ARCoSS

LNCS 8572

Javier Esparza
Pierre Fraigniaud
Thore Husfeldt
Elias Koutsoupias (Eds.)

Automata, Languages, and Programming

41st International Colloquium, ICALP 2014
Copenhagen, Denmark, July 8-11, 2014
Proceedings, Part I

1 Part I

EATCS

Springer

Lecture Notes in Computer Science 8572

Advanced Research in Computing and Software Science

Subline of Lectures Notes in Computer Science

Javier Esparza   Pierre Fraigniaud
Thore Husfeldt   Elias Koutsoupias (Eds.)

Automata, Languages, and Programming

41st International Colloquium, ICALP 2014
Copenhagen, Denmark, July 8-11, 2014
Proceedings, Part I

Springer

Volume Editors

Javier Esparza
Technische Universität München, Germany
E-mail: esparza@in.tum.de

Pierre Fraigniaud
LIAFA, Université Paris Diderot-Paris 7, France
E-mail: pierre.fraigniaud@liafa.univ-paris-diderot.fr

Thore Husfeldt
IT University of Copenhagen, Denmark
E-mail: thore@itu.dk

Elias Koutsoupias
University of Oxford, UK
E-mail: elias@cs.ox.ac.uk

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-662-43947-0 e-ISBN 978-3-662-43948-7
DOI 10.1007/978-3-662-43948-7
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014941765

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

This volume contains the papers presented at ICALP 2014: the 41st International Colloquium on Automata, Languages and Programming, held during July 8–11, 2014, at IT University of Copenhagen. ICALP is the main conference and annual meeting of the European Association for Theoretical Computer Science (EATCS) and first took place in 1972. This year the ICALP program consisted of three tracks:

- Track A: Algorithms, Complexity, and Games
- Track B: Logic, Semantics, Automata, and Theory of Programming
- Track C: Foundations of Networked Computation

In response to the call for papers, the three Program Committees received 484 submissions, a record number for ICALP. Track A received 319 submissions (another record), track B received 106 submissions, and track C received 59 submissions. Each submission was reviewed by at least three Program Committee members, aided by many subreviewers. The committee decided to accept 136 papers, which are collected in these proceedings. The selection was made by the Program Committees based on originality, quality, and relevance to theoretical computer science. The quality of the submissions was very high indeed, and many deserving papers could not be selected.

The EATCS sponsored awards for both a best paper and a best student paper for each of the three tracks, selected by the Program Committees.

The best paper awards were given to the following papers:

- Track A: Andreas Björklund and Thore Husfeldt, “Shortest Two Disjoint Paths in Polynomial Time”
- Track B: Joel Ouaknine and James Worrell. “Ultimate Positivity Is Decidable for Simple Linear Recurrence Sequences”
- Track C: Oliver Göbel, Martin Hoefer, Thomas Kesselheim, Thomas Schleiden, and Berthold Vöcking, “Online Independent Set Beyond the Worst-Case: Secretaries, Prophets, and Periods”

The best student paper awards, for papers that are solely authored by students, were given to the following papers:

- Track A: Sune K. Jakobsen, “Information Theoretical Cryptogenography”
- Track B: Michael Wehar, “Hardness Results for Intersection Non-Emptiness”
- Track C: Mohsen Ghaffari, “Near-Optimal Distributed Approximation of Minimum-Weight Connected Dominating Set”

Apart from the contributed talks, the conference included invited presentations by Sanjeev Arora, Maurice Herlihy, Viktor Kuncak, and Claire Mathieu. Abstracts of their talks are included in these proceedings as well.

The program of ICALP 2014 also included presentation of the Presburger Award 2014 to David Woodruff, the EATCS Award 2014 to Gordon Plotkin, and the Gödel Prize to Ronald Fagin, Amnon Lotem, and Moni Naor.

Two satellite events of ICALP were held on 7 July, 2014:

- Trends in Online Algorithms (TOLA 2014)
- Young Researcher Workshop on Automata, Languages and Programming (YR-ICALP 2014)

We wish to thank all the authors who submitted extended abstracts for consideration, the members of the three Program Committees for their scholarly efforts, and all additional reviewers who assisted the Program Committees in the evaluation process. We thank the sponsors Springer-Verlag, EATCS, CWI Amsterdam, and Statens Kunstfond for their support, and the IT University of Copenhagen for hosting ICALP 2014.

We are also grateful to all members of the Organizing Committee and to their support staff.

The conference-management system EasyChair was used to handle the submissions, to conduct the electronic Program Committee meetings, and to assist with the assembly of the proceedings.

May 2014

Javier Esparza
Pierre Fraigniaud
Thore Husfeldt
Elias Koutsoupias

Organization

Program Committee

Dimitris Achlioptas	UC, Santa Cruz, USA
Pankaj Agrawal	Duke University, USA
Paolo Baldan	Università di Padova, Italy
Nikhil Bansal	Eindhoven University of Technology, The Netherlands
Michele Boreale	Università di Firenze, Italy
Tomas Brazdil	Masaryk University, Czech Republic
Gerth Stølting Brodal	Aarhus University, Denmark
Véronique Bruyère	University of Mons, Belgium
Jean Cardinal	Université libre de Bruxelles, Belgium
Ning Chen	Nanyang Technological University, Singapore
Giorgos Christodoulou	University of Liverpool, UK
Andrea Clementi	University of Rome Tor Vergata, Italy
Veronique Cortier	CNRS, Loria, France
Anuj Dawar	University of Cambridge, UK
Xiaotie Deng	Shanghai Jiaotong University, China
Ilias Diakonikolas	University of Edinburgh, UK
Benjamin Doerr	MPI Saarbrücken, Germany
Chaled Elbassioni	Masdar Institute, Abu Dhabi
Javier Esparza	TU München, Germany
Kousha Etessami	University of Edinburgh, UK
Panagiota Fatourou	University of Crete, Greece
Michal Feldman	Hebrew University, Israel
Maribel Fernandez	Kings College London, UK
Antonio Fernández Anta	Universidad Rey Juan Carlos, Spain
Amos Fiat	Tel Aviv University, Israel
Pierre Fraigniaud	CNRS and University of Paris Diderot, France
David Frutos Escrig	Complutense University of Madrid, Spain
Pierre Ganty	IMDEA Software Institute, Spain
Leszek Gasieniec	University of Liverpool, UK
Phillip Gibbons	Intel Labs, USA
Leslie Goldberg	University of Oxford, UK
Vipul Goyal	Microsoft, India
Peter Habermehl	LIAFA, University of Paris 7, France
Magnus Halldorsson	Reykjavik University, Iceland
Giuseppe Italiano	University of Rome Tor Vergata, Italy
Marcin Kaminski	University of Warsaw, Poland

Haim Kaplan	Tel Aviv University, Israel
Anna Karlin	University of Washington, USA
Ioardanis Kerenidis	University of Paris Diderot, France
Anne-Marie Kermarrec	Inria Rennes, France
Robert Kleinberg	Cornell University, USA
Michal Koucky	Czech Academy of Sciences, Czech Republic
Elias Koutsoupias	University of Oxford, UK
Robert Krauthgamer	Weizmann Institute, Israel
Manfred Kufleitner	University of Stuttgart, Germany
Sławomir Lasota	Warsaw University, Poland
James Lee	University of Washington, USA
Oded Maler	CNRS-VERIMAG, France
Sebastian Maneth	NICTA and UNSW, Australia
Madhavan Mukund	Chennai Mathematical Institute, India
Ashwin Nayak	University of Waterloo, Canada
Jens Palsberg	UCLA, USA
Gopal Pandurangan	Nanyang Technological University, Singapore
Boaz Patt-Shamir	Tel Aviv University, Israel
Andrea Pietracaprina	Università di Padova, Italy
Andrea Richa	Arizona State University, USA
Luís Rodrigues	Universidade Técnica de Lisboa, Portugal
Jared Saia	University of New Mexico, USA
Piotr Sankowski	University of Warsaw, Poland
Christian Scheideler	Universität Paderborn, Germany
Thomas Schwentick	TU Dortmund, Germany
Maria Serna	UP Catalunya, Spain
Sonja Smets	University of Amsterdam, The Netherlands
Christian Sohler	TU Dortmund, Germany
Jiri Srba	Aalborg University, Denmark
Jukka Suomela	Aalto University, Finland
Ryan Williams	Stanford University, USA
Philipp Woelfel	University of Calgary, Canada
Steve Zdancewic	University of Pennsylvania, USA

Additional Reviewers

Aaronson, Scott
Abe, Masayuki
Abraham, Ittai
Aceto, Luca
Adler, Isolde
Adsul, Bharat
Afshani, Peyman
Agarwal, Alekh
Agarwal, Rachit
Aghazadeh, Zahra
Agrawal, Shweta
Ajwani, Deepak
Akutsu, Tatsuya
Al-Humaimeedy, Abeer
Alamdari, Soroush
Alglave, Jade

Allender, Eric
Alon, Noga
Althaus, Ernst
Alves, Sandra
An, Hyung-Chan
Anagnostopoulos, Aris
Ananth, Prabhanjan
Andoni, Alex
Andoni, Alexandr
Ardenboim, Alon
Arkhipov, Alex
Asarin, Eugene
Aspnes, James
Atig, Mohamed Faouzi
Atserias, Albert
Augustine, John
Avron, Haim
Babichenko, Yakov
Bacci, Giorgio
Bacci, Giovanni
Bach, Eric
Balabonski, Thibaut
Banerjee, Abhishek
Barrington, David
Bartoletti, Massimo
Basset, Nicolas
Bavarian, Mohammad
Beame, Paul
Becchetti, Luca
Bei, Xiaohui
Belmonte, Rémy
Ben Avraham, Rinat
Ben-Amram, Amir
Berger, Eli
Berry, Jonathan
Bertrand, Nathalie
Berwanger, Dietmar
Bhaskar, Umang
Bitansky, Nir
Blazy, Olivier
Blesa, Maria J.
Blömer, Johannes
Bodirsky, Manuel
Bodlaender, Hans L.
Bodlaender, Marijke
Bogdanov, Andrej
Bojanczyk, Mikolaj
Boker, Udi
Bollig, Beate
Bollig, Benedikt
Bonamy, Marthe
Bonchi, Filippo
Boneh, Dan
Bonifaci, Vincenzo
Bonnet, Edouard
Bonsangue, Marcello
Bonsma, Paul
Borgström, Johannes
Boutsidis, Christos
Boyar, Joan
Boyle, Elette
Brakerski, Zvika
Brandstadt, Andreas
Braverman, Mark
Bremner, Michael
Brettell, Nick
Briet, Jop
Brihaye, Thomas
Broadbent, Anne
Brody, Joshua
Bruni, Roberto
Brzuska, Christina
Buchbinder, Niv
Buchin, Kevin
Buhrman, Harry
Byrka, Jaroslaw
Böhl, Florian
Cai, Yang
Caltais, Georgiana
Canetti, Ran
Canonne, Clément
Cao, Yixin
Carraro, Alberto
Cash, David
Ceccarello, Matteo
Chakrabarti, Amit
Chakraborty, Supratik
Chalermsook, Parinya
Chan, Hubert
Chan, Siu On

Chan, Timothy
Chandran, Nishanth
Charatonik, Witold
Chase, Melissa
Chatterjee, Krishnendu
Chechik, Shiri
Chekuri, Chandra
Chen, Jing
Chen, Xujin
Chen, Zhou
Cheval, Vincent
Choudhury, Ashish
Chow, Sherman S.M.
Chrobak, Marek
Chung, Kai-Min
Ciancia, Vincenzo
Cicalese, Ferdinando
Clavier, Christophe
Clemente, Lorenzo
Codenotti, Paolo
Cohen, Edith
Cohen, Sarel
Cohn, Henry
Colcombet, Thomas
Colini Baldeschi, Riccardo
Costello, Craig
Crescenzi, Pierluigi
Cryan, Mary
Cygan, Marek
Czerwiński, Wojciech
Dalmau, Victor
Damaschke, Peter
Damgård, Ivan
Dang, Thao
Dani, Varsha
Dasgupta, Bhaskar
Datta, Samir
David, Alexandre
De Bonis, Annalisa
de Caro, Angelo
De Caro, Angelo
De Liguoro, Ugo
de Wolf, Ronald
Decker, Normann
Degorre, Aldric
Delahaye, Benoit
Delling, Daniel
Delvenne, Jean-Charles
Delzanno, Giorgio
Denysyuk, Oksana
Dereniowski, Dariusz
Devanur, Nikhil
Devroye, Luc
Diaz, Josep
Dietzfelbinger, Martin
Diks, Krzysztof
Dima, Catalin
Diochnos, Dimitris
Dobrev, Stefan
Doerr, Carola
Doyen, Laurent
Driemel, Anne
Duflot, Marie
Dumitrescu, Adrian
Dupuis, Frédéric
Durand, Arnaud
Durand-Gasselin, Antoine
Durnoga, Konrad
Dvir, Zeev
Dyer, Martin
Edmonds, Jeff
Efremenko, Klim
Efthymiou, Charilaos
Ehrgott, Matthias
Ehsanfar, Ebrahim
Elbassioni, Khaled
Elberfeld, Michael
Elmasry, Amr
Elsässer, Robert
Emmi, Michael
Ene, Alina
Enea, Constantin
Enqvist, Sebastian
Eppstein, David
Epstein, Leah
Erlebach, Thomas
Escoffier, Bruno
Even, Guy
Fahrenberg, Uli
Fanelli, Angelo

Farshim, Pooya
Fefferman, Bill
Feige, Uriel
Fekete, Sándor
Fernau, Henning
Fijalkow, Nathanaël
Filiot, Emmanuel
Filmus, Yuval
Fiorini, Samuel
Firmani, Donatella
Fisman, Dana
Flammini, Michele
Forbes, Michael A.
Forejt, Vojtech
Fortnow, Lance
Fotakis, Dimitris
Fountoulakis, Nikolaos
Franciosa, Paolo
Frati, Fabrizio
Frieze, Alan
Fu, Hu
Fu, Zhiguo
Fábregas, Ignacio
Gaboardi, Marco
Gadducci, Fabio
Gaertner, Bernd
Galanis, Andreas
Galesi, Nicola
Gambs, Sebastien
Garg, Ankit
Gaspers, Serge
Gastin, Paul
Gavinsky, Dmitry
Gawrychowski, Pawel
Geck, Gaetano
Geeraerts, Gilles
Gelles, Ran
Genest, Blaise
Ghaffari, Mohsen
Giakkoupis, George
Giannakopoulos, Yiannis
Giannopoulou, Archontia
Giaquinta, Emanuele
Gierasimczuk, Nina
Gilbert, Seth
Gille, Marc
Giunti, Marco
Gkatzelis, Vasilis
Glacet, Christian
Glen, Amy
Gmyr, Robert
Gogacz, Tomasz
Goldberg, Paul
Gonzalez Vasco, Maria Isabel
Gopalan, Parikshit
Gorbunov, Sergey
Gorecki, Pawel
Gorgunov, Sergey
Gorla, Daniele
Grandoni, Fabrizio
Greco, Gianluigi
Green, Oded
Grenet, Bruno
Grigorescu, Elena
Grigoryev, Dmitry
Grossi, Roberto
GualÃ, Luciano
Guo, Heng
Guo, Jiong
Guo, Siyao
Guruswami, Venkatesan
Gutwenger, Carsten
Göbel, Andreas
Haeupler, Bernhard
Haghpanah, Nima
Haitner, Iftach
Hajiaghayi, Mohammadtaghi
Hansen, Kristoffer Arnsfelt
Hansen, Thomas Dueholm
Hardt, Moritz
Harju, Tero
Harrow, Aram
Harsha, Prahladh
Hatami, Hamed
Haviv, Ishay
Hayes, Thomas
Hazay, Carmit
He, Meng
Heam, Pierre-Cyrille
Heggernes, Pinar

Helmi, Maryam
Hirschkoff, Daniel
Hlout, Loc
Hoefer, Martin
Hoffmann, Hella-Franziska
Hofheinz, Dennis
Hofman, Piotr
Huang, Chien-Chung
Huang, Sangxia
Huang, Xiangru
Huang, Zhiyi
Hunter, Paul
Husfeldt, Thore
Im, Hyeonseung
Indyk, Piotr
Iovino, Vincenzo
Irani, Sandy
Isopi, Marco
Ito, Takehiro
Jacob, Riko
Jain, Rahul
Jansen, Bart M.P.
Jao, David
Jerrum, Mark
Jeż, Artur
Jeż, Łukasz
Jiang, Minghui
Jiang, Zhansheng
Joret, Gwenaël
Joux, Antoine
Jurdzinski, Tomasz
Jørgensen, Allan Grønlund
Kakimura, Naonori
Kantor, Erez
Kao, Ming-Yang
Kapralov, Michael
Kapur, Deepak
Kara, Ahmet
Karakostas, George
Karhumäki, Juhani
Kausch, Jonathan
Kavitha, Telikepalli
Kawamura, Akitoshi
Kayal, Neeraj
Keller, Orgad
Kerber, Michael
Kesselheim, Thomas
Khandekar, Rohit
Kiefer, Stefan
King, Valerie
Kiraly, Tamas
Klauck, Hartmut
Klein, Philip
Klima, Ondrej
Klin, Bartek
Klivans, Adam
Kniesburges, Sebastian
Kobayashi, Yusuke
Kobourov, Stephen
Koebler, Johannes
Koiran, Pascal
Kolay, Sudeshna
Kolliopoulos, Stavros
Komjathy, Julia
Kontchakov, Roman
Kopczyński, Eryk
Kopelowitz, Tsvi
Kopparty, Swastik
Kortsarz, Guy
Kosowski, Adrian
Kosub, Sven
Kothari, Nishad
Kothari, Pravesh
Koutis, Ioannis
Koutsopoulos, Andreas
Kovacs, Annamaria
Kratsch, Stefan
Krcal, Jan
Kretinsky, Jan
Krishnaswamy, Ravishankar
Krivosija, Amer
Krug, Robert
Krysta, Piotr
Kucera, Antonin
Kulikov, Alexander
Kulkarni, Janardhan
Kulkarni, Raghav
Kumar, Akash
Kumar, Amit
Kumar, K. Narayan

Kuperberg, Denis
Kurz, Denis
Kyropoulou, Maria
Labourel, Arnauld
Lachish, Oded
Laekhanukit, Bundit
Lagniez, Jean Marie
Lanik, Jan
Laura, Luigi
Lauria, Massimo
Lauriere, Mathieu
Laursen, Simon
Lauser, Alexander
Le Gall, Francois
Le Scouarnec, Nicolas
Lee, James
Lee, Troy
Leonardos, Nikos
Lerays, Virginie
Leroux, Jerome
Levavi, Ariel
Levin, Asaf
Levy, Jean-Jacques
Lewenstein, Moshe
Li, Jian
Li, Minming
Li, Shi
Li, Yi
Li, Yingkai
Libert, Benoit
Libkin, Leonid
Lime, Didier
Lin, Anthony Widjaja
Lin, Chengyu
Liu, Feng-Hao
Llana, Luis
Lodaya, Kamal
Lohrey, Markus
Lopez-Ortiz, Alejandro
Loreti, Michele
Lotker, Zvi
Lovett, Shachar
Lozin, Vadim
Lu, Pinyan
Lu, Steve
Lucier, Brendan
Löding, Christof
M.S., Ramanujan
Ma, Minghui
Magniez, Frederic
Mahdian, Mohammad
Mahmoody, Mohammad
Makarychev, Konstantin
Makarychev, Yury
Maletti, Andreas
Malizia, Enrico
Mallmann-Trenn, Frederik
Manea, Florin
Maneva, Elitza
Mansour, Yishay
Mardare, Radu
Markey, Nicolas
Markou, Euripides
Martens, Wim
Martin, Barnaby
Martin, Russell
Marx, Dániel
Marx, Maarten
Masopust, Tomas
Mathieson, Luke
Matulef, Kevin
May, Alexander
Mayr, Richard
McColl, Robert
McGregor, Andrew
McSherry, Frank
Megow, Nicole
Meier, Arne
Meiklejohn, Sarah
Meir, O.
Mendel, Manor
Meng, Xianmeng
Mens, Irini-Eleftheria
Mertzios, George
Meunier, Pierre-Etienne
Miao, Peihan
Michail, Dimitrios
Michalak, Tomasz
Mignot, Ludovic
Milanic, Martin

Milchtaich, Igal
Miltersen, Peter Bro
Misra, Pranabendu
Molinero, Xavier
Monemizadeh, Morteza
Monmege, Benjamin
Montanaro, Ashley
Montecchiani, Fabrizio
Montenegro, Ravi
Moore, Cristopher
Moran, Tal
Morere, Philippe
Morris, Ben
Morsy, Ehab
Moseley, Benjamin
Movahedi, Mahnush
Mucha, Marcin
Munagala, Kamesh
Munteanu, Alexander
Murawski, Andrzej
Murlak, Filip
Muscholl, Anca
Mvprao
Nagaj, Daniel
Nanongkai, Danupon
Narayan Kumar, K.
Narodytska, Nina
Natale, Emanuele
Nathan, Lemons
Navara, Mirko
Navarra, Alfredo
Nederlof, Jesper
Neiman, Ofer
Nekrich, Yakov
Newman, Alantha
Nguyen, Hung Son
Nguyen, Huy
Nguyen, Trung Thanh
Niehren, Joachim
Nielsen, Jesper Sindahl
Niewerth, Matthias
Nikishkin, Vladimir
Nikoletseas, Sotiris
Nikolov, Aleksandar
Nissim, Kobbi
Niwinski, Damian
Nordstrom, Jakob
Novotný, Petr
Nowotka, Dirk
Nutov, Zeev
Nuñez Chiroque, Luis
O'Donnell, Ryan
O'Neill, Adam
Obdrzalek, Jan
Ogierman, Adrian
Olesen, Mads C.
Oliveira, Igor
Onak, Krzysztof
Ong, Luke
Ortega-Mallén, Yolanda
Ortmann, Mark
Ossona De Mendez, Patrice
Oualhadj, Youssouf
Paes Leme, Renato
Pagh, Rasmus
Palomino, Miguel
Paluch, Katarzyna
Pan, Jiangwei
Pan, Jiaxin
Panagiotou, Konstantinos
Panangaden, Prakash
Pandey, Omkant
Panigrahi, Debmalya
Papadopoulos, Dimitrios
Papakonstantinou, Periklis
Paparas, Dimitris
Parys, Pawel
Pasquale, Francesco
Pastro, Valerio
Patt-Shamir, Boaz
Paulusma, Daniel
Pauly, Arno
Pavan, A.
Peikert, Christopher
Peng, Pan
Peressotti, Marco
Peretz, Ron
Perez, Guillermo
Perifel, Sylvain
Perrin, Dominique

Peserico, Enoch
Pettie, Seth
Peña, Ricardo
Picaronny, Claudine
Pieris, Andreas
Pighizzini, Giovanni
Pilipczuk, Marcin
Pilipczuk, Michal
Pin, Jean-Eric
Plandowski, Wojciech
Polychroniadou, Antigoni
Pottier, Franois
Pottonen, Olli
Pous, Damien
Pozzato, Gian Luca
Prabhakar, Pavithra
Praveen, M.
Price, Eric
Pruhs, Kirk
Pucci, Geppino
Pulina, Luca
Pérez, Jorge A.
Qiang, Ruixin
Qiao, Youming
Quyen, Vuong Anh
Rabani, Yuval
Rabie, Mikael
Raecke, Harald
Raghavendra, Prasad
Raghunathan, Ananth
Raghvendra, Sharathkumar
Rahaman, Anisur
Rampersad, Narad
Raskin, Jean-François
Raz, Ran
Regev, Oded
Rehak, Vojtech
Reynier, Pierre-Alain
Riba, Colin
Richerby, David
Riondato, Matteo
Robinson, Peter
Roditty, Liam
Rodriguez, Ismael
Roetteler, Martin
Roland, Jérémie
Romano, Paolo
Ron, Dana
Rosa-Velardo, Fernando
Rosołek, Robert
Rossi, Gianluca
Rossmanith, Peter
Rosulek, Michael
Rothvoss, Thomas
Rubin, Natan
Rubio, Fernando
Ruppert, Eric
Saad, George
Sablik, Mathieu
Sack, Joshua
Sadrzadeh, Mehrnoosh
Saha, Chandan
Salvati, Sylvain
Sammartino, Matteo
Sangnier, Arnaud
Sankur, Ocan
Santaroni, Federico
Santhanam, Rahul
Santocanale, Luigi
Santos, Nuno
Saptharishi, Ramprasad
Sarkar, Susmit
Satti, Srinivasa Rao
Sau, Ignasi
Sauerwald, Thomas
Saurabh, Saket
Sawada, Joe
Saxena, Nitin
Scarpa, Giannicola
Scheder, Dominik
Schmidt, Melanie
Schmidt-Schauss, Manfred
Schmitz, Sylvain
Schneider, Stefan
Schroder, Dominique
Schröder, Lutz
Schuster, Martin
Schwartz, Roy
Schweikardt, Nicole
Schwiegelshohn, Chris

Schwoon, Stefan
Servais, Frédéric
Servedio, Rocco
Seshadhri, C.
Setzer, Alexander
Shah, Rahul
Shah, Simoni
Shamir, Ohad
Sharma, Vikram
Shen, Alexandre
Shenoy R., Gautham
Shpilka, Amir
Shraibman, Adi
Sidiropoulos, Anastasios
Siebertz, Sebastian
Sikdar, Somnath
Silva, Alexandra
Silvestri, Riccardo
Singh, Mohit
Sitchinava, Nodari
Sitters, Rene
Skowron, Piotr
Sokolova, Ana
Solomon, Shay
Sommer, Christian
Sousi, Perla
Spoerhase, Joachim
Sramek, Rastislav
Srinivasan, Srikanth
Srivastava, Piyush
Srivathsan, B.
Stachowiak, Grzegorz
Staiger, Ludwig
Stainer, Julien
Starikovskaya, Tatiana
Stefankovic, Daniel
Stehle, Damien
Stephan, Frank
Stergiou, Christos
Stoddard, Greg
Strassburger, Lutz
Straubing, Howard
Strefler, Mario
Strejcek, Jan
Strothmann, Thim
Struth, Georg
Su, Le
Suchy, Ondrej
Sun, Xiaoming
Sun, Xiaorui
Suomela, Jukka
Suresh, S.P.
Syrgkanis, Vasilis
Sénizergues, Géraud
Ta-Shma, Amnon
Tamaki, Suguru
Tamir, Tami
Tan, Li-Yang
Tang, Bo
Tao, Yufei
Tarjan, Robert
Tavenas, Sébastien
Telle, Jan Arne
Terhal, Barbara
Terui, Kazushige
Terzi, Evimaria
Thaler, Justin
Thanh, Nguyen
Thapper, Johan
Thiagarajan, P.S.
Thilikos, Dimitrios
Thorup, Mikkel
Thraves, Christopher
Toledo, Sivan
Toledoii, Sivan
Tompits, Hans
Torres Vieira, Hugo
Torunczyk, Szymon
Toruńczyk, Szymon
Trevisan, Luca
Trivedi, Ashutosh
Tschudi, Daniel
Tulsiani, Madhur
Uehara, Ryuhei
Ulus, Dogan
Umans, Chris
Umboh, Seeun
Uno, Yushi
Upadhyay, Jalaj
Valiant, Gregory

Valiente, Gabriel
Valiron, Benoît
van Breugel, Franck
van Melkebeek, Dieter
Van Melkebeek, Dieter
van Stee, Rob
Varacca, Daniele
Vassilevska Williams, Virginia
Vegh, Laszlo
Velickovic, Boban
Venkitasubramaniam, Muthuramakrishnan
Ventre, Carmine
Verschae, Jose
Vidick, Thomas
Viet Tung, Hoang
Viglietta, Giovanni
Vijayaraghavan, Aravindan
Vilaça, Xavier
Visconti, Ivan
Viswanathan, Mahesh
Vogler, Walter
Volkovich, Ilya
Vrgoc, Domagoj
Wachter-Zeh, Antonia
Wahlström, Magnus
Walter, Tobias
Walukiewicz, Igor
Wang, Juntao
Wang, Kainan
Wanka, Rolf
Watson, Thomas
Wee, Hoeteck
Weinstein, Omri
Weiss, Armin
Westermann, Matthias
Whistler, William
Wieder, Udi
Wiese, Andreas
Wilkinson, Bryan T.
Wilson, David
Winslow, Andrew
Witek, Maximilian
Witkowski, Piotr
Wollan, Paul
Wong, Prudence W.H.
Woodruff, David
Wootters, Mary
Wright, John
Wrochna, Marcin
Wu, Xiaodi
Wulff-Nilsen, Christian
Wullschleger, Juerg
Xia, Ge
Xiao, Tao
Xie, Ning
Xing, Chaoping
Xu, Xiaoming
Xue, Guoliang
Yamada, Shota
Yamakami, Tomoyuki
Yamauchi, Yukiko
Yang, Kaiyu
Yao, Penghui
Yaroslavtsev, Grigory
Ye, Tao
Yekhanin, Sergey
Yi, Ke
Yiannakopoulos, Yiannis
Yin, Yitong
Yoshida, Yuichi
Young, Max
Yu, Huacheng
Yuen, Tsz Hon
Zacharias, Thomas
Zamani, Mahdi
Zang, Wenan
Zeh, Norbert
Zhang, Bingsheng
Zhang, Chihao
Zhang, Hongyang
Zhang, Jialin
Zhang, Jie
Zhang, Jin
Zhang, Shengyu
Zhang, Wuzhou
Zhang, Yong
Zhao, Zhiguang
Zhou, Hong-Sheng
Zhou, Yuan

Zhu, Zeyuan Allen
Ziegler, Martin
Zimand, Marius
Ziv-Ukelson, Michal
Zivny, Stanislav
Zuckerman, David
Zwick, Uri
Zych, Anna

Invited Talks

Overcoming the Intractability Obstacle in Unsupervised Learning

Sanjeev Arora

Computer Science, Princeton University

Abstract. Unsupervised learning—*i.e.*, learning with unlabeled data—is increasingly important given today's data deluge. Most natural problems in this domain—*e.g.* for models such as mixture models, HMMs, graphical models, topic models and sparse coding/dictionary learning — are NP-hard. Therefore researchers in practice use either heuristics or convex relaxations with no concrete approximation bounds. Several nonconvex heuristics work well in practice, which is also a mystery.

Recently, a sequence of results has shown that rigorous approaches leading to polynomial running time are possible for several of these problems. These involve sidestepping worst-case complexity via special assumptions on the input. Some of this work—*e.g.* for topic models—even leads to practical running times (50x faster than previous approaches). It has even become possible to analyse nonconvex optimization heuristics such as alternating minimization or kSVD.

The talk will be a survey of these new results, including topic modeling, sparse coding, and deep learning.

On the Glass Ceiling Effect in Social Networks

Claire Mathieu

CNRS, École Normale Supérieure

Abstract. The glass ceiling may be defined as "the unseen, yet unbreakable barrier that keeps minorities and women from rising to the upper rungs of the corporate ladder, regardless of their qualifications or achievements." Although undesirable, it is well documented that many societies and organizations exhibit a glass ceiling. In this paper we formally define and study the glass ceiling effect in social networks and provide a natural mathematical model that (partially) explains it. We propose a biased preferential attachment model that has two type of nodes, and is based on three well known social phenomena:

i) rich get richer (preferential attachment),
ii) minority of females (or other group) in the network, and
iii) homophily (preference to bond with similar people).

We prove that our model exhibits a strong glass ceiling effect and that all three conditions are necessary, *i.e.*, removing any one of them, will cause the model not to exhibit a glass ceiling effect. Additionally we present empirical evidence of student–mentor networks of researchers that exhibits all the above properties: female minority, preferential attachment, homophily and a glass ceiling.

Joint work with Chen Avin, Barbara Keller, Zvi Lotker, David Peleg, and Yvonne-Anne Pignolet.

Table of Contents – Part I

Invited Talks

Track A: Algorithms, Complexity, and Games

Table of Contents – Part II

Track B: Logic, Semantics, Automata, and Theory of Programming

Track C: Foundations of Networked Computing

Sporadic Solutions to Zero-One Exclusion Tasks

Eli Gafni and Maurice Herlihy

Computer Science Department, Brown University,Providence, RI 02912
mph@cs.brown.edu

Abstract. Zero-one exclusion is a family of distributed tasks indexed by n-bit Boolean *signatures* $b[0], \ldots b[n-1]$. We are interested in asynchronous computations where at most $n+1$ asynchronous processes participate. They communicate with one another by reading and writing a shared memory, and halt after choosing a Boolean value. If $m < n+1$ processes participate, then they must not all choose the value $b[m-1]$. If all $n+1$ processes participate, then they must not all choose the same value.

It is easy to show that some instances of zero-one exclusion are computationally difficult, in the sense that they cannot be solved by any algorithm in which asynchronous processes communicate by reading and writing a shared memory. Can we characterize the Boolean signatures for which zero-one exclusion does have an asynchronous read-write algorithm? We give a partial answer, which we feel is interesting because of the way it ties together distributed computability, combinatorial topology, and elementary number theory.

1 Introduction

This paper poses a simple-sounding puzzle, and uses that puzzle as a window into some perhaps unexpected connections between distributed computing, combinatorial topology, and certain Diophantine equations. This paper is part tutorial, and part open problem presentation. We do not present proofs that are available in other papers, and the proofs we do present are informal sketches intended to convey the spirit of the material to non-specialists. Although the results presented here are new, our goal is to convey in elementary terms some of the connections between these apparently dissimilar areas, and to attract the attention of a broader community to this area.

2 Model of Computation

A *distributed system* is a collection of *processes*, together with a communication medium, assumed here to be shared read-write memory. Each process executes a finite *protocol*. It starts in an initial state, and takes steps until it either *fails*, meaning it halts and takes no additional steps, or it *halts*, usually because it has completed the protocol. Each other kind of step is a local state transition along with a read or write to the shared memory. Processes are deterministic:

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 1–10, 2014.

each transition is determined by the process's current state and the state of the memory.

As noted, a process can *fail*, meaning that it simply stops taking steps. Execution is *asynchronous*, meaning that processes run at arbitrary, unpredictable speeds, and there is no bound on process step time. A failed process cannot be distinguished from a slow process. As many as n out of the $n+1$ processes may fail before taking a step: such processes are said not to *participate* in the protocol.

In distributed computing, the basic unit of computation is called a *task*. In a task, each process is given its own input value, the tasks communicate for a bounded number of steps, and each task eventually halts with its own output value. The task specification states which sets of outputs can be produced in response to which sets of inputs.

We are interested in *classifying* tasks according to their relative power. Given two tasks, can one be used to implement the other, or are they incomparable? One way to classify tasks is by *consensus number* [7], the largest number of processes that can use that task to solve consensus. Nevertheless, tasks that are too weak to solve consensus for two processes still have a rich structure that remains poorly understood.

In this paper, we investigate a family of such "sub-consensus" tasks, called the *zero-one* exclusion tasks. We use combinatorial arguments to investigate which tasks in this family can be solved in read-write memory. These solutions correspond to solutions to certain Diophantine equations.

3 Zero-One Exclusion

Zero-one exclusion [5] is a *family* of tasks, indexed by an n-bit Boolean *signature* $b[0], \ldots b[n-1]$, where each $b[i]$ is either 0 or 1. Any subset of processes can participate, ranging from 1 to $n+1$. Each process has no inputs, and halts with a Boolean value, subject to the following restrictions. First, if the number m of participating processes does not exceed n, then they may not *all* choose $b[m-1]$. Second, if all $n+1$ processes participate, then they must disagree: some must choose 0 and some must choose 1.

With respect to classification, the family of zero-one exclusion tasks lies between two well-known tasks, *set-agreement* [4] and *weak symmetry-breaking* (WSB) [6]. In the *set-agreement* task, each of the $n+1$ processes starts with an input, and each must halt with some process's input, such that no more than n distinct inputs are chosen. It is known that set agreement has no read-write protocol [9]. In the WSB task, the processes must choose binary outputs so that if all $n+1$ processes participate, at least one chooses 0 and one chooses 1. WSB is strictly weaker than set agreement, and is closely associated with the classical *renaming* task [1]. It is known that WSB has a read-write protocol when $n+1$ is not a prime power [3,10].

Some instances of zero-one exclusion are computationally equivalent. Any zero-one exclusion task with signature b can be transformed into its *complement*:

$\overline{b}[k] = 1 - b[k]$. Any protocol for one task can be trivially transformed into a protocol for its complement.

4 Elements of Combinatorial Topology

Our results are expressed in the language of elementary combinatorial topology. Here we review some basic concepts.

Definition 1. *Given a set S, and a family $\mathcal{A}$ of finite subsets of S, we say that $\mathcal{A}$ is an* abstract simplicial complex *on S if the following are satisfied:*

(1) if $X \in \mathcal{A}$, and $Y \subseteq X$, then $Y \in \mathcal{A}$;
(2) $\{v\} \in \mathcal{A}$, for all $v \in S$.

An element of S is a called a *vertex*, and an element of $\mathcal{A}$ is called a *simplex.* The set of all vertexes of $\mathcal{A}$ is denoted by $V(\mathcal{A})$. A simplex $\sigma \in \mathcal{A}$ is said to have *dimension* $|\sigma| - 1$. In particular, vertexes are 0-dimensional simplexes. We sometimes mark a simplex's dimension with a superscript: σ^n. A simplex of dimension n is sometimes called an *n-simplex.*

We usually use lower-case Latin letters to denote vertexes $(x, y, z, \ldots)$, lower-case Greek letters to denote simplexes $(\sigma, \tau, \ldots)$, and calligraphic font to denote simplicial complexes $(\mathcal{A}, \mathcal{B}, \ldots)$.

A simplex τ is a *face* of σ if $\tau \subseteq \sigma$, and a *proper face* if $\tau \subset \sigma$. If τ has dimension k, then τ is a *k-face* of σ.

We often add one or more *labels* to vertexes, $\lambda : V \to D$, where D is an arbitrary domain. In particular, Π is a set of process names, and the label $\mathit{name} : V \to \Pi$ associates each vertex with a process name. Here, every simplex is *properly colored* by these names: if $u, v \in \sigma$ and $u \neq v$, then $\mathit{name}(u) \neq \mathit{name}(v)$. Unless stated otherwise, the complexes considered here are properly colored by process names. For $F \subseteq \Pi$, σ_F is the face of σ labeled by names from F. A *black-and-white* coloring β is a map $\beta : V \to \{0, 1\}$. A simplex is *monochromatic* if β maps its vertexes to the same color.

A *simplicial map* $\mathcal{M} : \mathcal{A} \to \mathcal{B}$ carries vertexes of complex $\mathcal{A}$ to vertexes of complex $\mathcal{B}$ so that every simplex of $\mathcal{A}$ maps to a simplex of $\mathcal{B}$. The simplicial map $\mathcal{M} : \mathcal{A} \to \mathcal{B}$ is *color-preserving* if $\mathit{name}(v) = \mathit{name}(\mathcal{M}(v))$ for every vertex v in $\mathcal{A}$.

Following Munkres [11], a *geometric n-simplex* is the convex hull of a set of $n + 1$ affinely-independent points in N-dimensional Euclidean space, for some $N \geq n$. A *geometric complex* is a collection of geometric simplexes closed under containment such that every pair of distinct simplexes has disjoint interiors. The point-set occupied by a simplex σ or complex $\mathcal{C}$ is denoted $|\sigma|$ or $|\mathcal{C}|$, and is called its *polyhedron.* A *subdivision* of a simplex σ is a complex $\mathcal{S}(\sigma)$ such that $|\mathcal{S}(\sigma)| = |\sigma|$.

Let Δ^n be an n-simplex properly colored by process names and $\mathit{Div}(\Delta^n)$ a subdivision, also properly colored. Define the *boundary complex* $\partial\Delta^n$ to be the set of faces of Δ of dimension less than n. Any subdivision $\mathit{Div}(\Delta^n)$ induces a subdivision $Div(\partial\Delta^n)$) of its boundary complex. An *orientation* of $Div(\Delta^n)$ is a way of assigning ±1 to each n-simplex so that two n-simplexes that share an $(n-1)$-face have opposite orientations. Any subdivision $Div(\Delta^n)$ has two possible orientations.

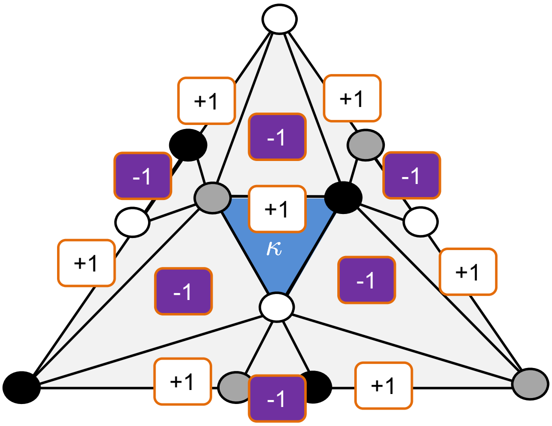


Fig. 1. Standard chromatic subdivision and standard orientation

For an n-simplex $\Delta^n = (s_0, \ldots, s_n)$, the *standard chromatic subdivision* $Ch(\Delta^n)$, illustrated in Fig. 1, is defined as follows. Each vertex of $Ch(\Delta^n)$ is a pair (s, ϕ), where *phi* is a face of Δ, and s a vertex of ϕ. A set of vertexes of $Ch(\Delta)$ define a simplex if for each pair (s, ϕ) and (s', ϕ'), $name(s)$ and $name(s')$ are distinct, and either $\phi \subseteq \phi'$, or $\phi' \subseteq \phi$. As s ranges over the vertexes of Δ^n, the vertexes (s, Δ^n) define the *central simplex* of $Ch(\Delta^n)$.

The *canonical* orientation of $Ch(\Delta^n)$ is defined by assigning a $+1$ orientation to the simplex $\{\langle s_0, \{s_0\}\rangle, \langle s_1, \{s_0, s_1\}\rangle, \ldots \langle s_n, \{s_0, \ldots, s_n\}\rangle\}$ as illustrated in Fig. 1. The orientation of any other simplex is determined by the parity of the number of "flips" across an $(n-1)$-face needed to get there from the starting simplex.

Lemma 1. *In the canonical orientation of $Ch(\Delta^n)$, the central simplex κ has orientation $(-1)^n$.*

Proof. As illustrated in Fig. 2, starting from the positively-oriented n-simplex α, where the vertex labeled with P_i lies in the central simplex of dimension i, it takes n flips to get to κ.

Lemma 2. *Let $\Delta^k \subset \Delta^n$ be a proper k-face, κ^n the central simplex of $Ch(\Delta^n)$, κ^k the central simplex of $Ch(\Delta^k)$, κ^{n-k-1} the face of κ^n labeled with the complement of the process names labeling κ^k, and $\kappa^n_k = \kappa^{n-k-1} \cup \kappa^k$ (see Fig. 3). In the canonical orientation of $Ch(\Delta^n)$, κ^n_k has orientation $(-1)^{n+1}$.*

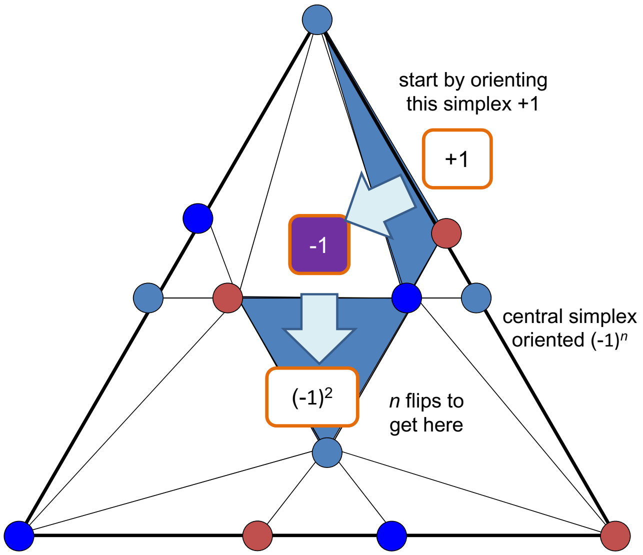


Fig. 2. The central simplex of $Ch(\Delta^n)$ has orientation $(-1)^n$

Proof. Starting from the central simplex κ^n, with orientation $(-1)^n$, we can reach κ^n_k with $2k+1$ flips as illustrated in Fig. 3, yielding an orientation of $(-1)^{n+2k+1} = (-1)^{n+1}$.

Let Δ^n be an n-simplex, $Div(\Delta^n)$ a subdivision, and $\chi : Div(\Delta^n) \to \{0,1\}$ a black-and-white coloring. The *content* of a c-monochromatic n-simplex $\sigma \in Div(\Delta^n)$ with orientation $d \in \{\pm 1\}$ is defined to be is defined to be $(-1)^{c \cdot n} d$. The content of the subdivision, $C(Div(\Delta^n), \chi)$ is sum of the contents of its monochromatic n-simplexes.

5 Read-Write Solvability

We now show that the question whether a particular zero-one exclusion task has a read-write protocol is equivalent to the question whether a particular black-and-white coloring exists. In this way, questions about concurrent computation, that is, a dynamic process that unfolds in time, can be reformulated in terms of questions about static combinatorial objects.

The next theorem, which follows from the *asynchronous computability theorem* of Herlihy and Shavit [9] [8, Ch.11], provides a first step.

Theorem 1. *An $(n+1)$-process zero-one exclusion task b has a read-write protocol if and only if there is a chromatic subdivision Div and a simplicial map χ,*

$$\chi : Div(\Delta^n) \to \{0,1\}$$

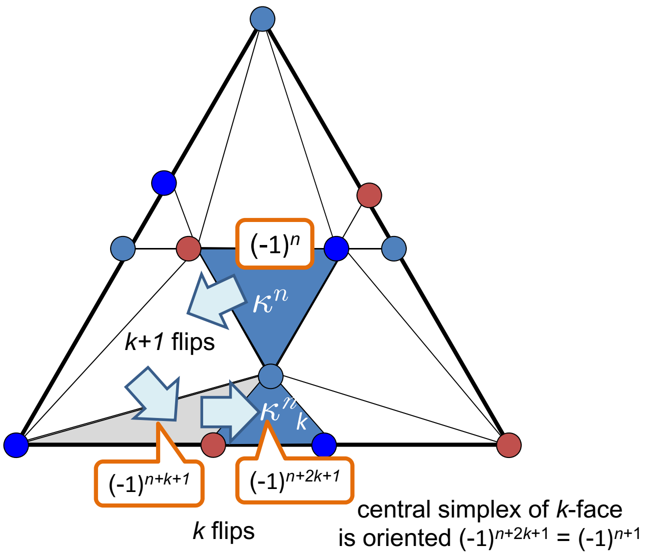


Fig. 3. It takes $k+1$ flips to carry κ^n to σ^n_k, and k more to get to κ^n_k

such that (1) for every proper k-face $\Delta^k \subset \Delta^n$, $Div(\Delta^k)$ has no $b[k]$-monochromatic k-faces, and (2) $Div(\Delta^n)$ contains no monochromatic n-faces.

Simplifying somewhat, the n-simplex Δ^n represents the starting configuration for the $n+1$ processes. The subdivision $Div(\Delta^n)$ represents all possible executions of a finite protocol. Each vertex in $Div(\Delta)$ represents a final process state when it halts at the end of its own execution. Each n-simplex $\sigma^n \in Div(\Delta^n)$ represents a possible execution in which all $n+1$ processes participate: the vertexes of σ^n are the participating processes' final states at the end of that execution. For a set of $(k+1)$ processes $F \subseteq \Pi$, recall that Δ^k_F is the k face of Δ^n labeled by names from F. Each k-simplex $\sigma^k \in Div(\Delta^k_F)$ represents a possible execution in which all and only the $k+1$ processes in F participate: the vertexes of σ^k are the participating processes' final states at the end of that execution. The map χ represents each process's decision: if a process finishes the protocol with final state (vertex) v, it chooses as Boolean output the value $\chi(v)$. The requirement that $Div(\Delta^k)$ has no $b[k]$-monochromatic k-faces is just a combinatorial way of stating that if k processes participate, they must not all choose output value $b[k]$, and the requirement that $Div(\Delta^n)$ contains no monochromatic n-faces just states that if all processes participate, they must not all choose the same output value.

While Theorem 1 is a complete characterization, it is not clear how one can use it to determine whether a particular zero-one exclusion task has a read-write protocol. Our next step is to shift from counting monochromatic simplexes to counting them by orientation, through the content $C(Div(\Delta^n), \chi)$, which is easier to compute. Clearly, if $C(Div(\Delta^n), \chi)$ is non-zero, then $Div(\Delta^n)$ contains at least one monochromatic n-simplex. In general, the converse is false: a content of zero does not mean there are no monochromatic n-simplexes, only that their orientations cancel. Nevertheless, Castanñeda and Rajsbaum [3] have shown that by adjusting the subdivision in a way that leaves the boundary unchanged, it is possible to make pairs of monochromatic n-simplexes of opposite orientation "cancel out", eventually leaving none. More precisely, if $C(Div(\Delta^n,)\chi)$ is zero, then there is a subdivision and black-and-white coloring

$$\chi' : Div'(\Delta^n) \to \{0, 1\}$$

where the boundaries agree: $Div'(\partial\Delta^n) = Div(\partial\Delta^n)$, χ and χ' agree on that boundary, but $Div'(\Delta^n)$ contains no monochromatic n-simplexes.

We can now reformulate Theorem 1, replacing the ban on monochromatic n-simplexes with a restriction on content.

Theorem 2. *An $(n+1)$-process zero-one exclusion task b has a read-write protocol if and only if there is a chromatic subdivision Div and a simplicial map χ,*

$$\chi : Div(\Delta^n) \to \{0, 1\}$$

such that (1) for every proper k-face $\Delta^k \subset \Delta^n$, $Div(\Delta^k)$ has no $b[k]$-monochromatic k-faces, and (2) $C(Div(\Delta^n), \chi)$ is zero.

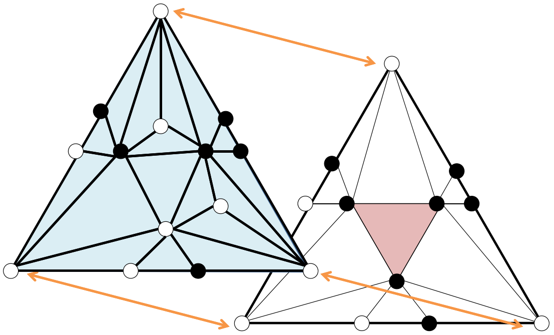


Fig. 4. Replacing the interior of a subdivision with a single simplex without changing the content

The next step is to replace the unknown subdivision $Div(\Delta^n)$ with a known subdivision, namely $Ch(\Delta^n)$, the standard chromatic subdivision shown in Fig. 1. A classical result, called the *Index Lemma* [1], states that the content $C(Div(\Delta^n), \chi)$ depends only on how the coloring χ behaves on $Div(\partial\Delta^n)$, the boundary of the subdivision. (The exact formula need not concern us here.) As a result, as long as

[1] The version of the classical Index Lemma used here is adapted from Herlihy et. al. ([8, Ch. 12].

we leave the subdivision of the boundary unchanged, we can rearrange the interior of $Div(\Delta^n)$ without changing the content.

Let

$$\chi : Div(\Delta^n) \to \{0,1\}$$

be a black-and-white coloring satisfying the conditions of Theorem 2. We give a new series of transformations changing $Div(\Delta^n)$ into $Ch(\Delta^n)$ and χ into χ_0 without changing the content. Let κ be an n-simplex properly colored with names from Π. For each set of process names $F \subset \Pi$, let $\Delta_{\overline{F}}$ be the face of Δ^n labeled by process names *not* in F. Define $Div_n(\Delta^n)$ to be the subdivision constructed by joining each face $\kappa_F \subset \kappa$ to $Div(\Delta_{\overline{F}})$. Define

$$\chi_n : Div_n(\Delta^n) \to \{0,1\}$$

to be 0 on vertexes of κ, and χ on $\partial Div(\Delta^n)$. Because the boundary complex is unaffected, the content is unchanged: $C(Div(\Delta^n), \chi) = C(Div_n(\Delta^n), \chi_n)$. Note that every n-simplex of $Div_n(\Delta^n)$ intersects κ, so $Div_n(\Delta^n)$ contains no 1-monochromatic n-simplexes.

Moving down one dimension, if $b[n]$ is 0, then for every $(n-1)$-face Δ^{n-1} of Δ^n, $Div_n(\Delta^{n-1})$ contains no 0-monochromatic $(n-1)$-simplexes, and contributes nothing to the content. We use the same construction to replace the interior of $Div_{n-1}(\Delta^{n-1})$ with a single 1-monochromatic simplex, leaving $\partial Div_{n-1}(\Delta^{n-1})$ unchanged. If $b[n]$ is 1, then for every $(n-1)$-face Δ^{n-1} of Δ^n, $Div_n(\Delta^{n-1})$ contains no 1-monochromatic $(n-1)$-simplexes, so each face's contribution to the content is its number of 0-monochromatic $(n-1)$-simplexes, counted by orientation, which is just the content $C(Div_n(\Delta^{n-1}), \chi_n)$. We replace the interior of $Div_{n-1}(\Delta^{n-1})$ with a single 0-monochromatic simplex, leaving $\partial Div_{n-1}(\Delta^{n-1})$ unchanged. We have defined a new subdivision and coloring

$$\chi_{n-1} : Div_{n-1}(\Delta^n) \to \{0,1\}$$

with content $C(Div(\Delta^n), \chi) = C(Div_{n-1}(\Delta^n), \chi_{n-1})$.

Continuing in this way, $Div_0(\Delta^n) = Ch(\Delta^n)$, the standard chromatic subdivision, and $\chi_0 = \beta$, yielding the following theorem.

Theorem 3. *The $(n+1)$-process zero-one exclusion task with signature b has a read-write protocol if and only if there exists a black-and-white coloring*

$$\beta : \ Ch(\Delta^n) \ \to \ \{0,1\}$$

such that (1) for each $\Delta^k \subset \Delta^n$, the central simplex of $Ch(\Delta^k)$ is $(1-b[k])$-monochromatic, and (2) the content of $Ch(\Delta^n)$ is zero.

This construction replaces the subdivision and coloring of Theorem 2 with a specific subdivision and coloring, making it possible to compute the content. For each face $\Delta^k \subseteq \Delta^n$, we can compute the content $C(Ch(\Delta^k, \beta))$ in terms of the contents $C(Ch(\Delta^i, \beta)$, for the proper faces $\Delta^i \subset \Delta^k$. Because $b[0] = 1$, $C(Ch(\Delta^0, \beta)) = 0$ for every vertex Δ^0 of Δ^k. By Lemma 1, the central simplex

κ^k of $Ch(\Delta^k)$ contributes $(-1)^k$ to the content $C(Ch(\Delta^n), \beta)$. For each $\Delta^i \subset \Delta^k$, if $b[k-i-1]$ is zero, then $Ch(\Delta^i)$ contributes nothing. If $b[k-i-1]$ is one, then by Lemma 2, $Ch(\Delta^i)$ contributes $(-1)^{k+1} \cdot C(Ch(\Delta^{k-i-1}), \beta)$. There are $\binom{k+1}{i+1}$ such i-faces, yielding:

$$C(Ch(\Delta^k), \beta) = (-1)^k \cdot (1 - \sum_{i=0}^{k-1} b[k-i-1] \cdot \binom{k+1}{i+1} \cdot C(Ch(\Delta^{n-i-1}), \beta)).$$

Corollary 1. *The zero-one exclusion task with signature b has a read-write protocol exactly when*

$$0 = 1 - \sum_{k=0}^{n-1} b[n-k-1] \cdot \binom{n+1}{k+1} \cdot C(Ch(\Delta^{n-k-1}), \beta) \tag{1}$$

We will use the following fact from number theory.

Fact 4. *The binomial coefficients $\binom{n+1}{1}, \ldots, \binom{n+1}{n}$ are relatively prime if and only if $n+1$ is not a prime power [2].*

Lemma 3. *An $(n+1)$-process read-write protocol for zero-one exclusion task b exists only if $n+1$ is not a prime power.*

Proof. A necessary (but not sufficient) condition to solve Equation 1 is that the binomial coefficients $\binom{n+1}{k+1}$ be relatively prime, for $0 \le k \le n-1$.

This condition can be strengthened slightly.

Lemma 4. *An $(n+1)$-process read-write protocol for zero-one exclusion task b exists only if the binomial coefficients $\binom{n+1}{k+1}$ can be partitioned into two sets S_0 and S_1 such that the values in each set are relatively prime.*

Proof. Let

$$S_0 = \left\{ \binom{n+1}{k+1} \quad : \quad b[k] = 1 \right\}.$$

These terms must be relatively prime to satisfy Equation 1. Since S_1 plays the same role for the complementary zero-one exclusion task $\overline{b}$, the same observation holds.

Lemma 5. *There exist zero-one exclusion tasks with read-write protocols.*

Proof. It is easy to write an inefficient program to search for solutions. The smallest dimension for which a solution exists is 29. Here is one:

1110 0110 0010 0110 1101 1100 1100 01

Counting complementary solutions, there are 16 solutions of dimension 29, none at dimensions 30 or 31 (31 and 32 are prime powers), 4 at dimension 32, 58 at dimension 33, 32 at dimension 34, and 80 at dimension 35, where our search stopped.

6 Conclusions

By presenting this only partially-solved puzzle, we intend to illustrate how the application of combinatorial topology to concurrent computing can lead in some perhaps surprising directions. We have barely touched on the most interesting problem: given a set of "black boxes" that solve instances of zero-one exclusion, can we compose them to construct solutions to other instances? We can ask this question of any family of tasks, and these problems are key to understanding concurrent computability.

References

1. Attiya, H., Bar-Noy, A., Dolev, D., Peleg, D., Reischuk, R.: Renaming in an Asynchronous Environment. Journal of the ACM (July 1990)
2. Attiya, H., Castañeda, A., Herlihy, M., Paz, A.: Upper bound on the complexity of solving hard renaming. In: Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing, PODC 2013, pp. 190–199. ACM, New York (2013)
3. Castañeda, A., Rajsbaum, S.: New combinatorial topology bounds for renaming: the lower bound. Distributed Computing 22, 287–301 (2010), 10.1007/s00446-010-0108-2
4. Chaudhuri, S.: Agreement Is Harder Than Consensus: Set Consensus Problems in totally asynchronous systems. In: Proceedings of The Ninth Annual ACM Symosium on Principles of Distributed Computing, pp. 311–234 (August 1990)
5. Gafni, E.: The 0—1-exclusion families of tasks. In: Baker, T.P., Bui, A., Tixeuil, S. (eds.) OPODIS 2008. LNCS, vol. 5401, pp. 246–258. Springer, Heidelberg (2008)
6. Gafni, E., Rajsbaum, S., Herlihy, M.P.: Subconsensus Tasks: Renaming Is Weaker Than Set Agreement. In: Dolev, S. (ed.) DISC 2006. LNCS, vol. 4167, pp. 329–338. Springer, Heidelberg (2006)
7. Herlihy, M.: Wait-free synchronization. ACM Trans. Program. Lang. Syst. 13(1), 124–149 (1991)
8. Herlihy, M., Kozlov, D., Rajsbaum, S.: Distributed Computing Through Combinatorial Topology. Elsevier Science (2013)
9. Herlihy, M., Shavit, N.: The topological structure of asynchronous computability. J. ACM 46(6), 858–923 (1999)
10. Kozlov, D.N.: Weak symmetry breaking and abstract simplex paths (2013) (preprint)
11. Munkres, J.: Elements of Algebraic Topology, 2nd edn. Prentice Hall (January 1984)

Verifying and Synthesizing Software with Recursive Functions

(Invited Contribution)

Viktor Kuncak*

École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
{firstname.lastname}@epfl.ch

Abstract. Our goal is to help people construct software that does what they wish. We develop tools and algorithms that span static and dynamic verification, constraint solving, and program synthesis. I will outline the current state our verification and synthesis system, *Leon*, which translates software into a functional language and uses SMT solvers to reason about paths in programs and specifications. Certain completeness results partly explain the effectiveness of verification and synthesis procedures implemented within Leon, in particular results on decidability of sufficiently surjective abstraction functions, and the framework of complete functional synthesis.

1 Introduction

Software is more widespread than ever, thanks to trends such as mass adoption of smartphones and tablets, as well as complex software controllers in, e.g. personal vehicles. At the same time, it is still too difficult to construct software that does something meaningful, let alone enforcing that software conforms to rigorous correctness standards. This motivates our current research on making software construction more accessible to large user bases, as well as increasing the confidence in software artifacts being constructed. These tasks require automated reasoning about requirements, specifications, and implementations. A core problem is automatically mapping users' requirements into efficiently executing systems. The problem has traditionally found home in programming languages, formal methods, software engineering, and design automation, but is also related to automated reasoning, human-computer interaction, machine learning, and natural language processing. We can evaluate our progress on addressing this problem by building software development tools that help software developers and users. Their development often required new algorithms for program verification, analysis, program synthesis, and new decision procedures.

In the sequel I use an example to illustrate several usage scenarios of interest and define the corresponding algorithmic questions. The examples use the syntax

* This work is supported in part by the European Research Council (ERC) Project *Implicit Programming* and the Swiss National Science Proposal *Constraint Solving Infrastructure for Program Analysis.*

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 11–25, 2014.

of the Scala programming language (`http://scala-lang.org/`, [59,60]) and are mostly based on the capabilities of the Leon verification and synthesis system for a subset of Scala (`http://leon.epfl.ch`) on which we have been working for the past few years [10,17,35,40,45–47,52,74–76], but we have also explored related ideas in several other works [15,16,24,26–28,31,32,34,41–44,49,50,56, 63,64,67,70,71,77–81].

2 Problem Definitions through Examples

Consider computations that take inputs $i \in I$ and produce outputs $o \in O$. We view a program as a function $f : I \to O$ from inputs to outputs. We view a specification $P : (I \times O) \to \{\text{false}, \text{true}\}$ as a predicate that takes a potential input and a potential output and returns true iff the output is considered acceptable for the given input. We fix this notation throughout this section.

2.1 Four Types of Problems

We consider the following categories of problems:

(RV) **Runtime Verification**: given P, f and a specific input $i_1 \in I$, compute the value $f(i_1)$, then compute whether $P(i_1, f(i_1))$ holds.

(SV) **Static Verification**: given P, f, either prove $\forall i \in I.\, P(i, f(i))$, or find a counterexample $i_1 \in I$ such that $\neg P(i_1, f(i_1))$.

(RC) **Runtime Constraint solving**: given P and an input $i_1 \in I$, find one corresponding output $o_1 \in O$ such that $P(i_1, o_1)$. This succeeds iff $\exists o.\, P(i_1, o)$.

(SC) **Static Constraint solving**: given P, find a computable function f such that $\forall i \in I.\ P(i, f(i))$. This is a form of *program synthesis* [46,54].

The classification uses two criteria. The first criterion (R/S) is whether the problem being solved takes place at *runtime* (R), that is, during program execution, or *statically* (S), at compile time, before the program runs. The second criterion (V/C) is whether the task is verification (V), when both the program and the specification are given, or constraint solving (C), when only the specification is given, and we aim to compute values that satisfy it.

A List Definition in Scala. To make the discussion more concrete, consider the simple example of describing operations on sorted lists of integers. We choose a purely functional subset of the Scala to implement these operations (`http://scala-lang.org/`, [59,60]). We rely only on a small subset of Scala, so our functions correspond to the mathematical notion of mutually recursive functions defined over discrete domains. Listing 1 shows the definition of lists as an algebraic data type with a zero-arity constructor *Nil* and a binary constructor *Cons* : (*Int* × *List*) → *List*. We define an algebraic data type in Scala using class inheritance; for pure functions this corresponds to a term algebra [30,53] with the corresponding constructors (here: *Nil* and *Cons*) and selectors (here: *head* and *tail*). The domain thus represents finite sequences of integers.

Insertion into the List. Listing 2 defines a recursive function *sortedIns* that inserts a given integer into the sorted list, while preserving the property of being sorted. This is a concrete example of a program denoted f above. The ***match*** construct performs the usual case analysis on whether the list is empty or not, and, in the non-empty case, binds the provided variables *x*, *xs* to the head and tail of the list. Such code follows a standard approach for defining recursively defined structures and functions. Our methodology uses this same executable language of recursive functions to also describe the desired properties P of programs.

```
abstract class List
case class Cons(head: Int, tail: List) extends List
case object Nil extends List
```

Listing 1. List of Integers Defined Using Custom Case Classes

```
def sortedIns(e: Int, l: List): List = l match {
    case Nil ⇒ Cons(e,Nil)
    case Cons(x,xs) ⇒ if (x ≤ e) Cons(x,sortedIns(e, xs)) else Cons(e, l) }
```

Listing 2. Conventional Implementation of Insertion into a Sorted List

```
def sortedIns(e: Int, l: List): List = {
  require(isSorted(l))
  l match {
    case Nil ⇒ Cons(e,Nil)
    case Cons(x,xs) ⇒ if (x ≤ e) Cons(x,sortedIns(e, xs)) else Cons(e, l) }
} ensuring(result ⇒ isSorted(res) && content(result) == content(l) ++ Set(e))
```

Listing 3. The Insertion into a Sorted List together with its Specification

Writing Specifications for Functions. Suppose that we wish to specify that *sortedIns* indeed inserts the given element into the set of elements it stores, and that it maintains the ordering of the elements in the list. Listing 3 shows how to write such specification in Scala. We indicate that the input list needs to be sorted using the ***require*** operator, which takes a predicate that should hold for function arguments (function precondition). In this example we use the *isSorted* predicate to define the precondition. Listing 4 shows the definition of *isSorted* as a recursive function. To specify the postcondition, we use the ***ensuring*** clause, which also takes a predicate, but this time involving not only parameters of the function, but also its result, here bound to the *result* variable. The specification in Listing 3 indicates that the result should also be a sorted list. Moreover, it says that the set of elements stored in the resulting list, *content(result)*, should be equal to the union of the content of the argument list *content(l)* and the singleton set $\{e\}$, denoted in Scala as *Set(e)*. Listing 5 defines the *content* function recursively. We call such function an abstraction function; it abstracts away from the list ordering and computes a set. Algebraically, it is a homomorphism, mapping values of the list algebraic data type with constructor operations into finite sets with union. The property P that we would like to ensure about the result of *sortedIns* when invoked with arguments e and l is thus:

$$\mathit{isSorted(l)} \rightarrow \big(\mathit{isSorted(result)} \land \mathit{content(result)} = \mathit{content(l)} \cup \{e\}\big)$$

In the terminology of our classification, the above predicate is the specification P. the input i is the tuple of arguments (e, l), the output o is *result*, and the function f is *sortedIns*. The overall correctness condition $P((e, l), f(e, l))$ is therefore:

$$\mathit{isSorted(l)} \rightarrow \big(\mathit{isSorted(sortedIns(e,l))} \land \mathit{content(sortedIns(e,l))} = \mathit{content(l)} \cup \{e\}\big) \tag{1}$$

Runtime Verification. Runtime verification checks the above correctness condition when the values of arguments (in our example, e and l) are known. In addition, before invoking the function *isSorted*, the program system checks its precondition, so the function body only executes when the assumption of the above implication is true. In general, we assume that both f and P are expressed in a language of computable recursive functions. Therefore, the applications of f and P to their arguments are simply computations according to the semantics of the language. As a result, runtime verification RV is a computable problem in our context. In fact, the ***require*** and ***ensuring*** are simply library functions of the Scala programming language [58], and runtime checking in principle requires no further support. Runtime checking is nonetheless not a trivial problem, because the specification P is often written aiming for clarity and provability, and not aiming for efficiency of evaluation. A naturally written specification often leads to naive and repeated computation if executed at runtime. Leon can substantially improve the predictability and performance of runtime checks using static techniques, as we discuss below.

```
def isSorted(l: List): Boolean = l match {
  case Cons(x, Cons(y, ys)) ⇒ x ≤ y && isSorted(Cons(y, ys))
  case _ ⇒ true }
```

Listing 4. Sortedness Property of a List

```
def content(l: List): Set[Int] = l match {
  case Cons(x, xs) ⇒ Set(x) ++ content(xs)
  case Nil ⇒ Set() }
```

Listing 5. Set of Elements Stored in a List

Static Verification. Whereas runtime verification checks (1) for the particular e, l, static verification attempts to prove it for all e, l. The Leon verifier takes the same input as for runtime checking, and attempts to either prove correctness of (1) or find a counterexample for its correctness. For this purpose, we build on the field of Satisfiability Modulo Theory (SMT) solvers [2,19,22], which contain decision procedures for many theories that support useful combinations of operations present in programs. For property (1), the unfolding of *isSorted* requires reasoning about algebraic data types, handled by the theory of recursive data types [3] and reasoning about the ordering on integers, handled by integer linear arithmetic on integers within Z3 [36]. Reasoning about operations on sets is handled using array combinators [20] although more expressive theories of sets with cardinalities were evaluated in previous versions of Leon [76]. A careful reader will observe that a specification using sets may be weaker than desired, because it allows a sorting routing to remove or introduce duplicates. A more precise specification can naturally be written using multisets, whose simple fragments can be also encoded using arrays combinators [20], and for which the development of decision procedures of optimal complexity is a result of relatively recent developments [62–64].

In contrast to operators built into Leon's language subset, recursively defined functions are not directly supported in SMT solvers, so Leon implements its own algorithm to handle them [73–75]. Leon's algorithm [75] is related to bounded model checking ideas [1,7] and k-induction [23,37], but applies to recursive functions. In our example, when proving correctness of the condition (1), the system performs satisfiability checks while increasingly unfolding the definitions. For a relatively small unfolding depth, in this example the formula becomes unsatisfiable. In other examples, the system finds a counterexample for certain unfolding depth. In general, it need not terminate because the problem is undecidable.

However, there are interesting classes of recursive functions for which the system is a decision procedure [73,74]. The class that we have considered have the form of homomorphism functions, such as the *content* function in Listing 5. We have identified a number of such functions, which we call "sufficiently surjective abstracts" in [74], thus deriving several families of extensions of term algebras for which satisfiability of quantifier-free formulas is decidable.

The work can also be viewed [34] from the point of view of Ψ-local theory extensions [33], where further decidable extensions of term algebras have been identified [68].

In general, particular recursive functions may require reasoning specialized to this class. What is remarkable is that for a class of sufficiently surjective abstraction functions, the unrolling (a form of bounded model checking for recursive functions) becomes a uniform decision procedure [73]. Therefore, we have a series of decidability results, but the underlying algorithm need not be aware of them, they simply ensure its completeness in some cases. Our experience suggests that the algorithm works in practice for many other cases as well, which suggests that there may be further completeness results to be discovered.

Function unfolding is justified for terminating functions, and corresponds to inductive reasoning according to the well-founded relation that implies termination of functions. Leon currently does not check termination by default; for some approaches to check termination, see [13,65]. Sufficiently surjective abstraction functions are, however, terminating by their syntactic structure. Note also that the properties that we are checking are safety properties, which means that we could generate logical encoding that uses relations instead functions and is not sensitive to termination. It remains to be investigated how much we would lose in practice by using relations instead of functions.

Inductive Generalization. In our experience, more properties turned out to be k-inductive than what we initially expected. The general-purpose algorithm of Leon, based on function unfolding, is therefore surprisingly effective in practice. For some cases, though, it fails to perform the required generalization and find an invariant that implies the desired property. To address these cases, we have started incorporating some of the ideas from model checking and constraint-based static analysis.

One approach is to search for invariants of a particular template form [6]. We have recently also implemented a refinement of such an approach in Leon and showed that it is able to compute worst-case execution bounds for sequential and parallel execution of functional programs [52], which often involve difficult-to-find numerical constants.

An alternative approach is to generalize predicate abstraction to recursive functions. State of the art methods use counterexample-guided refinement of the set of predicates, often based on interpolation. We have applied this approach to linear integer arithmetic models of programs [67]. The technique requires tree interpolants [29, 57], which generalizes interpolation problem to more complex cases of proving consistency of Horn clauses [8, 25, 66].

In both of these approaches to verification with inductive invariant inference, we were greatly influenced by the works of Andrey Rybalchenko.

Why an Executable Specification Language. The choice of executable predicates for specifications (as opposed to, for example, logic with quantifiers or dynamic logic) is somewhat restrictive but very practical. First, the class of properties is rather large, because we are allowed to use a Turing-complete language for predicates. Second, it is not an obscure language that happens to be Turing complete, but a functional language that is already used for implementations, and which many schools teach to undergraduates. Whereas software developers may be hesitant to use logical notation, here they just use assertions. Third, it is an executable language, which immediately enables runtime checking in program runs and test runs, as discussed above. It also leads to automated generate-and test approaches to bug finding, which are as complete as theoretically possible, given that the problem of finding counterexamples is recursively enumerable. The computable specifications approach is also related to reasons why bounded model checking is effective for such specifications. Finally, as we have seen, the absence of counterexamples can also be automated through inductive reasoning. For many of these reasons this has been a popular choice in other verification tools as well, most notably the ACL2 prover and its predecessors [12, 38, 39]. That said, given numerous other heavier-weight specification and verification approaches, we feel that the elegance and the advantages of the approach of using executable functions can never be emphasized enough.

Runtime Constraints Solving (Constraint Programming). Using the abstraction function and the invariant, we can concisely specify an *insert* operation for sorted lists using a constraint as in Listing 6.

```
def insert(l: List, v: Int) = {
  require(isSorted(l))
  choose{ (x: List) ⇒ isSorted(x) && (content(x) == content(l) ++ Set(v)) }
}
```

Listing 6. Insertion Specified using Constraint

Runtime constraint solving allows the developer to describe computations using predicates alone, avoiding the need to write an explicit function from inputs to inputs. In other words, they allow programming with "implicit" functions.

Observe that, given f we can define P that characterizes it by defining $P(x, y) \iff (f(x) = y)$. Therefore, input/output specifications (which are relations) subsume implementations (which we consider to be functions). They are more expressive because they can describe the desired properties of the output, without specifying it uniquely. This allows us to specify orthogonal properties separately and then combine them using conjunction to obtain a function as an intersection of several relations.

The advantages of specifications become even more apparent for more complex examples. The following method describes the insertion into a red-black tree.[1]

```
def insert(t: Tree, v: Int) = {
  require(isRedBlack(t))
  choose{ (x: Tree) ⇒ isRedBlack(x) && (content(x) == content(t) ++ Set(v)) }
}
```

In such scenario, the run-time waits until the argument t and the value v are known, and finds a new tree value x such that the constraint holds. Thanks to our constraint solver, which has a support recursive functions and also leverages the Z3 SMT solver, this approach works well for small red-black trees. It is therefore extremely useful for prototyping and testing and we have previously explored it as a stand-alone technique for constraint programming in Scala [43].

If we now considered writing a removal operation for trees, an approach based on conventional imperative or functional code would require writing a separate removal algorithm, which is non-trivial for red-black trees. Using specifications, the desired behavior is given simply by replacing ++ sign with -- sign:

```
def remove(t: Tree, v: Int) = {
  require(isRedBlack(t))
  choose{ (x: Tree) ⇒ isRedBlack(x) && (content(x) == content(t) -- Set(v)) }
}
```

[1] We omit here the definition of the tree invariant for brevity, which is non-trivial [14, 61], but still rather natural to describe using recursive functions.

Static Constraint Solving (synthesis). Analogously to verification, we would like to obtain efficiency and predictability advantages of static computation also in the case of implicitly defined computations. For this purpose, we aim to statically solve specifications and convert them into directly executable functions. This process is typically referred to as program synthesis [46, 54]. The synthesis techniques in Leon [40] heavily rely on the underlying verification techniques, but also on complete functional synthesis [35, 46–48]. Our current implementation of synthesis in Leon [40] is able to translate the specification into the complete implementation shown in Listing 7.

```
def insert(l: List, v: Int) = {
  require(isSorted(l))
  l match {
    case Cons(head, tail) ⇒
      if (v == head) {
        l
      } elseif (v < head) {
        Cons(v, l)
      } else {
        insert(t, v)
      }
    case Nil ⇒
      Cons(v, Nil)
  }
}
```

Listing 7. Result of Synthesis of Code Shown in Listing 6

Theoretical questions of completeness for synthesis have interesting connection to logic and automata. For example, synthesis for Presburger arithmetic turns out to be related to constructive quantifier elimination [46–48]. On the other hand, we can obtain better theoretical bounds for synthesized code using automata techniques [28,70]. Parameterized complexity of problems is likely to be important when theoretically characterizing when synthesis is useful, because we wish to consider the specification P and the input i as two distinct input parameters.

Our implemented technique [40] also builds on counterexample-guided approaches [69]. Techniques for learning representations in logic and automata [51] are a likely to have further fruitful applications in software synthesis.

In addition to synthesis over discrete domains, an important problem is synthesis of numerical computations that conform to the desired precision guarantees [17, 18].

2.2 Relationship between Different Problems

This classification gives an overview of typical different tasks, though some of the most interesting questions arise by considering combinations and relationships between these four problems.

Optimizing Runtime Checks Using Static Verification. The Leon verifier can remove the runtime checks that are provable statically, and transform the programs to avoid duplicate checks. This allows automatic static verification for as many properties as possible, but still allows the resulting programs to run and to detect if any leftover checks fail for the values arising during program use and testing. This is part of an ongoing work with Emmanouil Koukoutos at EPFL.

Invoking Static Verification at Runtime for Complex Programs. When program state is complex, static verification techniques face their limitations. It is therefore interesting to invoke static analyzers at runtime, in parallel with actual program executions. We have done this in the context of distributed system implementations [80] and Java programs (EPFL MSc thesis of Sebastian Gfeller), and it could also be done to perform eager checks of higher-order function contracts in functional programs.

Using Counterexamples of Static Verification for Constraint Solving. A very important connection shows that runtime constraint solving is a dual to static verification. We perform static verification in Leon, showing validity of $\forall i.P(i, f(i))$, by proving unsatisfiability of $C(x)$ defined as $\neg P(x, f(x))$. A satisfying assignment for $C(x)$ is a counterexample to the validity. Counterexample search is thus search for values that satisfy a constraint $C(x)$, expressed in logics that contain operations from theories, as well as recursive function invocations. On the other hand, runtime constraint solving tries to find, for a given i_1 a value o such that $P(i_1, o)$ holds. If we define $C'(x)$ as $P(i_1, x)$, then runtime constraint solving also corresponds to solving the constraints $C'(x)$ expressed in the same language as before. Therefore, in our work on constraint programming for Scala, we were able to use the same constraint solving implementation to solve constraints [43]. This mechanism is also avaible in the current Leon system [45], and allows us to execute very expressive specifications between inputs and outputs.

Combining Constraint Solving and Synthesis. Our deductive synthesis framework performs a search over different steps that transform specification into a set of potentially simpler ones. This architecture allows us to combine synthesis and run-time constraint solving. We illustrate this using an example of a *red-black tree with a cache.* Such a tree contains a red-black tree, but also redundantly stores one of its elements.

```
case class CTree(cache: Int, data: Tree)
```

The specification of the invariant *inv* formalizes the desired property: the cache value must be contained in the tree unless the tree is empty.

```
def inv(ct: CTree) = {
  isRedBlack(ct.data) &&
  (ct.cache ∈ content(ct.data)) || (ct.data == Empty)
}
```

The *contains* operation tests membership in the tree.

```
def contains(ct: CTree, v: Int): Boolean = {
  require(inv(ct))
  choose{ (x: Boolean) ⇒ x == (v ∈ content(ct)) }
}
```

While not being able to fully synthesize it, the deductive synthesis procedure decomposes the problem and partially synthesizes the constraint. One of its possible results is the following *partial* implementation that combines actual code and a sub-constraint:

```
def contains(ct: CTree, v: Int): Boolean = ct.data match {
  case n: Node ⇒
    if (ct.cache == v) {
      true
    } else {
      choose { (x: Boolean) ⇒ x == (v ∈ content(n)) }
    }
  case Empty ⇒
    false
}
```

We notice that this partial implementation makes use of the cache in accordance with the invariant. The code accurately reflects the fact that the cache may not be trusted if the tree is empty. The remaining constraint is in fact a simpler problem that only relates to standard red-black trees. Our system can then compile the resulting code, where the fast path is compiled as the usual Scala code, and the *choose* construct is compiled using the run-time solving approach.

Using Synthesis to Verify Existential Statements. Note that constraint solving by itself does not have static guarantees that the synthesized value can be produced. By successfully solving a synthesis problem, the system establishes the truth of an existentially quantified statement. Therefore, such synthesis capability can be used to prove quantified statements. Here we do not mean to imply that constructive interpretation of quantifiers is the only one possible, nor that it should be built into the semantics. We merely observe that synthesis is an interesting method for proving existential statements containing, for example recursive functions. It is interesting to note that the authors of classical deductive synthesis [54, 55] concluded that better inductive theorem proving is needed to enable synthesis of recursive programs. Given recent advances in software synthesis including ours and others [11, 72], it is interesting to re-examine the use of synthesis algorithms for theorem proving.

3 Towards Case Studies as Mathematical Statements

One challenge in this research is that software as manipulates inputs from very large or infinite domains, with rich algebraic structure. Examples include numerical domains such as integers or approximations of real numbers, as well symbolic domains, such as algebraic data types, sequences, sets, multisets, and maps. To handle this complexity, it is essential to further advance the field of Satisfiability Modulo Theories, including the development of new decision procedures for structures such as multisets and algebraic data types. Moreover, the applications in embedded and cyber-physical systems call for systematic support for reasoning about approximations of real numbers, something that we have recently started exploring as well [17], and that connects the area of verification to numerical analysis and to decision procedures for theories of real numbers [21].

Another challenge is that programs have complex control and language structure, including conditionals, recursion, higher-order functions, dynamic dispatch, and concurrency. Much of this complexity can be handled by translation into first-order side-effect-free functional or logic programming language. This supports the use of standardized formats for software analysis and program synthesis problems using either Horn clauses in the language of SMT-LIB theories [8], or pure subsets of popular programming languages such as Scala.

The format of Horn clauses has a particularly promising future as a way of taming the complexity of programming languages as well as the methodological complexity of precise and automated verification algorithms, as witnessed by a number of successful approaches in this direction. Apart from those mentioned already, exciting new work includes addressing more complex quantification patterns that go beyond verification of safety properties [4,5,9].

When Horn clauses are expressed in SMT-LIB2 format (`http://www.smt-lib.org`), rich theories present in the format, combined with the ability to encode complex control flow, result in a versatile format for precisely stating mathematical problems about software. Building tools that handle such benchmarks requires new theoretical insights as well as important software development and experimental work.

Acknowledgements. The authors would like to thank his group members, his collaborators. The author also thanks researchers gathered around the EU COST Action “Rich Model Toolkit” (`http://richmodels.epfl.ch`), 2009-2013.

References

1. Armando, A., Mantovani, J., Platania, L.: Bounded model checking of software using SMT solvers instead of SAT solvers. In: Valmari, A. (ed.) SPIN 2006. LNCS, vol. 3925, pp. 146–162. Springer, Heidelberg (2006)
2. Barrett, C., Conway, C.L., Deters, M., Hadarean, L., Jovanović, D., King, T., Reynolds, A., Tinelli, C.: CVC4. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 171–177. Springer, Heidelberg (2011)

3. Barrett, C., Shikanian, I., Tinelli, C.: An abstract decision procedure for satisfiability in the theory of recursive data types. Electronic Notes in Theoretical Computer Science 174(8), 23–37 (2007)
4. Beyene, T.A., Chaudhuri, S., Popeea, C., Rybalchenko, A.: A constraint-based approach to solving games on infinite graphs. In: POPL, pp. 221–234 (2014)
5. Beyene, T.A., Popeea, C., Rybalchenko, A.: Solving existentially quantified Horn clauses. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 869–882. Springer, Heidelberg (2013)
6. Beyer, D., Henzinger, T.A., Majumdar, R., Rybalchenko, A.: Invariant synthesis for combined theories. In: Cook, B., Podelski, A. (eds.) VMCAI 2007. LNCS, vol. 4349, pp. 378–394. Springer, Heidelberg (2007)
7. Biere, A., Cimatti, A., Clarke, E.M., Zhu, Y.: Symbolic model checking without bdds. In: Cleaveland, W.R. (ed.) TACAS 1999. LNCS, vol. 1579, pp. 193–207. Springer, Heidelberg (1999)
8. Bjørner, N., McMillan, K.L., Rybalchenko, A.: Program verification as satisfiability modulo theories. In: SMT@IJCAR, pp. 3–11 (2012)
9. Bjørner, N., McMillan, K.L., Rybalchenko, A.: On solving universally quantified Horn clauses. In: Logozzo, F., Fähndrich, M. (eds.) SAS 2013. LNCS, vol. 7935, pp. 105–125. Springer, Heidelberg (2013)
10. Blanc, R.W., Kneuss, E., Kuncak, V., Suter, P.: An overview of the Leon verification system: Verification by translation to recursive functions. In: Scala Workshop (2013)
11. Bodik, R.: Algorithmic program synthesis with partial programs and decision procedures. In: Palsberg, J., Su, Z. (eds.) SAS 2009. LNCS, vol. 5673, p. 1. Springer, Heidelberg (2009)
12. Boyer, R.S., Moore, J.S.: Proving theorems about LISP functions. J. ACM 22(1), 129–144 (1975)
13. Codish, M., Giesl, J., Schneider-Kamp, P., Thiemann, R.: SAT solving for termination proofs with recursive path orders and dependency pairs. J. Autom. Reasoning 49(1), 53–93 (2012)
14. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 2nd edn. MIT Press and McGraw-Hill (2001)
15. Darulová, E., Kuncak, V.: Trustworthy numerical computation in scala. In: OOPSLA (2011)
16. Darulova, E., Kuncak, V.: Certifying solutions for numerical constraints. In: Qadeer, S., Tasiran, S. (eds.) RV 2012. LNCS, vol. 7687, pp. 277–291. Springer, Heidelberg (2013)
17. Darulova, E., Kuncak, V.: Sound compilation for reals. In: ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL) (2014)
18. Darulova, E., Kuncak, V., Majumdar, R., Saha, I.: Synthesis of fixed-point programs. In: Embedded Software (EMSOFT) (2013)
19. de Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008)
20. de Moura, L., Bjørner, N.: Generalized, efficient array decision procedures. In: Formal Methods in Computer-Aided Design (November 2009)
21. de Moura, L.M., Passmore, G.O.: Computation in real closed infinitesimal and transcendental extensions of the rationals. In: Bonacina, M.P. (ed.) CADE 2013. LNCS, vol. 7898, pp. 178–192. Springer, Heidelberg (2013)
22. Detlefs, D., Nelson, G., Saxe, J.B.: Simplify: a theorem prover for program checking. J. ACM 52(3), 365–473 (2005)

23. Donaldson, A.F., Haller, L., Kroening, D., Rümmer, P.: Software verification using k-induction. In: Yahav, E. (ed.) SAS 2011. LNCS, vol. 6887, pp. 351–368. Springer, Heidelberg (2011)
24. Gligoric, M., Gvero, T., Jagannath, V., Khurshid, S., Kuncak, V., Marinov, D.: Test generation through programming in UDITA. In: International Conference on Software Engineering (ICSE) (2010)
25. Grebenshchikov, S., Lopes, N.P., Popeea, C., Rybalchenko, A.: Synthesizing software verifiers from proof rules. In: PLDI, pp. 405–416 (2012)
26. Gvero, T., Kuncak, V., Kuraj, I., Piskac, R.: Complete completion using types and weights. In: PLDI (2013)
27. Gvero, T., Kuncak, V., Piskac, R.: Interactive synthesis of code snippets. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 418–423. Springer, Heidelberg (2011)
28. Hamza, J., Jobstmann, B., Kuncak, V.: Synthesis for regular specifications over unbounded domains. In: FMCAD (2010)
29. Heizmann, M., Hoenicke, J., Podelski, A.: Nested interpolants. In: POPL (2010)
30. Hodges, W.: Model Theory. Encyclopedia of Mathematics and its Applications, vol. 42. Cambridge University Press (1993)
31. Hojjat, H., Iosif, R., Konečný, F., Kuncak, V., Rümmer, P.: Accelerating interpolants. In: Chakraborty, S., Mukund, M. (eds.) ATVA 2012. LNCS, vol. 7561, pp. 187–202. Springer, Heidelberg (2012)
32. Hojjat, H., Konečný, F., Garnier, F., Iosif, R., Kuncak, V., Rümmer, P.: A verification toolkit for numerical transition systems (tool paper). In: Giannakopoulou, D., Méry, D. (eds.) FM 2012. LNCS, vol. 7436, pp. 247–251. Springer, Heidelberg (2012)
33. Ihlemann, C., Jacobs, S., Sofronie-Stokkermans, V.: On local reasoning in verification. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 265–281. Springer, Heidelberg (2008)
34. Jacobs, S., Kuncak, V.: Towards complete reasoning about axiomatic specifications. In: Jhala, R., Schmidt, D. (eds.) VMCAI 2011. LNCS, vol. 6538, pp. 278–293. Springer, Heidelberg (2011)
35. Jacobs, S., Kuncak, V., Suter, P.: Reductions for synthesis procedures. In: Giacobazzi, R., Berdine, J., Mastroeni, I. (eds.) VMCAI 2013. LNCS, vol. 7737, Springer, Heidelberg (2013)
36. Jovanovic, D., de Moura, L.M.: Cutting to the chase - solving linear integer arithmetic. J. Autom. Reasoning 51(1), 79–108 (2013)
37. Kahsai, T., Tinelli, C.: Pkind: A parallel k-induction based model checker. In: PDMC, pp. 55–62 (2011)
38. Kaufmann, M., Manolios, P., Moore, J.S. (eds.): Computer-Aided Reasoning: ACL2 Case Studies. Kluwer Academic Publishers (2000)
39. Kaufmann, M., Manolios, P., Moore, J.S. (eds.): Computer-Aided Reasoning: An Approach. Kluwer Academic Publishers (2000)
40. Kneuss, E., Kuncak, V., Kuraj, I., Suter, P.: Synthesis modulo recursive functions. In: OOPSLA (2013)
41. Kneuss, E., Kuncak, V., Suter, P.: Effect analysis for programs with callbacks. In: Cohen, E., Rybalchenko, A. (eds.) VSTTE 2013. LNCS, vol. 8164, pp. 48–67. Springer, Heidelberg (2014)
42. Kneuss, E., Suter, P., Kuncak, V.: Runtime instrumentation for precise flow-sensitive type analysis. In: Barringer, H., et al. (eds.) RV 2010. LNCS, vol. 6418, pp. 300–314. Springer, Heidelberg (2010)

43. Köksal, A., Kuncak, V., Suter, P.: Constraints as control. In: ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL) (2012)
44. Kuncak, V., Blanc, R.: Interpolation for synthesis on unbounded domains. In: Formal Methods in Computer-Aided Design (FMCAD) (2013)
45. Kuncak, V., Kneuss, E., Suter, P.: Executing specifications using synthesis and constraint solving (invited talk). In: Legay, A., Bensalem, S. (eds.) RV 2013. LNCS, vol. 8174, pp. 1–20. Springer, Heidelberg (2013)
46. Kuncak, V., Mayer, M., Piskac, R., Suter, P.: Complete functional synthesis. In: ACM SIGPLAN Conf. Programming Language Design and Implementation (PLDI) (2010)
47. Kuncak, V., Mayer, M., Piskac, R., Suter, P.: Software synthesis procedures. Communications of the ACM (2012)
48. Kuncak, V., Mayer, M., Piskac, R., Suter, P.: Functional synthesis for linear arithmetic and sets. Software Tools for Technology Transfer (STTT) 15(5-6), 455–474 (2013)
49. Kuncak, V., Piskac, R., Suter, P.: Ordered sets in the calculus of data structures (invited paper). In: Dawar, A., Veith, H. (eds.) CSL 2010. LNCS, vol. 6247, pp. 34–48. Springer, Heidelberg (2010)
50. Kuncak, V., Piskac, R., Suter, P., Wies, T.: Building a calculus of data structures (invited paper). In: Barthe, G., Hermenegildo, M. (eds.) VMCAI 2010. LNCS, vol. 5944, pp. 26–44. Springer, Heidelberg (2010)
51. Lemay, A., Maneth, S., Niehren, J.: A learning algorithm for top-down xml transformations. In: PODS, pp. 285–296 (2010)
52. Madhavan, R., Kuncak, V.: Symbolic resource bound inference for functional programs. In: Computer Aided Verification (CAV) (2014)
53. Mal'cev, A.I.: Axiomatizable classes of locally free algebras of various types. In: The Metamathematics of Algebraic Systems. North-Holland (1971); (Translation, original in Doklady, 1961)
54. Manna, Z., Waldinger, R.: A deductive approach to program synthesis. ACM Trans. Program. Lang. Syst. 2(1), 90–121 (1980)
55. Manna, Z., Waldinger, R.J.: Toward automatic program synthesis. Commun. ACM 14(3), 151–165 (1971)
56. Mayer, M., Kuncak, V.: Game programming by demonstration. In: SPLASH Onward! (2013)
57. McMillan, K.L., Rybalchenko, A.: Solving constrained Horn clauses using interpolation. Technical Report MSR-TR-2013-6, Microsoft Research (January 2013)
58. Odersky, M.: Contracts for Scala. In: Int. Conf. Runtime Verification (2010)
59. Odersky, M., Rompf, T.: Unifying functional and object-oriented programming with Scala. Commun. ACM 57(4), 76–86 (2014)
60. Odersky, M., Spoon, L., Venners, B.: Programming in Scala: a comprehensive step-by-step guide. Artima Press (2008)
61. Okasaki, C.: Purely Functional Data Structures. Cambridge University Press (1998)
62. Piskac, R., Kuncak, V.: Fractional collections with cardinality bounds, and mixed integer linear arithmetic with stars. In: Kaminski, M., Martini, S. (eds.) CSL 2008. LNCS, vol. 5213, pp. 124–138. Springer, Heidelberg (2008)
63. Piskac, R., Kuncak, V.: Linear arithmetic with stars. In: Gupta, A., Malik, S. (eds.) CAV 2008. LNCS, vol. 5123, pp. 268–280. Springer, Heidelberg (2008)

64. Piskac, R., Kuncak, V.: Munch - automated reasoner for sets and multisets (system description). In: Giesl, J., Hähnle, R. (eds.) IJCAR 2010. LNCS, vol. 6173, pp. 149–155. Springer, Heidelberg (2010)
65. Podelski, A., Rybalchenko, A.: Transition predicate abstraction and fair termination. ACM Trans. Program. Lang. Syst. 29(3) (2007)
66. Rümmer, P., Hojjat, H., Kuncak, V.: Classifying and solving horn clauses for verification. In: Cohen, E., Rybalchenko, A. (eds.) VSTTE 2013. LNCS, vol. 8164, pp. 1–21. Springer, Heidelberg (2014)
67. Rümmer, P., Hojjat, H., Kuncak, V.: Disjunctive interpolants for horn-clause verification. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 347–363. Springer, Heidelberg (2013)
68. Sofronie-Stokkermans, V.: Locality results for certain extensions of theories with bridging functions. In: Schmidt, R.A. (ed.) CADE 2009. LNCS, vol. 5663, pp. 67–83. Springer, Heidelberg (2009)
69. Solar-Lezama, A., Tancau, L., Bodík, R., Seshia, S.A., Saraswat, V.A.: Combinatorial sketching for finite programs. In: ASPLOS, pp. 404–415 (2006)
70. Spielmann, A., Kuncak, V.: Synthesis for unbounded bitvector arithmetic. In: Gramlich, B., Miller, D., Sattler, U. (eds.) IJCAR 2012. LNCS, vol. 7364, pp. 499–513. Springer, Heidelberg (2012)
71. Spielmann, A., Nötzli, A., Koch, C., Kuncak, V., Klonatos, Y.: Automatic synthesis of out-of-core algorithms. In: SIGMOD (2013)
72. Srivastava, S., Gulwani, S., Foster, J.: From program verification to program synthesis. In: POPL (2010)
73. Suter, P.: Programming with Specifications. PhD thesis, EPFL (December 2012)
74. Suter, P., Dotta, M., Kuncak, V.: Decision procedures for algebraic data types with abstractions. In: ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL) (2010)
75. Suter, P., Köksal, A.S., Kuncak, V.: Satisfiability modulo recursive programs. In: Yahav, E. (ed.) SAS 2011. LNCS, vol. 6887, pp. 298–315. Springer, Heidelberg (2011)
76. Suter, P., Steiger, R., Kuncak, V.: Sets with cardinality constraints in satisfiability modulo theories. In: Jhala, R., Schmidt, D. (eds.) VMCAI 2011. LNCS, vol. 6538, pp. 403–418. Springer, Heidelberg (2011)
77. Wies, T., Muñiz, M., Kuncak, V.: An efficient decision procedure for imperative tree data structures. In: Bjørner, N., Sofronie-Stokkermans, V. (eds.) CADE 2011. LNCS, vol. 6803, pp. 476–491. Springer, Heidelberg (2011)
78. Wies, T., Muñiz, M., Kuncak, V.: Deciding functional lists with sublist sets. In: Joshi, R., Müller, P., Podelski, A. (eds.) VSTTE 2012. LNCS, vol. 7152, pp. 66–81. Springer, Heidelberg (2012)
79. Wies, T., Piskac, R., Kuncak, V.: Combining theories with shared set operations. In: Ghilardi, S., Sebastiani, R. (eds.) FroCoS 2009. LNCS, vol. 5749, pp. 366–382. Springer, Heidelberg (2009)
80. Yabandeh, M., Knežević, N., Kostić, D., Kuncak, V.: Predicting and preventing inconsistencies in deployed distributed systems. ACM Transactions on Computer Systems 28(1) (2010)
81. Yessenov, K., Piskac, R., Kuncak, V.: Collections, cardinalities, and relations. In: Barthe, G., Hermenegildo, M. (eds.) VMCAI 2010. LNCS, vol. 5944, pp. 380–395. Springer, Heidelberg (2010)

Weak Parity

Scott Aaronson[1,⋆], Andris Ambainis[2,⋆⋆], Kaspars Balodis[2,⋆⋆⋆], and Mohammad Bavarian[1,†]

[1] MIT
[2] University of Latvia
aaronson@csail.mit.edu, andris.ambainis@lu.lv, kbalodis@gmail.com, bavarian@mit.edu

Abstract. We study the query complexity of Weak Parity: the problem of computing the parity of an n-bit input string, where one only has to succeed on a $1/2+\varepsilon$ fraction of input strings, but must do so with high probability on those inputs where one does succeed. It is well-known that n randomized queries and $n/2$ quantum queries are needed to compute parity on *all* inputs. But surprisingly, we give a randomized algorithm for Weak Parity that makes only $O(n/\log^{0.246}(1/\varepsilon))$ queries, as well as a quantum algorithm that makes $O(n/\sqrt{\log(1/\varepsilon)})$ queries. We also prove a lower bound of $\Omega\left(n/\log\left(1/\varepsilon\right)\right)$ in both cases, as well as lower bounds of $\Omega(\log n)$ in the randomized case and $\Omega(\sqrt{\log n})$ in the quantum case for any $\varepsilon > 0$. We show that improving our lower bounds is intimately related to two longstanding open problems about Boolean functions: the Sensitivity Conjecture, and the relationships between query complexity and polynomial degree.[1]

1 Introduction

Given a Boolean input $X = (x_1, \ldots, x_n) \in \{0,1\}^n$, the Parity problem is to compute

$$\text{Par}\,(X) := x_1 \oplus \cdots \oplus x_n\,. \tag{1}$$

This is one of the most fundamental and well-studied problems in computer science.

Since $\text{Par}(X)$ is sensitive to all n bits at every input X, any classical algorithm for Parity requires examining all n bits. As a result, Parity is often considered a "maximally hard problem" for query or decision-tree complexity. In the quantum case, one can get a *slight* improvement to $\lceil n/2 \rceil$ queries, by applying the Deutsch-Jozsa algorithm [7] to successive pairs of coordinates ((x_1, x_2), (x_3, x_4), etc.) and then

[⋆] Supported by the National Science Foundation under Grant No. 0844626 and an Alan T. Waterman Award.

[⋆⋆] Supported by the European Commission under the project QALGO (Grant No. 600700) and the ERC Advanced Grant MQC (Grant No. 320731).

[⋆⋆⋆] Supported by the European Social Fund within the project "Support for Doctoral Studies at University of Latvia."

[†] Partially supported by NSF STC Award 0939370.

[1] A full version of this paper is available at http://www.scottaaronson.com/papers/weakparity.pdf

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 26–38, 2014.

XORing the results. However, that factor-of-two improvement is known to be the best possible by quantum algorithms [9,3].[2]

So we might wonder: can we learn *anything* about a string's parity by making a sublinear number of queries? One natural goal would be to compute the parity, not for all inputs, but merely for as many inputs as possible. This motivates the following problem, which will be the focus of this paper.

Problem 1 ($\textsc{Weak Parity}$ *or* $\textsc{WeakPar}_{n,\varepsilon}$*).* Given $\varepsilon > 0$, design an algorithm that queries a Boolean input $X \in \{0,1\}^n$ as few times as possible, and whose acceptance probability $p(X)$ satisfies

$$\Pr_{X \in \{0,1\}^n}\left[\left|p(X) - \textsc{Par}(X)\right| \leq \frac{1}{3}\right] \geq \frac{1}{2} + \varepsilon. \tag{2}$$

Equivalently, the algorithm should satisfy $|A| \geq (1/2 + \varepsilon)\, 2^n$, where $A \subseteq \{0,1\}^n$ is the set of all inputs X such that $|p(X) - \textsc{Par}(X)| \leq 1/3$.

We will sometimes refer to the above as "bounded-error" $\textsc{Weak Parity}$. In the "zero-error" variant, we instead want to satisfy the stronger condition

$$\Pr_{X \in \{0,1\}^n}\left[p(X) = \textsc{Par}(X)\right] \geq \frac{1}{2} + \varepsilon. \tag{3}$$

To build intuition, let's start with some elementary remarks about $\textsc{Weak Parity}$.

(i) Of course it's trivial to guess $\textsc{Par}(X)$ on a $1/2$ fraction of inputs X, for example by always outputting 0. (On the other hand, being *wrong* on a $1/2 + \varepsilon$ fraction of X's is just as hard as being right on that fraction.)

(ii) As usual, the constant $1/3$ in equation (2) is arbitrary; we can replace it by any other constant in $(0, 1/2)$ using amplification.

(iii) There is no requirement that the acceptance probability $p(X)$ approximate a total Boolean function. In other words, if $X \notin A$ then $p(X)$ can be anything in $[0,1]$.

(iv) It is not hard to see that $\textsc{Weak Parity}$ is completely uninteresting for deterministic classical algorithms. Indeed, any such algorithm that makes fewer than n queries correctly guesses $\textsc{Par}(X)$ on exactly half of the inputs.

(v) Even a randomized or quantum algorithm must be "uncorrelated" with $\textsc{Par}(X)$, if it always makes $T < n$ queries (in the randomized case) or $T < n/2$ queries (in the quantum case). In other words, we must have

$$\sum_{X \in \{0,1\}^n}\left(p(X) - \frac{1}{2}\right)\left(\textsc{Par}(X) - \frac{1}{2}\right) = 0, \tag{4}$$

where $p(X)$ is the algorithm's acceptance probability. The reason is just Fourier analysis: if we switch domains from $\{0,1\}$ to $\{1,-1\}$, then $\textsc{Par}(X) = x_1 \cdots x_n$. But for a randomized algorithm, $p(X)$ is a multilinear polynomial in $x_1, \ldots, x_n$ of degree at most $T < n$, while for a quantum algorithm, Beals et al. [3] showed that $p(X)$ is a multilinear polynomial of degree at most $2T < n$. And any such polynomial has correlation 0 with the degree-n monomial $x_1 \cdots x_n$.

[2] Moreover, this holds even for *unbounded-error* quantum algorithms, which only need to guess $\textsc{Par}(X)$ with *some* probability greater than $1/2$, but must do so for every X.

(vi) Crucially, however, equation (4) does *not* rule out sublinear randomized or quantum algorithms for WEAK PARITY (which exist for all $\varepsilon = o(1)$, as we will see!). The reason is a bit reminiscent of the famous *hat puzzle*:[3] suppose, for example, that an algorithm output $\text{PAR}(X)$ with probability exactly $2/3$ on a $3/4$ fraction of inputs X, and with probability 0 on the remaining $1/4$ fraction of inputs. Such an algorithm would succeed at WEAK PARITY for $\varepsilon = 1/4$, despite maintaining an overall correlation of 0 with $\text{PAR}(X)$.

(vii) The correlation argument does establish that, for the *zero-error* variant of WEAK PARITY, any randomized algorithm must make at least n queries, and any quantum algorithm must make at least $n/2$ queries, with *some* nonzero probability. Even then, however, an algorithm that makes an *expected* sublinear number of queries on each input X is not ruled out (and as we will see, such algorithms exist).

The regime of WEAK PARITY that interests us the most is where ε is very small—the extreme case being $\varepsilon = 1/2^n$. We want to know: *are there fast randomized or quantum algorithms to guess the parity of X on slightly more than half the inputs?*

Despite an immense amount of work on query complexity, so far as we know the above question was never asked before. Here we initiate its study, both by proving upper and lower bounds, and by relating this innocent-looking question to longstanding open problems in combinatorics, including the Sensitivity Conjecture. Even though WEAK PARITY might look at first like a curiosity, we will find that the task of understanding its query complexity is tightly linked to *general* questions about query complexity, and these links help to motivate its study. Conversely, WEAK PARITY illustrates how an old pastime in complexity theory—namely, understanding the largest possible gaps between query complexity measures for *arbitrary* Boolean functions—can actually have implications for the query complexities of *specific* problems.

2 Our Results

First, in Section 4, we prove an upper bound of $O(n/\log^{0.246}(1/\varepsilon))$ on the zero-error randomized query complexity of WEAK PARITY, and an upper bound of $O(n/\sqrt{\log 1/\varepsilon})$ on its bounded-error quantum query complexity. (For zero-error quantum query complexity, we get the slightly worse bound $O\left(n \cdot \frac{\left(\log\log\frac{1}{\varepsilon}\right)^2}{\sqrt{\log 1/\varepsilon}}\right)$.)

Our quantum algorithm is based on Grover's algorithm, while our randomized algorithm is based on the well-known $O\left(n^{0.754}\right)$ randomized algorithm for the complete binary AND/OR tree. For the zero-error quantum algorithm, we use a recent zero-error quantum algorithm for the complete binary AND/OR tree due to Ambainis et al. [1].

[3] In that puzzle, n players are each assigned a red hat or a blue hat uniformly at random, and can see the colors of every hat except their own. At least one player must guess the color of her own hat, and every guess must be correct. Surprisingly, even though each player has only a $1/2$ probability of being correct, it is possible for the players to win this game with probability $\sim 1 - 1/n$, by "conspiring" so that the cases where they are wrong coincide with each other. See `http://en.wikipedia.org/wiki/Hat_puzzle`

Then, in Section 5, we prove a not-quite-matching lower bound of $\Omega\left(n/\log\left(1/\varepsilon\right)\right)$ queries, by using random self-reducibility to reduce ordinary PARITY to WEAK PARITY. This lower bound is the same for randomized and quantum, and for zero-error and bounded-error.

The gap between our upper and lower bounds might seem tiny. But notice that the gap steadily worsens for smaller ε, reaching $O(n^{0.754})$ or $O(\sqrt{n})$ or $O(\sqrt{n}\log^2 n)$ versus the trivial $\Omega\left(1\right)$ when $\varepsilon = 1/2^n$. This leads us to ask whether we can prove a nontrivial lower bound that works for *all* $\varepsilon > 0$. Equivalently, can we rule out an $O\left(1\right)$-query randomized or quantum algorithm that computes PARITY on a subset $A \subseteq \{0,1\}^n$ of size $2^{n-1}+1$?

In Section 6, we show that we *can* (barely) rule out such an algorithm. In 1988, Chung et al. [6] showed that any induced subgraph of the Boolean hypercube $\{0,1\}^n$, of size at least $2^{n-1}+1$, must have at least one vertex of degree $\Omega\left(\log n\right)$. As a consequence, we deduce that for all $\varepsilon > 0$, any bounded-error randomized algorithm for WEAK PARITY must make $\Omega(\log n)$ queries, and any bounded-error quantum algorithm must make $\Omega(\sqrt{\log n})$ queries. For the $\Omega(\log n)$ randomized lower bound, we also include a self-contained proof due to Andy Drucker.

It has been conjectured that Chung et al.'s $\Omega\left(\log n\right)$ degree lower bound can be improved to $n^{\Omega(1)}$. Previously, however, Gotsman and Linial [10] showed that such an improvement would imply the notorious *Sensitivity Conjecture* in the study of Boolean functions. In Section 6, we observe that an $n^{\Omega(1)}$ lower bound for Chung et al.'s problem would *also* yield an $n^{\Omega(1)}$ lower bound on the bounded-error randomized and quantum query complexities of WEAK PARITY, for all $\varepsilon > 0$. Thus, while we do not have a direct reduction between WEAK PARITY and the Sensitivity Conjecture in either direction, it seems plausible that a breakthrough on one problem would lead to a breakthrough on the other.

Next, in Section 7, we connect WEAK PARITY to another longstanding open problem in the study of Boolean functions—and in this case, we give a direct reduction. Namely, suppose we could prove a lower bound of $\Omega\left(n/\log^{1-c}\left(1/\varepsilon\right)\right)$ on the bounded-error randomized query complexity of WEAK PARITY. We show that this would imply that $\mathrm{R}_2\left(f\right) = \Omega\left(\deg\left(f\right)^c\right)$ for all total Boolean functions $f : \{0,1\}^n \to \{0,1\}$, where $\mathrm{R}_2\left(f\right)$ is the bounded-error randomized query complexity of f, and $\deg\left(f\right)$ is its exact degree as a real polynomial. Similar statements hold for other kinds of query complexity (e.g., the bounded-error quantum query complexity $\mathrm{Q}_2\left(f\right)$, and the zero-error randomized query complexity $\mathrm{R}_0\left(f\right)$).

Nisan [13] showed that $\mathrm{R}_2\left(f\right) = \Omega(\deg\left(f\right)^{1/3})$ for all total Boolean functions f, while Beals et al. [3] showed that $\mathrm{Q}_2\left(f\right) = \Omega(\deg\left(f\right)^{1/6})$ for all f.[4] Meanwhile, the largest known separations are $\mathrm{R}_2\left(f\right) = O(\deg\left(f\right)^{0.753\ldots})$ if f is the complete binary AND/OR tree (see Section 3 for a definition), and $\mathrm{Q}_2\left(f\right) = O(\sqrt{\deg\left(f\right)})$ if f is the OR function. However, even improving on the 3^{rd}- and 6^{th}-power relations remains open. Our result says that, if there existed Boolean functions f with larger separations than are currently known, then we *could* improve our algorithms for WEAK

[4] More precisely, they showed respectively that $\mathrm{R}_2\left(f\right) = \Omega(\mathrm{D}\left(f\right)^{1/3})$ and $\mathrm{Q}_2\left(f\right) = \Omega(\mathrm{D}\left(f\right)^{1/6})$ for all f. However, the stated results follow by combining those results with the elementary fact $\deg\left(f\right) \leq \mathrm{D}\left(f\right)$.

Parity. And conversely, any randomized lower bound for Weak Parity better than $\Omega(n/\log^{2/3}(1/\varepsilon))$, or any quantum lower bound better than $\Omega(n/\log^{5/6}(1/\varepsilon))$, would improve the known relations between degree and query complexity for *all* Boolean functions.

In the full version of this paper, we also consider the weak query complexities of functions other than Parity. We show that, for *every* Boolean function f, it is possible to agree with $f(X)$ on $2^{n-1}+1$ inputs X using a bounded-error quantum algorithm that makes $O(\sqrt{n})$ queries, or a zero-error randomized algorithm that makes $O(n^{0.754})$ queries, or a zero-error quantum algorithm that makes $O(\sqrt{n}\log^2 n)$ queries.

3 Preliminaries

We assume some familiarity with classical and quantum query complexity; see Buhrman and de Wolf [5] for an excellent introduction. This section reviews the most relevant definitions and facts.

Given a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$, let $\mathrm{D}(f)$, $\mathrm{R}_0(f)$, and $\mathrm{R}_2(f)$ be the deterministic, zero-error randomized, and bounded-error randomized query complexities of f respectively. We have $n \geq \mathrm{D}(f) \geq \mathrm{R}_0(f) \geq \mathrm{R}_2(f)$ for every f. It is also well-known [13,3] that $\mathrm{D}(f) \leq \mathrm{R}_0(f)^2$ and $\mathrm{D}(f) = O(\mathrm{R}_2(f)^3)$ for all total Boolean functions f.

We will write $\mathrm{R}_2(\textsc{WeakPar}_{n,\varepsilon})$ to denote the minimum number of queries made by any randomized algorithm that, for at least a $1/2+\varepsilon$ fraction of inputs $X \in \{0,1\}^n$, outputs $\textsc{Par}(X)$ with probability at least $2/3$. We will also write $\mathrm{R}_0(\textsc{WeakPar}_{n,\varepsilon})$ to denote the minimum number of queries made by any randomized algorithm that satisfies the following two properties, for at least a $1/2+\varepsilon$ fraction of inputs X:

- The algorithm outputs $\textsc{Par}(X)$ with probability at least $2/3$.
- If the algorithm does *not* output $\textsc{Par}(X)$, then it outputs "don't know."

In both the R_2 and R_0 cases, for the remaining inputs X (i.e., those on which the algorithm fails), the algorithm's output behavior can be arbitrary, but the upper bound on query complexity must hold for *all* inputs $X \in \{0,1\}^n$.

Note that we could also define $\mathrm{R}_0'(\textsc{WeakPar}_{n,\varepsilon})$ as the minimum *expected* number of queries made by any randomized algorithm that, for at least a $1/2+\varepsilon$ fraction of inputs X, outputs $\textsc{Par}(X)$ with probability 1. In this case, the expected number of queries needs to be bounded only for those X's on which the algorithm succeeds. In the full version, we prove that $\mathrm{R}_0(\textsc{WeakPar}_{n,\varepsilon})$ and $\mathrm{R}_0'(\textsc{WeakPar}_{n,\varepsilon})$ are equal up to constant factors.

Let $\mathrm{Q}_0(f)$ and $\mathrm{Q}_2(f)$ be the zero-error and bounded-error quantum query complexities of f respectively. We have $\mathrm{R}_0(f) \geq \mathrm{Q}_0(f) \geq \mathrm{Q}_2(f)$ and $\mathrm{R}_2(f) \geq \mathrm{Q}_2(f)$ for every f. We will use the following results of Beals et al. [3] and Midrijanis [12] respectively:

Theorem 1 (Beals et al. [3]). $\mathrm{D}(f) = O(\mathrm{Q}_2(f)^6)$ *for all total Boolean* f.

Theorem 2 (Midrijanis [12]). $\mathrm{D}(f) = O(\mathrm{Q}_0(f)^3)$ *for all total Boolean* f.

We will write $\mathrm{Q}_2(\textsc{WeakPar}_{n,\varepsilon})$ and $\mathrm{Q}_0(\textsc{WeakPar}_{n,\varepsilon})$ for the bounded-error and zero-error quantum query complexities of $\textsc{WeakPar}_{n,\varepsilon}$; these are defined precisely analogously with $\mathrm{R}_2(\textsc{WeakPar}_{n,\varepsilon})$ and $\mathrm{R}_0(\textsc{WeakPar}_{n,\varepsilon})$.

Given a Boolean function f, the *degree* $\deg(f)$ is the degree of the (unique) real multilinear polynomial $p : \mathbb{R}^n \to \mathbb{R}$ that satisfies $p(X) = f(X)$ for all $X \in \{0,1\}^n$. It is not hard to see that $\deg(f) \leq \mathrm{D}(f)$ for all Boolean functions f. Combined with previous results, this implies that $\mathrm{R}_2(f) = \Omega(\deg(f)^{1/3})$ and $\mathrm{Q}_2(f) = \Omega(\deg(f)^{1/6})$, as stated in Section 2.

Given an input $X \in \{0,1\}^n$ and a subset $B \subseteq [n]$, let X^B denote X with all the bits in B flipped. Then for a Boolean function f, the *sensitivity* $\mathrm{s}^X(f)$ is the number of indices $i \in [n]$ such that $f\left(X^{\{i\}}\right) \neq f(X)$, while the *block sensitivity* $\mathrm{bs}^X(f)$ is the maximum number of pairwise-disjoint "blocks" $B_1, \ldots, B_k \subseteq [n]$ that can be found such that $f\left(X^{B_j}\right) \neq f(X)$ for all $j \in [k]$. We then define

$$\mathrm{s}(f) := \max_{X \in \{0,1\}^n} \mathrm{s}^X(f), \qquad \mathrm{bs}(f) := \max_{X \in \{0,1\}^n} \mathrm{bs}^X(f). \tag{5}$$

Clearly $\mathrm{s}(f) \leq \mathrm{bs}(f)$. The famous *Sensitivity Conjecture* (see Hatami et al. [11] for a survey) asserts that the gap between $\mathrm{s}(f)$ and $\mathrm{bs}(f)$ is never more than polynomial.

Nisan and Szegedy [14] showed that $\mathrm{bs}(f) \leq 2\deg(f)^2$ (recently improved by Tal [17] to $\mathrm{bs}(f) \leq \deg(f)^2$), while Beals et al. [3] showed that $\deg(f) \leq \mathrm{bs}(f)^3$. Thus, degree and block sensitivity are polynomially related. This implies that the Sensitivity Conjecture is equivalent to the conjecture that sensitivity is polynomially related to degree.

A particular Boolean function of interest to us will be the *complete binary AND/OR tree*. Assume $n = 2^d$; then this function is defined recursively as follows:

$$\mathrm{T}_0(x) := x, \tag{6}$$

$$\mathrm{T}_d(x_1, \ldots, x_n) := \begin{cases} \mathrm{T}_{d-1}\left(x_1, \ldots, x_{n/2}\right) \mathrm{AND}\, \mathrm{T}_{d-1}\left(x_{n/2+1}, \ldots, x_n\right) & \text{if } d > 0 \text{ is odd,} \\ \mathrm{T}_{d-1}\left(x_1, \ldots, x_{n/2}\right) \mathrm{OR}\, \mathrm{T}_{d-1}\left(x_{n/2+1}, \ldots, x_n\right) & \text{if } d > 0 \text{ is even.} \end{cases}$$

It is not hard to see that $\mathrm{D}(\mathrm{T}_d) = \deg(\mathrm{T}_d) = 2^d = n$. By contrast, Saks and Wigderson [15] proved the following.

Theorem 3 (Saks-Wigderson [15]). $\mathrm{R}_0(\mathrm{T}_d) = O\left(\left(\frac{1+\sqrt{33}}{4}\right)^d\right) = O(n^{0.753\ldots})$.

Saks and Wigderson [15] also proved a matching lower bound of $\mathrm{R}_0(\mathrm{T}_d) = \Omega(n^{0.753\ldots})$, while Santha [16] proved that $\mathrm{R}_2(\mathrm{T}_d) = \Omega(n^{0.753\ldots})$ even for bounded-error algorithms. Note that T_d gives the largest known gap between $\mathrm{D}(f)$ and $\mathrm{R}_2(f)$ for any total Boolean function f.

Recently, building on the breakthrough quantum walk algorithm for game-tree evaluation [8] (see also [2]), Ambainis et al. [1] proved the following.

Theorem 4 (Ambainis et al. [1]). $\mathrm{Q}_0(\mathrm{T}_d) = O(\sqrt{n}\log^2 n)$.

By comparison, it is not hard to show (by reduction from $\textsc{Parity}$) that $\mathrm{Q}_2(\mathrm{T}_d) = \Omega(\sqrt{n})$. Once again, Theorem 4 gives the largest known gap between $\mathrm{D}(f)$ and $\mathrm{Q}_0(f)$ for any total f.

Finally, the following fact (proved in the full version) will be useful to us.

Proposition 1. *Let $n = 2^d$. The number of inputs $X \in \{0,1\}^n$ such that $\mathrm{T}_d(X) = \mathrm{PAR}(X)$ is exactly $2^{n-1} + 1$ if d is even, and exactly $2^{n-1} - 1$ if d is odd.*

4 Algorithms for WEAK PARITY

We now prove our first result: that there exist nontrivial randomized and quantum algorithms for WEAK PARITY. For simplicity, we first consider the special case $\varepsilon = 2^{-n}$; later we will generalize to arbitrary ε.

Lemma 1. *We have*

$$\mathrm{Q}_2(\mathrm{WEAKPAR}_{n,2^{-n}}) = O(\sqrt{n}), \tag{7}$$

$$\mathrm{R}_0(\mathrm{WEAKPAR}_{n,2^{-n}}) = O(n^{0.754}), \tag{8}$$

$$\mathrm{Q}_0(\mathrm{WEAKPAR}_{n,2^{-n}}) = O(\sqrt{n}\log^2 n). \tag{9}$$

Proof. For Q_2, observe that the OR function, $\mathrm{OR}(X)$, agrees with the parity of X on $2^{n-1} + 1$ inputs $X \in \{0,1\}^n$: namely, all the inputs of odd Hamming weight, plus the input 0^n. Thus, simply computing $\mathrm{OR}(X)$ gives us an algorithm for $\mathrm{WEAKPAR}_{n,\varepsilon}$ with $\varepsilon = 2^{-n}$. And of course, OR can be computed with bounded error in $O(\sqrt{n})$ quantum queries, using Grover's algorithm.

For R_0, assume for simplicity that n has the form 2^d; this will not affect the asymptotics. By Proposition 1, if d is even then the AND/OR tree $\mathrm{T}_d(X)$ agrees with $\mathrm{PAR}(X)$ on $2^{n-1}+1$ inputs X, while if d is odd then $1 - \mathrm{T}_d(X)$ does. Either way, simply computing $\mathrm{T}_d(X)$ gives us an algorithm for $\mathrm{WEAKPAR}_{n,2^{-n}}$. Furthermore, by Theorem 3, there is a zero-error randomized algorithm for $\mathrm{T}_d(X)$ that makes $O(n^{0.754})$ queries.

For Q_0, we also compute either $\mathrm{T}_d(X)$ or $1 - \mathrm{T}_d(X)$ as our guess for $\mathrm{PAR}(X)$, except now we use the zero-error quantum algorithm of Theorem 4, which makes $O(\sqrt{n}\log^2 n)$ queries.

Next, we give a general strategy for converting a WEAK PARITY algorithm for small ε into an algorithm that works for larger ε, with the query complexity gradually increasing as ε does.

Lemma 2. $\mathrm{R}_2(\mathrm{WEAKPAR}_{kn,\varepsilon}) \leq k \cdot \mathrm{R}_2(\mathrm{WEAKPAR}_{n,\varepsilon})$ *for all positive integers k. So in particular, suppose* $\mathrm{R}_2(\mathrm{WEAKPAR}_{n,1/f(n)}) \leq T(n)$*. Then for all N and $\varepsilon > 0$,*

$$\mathrm{R}_2(\mathrm{WEAKPAR}_{N,\varepsilon}) \leq \frac{N \cdot T\left(f^{-1}(1/\varepsilon)\right)}{f^{-1}(1/\varepsilon)}. \tag{10}$$

Exactly the same holds if we replace R_2 *by* R_0, Q_2, *or* Q_0 *throughout.*

Proof. Let A be a randomized algorithm for $\mathrm{WEAKPAR}_{n,\varepsilon}$, and let X be an input to WEAKPAR of size kn. Then our strategy is to group the bits of X into n blocks $Y_1, \ldots, Y_n$ of k bits each, then run A on the input $\mathrm{PAR}(Y_1), \ldots, \mathrm{PAR}(Y_n)$, and output whatever A outputs. If A made $T(n)$ queries originally, then this strategy can be implemented using $k \cdot T(n)$ queries: namely, k queries to the underlying input X every time A queries a bit $\textsc{Par}(Y_i)$. Furthermore, let $p(Z)$ be A's success probability on input $Z \in \{0,1\}^n$. Then the strategy succeeds whenever

$$|p(\textsc{Par}(Y_1), \ldots, \textsc{Par}(Y_n)) - (\textsc{Par}(Y_1) \oplus \cdots \oplus \textsc{Par}(Y_n))| \leq \frac{1}{3}, \tag{11}$$

and by assumption, this occurs for at least a $1/2 + \varepsilon$ fraction of Z's.

The inequality (10) is just a rewriting of the hypothesis, if we make the substitutions $\varepsilon := 1/f(n)$ and $n := f^{-1}(1/\varepsilon)$ to get $\mathrm{R}_2(\textsc{WeakPar}_{f^{-1}(1/\varepsilon),\varepsilon}) \leq T\left(f^{-1}(1/\varepsilon)\right)$, followed by $k := N/f^{-1}(1/\varepsilon)$. Finally, since we never used that A was classical or bounded-error, everything in the proof still works if we replace R_2 by R_0, Q_2, or Q_0 throughout.

Combining Lemmas 1 and 2 now easily gives us our upper bounds:

Theorem 5. *For all n and $\varepsilon \in [2^{-n}, 1/2]$, we have*

$$\mathrm{Q}_2(\textsc{WeakPar}_{n,\varepsilon}) = O(n/\sqrt{\log 1/\varepsilon}), \tag{12}$$

$$\mathrm{R}_0(\textsc{WeakPar}_{n,\varepsilon}) = O\left(n/\log^{0.246} 1/\varepsilon\right), \tag{13}$$

$$\mathrm{Q}_0(\textsc{WeakPar}_{n,\varepsilon}) = O(n \cdot (\log\log 1/\varepsilon)^2 / \sqrt{\log 1/\varepsilon}). \tag{14}$$

We do not know any upper bound on $\mathrm{R}_2(\textsc{WeakPar}_{n,\varepsilon})$ better than our upper bound on $\mathrm{R}_0(\textsc{WeakPar}_{n,\varepsilon})$.

As a final note, all of our algorithms actually satisfy a stronger property than the definition of WEAK PARITY requires. Namely, the algorithms all compute a total Boolean function $f(X)$ that agrees with $\textsc{Par}(X)$ on a $1/2 + \varepsilon$ fraction of inputs. This means, for example, that we can obtain a randomized algorithm that outputs $\textsc{Par}(X)$ with probability 1 on a $1/2 + \varepsilon$ fraction of inputs $X \in \{0,1\}^n$, and that halts after $O(n/\log^{0.246}(1/\varepsilon))$ queries in expectation on *every* input X (not just those inputs for which the algorithm succeeds).

5 Lower Bound via Random Self-Reducibility

Our next result is a *lower* bound on the bounded-error randomized and quantum query complexities of WEAK PARITY. The lower bound matches our upper bounds in its dependence on n, though not in its dependence on ε.

Theorem 6. $\mathrm{Q}_2(\textsc{WeakPar}_{n,\varepsilon}) = \Omega(n/\log(1/\varepsilon))$ *for all* $0 < \varepsilon < \frac{1}{2}$.

Proof. Let C be a quantum algorithm for $\textsc{WeakPar}_{n,\varepsilon}$ that never makes more than T queries. Using C, we will produce a new quantum algorithm C', which makes $O\left(T \log \frac{1}{\varepsilon}\right)$ queries, and which guesses $\textsc{Par}(X)$ on *every* input $X \in \{0,1\}^n$ with probability stricter greater than $1/2$. But it is well-known that any quantum algorithm of the latter kind must make at least $n/2$ queries: in other words, that PARITY has unbounded-error quantum query complexity $n/2$ (this follows from the polynomial method [3]). Putting the two facts together, we conclude that $T = \Omega(n/\log(1/\varepsilon))$.

To produce C', the first step is simply to amplify C. Thus, let C^* be an algorithm that outputs the majority answer among $d \log 1/\varepsilon$ invocations of C. Then by a Chernoff bound, provided the constant d is sufficiently large,

$$\Pr_{X \in \{0,1\}^n} \left[\left| \Pr\left[C^*\left(X\right) \text{ accepts}\right] - \textsc{Par}\left(X\right) \right| \leq \varepsilon \right] \geq \frac{1}{2} + \varepsilon. \tag{15}$$

Next, C' chooses a string $Y \in \{0,1\}^n$ uniformly at random and sets $Z := X \oplus Y$. It then runs C^* to obtain a guess b about $\textsc{Par}(Z)$. Finally, C' outputs $\textsc{Par}(Y) \oplus b$ as its guess for $\textsc{Par}(X)$.

Clearly C' has the same quantum query complexity as C^*: it is easy to simulate a query to a bit z_i of Z, by querying the corresponding bit x_i of X and then XORing with y_i. Furthermore, notice that Z is uniformly random, regardless of X, and that if $b = \textsc{Par}\left(Z\right)$ then $\textsc{Par}(Y) \oplus b = \textsc{Par}(X)$. It follows that C' succeeds with probability at least

$$\left(\frac{1}{2} + \varepsilon\right)(1 - \varepsilon) = \frac{1}{2} + \frac{\varepsilon}{2} - \varepsilon^2 > \frac{1}{2} \tag{16}$$

for every X, which is what we wanted to show.

Of course, Theorem 6 implies that $\mathrm{Q}_0(\textsc{WeakPar}_{n,\varepsilon})$, $\mathrm{R}_2(\textsc{WeakPar}_{n,\varepsilon})$, and $\mathrm{R}_0(\textsc{WeakPar}_{n,\varepsilon})$ are $\Omega\left(n/\log\left(1/\varepsilon\right)\right)$ as well. It is curious that we do not get any lower bounds for Q_0, R_2, or R_0 better than for Q_2.

6 Lower Bound via Sensitivity

Theorem 6 shows that our algorithms from Theorem 5 are close to optimal when ε is reasonably large. Unfortunately, though, Theorem 6 gives nothing when $\varepsilon = 2^{-n}$. Equivalently, it does not even rule out a randomized or quantum algorithm making a *constant* number of queries (!), that correctly decides $\textsc{Parity}$ on a subset of size $2^{n-1} + 1$. We conjecture that $n^{\Omega(1)}$ randomized or quantum queries are needed for the latter task, but we are unable to prove that conjecture—a state of affairs that Section 7 will help to explain. In this section, we at least prove that $\Omega(\log n)$ randomized queries and $\Omega(\sqrt{\log n})$ quantum queries are needed to solve $\textsc{Weak Parity}$ for all $\varepsilon > 0$.

The key is a combinatorial quantity called $\Lambda\left(n\right)$, which was introduced by Chung, Füredi, Graham, and Seymour [6]. Abusing notation, we identify the set $\{0,1\}^n$ with the Boolean hypercube graph (where two vertices are adjacent if and only if they have Hamming weight 1), and also identify any subset $G \subseteq \{0,1\}^n$ with the induced subgraph of $\{0,1\}^n$ whose vertex set is G. Let $\Delta\left(G\right)$ be the maximum degree of any vertex in G. Then

$$\Lambda\left(n\right) := \min_{G \subseteq \{0,1\}^n \,:\, |G| = 2^{n-1}+1} \Delta\left(G\right) \tag{17}$$

is the minimum of $\Delta\left(G\right)$ over all induced subgraphs G of size $2^{n-1} + 1$.

The following proposition relates $\Lambda\left(n\right)$ to $\textsc{Weak Parity}$.

Proposition 2. $\mathrm{R}_2(\textsc{WeakPar}_{n,\varepsilon}) = \Omega(\Lambda\left(n\right))$ *and* $\mathrm{Q}_2(\textsc{WeakPar}_{n,\varepsilon}) = \Omega(\sqrt{\Lambda\left(n\right)})$ *for all* $\varepsilon > 0$.

Proof. Let U be an algorithm that decides PARITY (with bounded error probability) on a subset $A \subseteq \{0,1\}^n$. Then we claim that U must make $\Omega(\Delta(A))$ randomized or $\Omega(\sqrt{\Delta(A)})$ quantum queries, which is $\Omega(\Lambda(n))$ or $\Omega(\sqrt{\Lambda(n)})$ respectively if $|A| > 2^{n-1}$. To see this, let $X \in A$ be a vertex with degree $\Delta(A)$. Then PARITY, when restricted to X and its neighbors, already yields a Grover search instance of size $\Delta(A)$. But searching a list of N elements is well-known to require $\Omega(N)$ randomized or $\Omega(\sqrt{N})$ quantum queries [4].

To build intuition, it is easy to find an induced subgraph $G \subseteq \{0,1\}^n$ such that $|G| = 2^{n-1}$ but $\Delta(G) = 0$: consider the set of all points with odd Hamming weight. But adding a single vertex to that G increases its maximum degree $\Delta(G)$ all the way to n. More generally, Chung et al. [6] were able to prove the following.

Theorem 7 (Chung et al. [6]). $\Lambda(n) \geq \frac{1}{2}\log_2 n - \frac{1}{2}\log_2\log_2 n + \frac{1}{2}$.

Combining Theorem 7 with Proposition 2 tells us immediately that

$$\mathrm{R}_2(\textsc{WeakPar}_{n,\varepsilon}) = \Omega(\log n), \quad \mathrm{Q}_2(\textsc{WeakPar}_{n,\varepsilon}) = \Omega(\sqrt{\log n}) \tag{18}$$

for all $\varepsilon > 0$.

Now, the best-known *upper* bound on $\Lambda(n)$, also proved by Chung et al. [6], is $\sqrt{n}+1$, and it is conjectured that this is essentially tight. By Proposition 2, clearly a proof of that conjecture would imply

$$\mathrm{R}_2(\textsc{WeakPar}_{n,\varepsilon}) = \Omega(\sqrt{n}), \quad \mathrm{Q}_2(\textsc{WeakPar}_{n,\varepsilon}) = \Omega(n^{1/4}) \tag{19}$$

for all $\varepsilon > 0$—and more generally, proving $\Lambda(n) \geq n^{\Omega(1)}$ would imply $\mathrm{R}(\textsc{WeakPar}_{n,\varepsilon})$ and $\mathrm{Q}(\textsc{WeakPar}_{n,\varepsilon})$ are $n^{\Omega(1)}$.

Unfortunately, proving $\Lambda(n) \geq n^{\Omega(1)}$ will be challenging. To see why, recall the famous *Sensitivity Conjecture*, which says that $\mathrm{s}(f)$ is polynomially related to $\mathrm{bs}(f)$ (or equivalently, to $\deg(f)$). In 1992, Gotsman and Linial [10] showed that the Sensitivity Conjecture is equivalent to a statement about the maximum degrees of induced subgraphs of $\{0,1\}^n$:

Theorem 8 (Gotsman-Linial [10]). *Given any growth rate h, we have $\mathrm{s}(f) > h(\deg(f))$ for all Boolean functions $f : \{0,1\}^n \to \{0,1\}$, if and only if*

$$\max\{\Delta(G), \Delta(\{0,1\}^n \setminus G)\} \geq h(n) \tag{20}$$

for all subsets $G \subseteq \{0,1\}^n$ such that $|G| \neq 2^{n-1}$.

Notice that if $|G| \neq 2^{n-1}$, then $\max\{\Delta(G), \Delta(\{0,1\}^n \setminus G)\} \geq \Lambda(n)$. To see this, choose whichever of G or $\{0,1\}^n \setminus G$ is larger, and then discard all but $2^{n-1}+1$ of its elements. Thus, any lower bound on Chung et al.'s combinatorial quantity $\Lambda(n)$ implies the same lower bound on the function $h(n)$ of Theorem 8. For example, if $\Lambda(n) \geq n^{\Omega(1)}$, then $\mathrm{s}(f) \geq \deg(f)^{\Omega(1)}$.

But this means that *any proof of $\Lambda(n) \geq n^{\Omega(1)}$ would imply the Sensitivity Conjecture!*[5] Thus, the conjecture $\Lambda(n) \geq n^{\Omega(1)}$ could be seen as a "common combinatorial core" of the WEAK PARITY and sensitivity versus block sensitivity questions.

[5] Interestingly, we do not know the reverse implication.

As a final note, Andy Drucker (personal communication) found a self-contained proof for $\mathrm{R}_2(\text{WEAKPAR}_{n,\varepsilon}) = \Omega(\log n)$, one that does not rely on $\Lambda(n)$, and that indeed achieves a better constant than the $\Lambda(n)$-based proof. We include that proof in the full version. Interestingly, unlike with our argument based on $\Lambda(n)$, we do not know how to generalize Drucker's argument to prove any lower bound on *quantum* query complexity, nor do we know (even conjecturally) how to push the argument beyond $\Omega(\log n)$.

7 Connection to deg (f) vs. Q (f)

In the last section, we identified a known combinatorial conjecture ($\Lambda(n) \geq n^{\Omega(1)}$) that would imply that the randomized and quantum query complexities of WEAK PARITY are $n^{\Omega(1)}$ for all $\varepsilon > 0$. However, since $\Lambda(n) \geq n^{\Omega(1)}$ would also imply the Sensitivity Conjecture, it will clearly be difficult to prove.

So could there be a *different* way to prove tight lower bounds for $\mathrm{R}_2(\text{WEAKPAR}_{n,\varepsilon})$ and $\mathrm{Q}_2(\text{WEAKPAR}_{n,\varepsilon})$—a way that wouldn't require us to address any longstanding open problems about Boolean functions? Alas, in this section we largely close off that possibility. In particular, suppose we could prove a strong lower bound on $\mathrm{R}_2(\text{WEAKPAR}_{n,\varepsilon})$. We will show that this would imply a better polynomial relationship between $\deg(f)$ and $\mathrm{R}_2(f)$ for *all* total Boolean functions f than is currently known. Similar statements hold for R_0, Q_2, and Q_0.

Theorem 9. *Given a constant c, suppose there exists a sequence of functions* $\{f_n\}_{n\geq 1}$ *such that* $\deg(f_n) = n$ *and* $\mathrm{R}_2(f_n) = O(n^c)$. *Then*

$$\mathrm{R}_2(\text{WEAKPAR}_{n,\varepsilon}) = O\left(n/\log^{1-c} 1/\varepsilon\right). \tag{21}$$

The same holds if we replace R_2 *by* R_0, Q_2, *or* Q_0 *in both instances.*

Proof. In the full version.

So for example, suppose we could prove that $\mathrm{Q}_2(\text{WEAKPAR}_{n,\varepsilon}) = \Omega(n/\sqrt{\log 1/\varepsilon})$; i.e., that the quantum algorithm of Theorem 5 was optimal. Then we would prove the longstanding conjecture that $\mathrm{Q}_2(f) = \Omega(\sqrt{\deg(f)})$ for all Boolean functions f (the bound being saturated when $f = \mathrm{OR}$).

One might wonder: can we also go in the other direction, and use the known polynomial relationships between $\deg(f)$ and query complexity measures to prove better lower bounds for WEAK PARITY? At present, we cannot quite do that, but we can do something close. Recall from Section 1 that, in defining WEAK PARITY, we did not impose any requirement that our algorithm's acceptance probability $p(X)$ approximate a total Boolean function. However, suppose we *do* impose that requirement. Then we can easily show the following (in the full version):

Proposition 3. *Fix any* $\varepsilon > 0$. *Suppose an algorithm's acceptance probability must satisfy* $p(X) \in [0, 1/3] \cup [2/3, 1]$ *for all* $X \in \{0,1\}^n$. *Then any randomized algorithm for* $\text{WEAKPAR}_{n,\varepsilon}$ *makes* $\Omega\left(n^{1/3}\right)$ *queries, and any quantum algorithm makes*

$\Omega\left(n^{1/6}\right)$ queries. Suppose further that the acceptance probability must satisfy $p(X) \in \{0,1\}$ for all X. Then any randomized algorithm for WEAKPAR$_{n,\varepsilon}$ *makes $\Omega\left(n^{1/2}\right)$ queries in expectation, and any quantum algorithm makes $\Omega\left(n^{1/3}\right)$ queries.*

8 Open Problems

The obvious problem is to close the gaps between our upper and lower bounds on the query complexity of WEAK PARITY. We have seen that this problem is intimately related to longstanding open problems in the study of Boolean functions, including polynomial degree versus query complexity, the Sensitivity Conjecture, and lower-bounding Chung et al.'s [6] combinatorial quantity $\Lambda(n)$. Perhaps the surprising relationships among these problems could motivate renewed attacks. In the meantime, can we reprove our $\Omega(n/\log(1/\varepsilon))$ lower bound for WEAK PARITY (or better yet, improve it) *without* exploiting PARITY's random self-reducibility? How far can we get by using (say) the polynomial or adversary methods directly?

Acknowledgments. We thank Ronald de Wolf, both for helpful discussions and for his comments on a draft; Andy Drucker for allowing us to mention his proof; and the anonymous reviewers for their suggestions.

References

1. Ambainis, A., Childs, A., Le Gall, F., Tani, S.: The quantum query complexity of certification. Quantum Information and Computation 10(3-4) arXiv:0903.1291 (2010)
2. Ambainis, A., Childs, A.M., Reichardt, B.W., Špalek, R., Zhang, S.: Any AND-OR formula of size N can be evaluated in time $N^{1/2+o(1)}$ on a quantum computer. In: Proc. IEEE FOCS (2007); quant-ph/0703015 and arXiv:0704.3628
3. Beals, R., Buhrman, H., Cleve, R., Mosca, M., de Wolf, R.: Quantum lower bounds by polynomials. J. ACM 48(4), 778–797 (2001); Earlier version in IEEE FOCS 1998, pp. 352–361 (1998) quant-ph/9802049
4. Bennett, C., Bernstein, E., Brassard, G., Vazirani, U.: Strengths and weaknesses of quantum computing. SIAM J. Comput. 26(5), 1510–1523 (1997); quant-ph/9701001
5. Buhrman, H., de Wolf, R.: Complexity measures and decision tree complexity: a survey. Theoretical Comput. Sci. 288, 21–43 (2002)
6. Chung, F.R.K., Füredi, Z., Graham, R.L., Seymour, P.: On induced subgraphs of the cube. J. Comb. Theory Ser. A 49(1), 180–187 (1988)
7. Deutsch, D., Jozsa, R.: Rapid solution of problems by quantum computation. Proc. Roy. Soc. London A439, 553–558 (1992)
8. Farhi, E., Goldstone, J., Gutmann, S.: A quantum algorithm for the Hamiltonian NAND tree. Theory of Computing 4(1), 169–190 (2008); quant-ph/0702144
9. Farhi, E., Goldstone, J., Gutmann, S., Sipser, M.: A limit on the speed of quantum computation in determining parity. Phys. Rev. Lett. 81, 5442–5444 (1998); quant-ph/9802045
10. Gotsman, C., Linial, N.: The equivalence of two problems on the cube. J. Comb. Theory Ser. A 61(1), 142–146 (1992)
11. Hatami, P., Kulkarni, R., Pankratov, D.: Variations on the sensitivity conjecture. Theory of Computing Library Graduate Surveys 4 (2011)

12. Midrijanis, G.: On randomized and quantum query complexities (2005) quant-ph/0501142
13. Nisan, N.: CREW PRAMs and decision trees. SIAM J. Comput. 20(6), 999–1007 (1991)
14. Nisan, N., Szegedy, M.: On the degree of Boolean functions as real polynomials. Computational Complexity 4(4), 301–313 (1994)
15. Saks, M., Wigderson, A.: Probabilistic Boolean decision trees and the complexity of evaluating game trees. In: Proc. IEEE FOCS, pp. 29–38 (1986)
16. Santha, M.: On the Monte-Carlo decision tree complexity of read-once formulae. Random Structures and Algorithms 6(1), 75–87 (1995)
17. Tal, A.: Properties and applications of Boolean function composition. In: Proc. Innovations in Theoretical Computer Science (ITCS), pp. 441–454 (2013); ECCC TR12-163

Consequences of Faster Alignment of Sequences

Amir Abboud[1], Virginia Vassilevska Williams[1], and Oren Weimann[2]

[1] Stanford University, USA
{abboud,virgi}@cs.stanford.edu
[2] University of Haifa, Israel
oren@cs.haifa.ac.il

Abstract. The Local Alignment problem is a classical problem with applications in biology. Given two input strings and a scoring function on pairs of letters, one is asked to find the substrings of the two input strings that are most similar under the scoring function. The best algorithms for Local Alignment run in time that is roughly quadratic in the string length. It is a big open problem whether substantially subquadratic algorithms exist. In this paper we show that for all $\varepsilon > 0$, an $O(n^{2-\varepsilon})$ time algorithm for Local Alignment on strings of length n would imply breakthroughs on three longstanding open problems: it would imply that for some $\delta > 0$, 3SUM on n numbers is in $O(n^{2-\delta})$ time, CNF-SAT on n variables is in $O((2-\delta)^n)$ time, and Max Weight 4-Clique is in $O(n^{4-\delta})$ time. Our result for CNF-SAT also applies to the easier problem of finding the longest common substring of binary strings with don't cares. We also give strong conditional lower bounds for the more general Multiple Local Alignment problem on k strings, under both k-wise and SP scoring, and for other string similarity problems such as Global Alignment with gap penalties and normalized Longest Common Subsequence.

1 Introduction

Many basic string and pattern matching problems have overwhelming importance in current bioinformatics research. A well known such problem is the Local Alignment problem which asks to find the two substrings of two given strings that are most similar, under some given similarity measure. The fastest theoretical algorithm for this problem runs in time $O(n^2/\log n)$ [15,38,8] on n-length strings, and is not much faster than the classical dynamic programming algorithm of Smith and Waterman [47] which runs in $O(n^2)$ time. A faster algorithm for this problem, one that runs in, say $O(n^{1.5})$ time, would have tremendous impact, as witnessed by the 49,000 citations to the paper introducing the practical BLAST (Basic Local Alignment Search Tool) algorithm [3]. However, there seems to be very little optimism in the computational biology community that Local Alignment and other important string problems admit such "truly subquadratic" algorithms, i.e. running in time $O(n^{2-\varepsilon})$ for $\varepsilon > 0$. Yet, the theoretical computer science community has not provided any evidence for this impossibility, and in particular, it is yet to give an answer to this pressing question: can Local Alignment be solved in truly subquadratic time?

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 39–51, 2014.

Perhaps the main reason for this lack of an answer, is that we do not have a clear technique for providing negative answers to such questions. The state of the art on unconditional lower bounds seems far from proving any significant superlinear lower bounds in the near future. The theory of NP-completeness cannot distinguish between quadratic upper bounds and $n^{1.5}$ ones. The $W[t]$-hardness approach of parameterized complexity requires parameterization, and like NP-hardness, does not distinguish between differing polynomial runtimes. Another approach is to prove lower bounds for a restricted family of algorithms. However, it is unclear what an appropriate candidate family would be, and either way, a restricted model lower bound only gives a partial answer to the question.

Our approach. We follow an approach that can be viewed as a refinement of NP-hardness. The importance of showing NP-hardness for a certain problem lies in the consequence that a polynomial time algorithm for this problem would also imply a polynomial time algorithm for many other problems that are widely believed to require superpolynomial solutions. The goal of this work and previous works that are mentioned below is to develop such theory that is able to prove that improving the exact running times of certain problems would also imply surprising algorithms for many other problems and is therefore unlikely.

Using this approach we are able to provide the following answer to our pressing question, which is stated more formally in our theorems: A truly subquadratic algorithm for Local Alignment is unlikely because it would also give truly faster algorithms for other famous problems like CNF-SAT, 3-SUM and Max-Weight-4-Clique, implying breakthroughs in three different areas of computer science: the satisfiability algorithms and circuit lower bounds, the computational geometry and the graph algorithms communities!

To provide such answers, to this and other important questions about the optimality of current upper bounds for string problems, we devise careful reductions to the string problems from famous problems that are widely believed to require certain running times, not necessarily quadratic.

1.1 3-SUM Hardness

The most prominent example of this approach is the theory of 3-SUM hardness which was introduced by Gajentaan and Overmars [25] and has been used to show that subquadratic upper bounds for many problems in Computational Geometry are unlikely.

In the 3-SUM problem we are given three lists of n numbers and are asked whether we can pick a number from each list so that the sum is 0. A simple algorithm solves the problem in $\tilde{O}(n^2)$ time, and Baran, Demaine and Pătraşcu [6] were able to get a $O(n^2/\log^2 n)$ solution, yet any improvement beyond this seems unlikely and it is a widely believed conjecture that a polynomial improvement on the upper bound is impossible. Support for this belief comes from the $\Omega(n^2)$ lower bound for the depth of an algebraic decision tree for the problem [23].

Conjecture 1 (3-SUM Conjecture). *In the Word RAM model with words of $O(\log n)$ bits, any algorithm requires $n^{2-o(1)}$ time in expectation to determine* *whether three sets $A, B, C \subset \{-n^3, \ldots, n^3\}$ with $|A| = |B| = |C| = n$ integers contain three elements $a \in A, b \in B, c \in C$ with $a + b + c = 0$.*

Since [25], there have been many papers proving the hardness of computational geometry problems, based on 3SUM, e.g. [19,37,22,13,7]. More recently, the 3-SUM Conjecture has been used in surprising ways to show polynomial lower bounds for purely combinatorial problems in dynamic algorithms [40,2] and Graph algorithms [40,32,48]. The only previous work relating 3-SUM to a Stringology problem, to our knowledge, is the result of Chen et al. [12] showing that under the 3-SUM Conjecture, when the input strings are encodings of much longer strings, using Run-Length-Encoding, then the string matching with don't cares problem requires time that is quadratic in the lengths of the *compressions*. This string problem, however, is strongly related to geometric problems and is less "combinatorial" than the problems we consider here (e.g. it is solvable by a sweep-line algorithm). Thus our reductions require different techniques.

We expand the list of 3-SUM hard problems, showing a reduction from 3-SUM to the Local Alignment problem, proving that a truly subquadratic algorithm for Local Alignment is impossible under the 3-SUM Conjecture, provided the alphabet is large enough.

Theorem 1. *If for some $\varepsilon > 0$, $\delta \in (0, 1)$, one can solve the Local Alignment problem on two strings of length n over an alphabet of size $n^{1-\delta}$ in time $O(n^{2-\delta-\varepsilon})$, then the 3-SUM Conjecture is false.*

To prove Theorem 1 we combine a recent reduction from k-SUM to k-Vector-SUM [1] with an efficient self reduction for 3-SUM using hashing [6,40], then we carefully construct a scoring scheme. The details are given in Section 3.

We note that it is not hard to argue that there is an unconditional lower bound of $\min\{|\Sigma|^2, n^2\}$ for Local Alignment where Σ is the alphabet. When $|\Sigma| = n^{1-\delta}$, this lower bound is $\Omega(n^{2-2\delta})$. Our lower bound for such alphabets is essentially $n^{2-\delta}$ which is a *polynomial* improvement over $n^{2-2\delta}$. Nevertheless, our reduction requires alphabet size at least n^{ε} for some arbitrarily small but constant $\varepsilon > 0$, and the really interesting case of Local Alignment is when the alphabet size is constant, e.g. 4 for DNA and RNA sequences and 20 for protein sequences. We are not able to use the 3-SUM Conjecture to conclude a lower bound for this case. To handle the constant alphabet case, we turn to the presumed hardness of CNF-SAT to prove a lower bound even for *binary* strings.

1.2 Strong ETH Hardness

Despite hundreds of papers on faster exponential algorithms for NP-Hard problems in recent years (see the surveys by Woeginger for an exposition [52]), and despite the remarkable effort put into obtaining faster satisfiability algorithms, the best upper bounds for CNF-SAT on n variables and m clauses remain of the form $2^{n-o(n)}\text{poly}\ (m)$ (e.g. [28,41,46]). The Strong Exponential Time Hypothesis (Strong ETH) of Impagliazzo, Paturi and Zane, which has received a lot of attention recently, states that better algorithms do not exist.

Conjecture 2 (Strong ETH). *For every $\varepsilon > 0$, there exists a k, such that SAT on k-CNF formulas on n variables cannot be solved in $O^*(2^{(1-\varepsilon)n})$ time.*

Strong ETH is an extremely popular conjecture in the exact exponential time algorithms community [11,18,35,17], and Cygan et al. [16] even showed it to be equivalent to assuming that several other NP-hard problems essentially require exhaustive search. Recently, many surprising lower bounds in several different areas were shown to hold under the SETH, including lower bounds for approximating the diameter of a sparse graph [45], for maintaining the number of strongly connected components in a dynamic graph [2], and for the 3-party communication complexity of Set-Disjointness [43].

We show a reduction from CNF-SAT to the *longest common substring with don't cares* problem, which is one of the simplest string problems for which truly subquadratic algorithms are not known and is a very restricted version of the Local Alignment problem.

Definition 1 (The Longest Common Substring with don't Cares Problem). *Given a string S over alphabet $\Sigma = \{0,1\}$ and a string T over $\Sigma \cup \{\star\}$, find the length of the longest string that is a substring of both S and T, where a $\star$ in T can be treated as either 0 or 1.*

If don't care letters are not allowed, the problem can be solved using a generalized suffix tree in $O(n)$ time [27]. If instead of looking for the longest common substring, one wants to check whether a binary string with don't cares appears as a substring in a length n binary string, then there are $O(n \log n)$ time algorithms based on the Fast Fourier Transform [24,31,33]. Thus only slight variations of the longest common substring with don't cares problem admit almost *linear* time solutions, yet our reduction implies that a truly *subquadratic* algorithm for it refutes Strong ETH!

Theorem 2. *If for some $\varepsilon > 0$ one can solve either the Local Alignment problem on two binary strings of length n, or the longest common substring with don't cares problem in time $O(n^{2-\varepsilon})$, then Strong ETH is false.*

To prove Theorem 2 we represent a partial assignment to the variables of our formula with a substring, and we make sure that two substrings will match if and only if the two partial assignments satisfy all the clauses of our formula. The details are given in Section 2.

1.3 Multiple Sequence Alignment

In Gusfield's book [27], algorithms for comparing multiple strings are called "the holy grail" of current research in computational biology. One of the important tasks is the Multiple Local Alignment (MLA) problem [34] defined as follows. Given k strings $T_1, \ldots, T_k$ over some alphabet, find substrings $S_1, \ldots, S_k$ (where S_i is a substring of T_i) whose alignment has maximum score. There are multiple ways to define the alignment score between k substrings but we focus on two options, the most general k-wise scoring scheme and the most popular Sum of Pairs (SP) scoring scheme [5].

The k-wise scoring function $s(\cdot, \ldots, \cdot)$ which is given as a k dimensional matrix with $(|\Sigma|)^k$ entries. This case is called k-wise scoring, and the score of an alignment of k strings $s_1, \ldots, s_k$ is $\sum_i s(s_1[i], \ldots, s_k[i])$[1]. The second option, which is called sum of pairs (SP) scoring, is to use a pairwise scoring function $s(\cdot, \cdot)$ and to define the score of an alignment to be $\sum_i \sum_{k<\ell} s(s_k[i], s_\ell[i])$.

The MLA problem on $k \geq 3$ strings can be defined using either k-wise scoring or SP scoring. Local Alignment is the $k = 2$ case and both scoring rules coincide.

For both scoring schemes, unsurprisingly, the best upper bound for the problem is $O(n^k)$ using dynamic programming. The reduction of Wang and Jiang [49] from the shortest common superstring problem on k strings to a polynomial number of instances of the (global or local) alignment problem on $k + 2$ strings with SP scoring implies that our problem is NP-hard when the number of strings is unbounded. Moreover, the W[1]-hardness results of Bodlaender et al. [10] for unbounded alphabets and of Pietrzak [42] for constant size alphabets also carry on to our problems, implying that upper bounds of the form $f(k) \cdot n^c$ for a constant c independent of k and n are unlikely. Huang [29] strengthens Pietrzak's reduction from k-clique to the shortest common superstring problem and shows that $n^{o(k)}$ algorithms for our problems are not possible under the plausible Exponential Time Hypothesis [30].

These negative results deliver an important message to biologists, showing that an efficient algorithm for optimally aligning a hundred strings is not likely to exist, yet another pressing question remains widely open: is there an algorithm running in time $O(n^{k-1})$ or even $O(n^{k/5})$ for MLA?[2] Such algorithms would imply a major advance in our ability to analyze biological data.

We extend our reduction from CNF-SAT to show that Strong ETH implies a negative answer to our question, when we are interested in k-wise scoring, even when the strings are binary.

Theorem 3. *If for some $\varepsilon > 0$ the MLA on k binary strings of length n with k-wise scoring can be solved in time $O(n^{k-\varepsilon})$, then Strong ETH is false.*

As is stressed in Gusfield's book [27], the less general case of SP scoring has more applications in Bioinformatics. Our reduction from CNF-SAT requires the computation of an OR function, which is easy with k-wise scoring yet does not seem possible with SP scoring. We show, however, that the Weighted-k-Clique problem can explain the hardness of getting faster algorithms even for the SP case of MLA.

Weighted k-Clique. The Max-Weight k-Clique problem is as follows. Given a graph $G = (V, E)$ with integer edge weights, find a k-clique of maximum total

[1] In a more general alignment, one can align alphabet symbols with spaces, and the scoring function can take that into account. In this paper, we prove hardness even for the easier alignment problem where no spaces are allowed.

[2] In the reduction of Bodlaender et al. [10] k increases to k^2 and in Pietrzak's [42] reduction n increases to n^7.

weight, or determine that no k-clique exists. Since k-Clique is a special case of the problem, the Max-Weight k-Clique problem is NP-complete and $W[1]$-hard. The unweighted k-Clique admits an $O(n^{0.792k})$ time algorithm using matrix multiplication [39]. When the edge weights are small, one can obtain $O(n^{k-\varepsilon})$ time algorithms for Max-Weight k-Clique as well, by reducing to a small number of instances of k-Clique. However, when the weights are larger than n^k, the trivial $O(n^k)$ algorithm is essentially the best known (ignoring $n^{o(1)}$ improvements).

We show a tight reduction from Max-Weight $2k$-Clique to MLA on k strings with SP scoring, showing that improving on n^k time for MLA would also imply that Max-Weight $2k$-Clique has faster than n^{2k} algorithms.

Theorem 4. *If for some $\varepsilon > 0$ the MLA on k strings of length n over an alphabet of size $\sqrt{n}$ with SP scoring can be solved in time $O(n^{k-\varepsilon})$, then Max-Weight $2k$-Clique on n nodes graphs can be solved in time $O(n^{2k-\varepsilon})$.*

An immediate corollary of the above theorem is a conditional lower bound for the Local Alignment problem for two strings, based on the assumption that Max-Weight 4-Clique does not have improved algorithms.

Corollary 1. *If for some $\varepsilon > 0$ the Local Alignment on two strings of length n over an alphabet of size $\sqrt{n}$ can be solved in time $O(n^{2-\varepsilon})$, then Max-Weight 4-Clique on n nodes graphs can be solved in time $O(n^{4-\varepsilon})$.*

We note that the special case of Max-Weight k-clique for $k = 3$ is especially interesting. The Max-Weight 3-Clique problem on n-node graphs is known to be essentially equivalent to the All Pairs Shortest Paths (APSP) problem, in the sense that if Max-Weight 3-Clique has a truly subcubic algorithm, so does APSP, and vice versa. It is a longstanding open problem whether APSP on n-node graphs can be solved in $O(n^{3-\varepsilon})$ time [51]. It is thus a major open problem whether Max-Weight 3-Clique is in $O(n^{3-\varepsilon})$ time for some $\varepsilon > 0$. Our current reductions show that a $O(n^{1.5-\varepsilon})$ time algorithm for Local Alignment on two strings would give $O(n^{3-\varepsilon})$ time for APSP, but we suspect that they can be strengthened to show a tighter relationship.

Extensions. In the full version of the paper we also show quadratic lower bounds for well-known generalizations of the *Global* Alignment problem like Alignment with Gap penalties [26] and Alignment with moves [36], and for other string problems like Normalized LCS [4,21] and Partial Match [14,44,9].

2 From CNF-SAT to Alignment

In this section we give a reduction from CNF-SAT on n variables and m clauses to the longest common substring with don't cares problem on binary strings of length $N = O(2^{n/2} \cdot m)$ in $O^*(2^{n/2} \cdot m)$ time. Thus, given an algorithm for this problem that runs in time $O(N^{2-\varepsilon})$ for some $\varepsilon > 0$, we can solve CNF-SAT in time $O^*((2^{n/2} \cdot m)^{2-\varepsilon}) = O^*(2^{(1-\varepsilon/2)n} \cdot \text{poly }(m))$, refuting Strong ETH. In the full version of this paper we explain how to get a reduction to Local Alignment on binary strings, proving Theorem 2, and give the extension to MLA on k strings to prove Theorem 3.

Our reduction follows the split and list technique introduced by Williams [50]. In particular, our reduction from CNF-SAT to the longest common substring problem can be obtained by combining his reduction from CNF-SAT to the *orthogonal vectors* problem on binary vectors and a simple reduction from the latter problem to the longest common substring problem. Below we present a direct reduction from CNF-SAT that makes our extension to MLA on k strings follow more clearly.

Lemma 1. *CNF-SAT over formulas on n variables and m clauses can be reduced to the longest common substring with don't cares problem over strings of length $O(2^{n/2} \cdot m)$ and constant-size alphabet in $O^*(2^{n/2} \cdot m)$ time.*

Proof. Let $V = \{x_1, \dots, x_n\}$ be the n variables of the input CNF formula φ. We split V into two sets of $n/2$ variables, $U = \{x_1, \dots, x_{n/2}\}$ and $V \setminus U$. Let $A = \{\alpha_1, \dots, \alpha_N\}$ and $B = \{\beta_1, \dots, \beta_N\}$ be the sets of all $N = 2^{n/2}$ partial assignments of boolean values to the variables in U and $V \setminus U$, respectively. Combining two partial assignments $\alpha \in A$ and $\beta \in B$ gives an assignment $(\alpha \cdot \beta)$ to all the variables in the formula. We say that a partial assignment α satisfies a clause C if α either assigns TRUE to a variable that appears in C as a positive literal or it assigns FALSE to a variable that appears in C as a negative literal. The key idea of the reduction is the following simple observation, which gives a way of checking the satisfiability of ϕ: *The formula ϕ is satisfiable if and only if there are two partial assignments $\alpha \in A, \beta \in B$ such that for every clause C in the formula at least one of α and β satisfies the clause C.*

The reduction will generate two strings S and T. The string S will have N segments of length $5m$ corresponding to the N partial assignments in A and each segment will encode which clauses are satisfied by the partial assignment. Similarly, T will encode the partial assignments in B. A common substring of S and T will have to be entirely contained in these segments, and it will be able to be the whole segment only if the two corresponding assignments satisfy all the clauses of the formula. Therefore, the longest common substring will be of length $5m$ if and only if ϕ is satisfiable.

Let $C_1, \dots, C_m$ be the clauses of our CNF formula ϕ. For $\alpha \in A$ we define the segment string S_α to contain a different symbol in the $(5j-2)^{th}$ position $S_\alpha[j]$ according to whether α satisfies C_j. Then, $\forall j \in [m]$:

$$S_\alpha[(5j-4)\dots(5j)] = [01x_j01], \text{where } \ x_j = 1 \text{ if } \alpha \text{ satisfies } C_j, \text{ and 0 otherwise.}$$

Similarly, for $\beta \in B$ we define the segment T_β as follows. $\forall j \in [m]$:

$$T_\beta[(5j-4)\dots(5j)] = [\star 1 y_j \star 1], \text{where } \ y_j = \star \text{ if } \beta \text{ satisfies } C_j, \text{ and 1 otherwise.}$$

Note that we can construct these strings for every $\alpha \in A$ and $\beta \in B$ given ϕ in $O^*(2^{n/2}m)$ time. Finally, we create S by concatenating all the segment strings

S_{α_i} for all $\alpha_i \in A$ and placing "unmatchable" [000] segments between them, and we create T similarly by concatenating the T_{β_i} segment strings and placing [111] segments between them.

$$S = S_{\alpha_1} \circ 0^3 \circ S_{\alpha_2} \circ \cdots \circ 0^3 \circ S_{\alpha_N}, \; T = T_{\beta_1} \circ 1^3 \circ T_{\beta_2} \circ \cdots \circ 1^3 \circ T_{\beta_N}$$

Claim 1. *The longest common substring of S and T is of length $5m$ iff there are $\alpha \in A, \beta \in B$ such that every clause C_j in ϕ is satisfied by at least one of α, β.*

To prove Claim 1 we first observe that two segments S_α and T_β match, giving a common substring of length $5m$, if and only if every clause in ϕ is satisfied by at least one of α and β, which implies that the formula is satisfiable. Then, we need to argue that any common substring of length $5m$ cannot contain any of our "unmatchable" parts and must therefore correspond to two segments S_α and T_β that match. The details are given in the full version of this paper.

3 From 3-SUM to Alignment

In this section we prove Theorem 1 by proving the following lemma.

Lemma 2. *For any $0 < \delta < 1$, the 3-SUM problem on n numbers can be reduced in $n^{2-\delta+o(1)}$ time to $n^{\delta+o(1)}$ instances of the local alignment problem on two strings of length $\tilde{O}(n)$ over an alphabet Σ of size $\tilde{O}(n^{1-\delta})$.*

Proof. Given three lists $A, B, C \subseteq \{-n^3, \ldots, n^3\}$ of n numbers each, we want to find a triple of numbers $a \in A, b \in B, c \in C$ that sum to 0. We start by applying a hashing scheme that is due to Dietzfelbinger [20] and that has been used in recent works on 3-SUM [6,40,32]. Let $R = n^{1-\delta}$. There is a simple family $\mathcal{H}$ of hash functions $h : \{-n^3, \ldots, n^3\} \to [R]$ such that if we pick a function $h \in \mathcal{H}$ from the family at random, and hash each element x in our input sets $A \cup B \cup C$ to the bucket $B(h(x))$, we get the following properties:[3]

- *Almost linearity.* For any three numbers a, b, c, if $a + b + c = 0$ then $(h(a) + h(b) + h(c))$ modulo R is either 0 or 1.
- *Good load balancing.* The average number of elements x hashed into a bucket $B(x)$ is $3n/R$ and, in expectation, at most $O(R)$ elements are hashed to buckets with load exceeding $9n/R$.

The reduction picks a random hash function $h \in \mathcal{H}$ and hashes each element x in our lists to a bucket $B(h(x))$. For the $O(R)$ elements that fall in overloaded buckets we can run a brute force check to see if they participate in a 3-sum in $O(nR)$ time. Therefore, we can assume that we have at most R buckets $B(1), \ldots, B(R)$, each containing at most $t = 9n/R = O(n^\delta)$ elements. We order the elements in these buckets $B(i) = \{x_1, \ldots, x_t\}$, and for each index j from 1 to t, we will have a separate stage. In stage j we check whether there

[3] The value $h(x)$ is computed by taking a random odd integer a and returning the $\log R$ most significant bits from the number $a \cdot x$.

is any element c in C such that c is the j^{th} element of its bucket $B(h(c))$ and $a+b+c=0$ for some $a \in A, b \in B$. By the "almost linearity" property, it is enough to search for the pair $a \in A, b \in B$ among the elements a, b for which either $h(a)+h(b)=-h(c)$ or $h(a)+h(b)=-h(c)+1$ (modulo R). To do these checks, in every stage j we create $n^{o(1)}$ instances of the local alignment problem, as described below. The total number of instances is therefore $t \cdot n^{o(1)} = n^{\delta+o(1)}$.

In a recent result by Abboud, Lewi and Williams (Lemma 3.1 in [1]), the authors show that we can construct a set of simple $N = n^{O(1/\log\log n)} = n^{o(1)}$ mappings $f_1, \dots, f_N$ from numbers in $\{-n^3, \dots, n^3\}$ to vectors in $\{-p, \dots, p\}^d$ where $p = \log n$ and $d = O(\log n / \log\log n)$ with the following useful property[4]. If three elements $a, b, c \in \{-n^3, \dots, n^3\}$ sum to 0, then for some $i \in [N]$, the vectors $f_i(a), f_i(b), f_i(c)$ will sum to the all-zero vector $\mathbf{0}$, while if three numbers a, b, c do not sum to 0, then for every $i \in [N]$, the vectors $f_i(a), f_i(b), f_i(c)$ will not sum to the all-zero vector. Note that the entries in each coordinate of our vectors are very small since $p = \log n$.

For every triple of numbers (i, j, z) where $i \in [N], j \in [t], z \in \{0, 1\}$ we create an instance of the Local Alignment problem, i.e. two strings S, T over alphabet Σ and a scoring function $s(\cdot, \cdot)$. The optimal solution to the (i, j, z) local alignment instance will determine whether there are three numbers $a \in A, b \in B, c \in C$ such that

1. c is the j^{th} element in its bucket $B(h(c))$,
2. $h(a)+h(b)+h(c)=z \pmod R$, and
3. $f_i(a)+f_i(b)+f_i(c)=\mathbf{0}$.

If we find a triple satisfying these conditions then we have found a 3-sum, since by the property of the mappings from numbers to vectors described above, condition 3 can only happen if $a+b+c=0$. On the other hand, if there is a 3-sum $a \in A, b \in B, c \in C, a+b+c=0$ in our input set, then for some $(i, j, z) \in [N] \times [t] \times \{0, 1\}$, the above three conditions will hold and we will find this 3-sum. To see this, note that by the property of the mappings there must exist an $i \in [N]$ for which condition 3 holds, and by choosing $j \in [t]$ to be such that c is the j^{th} element in its bucket $B(h(c))$ we satisfy condition 2, and finally, by the "almost linearity" property of our hash function, $z = h(a)+h(b)+h(c) \pmod R$ is in the set $\{0, 1\}$ and condition 2 holds as well.

We now describe the Local Alignment instances that we generate for each triple $(i, j, z) \in [N] \times [t] \times \{0, 1\}$. Our alphabet Σ will contain a letter (h, y) for every pair of integers $h \in [R]$ (which will be used to indicate the value of a hash of a number) and $y \in \{-p, \dots, p\}$ (which will represent the value in a coordinate of our vectors). We also add two symbols $\$_1$ and $\$_2$ to the alphabet. Note that by our choices of $p = \log n$ and $R = n^{1-\delta}$, we get that $|\Sigma| = \tilde{O}(n^{1-\delta})$. As in the reduction from CNF-SAT, the strings will be composed of segments. For every number a in A we create the segment S_a which will have

[4] The idea is simple: each number is mapped to a vector containing the base-p representation of the number, then enumerate over all guesses for the carries when summing k numbers in their base-p representation.

length d and in its ℓ^{th} coordinate it will contain the letter $(h(a), f_i(a)[\ell])$. This letter encodes both the hash of a and the value in the ℓ^{th} entry of the vector $f_i(a)$ (corresponding to a in our current mapping i). Similarly, for every number b in B we define the segment T_b so that $T_b[\ell] = (h(b), f_i(b)[\ell])$ for every $\ell \in [d]$. The strings S, T of our instance (i, j, z) are constructed by concatenating the segments with \$ signs between them. Let $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$, then $S = S_{a_1} \circ \$_1 \circ S_{a_2} \circ \$_1 \circ \cdots \circ \$_1 \circ S_{a_n}$ and $T = T_{b_1} \circ \$_2 \circ T_{b_2} \circ \$_2 \circ \cdots \circ \$_2 \circ T_{b_n}$.

The scoring function $s(\cdot, \cdot)$ is defined as follows. Given two letters (h_1, y_1) and (h_2, y_2), the scoring function will lookup $c \in C$, where c is the j^{th} element in bucket number $-(h_1 + h_2) + z \pmod R$, and return a score of 1 if $f_i(c) = -(y_1 + y_2)$. In any other case, the returned score is $-\infty$. Formally, for any pair of letters $(h_1, y_1), (h_2, y_2) \in [R] \times \{-p, \ldots, p\}$,

$$s((h_1, y_1), (h_2, y_2)) = \begin{cases} 1 & \text{if } c \in C \text{ is the } j^{th} \text{ element in } B((-(h_1 + h_2) + z) \mod R) \\ & \text{and } f_i(c) = -(y_1 + y_2). \\ -\infty & \text{otherwise} \end{cases}$$

We also disallow \$ symbols and gaps in the optimal alignment by giving a score of $-\infty$ to any pair containing them.

The proof is completed with the following claim that shows that our construction will find a 3-sum if it exists. The details are given in the full version of this paper.

Claim 2. *There are two substrings of S, T in the (i, j, z) instance achieving a score of d if and only if there there is a triple $a \in A, b \in B, c \in C$ satisfying conditions 1 to 3 above.*

4 From Weighted Clique to Alignment

In this section we obtain efficient reductions for all $k \geq 2$ from the Max-Weight $2k$-Clique problem to the MLA on k strings *with SP scoring.* We can assume that the input graph is complete, by making each nonedge have weight $-\infty$.

Lemma 3. *Max-Weight $2k$-Clique on a weighted graph with n nodes and m edges with weights in $\{-M, \ldots, M\}$ can be reduced in $O(mk)$ time to MLA on k strings of length $O(m(k + \log M))$ and alphabet of size $O(n + \log M)$.*

An interesting observation about our reduction is that the length of the substrings in the optimal alignment is only $O(k + \log M)$, which is quite short, while one could have hoped for faster algorithms for the restricted problem in which we are only looking for short substrings. With more work, one can strengthen the reduction to make the length of the optimal substrings only $O(\log k + \log M)$. Theorem 4 in the introduction is an immediate consequence of Lemma 3.

We explain the idea of the proof next and give the formal details in the full version of this paper. Each of the k input strings will contain m segments, one for each edge in the graph. These segments will be separated by "unmatchable"

\$ symbols so that the scoring function $s(\cdot, \cdot)$ will have $s(\$, a) = -\infty$ for any symbol a in the alphabet. This will enforce that any solution of positive value must pick a substring from a single segment from each input string. If a solution picks an entire segment corresponding to an edge (u_i, v_i) from each input string i, then our construction will guarantee that the score of the segment alignment is exactly $\sum_{i,j} w(u_i, v_j)$. Notice that if the vertices in $\{u_i, v_i\}_i$ are distinct, i.e. the edges (u_i, v_i) form a matching, then $\sum_{i,j} w(u_i, v_j)$ is exactly the weight of the $2k$-clique formed by these vertices. We make sure that when we align two segments (u_i, v_i) and (u_j, v_j) of two different strings i, j, for each of the weights $w \in \{w(u_i, v_j), w(u_j, v_i), w(u_i, u_j), w(v_i, v_j), w(u_i, v_i) + w(u_j, v_j)\}$ there is some coordinate of the segment alignment in which the pairwise score contributes to w. We carefully devise these contributions to obtain that the sum of the scores is exactly the weight of the clique.

Finally, our construction will guarantee that the best solution always picks an entire segment for each input string. We ensure this by beginning and ending each segment with a symbol Λ, where $s(\Lambda, \Lambda) = k^2 M$ and $s(\Lambda, a) = -\infty$ if $a \neq \Lambda$. The choice of $s(\Lambda, \Lambda)$ guarantees that the optimum solution would always align full segments on top of each other in order to include the Λ, Λ alignments, and hence the optimum solution corresponds to the maximum weight of a $2k$-clique.

Acknowledgements. We would like to thank Kevin Lewi, Ryan Williams and the anonymous reviewers for helpful discussions and comments. The first and second authors were supported by a Stanford School of Engineering Hoover Fellowship, NSF Grant CCF-1417238 and BSF Grant BSF:2012338. The third author was supported by Israel Science Foundation grant 794/13.

References

1. Abboud, A., Lewi, K., Williams, R.: On the parameterized complexity of k-sum. CoRR, abs/1311.3054 (2013)
2. Abboud, A., Vassilevska Williams, V.: Popular conjectures imply strong lower bounds for dynamic problems. arXiv, arXiv:1402.0054 (2014)
3. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. Journal of Molecular Biology 215(3), 403–410 (1990)
4. Arslan, A.N., Eğecioğlu, Ö., Pevzner, P.A.: A new approach to sequence comparison: Normalized sequence alignment. Bioinformatics 17(4), 327–337 (2001)
5. David, J.: Bacon and Wayne F. Anderson. Multiple sequence alignment. Journal of Molecular Biology 191(2), 153–161 (1986)
6. Baran, I., Demaine, E.D., Pătraşcu, M.: Subquadratic algorithms for 3SUM. Algorithmica 50(4), 584–596 (2008); In: Dehne, F., López-Ortiz, A., Sack, J.-R. (eds.) WADS 2005. LNCS, vol. 3608, pp. 409–421. Springer, Heidelberg (2005)
7. Barequet, G., Har-Peled, S.: Some variants of polygonal containment and minimum hausdorff distance undertranslation are 3SUM-hard. In: SODA, pp. 862–863 (1999)
8. Bille, P., Farach-Colton, M.: Fast and compact regular expression matching. Theoretical Computer Science 409(3), 486–496 (2008)
9. Bille, P., Gørtz, I.L., Vildhøj, H.W., Vind, S.: String indexing for patterns with wildcards. In: Fomin, F.V., Kaski, P. (eds.) SWAT 2012. LNCS, vol. 7357, pp. 283–294. Springer, Heidelberg (2012)

10. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Wareham, H.T.: The parameterized complexity of sequence alignment and consensus. Theoretical Computer Science 147(1), 31–54 (1995)
11. Calabro, C., Impagliazzo, R., Paturi, R.: The complexity of satisfiability of small depth circuits. In: Chen, J., Fomin, F.V. (eds.) IWPEC 2009. LNCS, vol. 5917, pp. 75–85. Springer, Heidelberg (2009)
12. Chen, K.-Y., Hsu, P.-H., Chao, K.-M.: Approximate matching for run-length encoded strings is 3sum-hard. In: Kucherov, G., Ukkonen, E. (eds.) CPM 2009 Lille. LNCS, vol. 5577, pp. 168–179. Springer, Heidelberg (2009)
13. Cheong, O., Efrat, A., Har-Peled, S.: On finding a guard that sees most and a shop that sells most. In: SODA, pp. 1098–1107 (2004)
14. Cole, R., Gottlieb, L.-A., Lewenstein, M.: Dictionary matching and indexing with errors and don't cares. In: STOC, pp. 91–100 (2004)
15. Crochemore, M., Landau, G.M., Ziv-Ukelson, M.: A subquadratic sequence alignment algorithm for unrestricted scoring matrices. SIAM Journal on Computing 32, 1654–1673 (2003)
16. Cygan, M., Dell, H., Lokshtanov, D., Marx, D., Nederlof, J., Okamoto, Y., Paturi, R., Saurabh, S., Wahlstrom, M.: On problems as hard as CNFSAT. In: CCC, pp. 74–84 (2012)
17. Cygan, M., Kratsch, S., Nederlof, J.: Fast Hamiltonicity checking via bases of perfect matchings. In: STOC, pp. 301–310 (2013)
18. Dantsin, E., Wolpert, A.: On moderately exponential time for SAT. In: Strichman, O., Szeider, S. (eds.) SAT 2010. LNCS, vol. 6175, pp. 313–325. Springer, Heidelberg (2010)
19. de Berg, M., de Groot, M., Overmars, M.H.: Perfect binary space partitions. Computational Geometry: Theory and Applications 7(81), 81–91 (1997)
20. Dietzfelbinger, M.: Universal hashing and k-wise independent random variables via integer arithmetic without primes. In: Puech, C., Reischuk, R. (eds.) STACS 1996. LNCS, vol. 1046, pp. 569–580. Springer, Heidelberg (1996)
21. Efraty, N., Landau, G.M.: Sparse normalized local alignment. In: Sahinalp, S.C., Muthukrishnan, S.M., Dogrusoz, U. (eds.) CPM 2004. LNCS, vol. 3109, pp. 333–346. Springer, Heidelberg (2004)
22. Erickson, J.: New lower bounds for convex hull problems in odd dimensions. SIAM Journal on Computing 28(4), 1198–1214 (1999)
23. Erickson, J.: Bounds for linear satisfiability problems. Chicago J. Theor. Comput. Sci. (1999)
24. Fischer, M.J., Paterson, M.S.: String matching and other products. SIAM-AMS Proc. 7, 113–125 (1973)
25. Gajentaan, A., Overmars, M.: On a class of $o(n^2)$ problems in computational geometry. Computational Geometry 5(3), 165–185 (1995)
26. Gotoh, O.: An improved algorithm for matching biological sequences. J. Mol. Biol. 162, 705–708 (1982)
27. Gusfield, D.: Algorithms on strings, trees and sequences: computer science and computational biology. Cambridge University Press (1997)
28. Hirsch, E.A.: Two new upper bounds for SAT. In: SODA, pp. 521–530 (1998)
29. Huang, X.: Parameterized complexity and polynomial-time approximation schemes. PhD thesis, Citeseer (2004)
30. Impagliazzo, R., Paturi, R.: On the complexity of k-sat. J. Comput. Syst. Sci. 62(2), 367–375 (2001)
31. Indyk, P.: Faster algorithms for string matching problems: Matching the convolution bound. In: FOCS, p. 166 (1998)

32. Jafargholi, Z., Viola, E.: 3sum, 3xor, triangles. Electronic Colloquium on Computational Complexity (ECCC) 20, 9 (2013)
33. Kalai, A.: Efficient pattern-matching with don't cares. In: SODA, pp. 655–656 (2002)
34. Lawrence, C.E., Altschul, S.F., Boguski, M.S., Liu, J.S., Neuwald, A.F., Wootton, J.C.: Detecting subtle sequence signals: a gibbs sampling strategy for multiple alignment. Science 262(5131), 208–214 (1993)
35. Lokshtanov, D., Marx, D., Saurabh, S.: Known algorithms on graphs on bounded treewidth are probably optimal. In: SODA, pp. 777–789 (2011)
36. Lopresti, D., Tomkins, A.: Block edit models for approximate string matching. Theoretical Computer Science 181, 159–179 (1997)
37. Erickson, J., Soss, M., Overmars, M.H.: Preprocessing chains for fast dihedral rotations is hard or even impossible. Computational Geometry: Theory and Applications 26(3), 235–246 (2002)
38. Masek, W.J., Paterson, M.S.: A faster algorithm computing string edit distances. Journal of Computer and System Sciences 20 (1980)
39. Nešetřil, J., Poljak, S.: On the complexity of the subgraph problem. Comment. Math. Univ. Carolin. 26(2), 415–419 (1985)
40. Pătraşcu, M.: Towards polynomial lower bounds for dynamic problems. In: STOC, pp. 603–610 (2010)
41. Paturi, R., Pudlák, P., Saks, M.E., Zane, F.: An improved exponential-time algorithm for k-SAT. J. ACM 52(3), 337–364 (2005)
42. Pietrzak, K.: On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. Journal of Computer and System Sciences 67(4), 757–771 (2003)
43. Pătraşcu, M., Williams, R.: On the possibility of faster SAT algorithms. In: SODA, pp. 1065–1075 (2010)
44. Rahman, M.S., Iliopoulos, C., Lee, I., Mohamed, M., Smyth, W.F.: Finding patterns with variable length gaps or don't cares. In: Chen, D.Z., Lee, D.T. (eds.) COCOON 2006. LNCS, vol. 4112, pp. 146–155. Springer, Heidelberg (2006)
45. Roditty, L., Vassilevska Williams, V.: Fast approximation algorithms for the diameter and radius of sparse graphs. In: STOC, pp. 515–524 (2013)
46. Schöning, U.: A probabilistic algorithm for k-SAT and constraint satisfaction problems. In: FOCS, pp. 410–414 (1999)
47. Smith, T.F., Waterman, M.S.: The identification of common molecular subsequences. Journal of Molecular Biology 147, 195–197 (1981)
48. Vassilevska, V., Williams, R.: Finding, minimizing, and counting weighted subgraphs. In: STOC, pp. 455–464 (2009)
49. Wang, L., Jiang, T.: On the complexity of multiple sequence alignment. Journal of Computational Biology 1(4), 337–348 (1994)
50. Williams, R.: A new algorithm for optimal constraint satisfaction and its implications. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 1227–1237. Springer, Heidelberg (2004)
51. Vassilevska Williams, V., Williams, R.: Subcubic equivalences between path, matrix and triangle problems. In: FOCS, pp. 645–654 (2010)
52. Woeginger, G.J.: Space and time complexity of exact algorithms: Some open problems. In: Downey, R.G., Fellows, M.R., Dehne, F. (eds.) IWPEC 2004. LNCS, vol. 3162, pp. 281–290. Springer, Heidelberg (2004)

Distance Labels with Optimal Local Stretch

Ittai Abraham and Shiri Chechik

Microsoft Research Silicon Valley, Mountain View CA, USA
{ittaia,schechik}@microsoft.com

Abstract. This paper studies distance oracles and distance labeling scheme with local stretch. Informally we would like to provide stretch guarantees for the r closest nodes of each node. A distance oracle has *local stretch* k for r neighborhoods if for any u, v such that $v = M(u, r')$ and $r' \leq r$: $\mathbf{dist}(u, v) \leq \tilde{\mathbf{dist}}(u, v) \leq k\mathbf{dist}(u, v)$, where $M(u, r')$ is the r' closest node to u and $\tilde{\mathbf{dist}}(u, v)$ is the estimated distance returned by the distance oracle.

For parameters $r > 1$, $k > 1$, we obtain labels of size $O(r^{1/k} \ln^{1-1/k} r + \ln k)$, with local stretch of $2k - 1$ for r neighborhoods in $O(k)$ time, significantly improving the query time and stretch constant of [ABN09].

Moreover, our stretch guarantee of $2k - 1$ matches the best known (and conjectured optimal) guarantees of standard distance oracles.

1 Introduction

Given a graph $G = (V, E)$ with $|V| = n$, let $\mathbf{dist}(u, v, G)$ be the length of a shortest path from u to v on G. [To ease notation, we let $\mathbf{dist}(u, v) = \mathbf{dist}(u, v, G)$. Namely, when we refer to the distances in some subgraph H of G, we will always state the subgraph explicitly and write $\mathbf{dist}(u, v, H)$. Otherwise, if we write $\mathbf{dist}(u, v)$ we mean $\mathbf{dist}(u, v, G)$]. An approximate distance oracle is a succinct data structure capable of quickly reporting an estimation $\tilde{\mathbf{dist}}(u, v)$ of the distance between any two points u,v. A distance oracle has several parameters: its construction time (the running time of the algorithm to produce the data structure), its size (the worst case size of the data structure), its query complexity (the running time of the query algorithm, given two points), and its stretch guarantee. The worst case stretch of a distance oracle (perhaps the most studied notion) is defined as the least number k such that for all u, v:

$$\mathbf{dist}(u, v) \leq \tilde{\mathbf{dist}}(u, v) \leq k\mathbf{dist}(u, v)$$

For any $k > 1$, Thorup and Zwick [TZ05] show how to construct an approximate distance oracle with size $O(kn^{1+1/k})$ that obtains stretch $2k - 1$. Wulff-Nilsen [WN13] showed that the query time can be reduced to $O(\log k)$, while keeping the rest of the parameters the same. Most recently, Chechik [Che14] improved the query time to optimal of $O(1)$, namely, a distance oracle of size $O(kn^{1+1/k})$, $2k-1$ stretch and $O(1)$ query time. Mendel and Naor [MN06,MN07] show how to construct an approximate distance oracle with size $O(n^{1+1/k})$ that obtains stretch $128k$ and $O(1)$ query time. Wulff-Nilsen [WN13] shows that

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 52–63, 2014.

it is possible to improve the $128k$ stretch of Mendel and Naor's construction [MN06,MN07] to $(2+\epsilon)k$ at the cost of additional k-factor in the size, the query time of his distance oracle is $O(1/\epsilon)$. For the case of $k = O(\log n/\log\log n)$ and a fixed ϵ, Wulff-Nilsen show that it is possible to reduce the size back to $O(n^{1+1/k})$.

In this paper we study a more refined *local* stretch guarantee first suggested by Abraham, Bartal and Neiman [ABN07]. Informally, we wish to obtain smaller size distance oracles that guarantee $2k-1$ stretch for nearby pairs.

For each vertex u let $<_u$ be a total order on $V \setminus \{u\}$ such that $\mathbf{dist}(u,v) < \mathbf{dist}(u,w)$ implies $v <_u w$. Let $NN(u,\ell)$ be the ℓ nearest neighbors of u (the first ℓ elements of $<_u$). Let $M(u,r)$ be the r'th closest node to u (the r'th element in $<_u$) and let $D(u,r) = \mathbf{dist}(u, M(u,r))$ be their distance.

We study a notion of local stretch that is applicable to distance labels, that was first suggested by [ABN07,ABN09]. A distance label is a special type of distance oracle where the data structure is distributed into a label for each point and the distance between any two points can be computed by looking at their labels. The size of a distance label scheme is the maximal size of any label.

Definition 1. *We say that a distance label scheme has local stretch k for r neighborhoods if for any u, v such that $v = M(u,r')$ and $r' \leq r$:*

$$\mathbf{dist}(u,v) \leq \tilde{\mathbf{dist}}(u,v) \leq k\mathbf{dist}(u,v)$$

Note that a distance label with local stretch k for r neighborhoods gives no distance estimation guarantees when $v = M(u,r')$ and $r' > r$. Nevertheless, in some applications one may have access to a *locality filter* that can quickly determine if $r' \leq r$ (and if not then use some other distance estimation technique). For example, the Transit-Node distance oracle [BFSS07] for road networks uses such a two-phase approach. A rough distance estimation is obtained via a locality filter (by looking at the geometric coordinates), then hash based approach is used for long-range distance queries and a local search approach for short-range distance queries.

The best previous results of [ABN09] obtain the following distance label scheme with local stretch k for r neighborhoods:

Theorem 1. *[ABN09] There exists a labeling scheme with maximum label size of $O(r^{1/k}\log^2 r)$ such that given the labels of two nodes s and t, and either $s \in NN(t,r)$ or $t \in NN(s,r)$ then one can estimate the distance from s to t with stretch $O(k)$ in $O(r^{1/k}\log^2 r)$ time.*

We significantly improve the query time and the stretch constant:

Theorem 2. *There exists a labeling scheme with maximum label size of $O(r^{1/k}\ln^{1-1/k} r + \ln k)$ such that given the labels of two nodes s and t, and either $s \in NN(t,r)$ or $t \in NN(s,r)$ then one can estimate the distance from s to t with stretch $2k-1$ in $O(k)$ time.*

Up to the $\ln^{1-1/k} r$ factor and assuming the girth conjecture of Erdős [Erd64] our result is tight, see the full version for details.

In particular, when k is constant and r is n^ϵ then the best previous query time was $n^{\Omega(1)}$ and we improve query time to $O(k)$ (which is a constant). Moreover, the best previous stretch had a large hidden constant $O(k)$ and we obtain a tight stretch of $2k-1$. Note that stretch $2k-1$ is the best known stretch for the standard problem (when $r=n$), is known to be optimal for some values of k and is conjectured to be optimal under the girth conjecture (see [TZ05]).

Our tight stretch result is obtained using a non-trivial definition of good events in order to de-randomize the Thorup-Zwick scheme (using the local lemma) and by subtly adjusting the query algorithm in an asymmetrical manner to accommodate these good events. See Section 3 for a more detailed discussion on the high-level ideas and main challenges.

2 Easy Warm-Up: Distance Oracles with Local Stretch

As an easy warm-up for Theorem 2, we start by showing a simple modification to Thorup-Zwick distance oracle that gives a distance oracle with local stretch $2k-1$ for r neighborhoods with $O(nr^{1/k})$ size for a given positive integer k. The non-trivial result is obtaining a distance label scheme with worst case label size bounds. The construction presented in this section does not give a guarantee on the label size. We will later show using the Lovász local lemma the existence of label-based distance oracle with maximum label size of $O(r^{1/k}\ln^{1-1/k} r)$. Due to space limitation the proofs are deferred to the full version.

The construction is as follows. Construct the sets $V=A_0 \supseteq A_1 \supseteq \cdots \supseteq A_{k-1}$ similarly to the Thorup-Zwick construction but with sampling every node with probability $1/r^{1/k}$ rather than $1/n^{1/k}$. Namely, $A_0=V$ and the i-th level A_i is constructed by sampling the vertices of A_{i-1} independently at random with probability $1/r^{1/k}$ for $1 \le i \le k-1$.

For every vertex v, define the bunch $\mathbf{Bun}(v)$ of v exactly as the Thorup-Zwick definition, but with the slight change that the last level contains only nodes in $NN(v,r)$. More precisely,

$$\mathbf{Bun}(v) = \bigcup_{i=0}^{k-2} [\{u \in A_i \setminus A_{i+1} \mid \mathbf{dist}(v,u) < \mathbf{dist}(v,A_{i+1})\}] \cup (NN(v,r) \cap A_{k-1}).$$

The pivot $p_i(v)$ is also exactly as Thorup-Zwick's definition, namely $p_i(v)$ is the closest vertex to v in A_i (break ties arbitrarily).

Lemma 1. *The expected size of the data structure is* $O(knr^{1/k})$.

Consider two nodes s and t such that $t \in NN(s,r)$. The query algorithm for s and t is exactly as the Thorup-Zwick query algorithm, with the slight modification that if the algorithm reaches iteration k and yet $p_{k-1}(t) \notin \mathbf{Bun}(s)$ then we simply return $D(s,r)$.

We next analyze the stretch.

Lemma 2. *The stretch is* $2k-1$.

3 Distance Labels with Local Stretch and Small Maximum Label Size

The algorithm of Section 2 gives a distance oracle with a bound on its total size. This algorithm can be easily transformed into a label based scheme, by assigning each node v a label containing the bunch $\mathbf{Bun}(v)$ and the distances from v to the nodes in $\mathbf{Bun}(v)$. It is not hard to verify that given the labels of two nodes one can estimate their distance with $2k-1$ stretch. However, this does not give a bound on the maximum size of the bunches of the nodes and thus on the labels size. In fact, it is easy to prove that when r is very small ($r << \log n$), the maximum size of the bunches can be much larger than $r^{1/k}$. In this section, we modify the algorithm and use the Lovász local lemma in order to prove the existence of a distance oracle where all bunches are of size $O(r^{1/k} \ln^{1-1/k} r)$.

Let us start by stating tee Lovász local lemma [EL75].

Lemma 3. *Let $\{E_1, ..., E_i\}$ be a set of events such that each event $E_{i'}$ occurs with probability at most p and such that each event is independent on all other events except at most d of them. If $4pd < 1$ then there is a nonzero probability that none of the events occurs.*

We next explain the high level overview of our construction in Section 3.1. We will later describe in Section 3.2 the formal construction and analysis. Due to space limitation some of the proofs are deferred to the full version.

3.1 Overview and Main Challenges

Consider the sets $V = A_0 \supseteq A_1 \supseteq \cdots \supseteq A_{k-1}$ that are defined similarly as in the Thorup-Zwick construction, but with sampling every node with probability $1/(r^{1/k} \ln^{1/k} r)$. For every vertex v, define the bunch

$$\mathbf{Bun}(v) = \bigcup_{i=0}^{k-2} [\{u \in A_i \setminus A_{i+1} \mid \mathbf{dist}(v,u) < \mathbf{dist}(v, A_{i+1})\}] \cup (NN(v,r) \cap A_{k-1}).$$

The pivot $p_i(v)$ is also exactly as Thorup-Zwick's definition, namely $p_i(v)$ is the closest vertex to v in A_i (break ties arbitrarily).

Let $A(x,i)$ be the event that the node x belongs to A_i. We will define good events that depend only on the events $\{A(x,i)\}_{x \in V, 1 \le i \le k-1}$. When we say that an event E depends on a node z, we mean that the outcome of E may depend on one of the events $A(z,i)$ for $1 \le i \le k-1$.

Roughly speaking, the good event $E_{good}(v)$ is when the size of $\mathbf{Bun}(v)$ is small (we will later see that we actually need to further extend $E_{good}(v)$). In order to apply the Lovász local lemma we want that the good events $E_{good}(v)$ depend only on a "small" ball around v. In fact, to show that there are small dependencies between the good events, we need that the event $E_{good}(v)$ depends on a ball of radius even smaller than $D(v,r)$ (recall that $D(v,r)$ is the distance from v to it's r'th closest node). This would require that the bunch $\mathbf{Bun}(v)$

contains only nodes in a small radius ball around v. However, we still need to be able to answer distance queries between v and nodes at distance at most $D(v,r)$ from it. This raises several complications. For example, it might happen that a node $u \in NN(v,r)$, but the pivots of u are at a larger distance, namely, it could be that $\mathbf{dist}(v,p_i(u)) > D(v,r)$. On the one hand we want that the label $L(v)$ to contain $p_i(u)$ (so that the distance $\mathbf{dist}(u,v)$ can be estimated in the query phase), and on the other hand we want that the good event $E_{good}(v)$ not to depend on far away nodes, (which implies that $L(v)$ can not contain far away nodes). We now highlight the general ideas of our solution.

The following observation is crucial. If $\mathbf{dist}(v,u) \geq d_{res}(v) = D(v,r)/(2k-1)$ then it is safe to return $D(v,r)$, since the stretch is at most $2k-1$. So we only need to worry about the case where $\mathbf{dist}(v,u) < d_{res}(v)$. Let i be the minimal index such that either $p_i(u) \in \mathbf{Bun}(v)$ or $p_i(v) \in \mathbf{Bun}(u)$. By the Thorup-Zwick analysis we can show that $\mathbf{dist}(u,p_j(u)) \leq j \cdot \mathbf{dist}(u,v)$ and $\mathbf{dist}(v,p_j(v)) \leq j \cdot \mathbf{dist}(u,v)$ for every $j \leq i$. So at the end when $i = k-1$, we get that $\mathbf{dist}(v,p_{k-1}(u)) \leq k \cdot \mathbf{dist}(v,u)$. In the case where $\mathbf{dist}(v,u) < d_{res}(v)$, we have $\mathbf{dist}(v,p_{k-1}(u)) \leq kd_{res}(v)$. This implies that good event of v only needs to depend on nodes at distance at most $kd_{res}(v)$ from v.

Notice that $kd_{res}(v) = kD(v,r)/(2k-1)$ is slightly larger than $D(v,r)/2$. The fact that the good event of a node v may depend on nodes at distance larger than $D(v,r)/2$ (rather than smaller than $D(v,r)/2$) makes the analysis of small dependence non-trivial (it would be trivial otherwise).

We show that a node z cannot be in too many good events. The following scenario shows a potential challenge with weighted edges. Assume for simplicity that $D(y,r)$ is the same for all nodes $y \in V$. Consider a node z with many neighbors connecting to it with edges of weight $D(z,r)/2 + \epsilon$. The node z may appear in the good events of all of its neighbors and we thus may have a large overlap between the good events.

We therefore need to define the good events in a subtle manner. Recall again that our main difficulty is when $\mathbf{dist}(v,u) \leq d_{res}(v)$ and $i = k-1$ is the smallest index such that either $p_i(u) \in \mathbf{Bun}(v)$ or $p_i(v) \in \mathbf{Bun}(u)$. Recall also that in this case $\mathbf{dist}(v,p_i(u)) \leq k\mathbf{dist}(u,v)$ and $\mathbf{dist}(u,p_i(u)) \leq (k-1)\mathbf{dist}(u,v)$. Notice that there is a path between v and $p_{k-1}(u)$ of length at most $k\mathbf{dist}(u,v)$ that uses edges of weight at most $(k-1)\mathbf{dist}(u,v) \leq (k-1)d_{res}(v)$. This path can be obtained by taking the shortest path from v to u followed by the shortest path from u to $p_{k-1}(u)$. Therefore, in some sense v can ignore long edges and consider only edges of length at most $(k-1)d_{res}(v)$. So we define the good event of v to be dependent only on nodes that are at distance at most $(k-1)d_{res}(v)$ from some node in $B(v,d_{res}(v))$. This helps us (as we will see later on) to show small overlap between the events.

More technical difficulties arise from the fact that $D(v,r)$ and $D(u,r)$ may be different. The event of u is not familiar with the node v, therefore we cannot define its event in terms of $D(v,r)$. To overcome this issue, we roll the responsibility from u to v. The good event of the node v is responsible that all nodes u in $B(v,d_{res}(v))$ are well behaved, in the sense that u is familiar with the relevant pivots of v. More specifically, the label of u stores the $\ell = c \cdot r^{1/k} \ln^{1-1/k} r$ closest nodes to u in A_i for every $1 \leq i \leq k-1$, where c is some constant to be fixed later on. Let $NN_i(v)$ be the ℓ closest nodes in A_i to v.

The good event of the node v is responsible that for all nodes u in $B(v, d_{res}(v))$ one of the following happens. Either $|B(u,(k-1)d_{res}(v)) \cap A_i| \leq \ell$ and then the label of u contains $p_i(v)$ (as $p_i(v) < (k-1)d_{res}(v)$) or the pivot $p_{i+1}(u)$ is closer to u than $p_i(v)$. In the first case the label $L(u)$ contains $p_i(v)$ and we are done (as we can return $\mathbf{dist}(v, p_i(v)) + \mathbf{dist}(p_i(v), u)$). In the second case, the analysis is similar to the Thorup-Zwick analysis.

We modify the query algorithm of Thorup-Zwick, instead of checking if $p_i(v) \in \mathbf{Bun}(u)$, we check if $p_i(v) \in NN_i(u)$.

The main reason for this change is as follows. Recall that now we define the bunch of z to contain only nodes at small radius ball $\tilde{B}(z)$ around z. In fact, the radius of the ball $\tilde{B}(z)$ is defined according to $D(z,r)$. Since it may happen that $D(u,r) < D(v,r)$, and since $\tilde{B}(u)$ is defined according to $D(u,r)$ and not according to $D(v,r)$, the ball $\tilde{B}(u)$ may be "too small" in v's perspective. Thus to overcome this issue, we consider $NN_i(u)$ rather than $\mathbf{Bun}(u)$.

3.2 Formal Construction

We start by defining a sample space for constructing the sets $A_0, ..., A_k$. We define a set of good events $E_{good}(v)$ for every $v \in V$ that depend on the sets $A_0, ..., A_k$. We then show using the Lovász local lemma that there is a nonzero probability that all the good events $E_{good}(v)$ occur. Finally, we show that given sets $A_0, ..., A_k$ that satisfy the events $E_{good}(v)$ for every $v \in V$, one can construct a labeled-based r-NN distance oracle with local stretch $2k-1$ (distance oracle with local stretch $2k-1$ for r neighborhoods) and labels size $O(r^{1/k} \ln^{1-1/k} r)$.

Let us start with defining the sample space for constructing the sets $A_0, ..., A_k$ and the events $E_{good}(v)$ for every $v \in V$.

Consider the sets $V = A_0 \supseteq A_1 \supseteq \cdots \supseteq A_{k-1}$ that are defined similarly as in the Thorup-Zwick construction, but with sampling every node with probability $1/(r^{1/k} \ln^{1/k} r)$. Namely, $A_0 = V$ and the i-th level A_i is constructed by sampling the vertices of A_{i-1} independently at random with probability $1/(r^{1/k} \ln^{1/k} r)$ for $1 \leq i \leq k-1$.

The pivots $p_i(v)$ for $1 \leq i \leq k-1$ are defined as in the Thorup-Zwick, namely, $p_i(v)$ is the closest node to v in A_i (break ties arbitrarily).

Let $G|_d$ be the graph obtaining by deleting all edges from G of weight d or more. Let $d_{res}(v) = D(v,r)/(2k-1)$ (recall that $D(v,r)$ is the distance from v to it's r'th closest node). Let $\tilde{G}(v)$ be the graph $G|_{(k-1)d_{res}(v)}$.

Definition 2. *Let $\tilde{B}(v)$ be the set of nodes in $B(v, k \cdot d_{res}(v), \tilde{G}(v))$, in other words all nodes x such that there is a path from v to x of length at most $k \cdot d_{res}(v)$, where all edges are of weight at most $(k-1)d_{res}(v)$.*

Definition 3. *Define the bunches as described earlier, while storing only nodes in $\tilde{B}(v)$. More precisely,*

$$\mathbf{Bun}(v) = \bigcup_{i=0}^{k-2} \Big[\{u \in (A_i \setminus A_{i+1}) \cap \tilde{B}(v) \mid \mathbf{dist}(v,u) < \mathbf{dist}(v, A_{i+1})\}\Big] \cup (A_{k-1} \cap \tilde{B}(v)).$$

We now formally define the events.

Defining the Events:

1. Let $\ell = 6r^{1/k} \ln^{1-1/k} r + \ln k$.
2. Let $NN_i(v)$ be the ℓ closest nodes in A_i to v.
3. Let $E_i(u,d)$ be the event that either $|B(u,d) \cap A_i| \le \ell$ or $NN_i(u) \cap A_{i+1} \ne \emptyset$.
4. Let $E(u,d)$ be the event that for every i, $E_i(u,d)$ occurs.
5. Let $E_{size}(v)$ be the event that $|\mathbf{Bun}(v)| \le 46r^{1/k} \ln^{1-1/k} r$.
6. Let $E_{neighbors}(v)$ be the event that for every node u: if $\mathbf{dist}(u,v) < d_{res}(v)$ and $D(u,r) \le D(v,r)$, then $E(u,(k-1)d_{res}(v))$ occurs.
7. Let $E_{good}(v)$ be the intersection of the events $E_{size}(v)$ and $E_{neighbors}(v)$.

The Distance labels Scheme:
We now show our distance-labeling scheme assuming the sets $A_0, ..., A_k$ satisfy events $E_{good}(v)$ for every $v \in V$.

The label of a node v is as follows. If $r \le 20$, just store $NN(v,r)$, otherwise do the following. For every $0 \le i \le k-1$, store the set $NN_i(v)$ and the distances from v to the nodes in $NN_i(v)$. Store an indication if $D(v,\ell) \ge d_{res}(v)$, and the distance $D(v,r)$. Store the pivots $p_i(v)$ and the bunch $\mathbf{Bun}(v)$ and the distances from v to these nodes. This concludes the construction of the labels.

The query algorithm for given nodes s and t is as follows.

1. Assume w.l.o.g. that $D(t,r) \le D(s,r)$ (otherwise switch s and t).
2. The query algorithm is similar to the Thorup-Zwick query algorithm with the following subtle modifications.
3. First, instead of checking if $p_i(s) \in \mathbf{Bun}(t)$, check if $p_i(s) \in NN_i(t)$. We keep the check $p_i(t) \in \mathbf{Bun}(s)$ as is.
4. Second, if k is even then the algorithm starts by checking if $s \in NN_0(t)$, otherwise it starts by checking $t \in \mathbf{Bun}(s)$. The reason for the second modification is to make sure that the algorithm finishes with the check $p_{k-1}(t) \in \mathbf{Bun}(s)$, rather than $s \in NN_{k-1}(t)$, for reasons that will become clearer later on.
5. Third, if the algorithm finishes and yet $p_{k-1}(t) \notin \mathbf{Bun}(s)$, then return $D(s,r)$.

See Figure 1 for the pseudo-code.
To ease presentation, let $p_k(v) = nil$.

Lemma 4. *If $t \in NN(s,r)$ or $s \in NN(t,r)$ and events $E_{good}(s)$ and $E_{good}(t)$ occur then the distance $\hat{\mathbf{dist}}(s,t)$ returned by the query algorithm satisfies, $\mathbf{dist}(s,t) \le \hat{\mathbf{dist}}(s,t) \le (2k-1)\mathbf{dist}(s,t)$.*

```
algorithm $\mathbf{dist}(s,t)$

$i \leftarrow 0$
If $D(t,r) > D(s,r)$ then $(s,t) \leftarrow (t,s)$.
If $r \leq 20$, return $\mathbf{dist}(s,t)$.
While $i \leq k-1$ do:
    If $k-i$ is odd then do:
        If $p_i(t) \in \mathbf{Bun}(s)$ then return $\mathbf{dist}(s,p_i(t)) + \mathbf{dist}(p_i(t),t)$.
    Else ($k-i$ is even) do:
        If $p_i(s) \in NN_i(t)$ then return $\mathbf{dist}(s,p_i(s)) + \mathbf{dist}(p_i(s),t)$.
    $i \leftarrow i+1$
Return $D(s,r)$.
```

Fig. 1. Answering a distance query for nodes (s,t)

Proof: Consider s and t such that either $t \in NN(s,r)$ or $s \in NN(t,r)$. Assume w.l.o.g. that $D(t,r) \leq D(s,r)$ (otherwise switch s and t). First we note that if either $s \in NN(t,r)$ or $t \in NN(s,r)$ then the returned distance $\hat{\mathbf{dist}}(s,t)$ satisfies $\mathbf{dist}(s,t) \leq \hat{\mathbf{dist}}(s,t)$. To see this, notice that the algorithm either returns $\mathbf{dist}(s,w) + \mathbf{dist}(w,t)$ for some node w or it returns $D(s,r)$. In the first case, clearly by triangle inequality, $\mathbf{dist}(s,t) \leq \hat{\mathbf{dist}}(s,t)$. In the second case, note that if $t \in NN(s,r)$ then by definition of $D(s,r)$, $\mathbf{dist}(s,t) \leq D(s,r)$ and if $s \in NN(t,r)$ then by definition of $D(t,r)$, $\mathbf{dist}(s,t) \leq D(t,r)$ and since $D(s,r) \geq D(t,r)$ then $\mathbf{dist}(s,t) \leq D(s,r)$. We get that $\mathbf{dist}(s,t) \leq \hat{\mathbf{dist}}(s,t)$. We are left with showing the second direction, that is, $\hat{\mathbf{dist}}(s,t) \leq (2k-1)\mathbf{dist}(s,t)$.

Note that the maximum distance returned by the algorithm is $D(s,r)$ and thus if $\mathbf{dist}(s,t) \geq D(s,r)/(2k-1)$ then $\hat{\mathbf{dist}}(s,t) \leq (2k-1)\mathbf{dist}(s,t)$ and we are done. So assume $\mathbf{dist}(s,t) < D(s,r)/(2k-1)$.

Let $\Delta = \mathbf{dist}(s,t)$.

Let i' be the number of iterations in the while loop of Algorithm **dist**. We prove by induction on i for every $i \leq i'$ the following. If $(k-i)$ is odd then $\mathbf{dist}(t,p_i(t)) \leq i\Delta$, else $\mathbf{dist}(s,p_i(s)) \leq i\Delta$. For $i=0$, the claim follows from the fact that $\mathbf{dist}(t,p_0(t)) = \mathbf{dist}(s,p_0(s)) = 0$. Assume the claim holds for every $i'' \leq i$ and consider $i+1$.

If $k-i$ is odd then the algorithm checks in the i'th iteration if $p_i(t) \in \mathbf{Bun}(s)$. we claim that $p_i(t) \in \tilde{B}(s)$. To see this, note that by induction hypothesis $\mathbf{dist}(t,p_i(t)) \leq i\Delta$. Consider the path P that is obtained by concatenating the path from s to t and the path from t to $p_i(t)$. It is not hard to verify that this path is of length at most $(i+1)\Delta \leq k\Delta/(2k-1)$ and in addition all edges on that path are of weight at most $(k-1)\Delta/(2k-1)$. We get that $p_i(t) \in \tilde{B}(s)$.

Hence, by definition if $p_i(t) \notin \mathbf{Bun}(s)$ then $\mathbf{dist}(s,p_{i+1}(s)) \leq \mathbf{dist}(s,p_i(t)) \leq (i+1)\Delta$, as required.

If $(k-i)$ is even then the algorithm checks in the i'th iteration if $p_i(s) \in NN_i(t)$. If $p_i(s) \notin NN_i(t)$, then by definition $\mathbf{dist}(p_i(s),t) \geq \max_{x \in NN_i(t)} \mathbf{dist}(t,x)$. In addition, $\mathbf{dist}(t,p_i(s)) \leq \mathbf{dist}(s,t) + \mathbf{dist}(t,p_i(s)) \leq i\Delta \leq (k-1)\Delta/(2k-1)$.

We assume that the event $E_{good}(s)$ occurs and hence, by definition event $E_{neighbors}(s)$ also occurs. Recall that the event $E_{neighbors}(s)$ is the event that for every node u: if $\mathbf{dist}(u,s) < d_{res}(s)$ and $D(u,r) \leq D(s,r)$, then $E(u,(k-1)d_{res}(s))$ occurs. Specifically, event $E(t,(k-1)d_{res}(s))$ occurs as $\mathbf{dist}(s,t) < d_{res}(s)$ and $D(t,r) \leq D(s,r)$.

Recall that by definition of event $E(t,(k-1)d_{res}(s))$, either $|B(t,(k-1)d_{res}(s)) \cap A_i| \leq \ell$ or $NN_i(t) \cap A_i \neq \emptyset$.

We claim that $|B(t,(k-1)d_{res}(s)) \cap A_i| > \ell$. To see this, recall that $p_i(s) \notin NN_i(t)$ and that $p_i(s) \in (B(t,(k-1)d_{res}(s)) \cap A_i)$. It follows that there must be ℓ nodes in A_i closer to t than $p_i(s)$. Hence $|B(t,(k-1)d_{res}(s)) \cap A_i| > \ell$.

It follows that $NN_i(u) \cap A_{i+1} \neq \emptyset$. We get that, $\mathbf{dist}(t,p_{i+1}(t)) \leq \max_{x \in NN_i(t)} \mathbf{dist}(t,x) \leq \mathbf{dist}(p_i(s),t) \leq \mathbf{dist}(s,t) + \mathbf{dist}(p_i(s),s) \leq (i+1)\Delta$, as required. ∎

Proving a Non-Zero Probability That All Events $E_{good}(v)$ Occur Simultaneously:
The next Lemma is strongly based on Lemma 3.5 from [TZ05] and is given here for completeness, the proof is deferred to the full version.

Lemma 5. *The probability that $E_{size}(v)$ does not occur is at most $1/(10r^4)$, for any node $v \in V$ and $r \geq 20$.*

Lemma 6. *The probability that $E_{neighbors}(v)$ does not occur is at most $1/(10r^4)$, for any $r \geq 10$.*

We now show that every node v belongs to at most r^3 sets $\tilde{B}(u)$, namely, $|\{u \in V \mid v \in \tilde{B}(u)\}| \leq r^3$. This is the main part of the analysis. This allows us later to apply the Lovász local lemma.

For nodes z_1, z_2 and graph H, let $P(z_1,z_2,H)$ be the shortest path from z_1 to z_2 in H.

Lemma 7. *Every node $\hat{x}$ satisfies $|\{u \in V \mid \hat{x} \in \tilde{B}(u)\}| \leq r^3$.*

Proof: Seeking a contradiction assume that there exists a node $\hat{x}$ such that $|\{u \in V \mid \hat{x} \in \tilde{B}(u)\}| > r^3$. Let $\hat{v}$ be the node with the maximal $D(\hat{v},r)$ such that $\hat{x} \in \tilde{B}(\hat{v})$.

Let $S = \{u \in V \mid \hat{x} \in \tilde{B}(u)\}$, namely, all the nodes such that their events may depend on $\hat{x}$.

For a node $u \in S$, let $q(u)$ be the first node (closest to u) in $P(u,\hat{x},\tilde{G}(u))$ that belongs to $NN(\hat{v},r)$. Note that such a node must exists as $\hat{x} \in NN(\hat{v},r)$.

For a node $x \in NN(\hat{v},r)$, let $S(x) = \{u \in S \mid x = q(u)\}$. Note that the sets $S(x)$ for $x \in NN(\hat{v},r)$ form a partition of S. By simple counting there must be a node x in $NN(\hat{v},r)$ such that $|S(x)| \geq r^2$.

For a node $w' \in S(x)$, let $q_2(w') = q_2(x,w',\tilde{G}(w'))$ be the second node of $P(x,w',\tilde{G}(w'))$. Note that $q_2(w')$ is a neighbor of x.

Let $N(S(x),x) = \{q_2(w') \mid w' \in S(x)\}$.

Consider two cases, the first case is when $|N(S(x),x)| \geq r$ (see Figure 2 for illustration). The second case is when $|N(S(x),x)| < r$ (see Figure 3 for illustration). Consider the first case, let $w \in S(x)$ be a node with maximal $\omega(x, q_2(w))$, where $\omega(w_1, w_2)$ for pair of nodes (w_1, w_2) is the weight of the edge (w_1, w_2) in the original graph G (or null if no such exists).

Let $d_2 = \omega(x, q_2(w))$. Let $d_1 = \mathbf{dist}(\hat{x}, x)$ and let $d_3 = \mathbf{dist}(x, w) - d_2$.

Notice that x has at least r neighbors at distance at most d_2 from it. As $\mathbf{dist}(w, x) = d_3 + d_2$, we get that $D(w, r) \leq d_3 + 2d_2$. In addition, recall that the edge $(x, q_2(w))$ appears in the graph $\tilde{G}(w)$. Therefore, by definition of $\tilde{G}(w)$, we have $d_2 = \omega(w, x) < (k-1)d_{res}(w) \leq (k-1)(d_3 + 2d_2)/(2k-1)$. Hence, $d_2 < (k-1)d_3$.

Since $\hat{x} \in \tilde{B}(w)$, we have $\mathbf{dist}(w, \hat{x}, \tilde{G}(w)) < kd_{res}(w)$. Note that, $\mathbf{dist}(w, \hat{x}, \tilde{G}(w)) = d_1 + d_2 + d_3$ and that $d_{res}(w) \leq (d_3 + 2d_2)/(2k-1)$.

Hence, $d_3 + d_2 + d_1 < k(d_3 + 2d_2)/(2k-1)$, therefore $(k-1)d_3 + (2k-1)d_1 < d_2$. It follows that $(k-1)d_3 < d_2$, contradiction.

Consider now the second case where $|N(S(x),x)| < r$. For a node w let $Q(w) = \{z \in S(x) \mid w \in P(x, z, \tilde{G}(z))\}$. Note that there must be a node $w \in N(S(x), x)$ such that $|Q(w)| \geq r$ and $w \notin NN(\hat{v}, r)$. Note that $\mathbf{dist}(w, \hat{x}) \geq (k-1)d_{res}(\hat{v})$.

Consider a node $z \in Q(w)$ of maximal $\mathbf{dist}(z, w)$. Recall that $D(z, r) \leq D(\hat{v}, r)$. Recall also that w appears on the shortest path $P(z, \hat{x}, \tilde{G}(z))$. In addition, as $\hat{x} \in \tilde{B}(z)$, we have $\mathbf{dist}(z, \hat{x}, \tilde{G}(z)) \leq kd_{res}(z)$. Moreover, note that as $\mathbf{dist}(w, \hat{x}) \geq (k-1)d_{res}(\hat{v})$, we get that $\mathbf{dist}(z, w) \leq \mathbf{dist}(z, w, \tilde{G}(z)) \leq \mathbf{dist}(z, \hat{x}, \tilde{G}(z)) - \mathbf{dist}(w, \hat{x}, \tilde{G}(z)) \leq kd_{res}(\hat{v}) - (k-1)d_{res}(\hat{v}) = d_{res}(\hat{v})$.

If follows that $\mathbf{dist}(z, w) \leq \mathbf{dist}(w, \hat{x})$. Note that w has at least r nodes at distance $\mathbf{dist}(z, w)$ from it.

We get that $D(w, r) \leq \mathbf{dist}(z, w)$. In addition, $D(z, r) \leq \mathbf{dist}(z, w) + D(w, r) \leq 2\mathbf{dist}(z, w)$.

Since $\hat{x} \in \tilde{B}(z)$, we also have, $\mathbf{dist}(\hat{x}, z) \leq kd_{res}(z)$.

We get, $2\mathbf{dist}(z, w) \leq \mathbf{dist}(z, \hat{x}) \leq kd_{res}(z) \leq k2\mathbf{dist}(z, w)/(2k-1) < 2\mathbf{dist}(z, w)$, contradiction. ▮

The following lemma shows that each event $E_{good}(v)$ depends on at most r^4 other events.

Lemma 8. *The event $E_{good}(v)$ for some node $v \in V$ is independent on all other events $E_{good}(u)$ but at most r^4 of them.*

Proof: Let $A(x, i)$ for some node x be the event that x belongs to A_i. It is not hard to see that the events $A(x_1, i)$ and $A(x_2, j)$ are independent in the case where $x_1 \neq x_2$.

First, we claim that the event $E_{good}(z)$ for some node z is determined by the set of the events $\{A(x, i) \mid x \in \tilde{B}(z), 1 \leq i \leq k-1\}$. To see this, recall that the event $E_{good}(z)$ is the intersection of the events $E_{size}(z)$ and $E_{neighbors}(z)$. The event $E_{size}(z)$ is the event such that $\mathbf{Bun}(z)$ is of size at most $46r^{1/k} \ln^{1-1/k} r$. Recall that $\mathbf{Bun}(z)$ includes only nodes nodes in $\tilde{B}(z)$ and it is not hard to see that $\mathbf{Bun}(z)$ is determined by the events $\{A(x,i) \mid x \in \tilde{B}(z), 1 \leq i \leq k-1\}$. The event $E_{neighbors}(z)$ is the event such that for every node u such that $\mathbf{dist}(u,z) < d_{res}(z)$ and $D(u,r) \leq D(z,r)$, $E(u,(k-1)d_{res}(z))$ occurs. It is not hard to verify that the event $E(u,(k-1)d_{res}(z))$ is determined by the events $\{A(x,i) \mid \mathbf{dist}(x, u_{\leq}(k-1)d_{res}(z)$. Note that the set $\{x \mid \mathbf{dist}(x,u) \leq (k-1)d_{res}(z)\} \subseteq \tilde{B}(z)$. We get that the event $E_{neighbors}(z)$ is determined by the events $\{A(x,i) \mid x \in \tilde{B}(z), 1 \leq i \leq k-1\}$. Hence, the event $E_{good}(z)$ is determined by the set of the events $\{A(x,i) \mid x \in \tilde{B}(z), 1 \leq i \leq k-1\}$.

We conclude that two events $E_{good}(v_1)$ and $E_{good}(v_1)$ are dependent on one another only if $\tilde{B}(v_1) \cap \tilde{B}(v_2) \neq \emptyset$. By Lemma 7 every node in $\tilde{B}(v)$ belongs to at most r^3 sets $\tilde{B}(v')$. The number of nodes in $\tilde{B}(v)$ is at most r, therefore there are at most r^4 nodes v' such that $\tilde{B}(v) \cap \tilde{B}(v') \neq \emptyset$. The lemma follows. $\blacksquare$

Let $\overline{E_{good}(v)}$ the event that $E_{good}(v)$ does not occur. For every v, $\overline{E_{good}(v)}$ occurs with probability at most $r^{-4}/5$. Each event $\overline{E_{good}(v)}$ depends on at most r^4 other events $\overline{E_{good}(u)}$. We thus have by Lemma 3 that there is a nonzero probability that node of the events $\overline{E_{good}(v)}$ occurs, or in other words, that all the events $E_{good}(v)$ occur.

We conclude the following.

Theorem 3. *There exists a labeling scheme with maximum label size of* $O(r^{1/k} \ln^{1-1/k} r + \ln k)$ *such that given the labels of two nodes* s *and* t*, if either* $s \in NN(t,r)$ *or* $t \in NN(s,r)$ *then one can estimate the distance from* s *to* t *with stretch* $2k-1$ *in* $O(k)$ *time.*

References

ABN07. Abraham, I., Bartal, Y., Neiman, O.: Local embeddings of metric spaces. In: Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing, STOC 2007, pp. 631–640. ACM, New York (2007)

ABN09. Abraham, I., Bartal, Y., Neiman, O.: On low dimensional local embeddings. In: SODA 2009: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, Philadelphia, PA, USA, pp. 875–884. Society for Industrial and Applied Mathematics (2009)

BFSS07. Bast, H., Funke, S., Sanders, P., Schultes, D.: Fast Routing in Road Networks with Transit Nodes. Science 316, 566 (2007)

Che14. Chechik, S.: Approximate distance oracles with constant query time (2014) (to appear in STOC 2014)

EL75. Erdős, P., Lovász, L.: Problems and results on 3-chromatic hypergraphs and some related questions. Infinite and Finite Sets 10(2), 609–627 (1975)

Erd64. Erdős, P.: Extremal problems in graph theory. In: Theory of Graphs and Its Applications, Proc. Sympos. Smolenice, pp. 29–36 (1964)

MN06. Mendel, M., Naor, A.: Ramsey partitions and proximity data structures. In: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, pp. 109–118. IEEE Computer Society, Washington, DC (2006)

MN07. Mendel, M., Naor, A.: Ramsey partitions and proximity data structures. Journal of the European Mathematical Society 9(2), 253–275 (2007)

TZ05. Thorup, M., Zwick, U.: Approximate distance oracles. J. ACM 52(1), 1–24 (2005)

WN13. Wulff-Nilsen, C.: Approximate distance oracles with improved query time. In: Proceedings of the Twenty-Forth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013. SIAM (2013)

A Figures

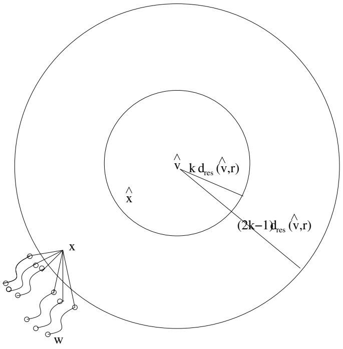


Fig. 2. Illustration of the first case

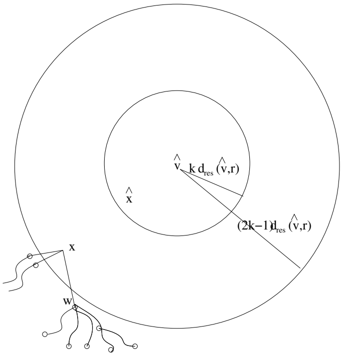


Fig. 3. Illustration of the second case

Time-Expanded Packings

David Adjiashvili, Sandro Bosio, Robert Weismantel, and Rico Zenklusen*

Institute for Operations Research, ETH Zürich, Zürich
{david.adjiashvili,sandro.bosio,robert.weismantel,
rico.zenklusen}@ifor.math.ethz.ch.

Abstract. We introduce a general model for time-expanded versions of packing problems with a variety of applications. Our notion for time-expanded packings, which we introduce in two natural variations, requires elements to be part of the solution for several consecutive time steps. Despite the fact that the time-expanded counterparts of most combinatorial optimization problems become computationally hard to solve, we present strong approximation algorithms for general dependence systems and matroids, respectively, depending on the considered variant. More precisely, for both notions of time-expanded packings that we introduce, the approximation guarantees we obtain are at most a small constant-factor worse than the best approximation algorithms for the underlying problem in its non-time-expanded version.

1 Introduction

The need to model the temporal dimension in real-world applications has sprung the development of several new branches in combinatorial optimization. While such extensions have been studied in depth in some fields, scheduling and network flow theory being notable examples, other areas of combinatorial optimization did not receive a similar attention. This paper presents a new model for extending combinatorial optimization problems over time with natural applications in scheduling, packing, and knapsack problems.

Consider the classical setting of combinatorial optimization problems, defined by a set system $(A, \mathcal{X})$, where A is a finite *ground set* and $\mathcal{X} \subseteq 2^A$ describes the *feasible subsets* of A. Furthermore, weights $w_a \in \mathbb{Z}_{\geqslant 0}$ are given for $a \in A$. We denote by $P = (A, \mathcal{X}, w)$ the classical linear optimization problem associated with the set system $(A, \mathcal{X})$, which asks to find a feasible set $S \in \mathcal{X}$ maximizing $w(S) := \sum_{a \in S} w_a$. We define two natural time-expanded versions of $P(A, \mathcal{X}, w)$, which we call *integer time-expanded packings* and *binary time-expanded packings*. In both versions we are given—in addition to $(A, \mathcal{X}, w)$—a time horizon $T \in \mathbb{Z}_{>0}$ and durations $\tau_a \in \mathbb{Z}_{>0}$ for $a \in A$ that satisfy $\tau_a \leqslant T$. A solution to both of our time-expanded versions is a collection $\mathcal{S} = \{S_t \in \mathcal{X} : t \in [T]\}$, where $[T] := \{1, \ldots, T\}$, such that for each $a \in A$, we have

- for *integer time-expanded packings*: $\{t \in [T] : a \in S_t\}$ is either empty, or can be partitioned into intervals of length τ_a;

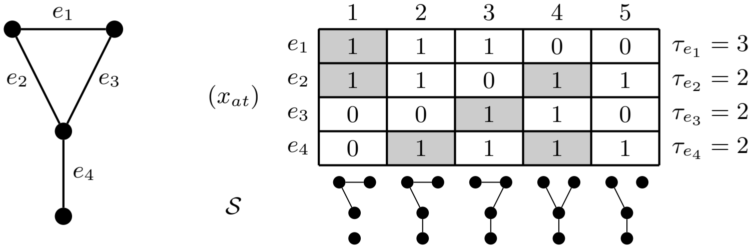
* Partially supported by NSF grant CCF-1115849.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 64–76, 2014.

– for *binary time-expanded packings*: $\{t \in [T] : a \in S_t\}$ is either empty, or is a single interval of length τ_a.

The objective in both variants of the problem is to maximize the total weight $w(\mathcal{S}) = \sum_{t \in [T]} w(S_t)$ of the solution.

A time-expanded solution $\mathcal{S}$ can be equivalently represented by the $|A| \times T$ binary matrix (x_{at}) where each column $x_{\cdot t}$ is the incidence vector of $S_t \in \mathcal{S}$ (see Figure 1 for an example of integer time-expanded packing)[1]. In this case the duration property states that in each row $x_{a\cdot}$, the length of any maximal consecutive-one sequence is a multiple of τ_a in the integer case, or precisely τ_a in the binary case. An informal phrasing of this condition is that whenever an element is *activated* it must remain in the solution for exactly τ_a time units.



	1	2	3	4	5	
e_1	1	1	1	0	0	$\tau_{e_1} = 3$
e_2	1	1	0	1	1	$\tau_{e_2} = 2$
e_3	0	0	1	1	0	$\tau_{e_3} = 2$
e_4	0	1	1	1	1	$\tau_{e_4} = 2$

Fig. 1. A graph (left) and a corresponding integer time-expanded forest (right) for $T = 5$. The matrix elements corresponding to activations are shaded in gray.

Time-expanded variants of classical combinatorial optimization problems are typically NP-hard. For example, even the trivial optimization problem that asks to pick at most one element of A—i.e., the feasible sets are independent sets in a uniform matroid of rank one—transforms into the integer/binary knapsack problem in its integer/binary time-expanded version. Whereas this only shows weak NP hardness, time-expanded variants of more complicated problems turn out to be more difficult. In particular, binary time-expanded matroid optimization is strongly NP-hard, while integer time-expanded bipartite matching is APX-hard. We present these results in the full version of the paper. In light of the latter results, we concentrate our efforts on finding constant-factor approximations. The following theorems summarize our main contribution. Recall that $(A, \mathcal{X})$ is an independence system if $X \subseteq Y \in \mathcal{X}$ implies $X \in \mathcal{X}$.

Theorem 1. *Let $P = (A, \mathcal{X}, w)$ be matroid optimization problem. Then, the binary time-expanded variant of P admits a 4-approximation algorithm.*

Theorem 2. *Let $P = (A, \mathcal{X}, w)$ be a combinatorial optimization problem admitting a polynomial β-approximation algorithm for $\beta = \beta(A, \mathcal{X})$, with $(A, \mathcal{X})$*

[1] We remark that since T is an integer parameter in the input, an explicit description of a time-expanded solution is in general exponential in the input size. To avoid confusion with output-polynomial algorithms, we stress here that all algorithms described in this paper run in *input*-polynomial time. In particular, the time-expanded solutions they provide can be encoded efficiently.

an independence system. Then, the integer time-expanded variant of P admits an $\alpha\beta$-approximation algorithm, with $\alpha \approx 1.691030$.

The exact value of α in Theorem 2 is $\alpha = \sum_{i=1}^{\infty} 1/s_i$, where $\{s_i\}_{i \in \mathbb{Z}_{\geqslant 1}}$ is the sequence defined recursively by $s_1 = 1$ and $s_{n+1} = s_n(s_n + 1)$ for $n \geqslant 2$. The sequence $\{s_i + 1\}_{i \in \mathbb{Z}_{\geqslant 1}}$ is known as Sylvester's sequence [19], and the constant α appears as the approximation guarantee of several well-known algorithms [12,14,5]. By a matroid optimization problem we mean that the set system $(A, \mathcal{X})$ is a matroid. The proofs of Theorems 1 and 2 are presented in Sections 2 and 3, respectively, while some details are deferred to the full version of the paper. In the remainder of this section we mention a number of applications of time-expanded packings, together with the implications we can derive from Theorems 1 and 2, and discuss some further related work.

Applications

Orthogonal rectangle packing. Here, we are given a set of weighted axis-parallel rectangles in $\mathbb{R}^2$ and a large rectangle Q. The task is to find a maximum weight packing of the rectangles into Q. A packing is an axis-parallel placement of rectangles, so that no two packed rectangles intersect in their interior. It is easy to see that if one is allowed to pack any arbitrary number of copies of every rectangle, then the problem can be modeled as the time-expanded integer knapsack problem, where the heights and widths of the given rectangles correspond to item sizes for the knapsack problem and durations of these items, respectively. The height of Q corresponds to the size of the knapsack, and its width corresponds to the total duration T. By applying Theorem 2 on this instance, we obtain an $(\alpha + \epsilon)$-approximation algorithm for this problem, as the set system corresponding to solutions of the knapsack problem is an independence system, and the integer knapsack problem admits an FPTAS (see e.g. [9,13]). Jansen et al. [10] gave a $(2 + \epsilon)$-approximation for the harder binary variant of this problem. Our result improves this approximation for the integer case. In fact, by applying Theorem 2 recursively d times, we can obtain the following result for the more general, *Box Packing* problem, consisting of packing d-dimensional boxes into a large d-dimensional box.

Corollary 1. *The d-dimensional Box Packing problem admits an $\alpha^{d-1}(1 + \epsilon)$-approximation algorithm for any $\epsilon > 0$.*

Cooperative scheduling. A second application of Theorem 2, which we call *Cooperative Scheduling*, models the following production problem. Consider a factory with a set M of machines where a set A of goods can be produced. The production of one unit of good a gives a profit w_a, but requires the cooperation (or joint operation) of a specific subset $M_a \subseteq M$ of the machines for a certain production time τ_a (no preemption is allowed). Goods can be produced in any quantity, and each machine can work on at most one good at any given time. Assuming that the factory is rented for T time units, the goal is to find a production schedule that maximizes the total profit within the time window, where a production schedule specifies which goods should be produced and when. By identifying the machine set M with the vertices of an hypergraph H, and each M_a with an hyperedge, Cooperative Scheduling can be easily seen to be equivalent to the time-expanded version of hypergraph matching (also known as set packing). Indeed, a subset of goods can be simultaneously produced if and only if the corresponding hyperedges form a matching in H, and the production times for the goods, together with the no-preemption assumption, imply the duration property. Theorem 2 alongside the $(\frac{k+1}{2}+\epsilon)$-approximation algorithm of Berman [4] for hypergraph matching with hyperedges with at most k elements, and the polynomiality of the maximum matching problem in graphs implies the following result.

Corollary 2. *Cooperative Scheduling for instances with* $|M_a| \leqslant k$ *for all goods* $a \in A$ *admits a* $(\alpha\frac{k+1}{2}+\epsilon)$*-approximation algorithm. If* $|M_a| \leqslant 2$ *for all goods the approximation ratio improves to* α*.*

We note that when each machine $i \in M$ can simultaneously process (at most) $b_i \in \mathbb{Z}_{\geqslant 1}$ goods and $|M_a| \leqslant 2$ we obtain the time-expanded b-matching problem, and hence this variant also admits an α-approximation algorithm.

Machine maintenance. We now mention an application of binary time-expanded packings. Consider the following *Machine Maintenance* problem. A factory containing n machines I needs to continuously perform a set of $m \leqslant n$ jobs J. Each machine can process at most one job at any given time. Furthermore, each machine can process only a subset of the jobs, expressed naturally via a bipartite graph $G = (I \cup J, E)$. Next, assume that machines need to undergo *maintenance*, during which they must be idle. Profits are assigned to machines to reflect how important it is to maintain them. For example, profits can reflect the time that elapsed after the last maintenance of the machine. Assuming that maintenance of machine i requires τ_i time units, the goal is to maintain the maximum profit set of machines within a given working cycle, consisting of T time units, while maintaining a constant processing of all jobs.

The latter maintenance problem can be cast as the binary time-expanded matroid optimization problem, with the matroid corresponding to the dual of the *scheduling matroid* associated with the graph G. Recall that the scheduling matroid corresponding to G has as bases all minimal subsets of machines $I' \subseteq I$ with the property that the machines in I' suffice to schedule all jobs in J. Hence, the dual of this matroid has as independent sets all sets of machines $I'' \subseteq I$, so that all jobs can be scheduled only using the machines in $I \setminus I''$. It is now evident that two problems are indeed the same.

We note that the latter problem can have another application, in which machines $a \in I$ that are not used can be *rented* for τ_a consecutive time steps at a given profit. This variant, denote by *Machine Renting*, can also be meaningful in the integer setup, where a machine can be rented in several periods to the same client. We conclude with the following corollary of Theorems 1 and 2.

Corollary 3. *Machine Maintenance and Machine Renting admit approximation algorithms with approximation guarantees* 4 *and* α*, respectively.*

Further Related Work

We review here some work related to temporal extensions of combinatorial problems. Flows over time are a well-known temporal extension of network flows. The field originated from a seminal paper by Ford and Fulkerson [7] which extended the standard maximum flow problem. For an extensive review of flows over time we refer to [18]. We refer to [11] for a recent development combining discrete and continuous flows over time in a unique framework.

In scheduling theory the dimension of time is inherent in most problems. We refer the reader to the book of Pinedo [16] for a thorough treatment of this topic. The scheduling problems which are most relevant to this work are profit maximization scheduling problems [3,2], in which not all given jobs need to be processed. Instead, each job is associated with a certain profit that is gained in case it is completed before its deadline.

2 Binary Time-Expanded Packings on Matroids

In this section we will present an algorithm proving Theorem 1. Thus, we assume that $(A, \mathcal{X})$ is a matroid. We first give a roadmap of our proof. We start by defining a relaxation of binary time-expanded packings, which we call *weak binary time-expanded packings*, and show that any weak binary time-expanded packing can be transformed into a binary time-expanded packing by losing at most a factor of 2. We then design a 2-approximation for finding a weak binary time-expanded packing. To prove that our algorithm is indeed a 2-approximation for weak binary time-expanded packings, we show that it is actually a 2-approximation even for a relaxed version of weak binary time-expanded packings, which we call *non-consecutive binary time-expanded packings*.

Definition 1 (Weak Binary Time-Expanded Packing). *A collection $\mathcal{S} = \{S_t \in \mathcal{X} : t \in [T]\}$ is a* weak time-expanded packing *if for each $a \in A$, the set $\{t \in [T] : a \in S_t\}$ is either*

(i) the empty set, or
(ii) an interval of size τ_a, or
(iii) an interval of size $< \tau_a$ that contains T.

Notice that a binary time-expanded packing is the same as a weak binary time-expanded packing without (iii). Hence, weak binary time-expanded packings can be considered as a temporal restriction of a bigger binary time-expanded packing to a smaller horizon T. We say that an element $a \in A$ appears in a weak binary time-expanded packing $\mathcal{S}^w$ if it is contained in at least one S_t^w. An element $a \in A$ is a *partial element* of $\mathcal{S}^w$ if it appears in the packing for strictly less than τ_a time steps, i.e., it fulfills condition (iii) of the definition of a weak binary time-expanded packing.

Lemma 1. *Any weak binary time-expanded packing $\mathcal{S}^w$ can be transformed efficiently into a binary time-expanded packing $\mathcal{S}$ with $w(\mathcal{S}) \geqslant \frac{1}{2}w(\mathcal{S}^w)$.*

Proof. Given a weak binary time-expanded packing $\mathcal{S}^w = \{S_t^w \in \mathcal{X} : t \in [T]\}$, we define two binary time-expanded packings $\mathcal{S}^1, \mathcal{S}^2$. $\mathcal{S}^1$ is obtained from $\mathcal{S}^w$ by removing all partial elements. Furthermore, $\mathcal{S}^2$ is a packing containing only the partial elements $a \in A$, each from $t = 1$ to $t = \tau_a$. Notice that $\mathcal{S}^2$ is indeed a binary time-expanded packing since the set $A_p \subseteq A$ of all partial elements of $\mathcal{S}^w$ is a subset of $\mathcal{S}_T^w$, and therefore $A_p \in \mathcal{X}$. It remains to observe $w(\mathcal{S}^1) + w(\mathcal{S}^2) \geqslant w(\mathcal{S}^w)$. Hence, the better of $\mathcal{S}^1$ and $\mathcal{S}^2$ is a 2-approximation of $\mathcal{S}$. □

We now introduce a 2-approximation for finding a weak binary time-expanded packing. Algorithm 1 describes our procedure, which starts with an empty packing and iteratively includes elements in non-increasing order of their weights. For readability, we use '+' and '−' to denote the addition and subtraction of a single element from a set, i.e., $S + a_1 - a_2 = (S \cup \{a_1\}) \setminus \{a_2\})$.

Algorithm 1. 2-approximation for weak binary time-expanded packings

```
1. Initialize $S_t := \emptyset$ for $t \in [T]$.
2. For $a \in A$, in order of non-increasing weights:
     if $\exists t \in [T]$ with $S_t + a \in \mathcal{X}$:
       Let $t_1 := \min\{t \in [T] : S_t + a \in \mathcal{X}\}$.
       Let $t_2 := \min\{t_1 + \tau_a - 1, T\}$.
       Update $S_t := S_t + a$ for $t \in \{t_1, \dots, t_2\}$.
     end if
   end for
3. Return $\{S_t : t \in [T]\}$.
```

We start by showing that Algorithm 1 returns a feasible solutions, i.e., a weak binary time-expanded packing. From the definition of t_1 and t_2 in each iteration of the for-loop, it is clear that conditions (i)–(iii) of the definition of a weak binary time-expanded packing are fulfilled. For feasibility, the only property that remains to be checked is whether the returned solution $\{S_t : t \in [T]\}$ satisfies $S_t \in \mathcal{X}$ for $t \in [T]$. We prove the following lemma which implies feasibility.

We denote by rank : $2^A \to \mathbb{Z}_{\geqslant 0}$ the rank function of the matroid $(A, \mathcal{X})$, and by span : $2^A \to 2^A$ its span function. Recall that the rank and span are defined by $\text{rank}(S) := \max\{|I| : I \in \mathcal{X}, I \subseteq S\}$ and $\text{span}(S) := \{a \in A : \text{rank}(S + a) = \text{rank}(S)\}$, where $S \subseteq A$.

Lemma 2. *At the beginning of any iteration of the for-loop of Algorithm 1, we have* $\text{span}(S_t) \supseteq \text{span}(S_{t+1})$ *for* $t \in [T-1]$.

Before proving Lemma (2), we notice that it indeed implies that Algorithm 1 returns a feasible solution. Whenever an element $a \in A$ is inserted in time steps $\{t_1, \dots, t_2\}$, we have that the set S_{t_1} before insertion satisfies $S_{t_1} + a \in \mathcal{X}$ by choice of t_1. In other words $a \notin \text{span}(S_{t_1})$. By Lemma 2, $a \notin \text{span}(S_t)$ for any $t \in \{t_1, \dots, t_2\}$, and hence, $S_t + a \in \mathcal{X}$. Thus, the insertion of elements $a \in A$ during Algorithm 1 preserves independence of the sets S_t, and hence, the returned set is feasible.

Proof (Lemma 2). We prove the lemma by showing that the claimed property is preserved whenever an element $a \in A$ is added in the for-loop of Algorithm 1. The lemma clearly holds at the beginning of the algorithm, when all sets S_t are empty. Hence, let S_t for $t \in [T]$ be the sets before the element $a \in A$ is considered in the for-loop of the algorithm, and let G_t for $t \in [T]$, be the sets after the update step corresponding to a. Thus, $G_t = S_t$ for $t < t_1$ or $t > t_2$, and $G_t = S_t + a$ for $t \in \{t_1, \ldots, t_2\}$. Assuming that the lemma holds before considering a, i.e., $\text{span}(S_{t+1}) \subseteq \text{span}(S_t)$ for $t \in [T]$, we have to show $\text{span}(G_{t+1}) \subseteq \text{span}(G_t)$ for $t \in [T]$. For $t \in [t_1 - 1]$, we have

$$\text{span}(G_t) = \text{span}(S_t) = \text{span}(S_t + a) \supseteq \text{span}(S_{t+1} + a) \supseteq \text{span}(G_{t+1}), \qquad (1)$$

where the second equality follows by the fact that $t < t_1$ and therefore $S_t + a \notin \mathcal{X}$, and the first '$\supseteq$' follows from $\text{span}(S_t) \supseteq \text{span}(S_{t+1})$. For the remaining case $t \in \{t_1, \ldots, T\}$, we have that if $G_{t+1} = S_{t+1} + a$ then also $G_t = S_t + a$. Hence, either $G_{t+1} = S_{t+1}$, in which case

$$\text{span}(G_{t+1}) = \text{span}(S_{t+1}) \subseteq \text{span}(S_t) \subseteq \text{span}(G_t),$$

or $G_{t+1} = S_{t+1} + a$, and

$$\text{span}(G_{t+1}) = \text{span}(S_{t+1} + a) \subseteq \text{span}(S_t + a) = \text{span}(G_t).$$

□

To show that Algorithm 1 is a 2-approximation for finding a weak binary time-expanded packing, we introduce an even weaker notion of packings, which we name *non-consecutive binary time-expanded packings*, and show that Algorithm 1 is even a 2-approximation for this weaker notion.

Definition 2 (Non-Consecutive Binary Time-Expanded Packings). *A family* $\{S_t \in \mathcal{X} : t \in [T]\}$ *is a* non-consecutive binary time-expanded packing *if* $|\{t \in [T] : a \in S_t\}| \leqslant \tau_a$ *for all* $a \in A$.

Hence, elements need not appear consecutively in a non-consecutive binary time-expanded packing. The only remaining relation between the time steps, is that an element $a \in A$ can not occur in more than τ_a time steps.

Lemma 3. *Algorithm 1 is a* 2*-approximation for the non-consecutive binary time-expanded packing problem.*

Proof. To prove the lemma, we show that finding a maximum weight non-consecutive binary time-expanded packing can be formulated as a matroid intersection problem, and Algorithm 1 can be interpreted as the natural greedy algorithm for this problem. The result then follows by the fact that the greedy algorithm for matroid intersection problems is a 2-approximation.

For each $t \in [T]$, let $A^t = \{a^t : a \in A\}$ be an independent copy of A, and let $\mathcal{A} = \cup_{t \in [T]} A^t$. Consider the following two matroids $M_1 = (\mathcal{A}, \mathcal{X}_1)$ and $M_2 = (\mathcal{A}, \mathcal{X}_2)$ over the common ground set $\mathcal{A}$. A set $W \subseteq \mathcal{A}$ is independent in M_1 if for every $t \in [T]$, we have $\{a \in A : a^t \in W \cap A^t\} \in \mathcal{X}$. M_1 is clearly a matroid, since it is the disjoint union of independent copies of the matroid $(A, \mathcal{X})$. Furthermore, a set $W \subseteq \mathcal{A}$ is independent in M_2 if for every $a \in A$, we have $|\{t \in [T] : a^t \in W\}| \leqslant \tau_a$. Hence, M_2 is a partition matroid. Observe that there is a one-to-one correspondence between common independent sets $W \subseteq \mathcal{A}$ of both matroids M_1 and M_2, and non-consecutive binary time-expanded packings, where we interpret W as the packing $\{W_t : t \in [T]\}$, where $W_t = \{a \in A : a^t \in W\}$.

Let $A = \{a_1, \ldots, a_n\}$ be the numbering that corresponds to the order in which Algorithm 1 considers the elements of A in the for-loop. Hence, $w(a_1) \geqslant \ldots \geqslant w(a_n)$. Thus, Algorithm 1 can naturally be interpreted as the greedy algorithm for the matroid intersection problem on $\mathcal{A}$, considering the elements in the order $a_1^1, \ldots, a_1^T, \ldots, a_n^1, \ldots, a_n^T$, where element a_i^t has weight $w(a_i^t) = w(a_i)$. Since the greedy algorithm for matroid intersection is a 2-approximation (see e.g. [17]), the result follows. □

Since every weak binary time-expanded packing is as well a non-consecutive binary time-expanded packing, Lemma 3 implies the following.

Corollary 4. *Algorithm 1 is a 2-approximation for the weak binary time-expanded packing problem.*

Finally, combining Lemma 3 with Lemma 1, we obtain a 4-approximation for the binary time-expanded packing problem, as claimed by Theorem 1. It is an interesting open question if this analysis can be tightened.

3 Integer Time-Expanded Packing on Independence Systems

We sketch next the proof of Theorem 2. To state the algorithm that achieves the claimed approximation guarantee we establish some further notation. For an item $a \in A$ let $T_a := \lfloor T/\tau_a \rfloor$ and $\bar{w}_a := w_a \tau_a T_a$ denote the maximal possible number of activations of item a in any feasible solution, and the total value attainable by this many activations, respectively.

Our algorithm, which we call *Total Value Greedy*, determines first a β-approximate solution S^* to the static maximization problem *with respect to the weight function* $\bar{w}$. The time expanded solution $\mathcal{S}$ returned by the algorithm is then defined by setting $S_t = \{a \in S^* : t \leqslant T_a \tau_a\}$ for every $t \in [T]$. Hence, each element of S^* is repeated as often as possible, starting at the first time step.

Lemma 4. *Total Value Greedy is a $\alpha\beta$-approximation for the integer time-expanded packing problem.*

Before proving Lemma 4, which implies Theorem 2, we briefly mention another, very related, greedy approach, which we term *Simple Greedy*. The Simple Greedy algorithm first determines a β-approximation of the maximum-weight static solution $S^* \in \mathcal{X}$ *with respect to the weight function* w, and then returns the solution $\mathcal{S} = \{S_t \in \mathcal{X} : t \in [T]\}$ defined by $S_t = \{a \in S^* : T_a \tau_a \leqslant t\}$, analogously to the Total Value Greedy. It is quite easy to show that Simple Greedy is a 2β-approximation, and hence, the challenge in finding strong approximations for integer time-expanded packings lies in beating the approximation factor of 2β. This is in contrast to the binary time-expanded packing problem, where even getting any $O(1)$-approximation is nontrivial.

We highlight that the Total Value Greedy was already considered by Kohli and Krishnamurti [12] for the knapsack problem, which—as we already discussed—is a special case of integer time-expanded packings. More precisely, they showed that the Total Value Greedy is an α-approximation for the integer knapsack problem, thus proving Lemma 4 for the special case when the underlying independence system is a uniform matroid of rank one. Furthermore, the analysis in [12] was shown to be tight, i.e., for any $\epsilon > 0$, there is an instance of the integer knapsack problem for which the solution returned by the Total Value Greedy is *not* an $(\alpha - \epsilon)$-approximation. Since integer time-expanded packings generalize the integer knapsack problem, our analysis is also tight. We stress that our analysis differs significantly from that in [12]. The analysis in [12] relies on the inequality $T_{a^*}\tau_{a^*}w_{a^*} \geqslant T_a \tau_a w_a$ for all $a \in A$, where a^* is the optimal element to repeat. In our setup, the latter inequality can only be obtained for *sums* of elements with various duraitions. This results in various new difficulties, from the derivation of a linear program bounding the approximation guaranetee (In [12] an *integer program* is used), to the analysis of the resulting linear program. The rest of this section is devoted to the proof of Lemma 4.

Proof of Lemma 4. The proof follows four main steps, which we outline next. First, we present a combinatorial argument that allows for deducing an approximation guarantee that depends on a set of problem parameters. For each horizon $T \in \mathbb{Z}_{>0}$, the worst-case set of parameters can be determined through a linear program P_T, whose optimal value thus represents an approximation guarantee that Total Value Greedy satisfies for any instance with horizon T. Next, an *infinite* linear program P_∞ is derived, whose optimal value bounds the optimal value of P_T for all $T \in \mathbb{Z}$ simultaneously. In the third step, the linear programming dual D_∞ of P_∞ is obtained, and weak duality is established, implying that any solution of D_∞ provides an approximation guarantee for Total Value Greedy. Finally, we explicitly construct a solution with value $\alpha\beta$ for D_∞, proving the theorem. We give a complete description of the first step only.

Let $\mathcal{G}$ be the time-expanded solution returned by Total Value Greedy, which repeats a solution $G \in \mathcal{X}$. The value of $\mathcal{G}$ is $\textsc{Alg} := w(\mathcal{G}) = \sum_{a \in G} T_a \tau_a w_a$, and $\beta \bar{w}(G) \geqslant \bar{w}(S)$ for all $S \in \mathcal{X}$. Let also $\mathcal{S}^* = (S_1, \ldots, S_T)$ be an optimal time-expanded solution, with value $\textsc{Opt} := w(\mathcal{S}^*)$, and let $\delta := \frac{1}{\beta}\frac{\textsc{Opt}}{\textsc{Alg}}$. Our goal is to show that $\delta \leqslant \alpha$. If $T = 2$, one can observe that an optimal solution $\mathcal{S}^*$ can be chosen such that $S_1 = S_2$, which implies $\delta = 1$. For the same reasong we also have $\delta = 1$ for $T = 1$. Hence, in the following we assume $T \geqslant 3$.

For each $k \in [T]$, we define the subset of elements that can be repeated k times but not $k+1$ times by

$$A^k := \big\{a \in A : \tau_a \in \big(T/(k+1), T/k\big]\big\}\,.$$

As the durations τ_a are integers in $[T]$, the sets A^k partition A. By construction, for each $k \in [T]$ one has $T_a = k$ and $\tau_a \geqslant \Gamma_k$ for all $a \in A^k$, where $\Gamma_k := \lfloor T/(k+1) \rfloor + 1$. We then similarly partition each solution set S_t into the sets $S_t^k = S_t \cap A^k$, for $k, t \in [T]$.

In the following, we create a family of upper bounds for δ, parameterized by an integer $q \in [T]$. More precisely, for each $q \in [T]$, we define sets $M_1, \dots, M_q \in \mathcal{X}$ by choosing $M_j = S_{t_j}$ for $t_j := \left\lceil j \frac{T}{q+1} \right\rceil$. Now for each fixed $q \in [T]$ and $j \in [q]$,

$$\beta \cdot \textsc{Alg} = \beta \sum_{a \in G} T_a \tau_a w_a \geqslant \sum_{a \in M_j} T_a \tau_a w_a = \sum_{k=1}^{q} \sum_{a \in M_j \cap A^k} T_a \tau_a w_a = \sum_{k=1}^{q} \sum_{a \in M_j \cap A^k} k \tau_a w_a,$$

so that summing over all sets M_j for $j \in [q]$ we get

$$q\beta \cdot \textsc{Alg} \geqslant \sum_{j=1}^{q} \sum_{k=1}^{q} \sum_{a \in M_j \cap A^k} k \tau_a w_a = \sum_{k=1}^{q} k \sum_{j=1}^{q} \sum_{a \in M_j \cap A^k} \tau_a w_a. \tag{2}$$

In the last term, when we sum over $j \in [q]$, we are counting the elements $a \in S_t^k$ for all t with some repetitions.

Consider a fixed $k \leqslant q$ and a $t \in [T]$. Since elements in S_t^k have duration at least Γ_k, and the (fractional) distance in time between two solutions M_j is $T/(q+1)$, each element $a \in S_t^k$ will belong to at least $\lfloor \Gamma_k (q+1)/T - 1/T \rfloor$ different solutions M_j. The term $-1/T$ is needed to correct the counting when $\Gamma_k(q+1)/T$ is integer, in which case elements $a \in S_T^k$ might belong to $\Gamma_k(q+1)/T - 1$ different solutions M_j only. Note also that we stop at $k = q$ because, due to the choice of the solutions M_j, for $k > q$ some (or all) of the elements in S_t^k might never appear in the sets M_j. Let $z_k := (\sum_{t \in [T]} w(S_t \cap A^k))/\textsc{Opt}$ be the fraction of $\textsc{Opt}$ due to elements in A^k. Using the counting argument outlined above, for every $k \leqslant q$ we can write

$$\sum_{j=1}^{q} \sum_{a \in M_j \cap A^k} \tau_a w_a \geqslant \left\lfloor \Gamma_k \frac{q+1}{T} - \frac{1}{T} \right\rfloor \sum_{t=1}^{T} w(S_t^k) = \left\lfloor \Gamma_k \frac{q+1}{T} - \frac{1}{T} \right\rfloor z_k OPT. \tag{3}$$

Finally, combining (2) and (3) we get the bound

$$\sum_{k=1}^{q} \frac{1}{q} \left\lfloor \Gamma_k \frac{q+1}{T} - \frac{1}{T} \right\rfloor k z_k \leqslant \frac{1}{\delta}. \tag{4}$$

Let $T \in \mathbb{Z}_{\geqslant 0}$, and consider replacing δ by a variable Δ in (4). The latter derivation implies that to obtain an upper bound on δ one can maximize Δ

subject to the constraints (4) for all $q \in [T]$, the constraint $\sum_{k=1}^{T} z_k = 1$ and $z_k \geqslant 0$ for $k \in [T]$. By substituting $y_k := k\delta z_k$ and observing that $\Delta k > 0$ for all k, one arrives at the following linear program, providing an upper bound on δ.

$$
\begin{array}{llll}
 & \delta_T = \max & \displaystyle\sum_{k=1}^{T} \frac{1}{k} y_k & \\
(P_T) & s.t. & \displaystyle\sum_{k=1}^{q} \frac{1}{q} \left\lfloor \Gamma_k \frac{q+1}{T} - \frac{1}{T} \right\rfloor y_k \leqslant 1 & q \in [T] \qquad (5) \\
 & & y_k \geqslant 0 & k \in [T].
\end{array}
$$

Our goal is to create a uniform upper bound of α for the above family of linear problems, that holds for all $T \in \mathbb{Z}_{\geqslant 1}$. A natural approch is to exhibit for each $T \in \mathbb{Z}_{\geqslant 1}$ a feasible dual solution to P_T of value α. Finding a strong dual solution to P_T turns out to be a challenging task. One reason for this may be the somewhat surprising fact that the sequence of optimal values δ_T of P_T is not monotone in T. We therefore create another, infinite-dimensional, linear program P_∞ whose optimal value upper bounds P_T for all $T \geqslant 1$, and for which we can give a relatively simple dual solution of value α.

$$
\begin{array}{llll}
 & \delta_\infty = \max & \displaystyle\sum_{k \in \mathbb{Z}_{>0}} \frac{1}{k} y_k & \\
(P_\infty) & s.t. & \displaystyle\sum_{k=1}^{q} \frac{1}{q} \left\lfloor \frac{q+1}{k+1} \right\rfloor y_k \leqslant 1 & q \in \mathbb{Z}_{>0} \qquad (6) \\
 & & \displaystyle\sum_{k \in \mathbb{Z}_{>0}} \frac{1}{k+1} y_k \leqslant 1 & \qquad (7) \\
 & & y_k \geqslant 0 & k \in \mathbb{Z}_{>0}.
\end{array}
$$

Lemma 5. *$\delta_T \leqslant \delta_\infty$ for every $T \in \mathbb{Z}_{>0}$.*

To show $\delta_\infty \leqslant \alpha$, we make use of duality for infinite-dimensional linear programming. Let v_q be the dual variable to the q-th constraint (6), and let w be the dual variable corresponding to constraint (7). The standard linear programming dual of P_∞ reads as follows.

$$
\begin{array}{llll}
 & \delta'_\infty = \min & w + \displaystyle\sum_{q \in \mathbb{Z}_{>0}} v_q & \\
(D_\infty) & s.t. & \displaystyle\frac{1}{k+1} w + \sum_{q \geqslant k} \frac{1}{q} \left\lfloor \frac{q+1}{k+1} \right\rfloor v_q \geqslant \frac{1}{k} & k \in \mathbb{Z}_{>0} \qquad (8) \\
 & & w \geqslant 0, v_q \geqslant 0 & q \in \mathbb{Z}_{>0}.
\end{array}
$$

In the following lemma we show that weak duality holds for this pair of infinite linear programs, implying $\delta'_\infty \geqslant \delta_\infty$. We note that this is not true for *any* pair of dual infinite linear programs.

Lemma 6. $\delta'_\infty \geqslant \delta_\infty$.

To prove $\delta'_\infty \leqslant \alpha$, and hence conclude the proof, we now only need to provide a feasible dual solution of value α. To this end, recall the sequence $\{s_n\}_{n \in \mathbb{Z}_{>0}}$ used to define α. Our dual solution $(\tilde{w}, \tilde{v})$ is defined by setting $\tilde{w} = 1$ and

$$\tilde{v}_{s_j - 1} = 1/s_{j-1} - 1/s_{j-1}^2 + 1/s_j^2$$

for $j \geqslant 2$, and 0 elsewhere. This solution is nonzero only for variables v_q where $q = s_j - 1$ for some integer $j \geqslant 2$. The following lemma concludes the proof.

Lemma 7. *The solution $(\tilde{w}, \tilde{v})$ is feasible for D_∞, and its objective value is α.*

Finally, we note that, in fact, $\delta_\infty = \delta'_\infty = \alpha$, since, as we stated before, the result in [12] for the knapsack problem implies that our analysis is tight.

References

1. Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V.V., Marchetti-Spaccamela, M., Protasi, M.: Complexity and Approximation. Springer (1999)
2. Bar-Noy, A., Bar-Yehuda, R., Freund, A., Naor, J.S., Schieber, B.: A unified approach to approximating resource allocation and scheduling. J. ACM, 735–744 (2000)
3. Bar-Noy, A., Guha, S.: Approximating the throughput of multiple machines in real-time scheduling. SIAM J. Comput. 31, 331–352 (2002)
4. Berman, P.: A d/2 approximation for maximum weight independent set in d-claw free graphs. Nordic J. of Computing 7, 178–184 (2000)
5. Caprara, A.: Packing 2-dimensional bins in harmony. In: Proceedings of the 43rd Symposium on Foundations of Computer Science, FOCS 2002, pp. 490–499 (2002)
6. Duffin, R.J., Karlovitz, L.A.: An infinite linear program with a duality gap. Management Science 12(1), 122–134 (1965)
7. Ford, L.R., Fulkerson, D.R.: Constructing maximal dynamic flows from static flows. Operations Research 6(3), 419–433 (1958)
8. Garey, M.R., Johnson, D.S.: Computers and Intractability. A guide to the theory of NP-completeness. W.H. Freeman and Co. (1979)
9. Ibarra, O.H., Kim, C.E.: Fast approximation algorithms for the knapsack and sum of subset problems. J. ACM 22, 463–468 (1975)
10. Jansen, K., Zhang, G.: On rectangle packing: maximizing benefits. In: Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, pp. 204–213 (2004)
11. Koch, R., Nasrabadi, E., Skutella, M.: Continuous and discrete flows over time: A general model based on measure theory. Mathematical Methods Of Operations Research 73(3), 301–337 (2011)
12. Kohli, R., Krishnamurti, R.: A total-value greedy heuristic for the integer knapsack problem. Operations Research Letters 12(2), 65–71 (1992)
13. Lawler, E.L.: Fast approximation algorithms for knapsack problems. In: Proceedings of the 18th Symposium on Foundations of Computer Science, FOCS 1977, pp. 206–213 (1977)
14. Lee, C.C., Lee, D.T.: A simple on-line bin-packing algorithm. J. ACM 32, 562–572 (1985)

15. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley (1994)
16. Pinedo, M.L.: Scheduling: Theory, Algorithms and Systems, 3rd edn. Springer (2008)
17. Schrijver, A.: Combinatorial Optimization, Polyhedra and Efficiency. Springer (2003)
18. Skutella, M.: An introduction to network flows over time. Research Trends in Combinatorial Optimization, pp. 451–482. Springer (2009)
19. Sloane, N.J.A.: Sequences A000058 and A007018. The On-Line Encyclopedia of Integer Sequences, http://oeis.org

Deterministic Rectangle Enclosure and Offline Dominance Reporting on the RAM

Peyman Afshani[1,⋆], Timothy M. Chan[2, ⋆⋆], and Konstantinos Tsakalidis[3]

[1] MADALGO, Department of Computer Science, Aarhus University, Denmark
peyman@madalgo.au.dk

[2] David R. Cheriton School of Computer Science, University of Waterloo, Canada
tmchan@uwaterloo.ca

[3] Computer Science and Engineering Department, CUHK, Hong Kong
tsakalid@cse.cuhk.edu.hk

Abstract. We revisit a classical problem in computational geometry that has been studied since the 1980s: in the *rectangle enclosure* problem we want to report all k enclosing pairs of n input rectangles in 2D. We present the first deterministic algorithm that takes $O(n \log n + k)$ worst-case time and $O(n)$ space in the word-RAM model. This improves previous deterministic algorithms with $O((n \log n + k) \log \log n)$ running time. We achieve the result by derandomizing the algorithm of Chan, Larsen and Pătraşcu [SoCG'11] that attains the same time complexity but in expectation.

The 2D rectangle enclosure problem is related to the *offline dominance range reporting* problem in 4D, and our result leads to the currently fastest deterministic algorithm for offline dominance reporting in any constant dimension $d \geq 4$.

A key tool behind Chan et al.'s previous randomized algorithm is *shallow cuttings for 3D dominance ranges*. Recently, Afshani and Tsakalidis [SODA'14] obtained a deterministic $O(n \log n)$-time algorithm to construct such cuttings. We first present an improved deterministic construction algorithm that runs in $O(n \log \log n)$ time in the word-RAM; this result is of independent interest. Many additional ideas are then incorporated, including a linear-time algorithm for *merging* shallow cuttings and an algorithm for an offline *tree point location* problem.

1 Introduction

We study the problem of *rectangle enclosure*: given a set of n axis-aligned rectangles on the plane, report all k pairs (r_1, r_2) of input rectangles where r_1 completely encloses r_2. This is a classic problem in the field of computational geometry [19] with applications to VLSI design, image processing, computer graphics and databases [21,15,12,6].

[⋆] Work supported in part by the Danish National Research Foundation grant DNRF84 through Center for Massive Data Algorithmics (MADALGO).

[⋆⋆] The work of this author was done during his visit to the Department of Computer Science and Engineering, Hong Kong University of Science and Technology.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 77–88, 2014.

Previous Results. An early paper by Bentley and Wood [3] presented an $O(n \log n + k)$ worst-case time and linear-space algorithm for the related *rectangle intersection* problem (reporting all k pairs (r_1, r_2) where r_1 intersects r_2), raising the question whether the same bound could be achieved for rectangle enclosure. Vaishnavi and Wood [21] first addressed the question presenting an $O(n \log^2 n + k)$-time algorithm that uses $O(n \log^2 n)$ space. Lee and Preparata [15] improved the space bound to linear.

Further improvements were discovered in the 1990s. The linear-space algorithm of Gupta, Janardan, Smid and Dasgupta [12] and an alternative implementation by Lagogiannis, Makris and A. Tsakalidis [14] take $O((n \log n + k) \log\log n)$ worst-case time. Recently, Chan, Larsen and Pătraşcu [6] succeeded in improving the running time to the desired bound of $O(n \log n + k)$ using linear space. However their algorithm uses randomization and thus the time bound holds in expectation. All presented algorithms operate in the word-RAM model with word size $w \geq \log n$. (In fact, for all time bounds that are $\Omega(n \log n)$, they hold in the standard RAM model with $w = \log n$, since we can pre-sort and reduce to rank space.)

It is well-known that the rectangle enclosure problem is reducible to the 4D version of the *offline dominance reporting* problem: given n input and query points in $\mathbb{R}^d$, report the input points that are dominated by each query point ($(p_1, \ldots, p_d)$ is dominated by $(q_1, \ldots, q_d)$ if $p_i < q_i$ for all i). For the reduction it suffices to map each input rectangle $[x_1, x_2] \times [y_1, y_2]$ to a 4D point $(x_1, y_1, -x_2, -y_2)$ and equate the query points with the input points.

Offline dominance reporting is a fundamental problem in the area of orthogonal range searching; it has even found applications outside of computational geometry [4]. Chan, Larsen and Pătraşcu's result implies an algorithm with $O(n \log^{d-3} n + k)$ expected time for any constant dimension $d \geq 4$, where k is the total number of reported points. However their algorithm is randomized. The best deterministic algorithm known requires $O(n \log^{d-2} n + k)$ or $O((n \log^{d-3} n + k) \log\log n)$ time.

Our Contributions. We present the first deterministic algorithm for rectangle enclosure that takes $O(n \log n + k)$ worst-case time and $O(n)$ space in the standard word-RAM model. Our result thus improves over the previous deterministic algorithms of Gupta et al. [12] and Lagogiannis et al. [14] and removes randomization from the algorithm of Chan et al. [6].

Our approach also gives the currently fastest deterministic algorithm for the offline dominance reporting problem for any constant dimension $d \geq 4$, with worst-case running time $O(n \log^{d-3} n + k)$.

Our Approach. Our algorithm may be viewed as a derandomization of Chan, Larsen and Pătraşcu's [6], but significant new ideas are required.

In Chan et al.'s algorithm, randomization was used to construct combinatorial objects that have properties similar to those of *shallow cuttings for 3D dominance ranges.* Shallow cuttings were introduced by Matoušek [18], and a complicated randomized $O(n \log n)$-time algorithm was given by Ramos [20] for constructing shallow cuttings in the more general setting of 3D halfspace ranges. In a recent SODA'14 paper, Afshani and K. Tsakalidis [2] presented the first deterministic $O(n \log n)$-time algorithm for constructing shallow cuttings for 3D dominance ranges using linear space in the pointer machine model. In Section 2 we begin by improving their algorithm to run in $O(n \log \log n)$ worst-case time on the word-RAM. As an immediate consequence we obtain a deterministic algorithm for offline 3D dominance reporting that takes $O(n \log \log n + k)$ worst-case time and linear space in the word-RAM (Section 4); this result is new. Previously only $O(n \log n + k)$ worst-case time could be achieved using linear space [1,17,2].

Much further work is needed to derive our result on offline 4D dominance reporting and 2D rectangle enclosure. The crucial new ingredient is an algorithm that can *merge* two shallow cuttings for 3D dominance ranges in *linear* time; this result is obtained by modifying our shallow cutting construction algorithm in interesting ways and is described in Section 3. Then in Section 5 we use an intricate combination of Chan et al.'s approach with the deterministic shallow cutting construction and merging subroutines to achieve our final result. The combination requires a re-organization of the previous algorithm. In particular we isolate a subproblem we call *tree point location* for which we obtain a deterministic algorithm by incorporating planar separators [16] to ideas from Chan et al. This problem may be viewed as a new 2D variant of *fractional cascading* [9] and is thus of independent interest.

Notation and Definitions. Point p *dominates* point q if and only if each coordinate of p is greater than or equal to that of q. To avoid ambiguity for points on the plane we use the term "*covers*" (instead of "dominates"). Let P be a set of n points in $\mathbb{R}^d$. The *level* of any point $p \in \mathbb{R}^d$ (with respect to P) is the number of points in P that are dominated by p. The region of $\mathbb{R}^d$ dominated by p is called a *cell*. The conflict list of a cell is the subset of P that lies inside the cell.

A *k-shallow cutting* for dominance ranges on point set P is a set of *vertices* (points) S such that (i) every vertex in S has level at most $c_{\max}k$ in P for a constant $c_{\max} > 1$ and (ii) any point in $\mathbb{R}^d$ with level in P at most k is dominated by some vertex of S. Shallow cutting S is *optimal* when it contains at most $c_{\max}\frac{n}{k}$ vertices. A planar shallow cutting has the shape of an orthogonal *staircase* curve $c_1 d_1 c_2 \cdots d_{t-1} c_t$ of alternating vertical line segments $c_i d_i = [c_i(x)] \times [c_i(y), d_i(y)]$ and horizontal line segments $d_i c_{i+1} = [d_i(x), c_{i+1}(x)] \times [d_i(y)]$ (Fig. 1a). We call points c_i, d_i, the *corners* of the planar shallow cutting.

2 Construction of 3D Dominance Shallow Cuttings

Theorem 1. *An optimal k-shallow cutting for 3D dominance ranges on n input points and for any integer k can be constructed deterministically in $O(n \log \log n)$ worst-case time and $O(n)$ space.*

Algorithm Sketch. Following [2] we sort the points and sweep a plane parallel to the xy-plane considering the points in non-decreasing z-coordinate. This reduces

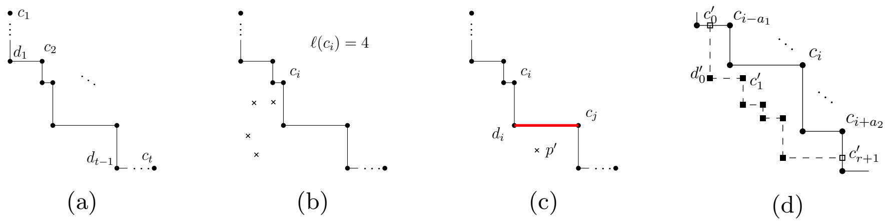


Fig. 1

the problem to the problem of maintaining a 2D shallow cutting under only insertions, specifically the planar shallow cutting of the xy-projection of the points of P that lie *below* the sweep plane. When the point with the next highest z-coordinate is considered (i.e., the next insertion), we update the corresponding 2D shallow cutting. The idea is that we do not need to change the planar shallow cutting for most insertions. However each insertion can increase the level of the shallow cutting corners. Thus, once in a while, the planar shallow cutting needs to be fixed. This is done by removing some consecutive parts of it and then adding new staircase "patches" that are covered by the parts just removed (details will follow). Every time a planar shallow cutting cell is removed, a 3D shallow cutting cell is created using the z-coordinate of the sweep plane (i.e., when a planar staircase corner with coordinates (x, y) is removed, a 3D vertex (x, y, z) is created where z is the coordinate of the sweep plane). Finally when the sweep terminates, the remaining planar shallow cutting cells are turned into 3D shallow cutting cells using $+\infty$ as the z-coordinate. It is easily verified that the produced 3D shallow cutting of P is correct and its size is equal to the number of planar shallow cutting corners removed throughout the algorithm plus the number of planar shallow cutting corners that remain when the sweep terminates.

Remark. The sketched algorithm is a variant of Afshani and Tsakalidis' [2, Section 3] with the significant difference that it is insertion-only (sweeping upwards) as opposed to their deletion-only algorithm (sweeping downwards). Their algorithm has the advantage that it can compute $O(\log n)$ different shallow cuttings in total $O(n \log n)$ time. However a crucial ingredient of that algorithm is an auxiliary data structure ([2, Lemma 2]) that needs to be updated at every sweep point. Unfortunately, we cannot see a way to update the auxiliary data structure in $O(\log \log n)$ time in the word RAM model. Fortunately, as we shall see later, we can achieve the desired $O(n \log \log n)$ running time without any auxiliary data structures, by just changing the direction of the sweep.

The Invariant. The planar shallow cutting is maintained in the form of a staircase $S = c_1 d_1 \cdots c_t$, composed of *inner* corners $d_1, \ldots, d_{t-1}$ and *outer* corners $c_1, \ldots, c_t$. The outer corners c_1 and c_t are (conceptually) at infinity, i.e. $c_1(y) = +\infty$ and $c_t(x) = +\infty$ (Fig. 1a). We maintain the invariant that the inner corners dominate at least k and the outer corners at most $10k$ points.

Details. We now discuss the details of the algorithm. Let $p = (x, y, z)$ be the next point swept by the sweep plane. Remember that we need to insert the point $p' = (x, y)$ into the dynamic planar shallow cutting. To do that we maintain the following structures: first, we use a dynamic van Emde Boas tree on the x-coordinates of the staircase, and second, for every outer corner c of the staircase, we keep track of the number of points it covers, $\ell(c)$ (Fig. 1b), as well as a linked list containing them, $L(c)$ (i.e., the conflict list and its size). Using these we can now insert the point p'. The dynamic van Emde Boas tree enables us to find the inner corner d_i immediately to the left of the point p', helping us decide whether p' lies above or below the staircase (Fig. 1c). In the former case, we are done. In the latter case, for every outer corner c_j that covers p', we increase $\ell(c_j)$ by one and then append p' to $L(c_j)$. If for all such corners we still have $\ell(c_j) \leq 10k$, then the invariant is maintained and thus we are done. However, it is possible that for some corners this invariant is violated. Below we describe how to "patch" such violated invariants.

Complexity. Sorting by z-coordinate takes $O(n \log\log n)$ time in total [13]. Finding d_i takes $O(\log\log n)$ time [22] and thus $O(n \log\log n)$ time in total. It turns out the rest of the algorithm consumes linear time. If there are $m(p')$ corners that cover p', then updating their relevant information takes $O(m(p'))$ time. Note that p' is now added to the conflict lists of $m(p')$ corners. Notice that since the size of each conflict list is $O(k)$, the total running time of this step is $O(Tk)$ where T is size of the shallow cutting. If we can prove that $T = O(n/k)$, then this running time is linear. Now we describe how to maintain the invariant which also guarantees the upper bound on T.

Patching. Let c_i be the leftmost outer corner whose invariant is violated. Let a_1 and a_2 be the largest integers such that all the outer corners $c_{i-a_1}, c_{i-a_1+1}, \ldots, c_{i+a_2}$ have levels greater than $3k$. To patch the staircase we begin by finding a new outer corner c'_0 at the same y-coordinate as c_{i-a_1}, such that it covers $3k$ points, as depicted in Fig. 1d. c'_0 can be found in $O(k)$ time using a linear time selection algorithm on the conflict list of c_{i-a_1}. Next, we find the inner corner d'_0 directly below c'_0 that covers k points. Now, we alternate between finding new outer and inner corners: at the j-th step, we find the outer corner c'_j that dominates $2k$ points, and then the inner corner d'_j that dominates k points. As before, using the right conflict list, each of these corners can be found in $O(k)$ time. The patching is terminated at a point c'_{r+1} with level $3k$ that lies below the outer corner c_{i+a_2} (Fig. 1d). Finally, the new outer and inner corners (from c'_0 to c'_{r+1}) are incorporated into the staircase, the old corners $(c_{i-a_1}, \ldots, c_{i+a_2})$ are removed, and the van Emde Boas tree is also updated to reflect the changes in the staircase.

Analysis of Patching. It easy to see that the overall cost of patching is $O(Tk)$ since each new inner or outer corner can be found in $O(k)$ time. Moreover, the facts that the levels of the removed corners differ from the levels of the created ones by at least k (except for only c'_0 and c'_{r+1}) and that at least 5 corners are created for every patch, suffice to claim that $T \leq C_0 \frac{n}{k}$ for a positive constant C_0. We set $c_{\max} := \max\{C_0, 10\}$. A detailed proof on the symmetric approach is found in [2, Section 3].

3 Merging Two 3D Dominance Shallow Cuttings

We begin by a naive merging algorithm.

Lemma 1. *Consider two point sets P_1 and P_2 that contain n_1 and n_2 points respectively. For $i = 1, 2$, assume we are given a k_i-shallow cutting C_i on P_i of size m_i such that the conflict list of every cell contains at most $\beta_i k_i$ points of P_i. A $(\min\{k_1, k_2\})$-shallow cutting C on the union point set $P = P_1 \cup P_2$ can be built in $O((m_1 + m_2)\log\log(m_1 + m_2))$ time, such that C contains $O(m_1 + m_2)$ cells and the conflict list of every cell in C contains $\leq \beta_1 k_1 + \beta_2 k_2$ input points.*

Proof. Let R be the subset of $\mathbb{R}^3$ that is dominated by at least one vertex in C_1 and at least one vertex in C_2. It is easily seen that R is orthogonally convex and in fact any orthogonal ray to $y = -\infty$ or $x = -\infty$ crosses the boundary of R at most one. Thus, the complexity of the boundary of R is $O(|C_1| + |C_2|) = O(m_1 + m_2)$. Furthermore, the boundary of R can be computed with a straightforward sweep plane algorithm in $O((m_1 + m_2)\log\log(m_1 + m_2))$ time by employing a van Embe Boas tree as the search structure [22]. The shallow cutting C is defined by the vertices of the boundary of R. It is clear that every such vertex dominates either k_1 points from P_1 or k_2 points from P_2 and thus it dominates at least $\min\{k_1, k_2\}$ points of P. Similarly, each vertex on R can dominate at most $\beta_1 k_1$ points of P_1 and $\beta_2 k_2$ points of P_2. □

While the above merging algorithm is quite fast, it worsens the constants behind the parameters of the shallow cutting and thus it cannot be applied more than a constant number of times. In the next theorem, we show how such a "bad" shallow cutting can be refined into an optimal one. For this purpose we also use the following lemma.

Lemma 2. *[5, Theorem 4.3] Online point location queries on a planar orthogonal subdivision of size n can be supported in $O(\log\log n)$ worst-case time and $O(n)$ space.*

Theorem 2. *Let P be a set of n points in $\mathbb{R}^3$ with presorted z-coordinates and let C be a k-shallow cutting on P of size $\alpha n/k$, where the conflict list of every cell has size at most βk, for arbitrary constants $\alpha, \beta > 0$. Then C can be refined into an optimal k'-shallow cutting C' on P in $O(n + \frac{n}{k}\log\log n)$ time, such that $k' = \frac{k}{c_{\max}}$ for a universal constant $c_{\max}$ that does not depend on α and β. Furthermore, C' contains at most $c_{\max}\frac{n}{k'}$ cells and the conflict list of every cell in C' contains at most $c_{\max}k'$ points of P.*

Proof. We build C' using the plane sweep algorithm from the previous section. To review, the algorithm maintains a planar shallow cutting in the form of a staircase S'. To process the next point p, a predecessor query is used to find one corner of the staircase that covers the projection p' of p. As we noted during the proof of Theorem 1, other than this predecessor search, the rest of the algorithm runs in linear time. To remove this bottleneck, we use C to augment each point of P with a correct pointer to a staircase corner that covers it, thus negating the need for the predecessor search.

Hence we project the cutting C on the xy-plane and obtain an orthogonal planar decomposition of disjoint polygonal *regions* in order to support online planar orthogonal point location queries, i.e. report the region that any given query point lies in. Thus, Lemma 2 enables us to perform the following operation in $O(\log\log n)$ time: given a point q in the xy-plane, find the shallow cutting vertex $C(q)$ in C with the largest z-coordinate whose projection covers q.

Observe that the level of every vertex in C' is at most $c_{\max}k' = k$; thus, every vertex of C' is contained in at least one cell of C. We thus maintain one additional invariant in our sweep. Consider a staircase corner $v \in S'$, the shallow cutting vertex $C(v) \in C$ and its conflict list $\ell(C(v))$. We maintain the invariant that if the projection p' of a point $p \in \ell(C(v))$ lies below the staircase S', then p is assigned a pointer to a staircase corner that covers p'. This invariant removes the need for the predecessor search during the sweep.

By looking at the algorithm in Section 2, it is clear that the invariant can only be violated when a new staircase corner v is created on S' (during the patching phase). To fix the invariant, by Lemma 2 we can find the vertex $C(v)$ and its conflict list $\ell(C(v))$. We go through each point in $\ell(C(v))$ and if its projection is covered by v, then we assign it a pointer to v. This takes $O(k)$ time, which is proportional to the time required for creating the staircase corner v. Thus, this incurs only an additive $O(\log\log n)$ time factor per shallow cutting vertex. □

Corollary 1. *Given two point sets $A, B \in \mathbb{R}^3$ presorted by z-coordinate and their respective k-shallow cuttings, for an integer $k = \Omega(\log\log(|A| + |B|))$, an optimal $\frac{k}{c_{\max}}$-shallow cutting on the union point set $A \cup B$ can be constructed deterministically in $O(|A| + |B|)$ worst-case time.*

4 Offline 3D Dominance Reporting

Theorem 3. *Offline 3D dominance reporting on n input points, n query points and k reported points can be solved deterministically in $O(n \log\log n + k)$ worst-case time and $O(n)$ space.*

Proof. We follow the approach of Afshani [1] for online 3D dominance reporting queries in internal memory. We presort all coordinates [13] and construct a single $(\log n)$-shallow cutting for 3D dominance ranges by using the algorithm of Theorem 1 in $O(n \log\log n)$ worst-case time. We *assign* every query point to a cell of the cutting whose vertex dominates it by (offline) point location in a planar orthogonal subdivision obtained from the projection of the cutting. By Lemma 2 this takes $O(n \log\log n)$ worst-case time [5]. We resolve all the assigned queries by solving independently for each of the $O(\frac{n}{\log n})$ cells an offline 3D dominance reporting *subproblem* on the conflict list of the cell. The 3D dominance reporting algorithm of [17] reports all (at most k) output points in $O(\frac{n}{\log n} \log n \log\log n + k) = O(n \log\log n + k)$ total worst-case time. This leaves at most $O(\frac{k}{\log n})$ queries *unresolved*, since each unresolved query reports $\Omega(\log n)$ points. We can also decrease the number of input points to $O(\frac{k}{\log n})$ in the same way by repeating the above with the roles of the input and query points reversed. Finally we solve a single offline 3D dominance reporting problem on all the remaining input points and unresolved queries by the more expensive $O(n \log n + k)$-time algorithm of [17], which now takes $O(k)$ time. □

Corollary 2. *Offline 3D dominance reporting on n input points, n query points and k reported points can be solved deterministically in $O(n+k)$ worst-case time and $O(n)$ space, if $n < \bar{w}^{O(1)}$ and a global look-up table has been constructed in $o(2^{\bar{w}})$ worst-case time for a parameter $\bar{w} \leq w$.*

Proof. We modify the proof of Theorem 3. In the construction of the $(\log n)$-shallow cutting we replace van Emde Boas trees [22] with atomic heaps [10], which decreases the $O(\log\log n)$ search cost to $O(\log_{\bar{w}} n) = O(1)$. The offline planar point location queries can be answered in $O(\log_{\bar{w}} n) = O(1)$ time per query, by a straightforward plane sweep implemented using an atomic heap. The conflict list of each cell has size $O(\log n) = O(\log(\bar{w}^{O(1)})) = o(\bar{w})$. Thus by table lookup, the subproblem on each conflict list can be solved in $O(1)$ time. □

5 Offline 4D Dominance Reporting

A preliminary $O(n \log n \log\log n + k)$ worst-case time and linear-space algorithm for the rectangle enclosure problem is implied by Theorem 3. We obtain a faster deterministic algorithm for rectangle enclosure.

Theorem 4. *Offline 4D dominance reporting problem on n input points, n query points and k reported points can be solved deterministically in $O(n \log n + k)$ worst-case time and $O(n)$ space.*

Algorithm. We follow the approach of Chan, Larsen and Pătraşcu [6, Section 4.3]. We build a complete binary range tree $\mathcal{T}$ over the fourth coordinate of the input points. For each query point, we consider the path from the root of $\mathcal{T}$ to the leaf that contains its successor input point; we associate the query point with the left siblings (if they exist) of the nodes along this path. It then suffices to solve in every node of $\mathcal{T}$ an offline 3D dominance reporting problem between the 3D projections of its input point set and its associated query point set.

To this end, we first *equip* each node at level i of $\mathcal{T}$ with an optimal K_i-shallow cutting for 3D dominance ranges of its input points for some K_i between $\log^2 n$ and $\log^3 n$. At each node every associated query point is *assigned* to a cell of the equipped cutting whose vertex dominates it. All assigned queries are resolved by solving independently for each cell in $\mathcal{T}$ an offline 3D dominance reporting *subproblem* on the cell's conflict list. This leaves at most $O(\frac{k}{\log^2 n})$ queries *unresolved*, since each unresolved query reports $\Omega(K_i) \geq \Omega(\log^2 n)$ points. We can also decrease the number of input points to $O(\frac{k}{\log^2 n})$ in the same way by repeating the above with the roles of the input and query points reversed. Finally we solve a single offline 4D dominance reporting problem on all the remaining input points and unresolved queries.

Complexity. There are $O(\frac{n}{K_i})$ subproblems of size $O(K_i)$ at each level i of $\mathcal{T}$. Hence the total cost $T_{4D}(n, k)$ of the algorithm on n input and query points and k reported points is at most 2 times

$$T_E(n) + T_A(n) + \sum_{i=1}^{\log n} \sum_{j=1}^{O(\frac{n}{K_i})} T_{3D}(O(K_i), k_{ij}) + T_{4D}(O\left(\frac{k}{\log^2 n}\right), k)$$

where (i) $T_E(n)$ represents the cost of equipping the nodes with shallow cuttings, (ii) $T_A(n)$ represents the cost of assigning the queries to cells of the cuttings, (iii) $T_{3D}(O(K_i), k_{ij})$ represents the cost of solving a subproblem on a conflict list of size $O(K_i)$ with output size k_{ij}, and (iv) $T_{4D}(O\left(\frac{k}{\log^2 n}\right), k)$ represents the cost of handling the remaining input points and unresolved queries.

For (i) and (ii), we will show that $T_E(n), T_A(n) = O(n \log n)$. For (iii), we have $T_{3D}(O(K_i), k_{ij}) = O(K_i + k_{ij})$ by applying the algorithm of Corollary 2 with $\bar{w} = \log n$, since $K_i \leq \log^3 n \leq \bar{w}^{O(1)}$. Summing the cost over all i and j yields $O(\sum_{i=1}^{\log n} \frac{n}{K_i} K_i + \sum_{i,j} k_{ij}) = O(n \log n + k)$. For (iv), we can use the more expensive algorithm of [15] with $O(n \log^2 n + k)$ running time, which gives $T_{4D}(O\left(\frac{k}{\log^2 n}\right), k) = O(k)$. The overall running time is thus $O(n \log n + k)$.

Equipping Nodes with Optimal Shallow Cuttings. To show that $T_E(n) = O(n \log n)$, we first construct optimal $(\log^3 n)$-shallow cuttings for the nodes at every level of $\mathcal{T}$ that is a multiple of $\delta \log\log n$, for a constant $\delta > 0$, using the algorithm of Theorem 1. In total this takes $O(n \log\log n \cdot \frac{\log n}{\delta \log\log n}) = O(n \log n)$ time.

For each j that is not a multiple of $\delta \log\log n$, we equip the nodes at level i of $\mathcal{T}$ with an optimal K_i-shallow cutting on its input points in $\mathcal{T}$ by merging the optimal K_{i-1}-shallow cuttings of its two children nodes with the algorithm of Corollary 1. Here, $K_i = \frac{K_{i-1}}{c_{\max}} \implies \log^3 n \geq K_i \geq \frac{\log^3 n}{(c_{\max})^{\delta \log\log n}} \geq \log^2 n$ as desired, if we make δ sufficiently small. The entire merging process takes $O(n)$ time per level of $\mathcal{T}$ (since $K_i = \Omega(\log\log n)$) and thus $O(n \log n)$ total time.

Assigning Queries to Cells. To show that $T_A(n) = O(n \log n)$ we formulate a general problem of independent interest.

Problem 1. [Offline 2D Tree Point Location] Given a binary tree where every node contains a planar orthogonal subdivision with rectangular cells, and given a set of query points each of which is associated with a root-to-leaf path in the tree, locate the cell containing q in the subdivisions at all the nodes along the path associated with q, for every query point q.

Lemma 3. *Offline 2D tree point location on n query points and a tree of subdivisions with total size N and tree height $O(\log N)$ can be solved deterministically in $O(N + n\log N)$ worst-case time and $O(n)$ space, assuming pre-sorted coordinates.*

Since the output to the problem consists of $O(\log N)$ cells per query, the above result is optimal. Directly answering each of the $O(n \log N)$ 2D orthogonal point location queries by known results (Lemma 2) would yield a slower $O(N + n\log N \log\log N)$ running time. On the other hand, if the subdivisions in the tree are one-dimensional, then the problem can be solved by the well known technique of *fractional cascading* [9], achieving the same bound as in Lemma 3. Thus, Lemma 3 may be viewed as a generalization of fractional cascading to 2D; no such generalizations were known before (although to be fair our lemma works only in the offline setting).

We will prove Lemma 3 in Section 6, but for now let's see how the tree point location is relevant to our original problem. At each node of $\mathcal{T}$, we form a planar orthogonal subdivision from the projection of the shallow cutting equipped at the node's left sibling. Then assigning queries to cells of the shallow cuttings of their associated nodes is precisely the above tree point location problem. Here, the total size of the planar subdivisions is $N := O\left(\sum_{i=1}^{\log n} \frac{n}{K_i}\right) \leq O(\log n \cdot \frac{n}{\log^2 n}) = o(n)$. The tree height is $\log n$, thus by Lemma 3 we get $T_A(n) = O(N+n\log N) = O(n\log n)$.

Remarks. The above algorithm description is actually conceptually cleaner than Chan, Larsen and Pătraşcu's [6], which worked with shallow cuttings of two different ranges of K, namely $K \approx \text{polylog}\, n$ and $K \approx 2^{\sqrt{w}}$. We only need the former, although the expression $2^{\sqrt{w}}$ will appear later in the proof of Lemma 3.

As an immediate corollary to Theorem 4, we can then solve the 2D rectangle enclosure problem in $O(n\log n + k)$ time. Also, we can solve the offline d-dimensional dominance reporting problem in $O(n\log^{d-3} n + k)$ time by a straightforward divide-and-conquer, using the new algorithm for $d = 4$ as the base case.

6 Offline 2D Tree Point Location

To complete the presentation we now prove Lemma 3 and solve the offline 2D tree point location problem. We adapt key ideas from Chan, Larsen and Pătraşcu [6], but in addition incorporate planar separators to get a deterministic algorithm.

Preliminaries. An *r-separator* of a planar graph with n vertices is a subset of $O(\sqrt{rn})$ vertices whose removal yields components of $O(n/r)$ size each. Given d-dimensional input points and query hyper-rectangles (*boxes*), the *offline d-dimensional orthogonal range reporting problem* asks to report the input points that are contained in every query box. Given an orthogonal subdivision of the plane into disjoint rectangles and query points, the *offline 2D orthogonal point location problem* asks report the rectangle that every query point lies in.

Lemma 4. *[11, Theorem 2.2] An r-separator of a planar graph of size n can be computed deterministically in $O(n)$ worst-case time and space.*

Lemma 5. *[6, Lemma 4.2] Offline d-dimensional orthogonal range reporting on n input points, m query boxes and k reported points can be solved deterministically in $O(n \log_b^{d-1} n + b^{d-1} m \log_b^{d-1} n + k)$ worst-case time and $O(n)$ space for a given parameter $b \geq 2$, assuming pre-sorted coordinates.*

Lemma 6. *[6, Lemma 4.1] Offline planar point location for n query points and a orthogonal subdivision of size n can be solved deterministically in $O(n)$ worst-case time and space when $n \leq 2^{O(\sqrt{w})}$, assuming pre-sorted coordinates.*

Lemma 5 follows from a b-ary variant of the standard range tree, while Lemma 6 requires a bit-packing technique of Chan and Pătraşcu [8].

Proof of Lemma 3. First we compute a *sparser* subdivision at every node v of $\mathcal{T}$. Namely, if the subdivision at v has size n_v, we compute a $\left(\frac{n_v}{A}\right)$*-separator* of the subdivision for a parameter $A := 2^{\sqrt{w}}$ with $w = \log N$. This gives $O(\sqrt{n_v \cdot \frac{n_v}{A}}) = O(n_v/\sqrt{A})$ separating rectangles; each remaining hole can be further decomposed into rectangles to yield a subdivision of size $O(n_v/\sqrt{A})$. By Lemma 4 the computation of the separators takes $O(N)$ time total. (This idea of using separators for point location has been used before [5,7].)

Now we locate each query point q in the sparser subdivisions at the nodes along q's path. The key idea from Chan, Larsen and Pătraşcu [6] is to view all these 2D point location queries collectively as a single offline orthogonal range reporting problem in 3D. Namely, we lift each rectangle in the sparser subdivision at node v to a box in 3D, where the range of the box in the third coordinate corresponds to the range of v in the binary tree. This 3D problem involves n points, $O(\frac{N}{\sqrt{A}})$ boxes and output size $k = O(n \log N)$. By Lemma 5 the problem can be solved in $O(n \log_b^2 N + b^2 (\frac{N}{\sqrt{A}}) \log_b^2 N + n \log N)$ time. By choosing the parameter $b := A^{1/2-\delta} = 2^{\Theta(\sqrt{\log N})}$, we have $\log_b^2 N = O(\log N)$ and the time bound becomes $O(n \log N)$.

Knowing the cell of the sparser subdivision containing a query point q, we still need to locate the cell of the original subdivision containing q. This reduces to point location in a component, but since each component has size $O(A) = O(2^{\sqrt{w}})$ we can apply the offline point location algorithm of Lemma 6, to finish in time linear to the size of the subdivisions $O(N)$ and the number of queries $O(n \log N)$. □

Acknowledgements. We thank Christos Makris from the University of Patras, Greece for preliminary work on this problem.

References

1. Afshani, P.: On dominance reporting in 3D. In: Halperin, D., Mehlhorn, K. (eds.) ESA 2008. LNCS, vol. 5193, pp. 41–51. Springer, Heidelberg (2008)
2. Afshani, P., Tsakalidis, K.: Optimal deterministic shallow cuttings for 3D dominance ranges. In: Proc. of the 25th An. SODA, pp. 1389–1398. ACM-SIAM (2014)
3. Bentley, J.L., Wood, D.: An optimal worst case algorithm for reporting intersections of rectangles. IEEE Trans. Computers 29(7), 571–577 (1980)
4. Chan, T.M.: All-pairs shortest paths with real weights in $O(n^3/\log\ n)$ time. Algorithmica 50(2), 236–243 (2008)
5. Chan, T.M.: Persistent predecessor search and orthogonal point location on the word RAM. ACM Transactions on Algorithms 9(3), 22 (2013)
6. Chan, T.M., Larsen, K.G., Pătraşcu, M.: Orthogonal range searching on the RAM, revisited. In: Proc. of the 27th SoCG, pp. 1–10. ACM (2011)
7. Chan, T.M., Pătraşcu, M.: Transdichotomous results in computational geometry, I: Point location in sublogarithmic time. SIAM J. Comp. 39(2), 703–729 (2009)
8. Chan, T.M., Pătraşcu, M.: Counting inversions, offline orthogonal range counting, and related problems. In: Proc. of the 21st An. SODA, pp. 161–173. ACM-SIAM (2010)
9. Chazelle, B., Guibas, L.J.: Fractional cascading: I. A data structuring technique. Algorithmica 1(2), 133–162 (1986)
10. Fredman, M.L., Willard, D.E.: Trans-dichotomous algorithms for minimum spanning trees and shortest paths. J. Comp. Syst. Sci. 48(3), 533–551 (1994)
11. Goodrich, M.T.: Planar separators and parallel polygon triangulation. J. Comp. Syst. Sci. 51(3), 374–389 (1995)
12. Gupta, P., Janardan, R., Smid, M., DasGupta, B.: The rectangle enclosure and point-dominance problems revisited. I. J. C. Geom. & Appl. 7(5), 437–455 (1997)
13. Han, Y.: Deterministic sorting in $O(n\text{loglog}n)$ time and linear space. J. Algorithms 50(1), 96–105 (2004)
14. Lagogiannis, G., Makris, C., Tsakalidis, A.K.: A new algorithm for rectangle enclosure reporting. Inf. Process. Lett. 72(5-6), 177–182 (1999)
15. Lee, D.T., Preparata, F.P.: An improved algorithm for the rectangle enclosure problem. J. Algorithms 3(3), 218–224 (1982)
16. Lipton, R.J., Tarjan, R.E.: Applications of a planar separator theorem. SIAM J. Comp. 9(3), 615–627 (1980)
17. Makris, C., Tsakalidis, K.: An improved algorithm for static 3D dominance reporting in the pointer machine. In: Chao, K.-M., Hsu, T.-s., Lee, D.-T. (eds.) ISAAC 2012. LNCS, vol. 7676, pp. 568–577. Springer, Heidelberg (2012)
18. Matoušek, J.: Reporting points in halfspaces. Comp. Geom. 2, 169–186 (1992)
19. Preparata, F.P., Shamos, M.I.: Computational Geometry - An Introduction. Springer (1985)
20. Ramos, E.A.: On range reporting, ray shooting and k-level construction. In: Proc. of the 15th SoCG, pp. 390–399. ACM (1999)
21. Vaishnavi, V., Wood, D.: Data structures for the rectangle containment and enclosure problems. Comp. Graphics and Image Processing 13(4), 372–384 (1980)
22. van Emde Boas, P., Kaas, R., Zijlstra, E.: Design and implementation of an efficient priority queue. Mathematical Systems Theory 10(1), 99–127 (1976)

The Tropical Shadow-Vertex Algorithm Solves Mean Payoff Games in Polynomial Time on Average*

Xavier Allamigeon, Pascal Benchimol, and Stéphane Gaubert

INRIA and CMAP, École Polytechnique, CNRS, France
{firstname.lastname}@inria.fr

Abstract. We introduce an algorithm which solves mean payoff games in polynomial time on average, assuming the distribution of the games satisfies a flip invariance property on the set of actions associated with every state. The algorithm is a tropical analogue of the shadow-vertex simplex algorithm, which solves mean payoff games via linear feasibility problems over the tropical semiring $(\mathbb{R} \cup \{-\infty\}, \max, +)$. The key ingredient in our approach is that the shadow-vertex pivoting rule can be transferred to tropical polyhedra, and that its computation reduces to optimal assignment problems through Plücker relations.

1 Introduction

A *mean payoff game* involves two opponents, "Max" and "Min", who alternatively move a pawn over the nodes of a weighted bipartite digraph. The latter consists of two classes of nodes, represented by squares and circles, and respectively indexed by $i \in [m]$ and $j \in [n]$ (we use the notation $[k] := \{1, \dots, k\}$). The weight of the arc (i, j) (resp. (j, i)) is a real number denoted by A_{ij} (resp. B_{ij}). We set $A_{ij} := -\infty$ (resp. $B_{ij} := -\infty$) when there is no such arc. An example of game is given in Figure 1.

When the pawn is placed over a square node i, Player Max selects an outgoing arc (i, j), and then moves the pawn to circle node j and receives the payment A_{ij} from Player Min. Conversely, when the pawn is located on a circle node j, Player Min chooses an arc (j, i'), moves the pawn to square node i', and Player Max pays her the amount $B_{i'j}$. We assume that A (resp. B) does not have any identically $-\infty$ row (resp. column), so that both players have at least one possible action at every node. The game starts from a circle node $j_0 = j$, and then the two players make infinitely many moves, visiting a sequence $j_0, i_1, j_1, i_2, \dots$ of nodes. The objective of Player Max is to maximize his mean payoff, defined as the liminf of the following ratio when $p \to +\infty$:

$$(-B_{i_1 j_0} + A_{i_1 j_1} - B_{i_2 j_1} + A_{i_2 j_2} + \cdots - B_{i_p j_{p-1}} + A_{i_p j_p})/p \; . \tag{1}$$

* X. Allamigeon and S. Gaubert are partially supported by the PGMO program of EDF and Fondation Mathématique Jacques Hadamard. P. Benchimol is supported by a PhD fellowship of DGA and École Polytechnique.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 89–100, 2014.

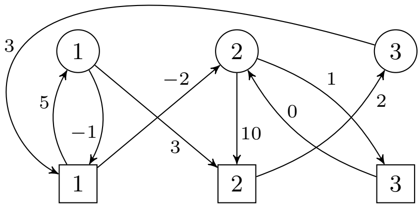


Fig. 1. An example of mean payoff game. Circle node 1 is a winning initial state for Player Max, while circle node 2 is losing.

Symmetrically, Player Min aims at minimizing her mean loss, defined as the limsup of (1) when $p \to +\infty$. Mean payoff games can be defined more generally over arbitrary (not necessarily bipartite) digraphs. This situation can be reduced to the present one [1].

Mean payoff games were first studied by Ehrenfeucht and Mycielski in [2], where they proved that these games have a value and positional optimal strategies. In more detail, for every initial state $j \in [n]$, there exists a real χ_j and positional strategies $\sigma : [m] \to [n]$ and $\tau : [n] \to [m]$, such that: (i) Player Max is certain to win a mean payoff greater than or equal to χ_j with the strategy σ (*i.e.* by choosing the arc $(i, \sigma(i))$ every time the pawn is on a square node $i \in [m]$), (ii) Player Min is sure that her mean loss is less than or equal to χ_j with the strategy τ.

The decision problem associated with mean payoff games consists in determining whether the initial state j is *winning* for Player Max, *i.e.* $\chi_j \geq 0$. The question of the existence of a polynomial time algorithm solving this problem was first raised by Gurvich, Karzanov and Khachiyan in [3]. This problem was shown to be in NP ∩ co-NP by Zwick and Paterson in [1]. While mean payoff games (and the related class of parity games) received an important attention over the past years [3,1,4,5,6,7,8,9,10], all the algorithms developed so far are superpolynomial, and the question raised by Gurvich *et al.* is still open.

The present work exploits the equivalence between mean payoff games and linear feasibility problems in tropical algebra. Indeed, it was shown in [11] that the initial state n is winning for Player Max in the game with payments matrices A, B if, and only if, there exists a solution $x \in (\mathbb{R} \cup \{-\infty\})^{n-1}$ to the following system of inequalities:

$$\forall i \in [m], \qquad \max(A_{i1} + x_1, \dots, A_{i(n-1)} + x_{n-1}, A_{in}) \\ \geq \max(B_{i1} + x_1, \dots, B_{i(n-1)} + x_{n-1}, B_{in}) \ . \quad (2)$$

The constraints of the form (2) correspond to affine inequalities over the tropical (max-plus) semiring $\mathbb{T} := \mathbb{R} \cup \{-\infty\}$ endowed with the operations $x \oplus y := \max(x, y)$ as addition, and $x \odot y := x + y$ as multiplication. The conjunction of finitely many such inequalities defines a *tropical polyhedron.* Solving a mean payoff game consequently amounts to determine whether a tropical polyhedron is

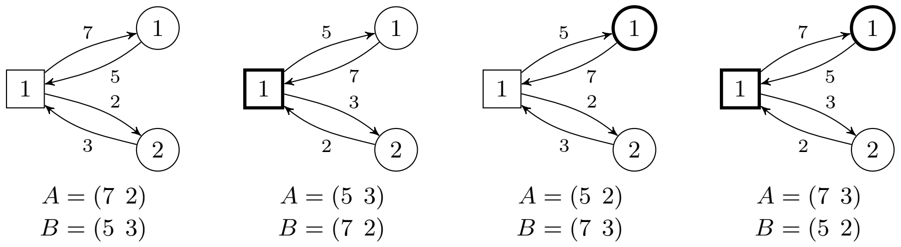


Fig. 2. A distribution of game satisfying the flip invariance property (with $m = 1$ and $n = 2$), together with the payment matrices. The four configurations are supposed to be equiprobable. The nodes on which the flip operations have been performed are depicted in bold.

empty, which can be thought of as the tropical analogue of the feasibility problem in linear programming. This is among the motivations leading to the development of a tropical simplex method in [12]. Then, complexity results known for the classical simplex algorithm can be potentially transferred to the tropical setting. However, the main obstacle is to "tropicalize" the pivoting rule involved, *i.e.* to define a tropical pivoting rule which is both compatible with the classical one, and computable, if possible, in a reasonable time complexity. So far [13], the only pivoting rules which have been tropicalized are *combinatorial*, *i.e.* they are defined in terms of the neighborhood of the current basic point in the vertex/edge graph of the polyhedron.

Contributions. We prove that the shadow-vertex simplex algorithm can be tropicalized. Following the average-case analysis of the shadow-vertex algorithm due to Adler, Karp and Shamir [14], we deduce that the tropical algorithm solves mean payoff games in polynomial time on average (Section 4). The complexity bound holds when the distribution of the games satisfies a *flip invariance* property. The latter requires that the distribution of the games is left invariant by every transformation consisting, for an arbitrary node of the game, in flipping the orientation of all the arcs incident to this node. Equivalently, the probability distribution on the set of payment matrices A, B is invariant by every transformation consisting in swapping the ith row of A with the ith row of B, or the jth column of A with the jth column of B. Figure 2 provides the illustration of a discrete distribution of games satisfying the property.

The key difficulty in our approach is to show that the computation of the tropical shadow-vertex pivoting rule can be done in polynomial time (Section 3). To this end, we exploit the fact that the shadow-vertex rule is semi-algebraic, *i.e.* it is defined in terms of the signs of finitely many polynomials. Under some genericity conditions, we deduce that the tropical rule reduces to classical linear programs over some Newton polytopes, which are actually (Minkowski sums of) bipartite perfect matching polytopes.

Related Work. We are not aware of other works with such average-case complexity results on mean payoff games. In [15], Roth *et al.* made a probabilistic analysis of $n \times n$ bi-matrix games with weights chosen independently uniformly in $[0,1]$. Under this assumption, they showed that with high probability (greater than $1 - f(n)$, with $f(n) = o(1/n^c)$ for all constant c), such games admit a pure stationary strategy equilibrium parametrized by only 4 actions. The latter can be consequently found in polynomial time. While this result indicates that complex instances of games are rare, it does not seem to us that it can be used to deduce an average-case complexity bound.

2 Preliminaries

2.1 Tropical Arithmetic and Generalized Puiseux Series

As previously discussed, $(\mathbb{T}, \oplus, \odot)$ forms a semiring, and the elements $\mathbb{0} := -\infty$ and $\mathbb{1} := 0$ correspond to the zero and unit respectively. The tropical operations can be extended to matrices with entries in $\mathbb{T}$ in a usual way, by defining $A \oplus B := (A_{ij} \oplus B_{ij})_{ij}$ and $A \odot B := (\bigoplus_k A_{ik} \odot B_{kj})_{ij}$. We also introduce the exponentiation $x^{\odot k}$ for any $x \in \mathbb{T}$ and $k \in \mathbb{N}$, which is defined as the product $x \odot x \odot \ldots \odot x$ of k occurrences of x (if $k = 0$, it is set to $\mathbb{1}$).

Even if the addition $\oplus$ does not have an inverse, it is convenient to consider tropical numbers with a negative sign. The sign is encoded in a formal way, by introducing two copies $\mathbb{T}_+$ and $\mathbb{T}_-$ of $\mathbb{T} \setminus \{\mathbb{0}\}$, respectively consisting of the positive and negative elements. The set $\mathbb{T}_\pm$ of *tropical signed numbers* is defined as $\mathbb{T}_+ \cup \mathbb{T}_- \cup \{\mathbb{0}\}$. The elements of $\mathbb{T}_+$ are simply denoted by elements $a \in \mathbb{T} \setminus \{\mathbb{0}\}$, while the elements of $\mathbb{T}_-$ are denoted by $\ominus a$. The *modulus* of $x \in \mathbb{T}_\pm$ is defined as $|x| := a$ if $x = a$ or $x = \ominus a$, and $|\mathbb{0}| := \mathbb{0}$. Similarly, we set $\operatorname{sign}(x) = +1$ if $x \in \mathbb{T}_+$, $\operatorname{sign}(x) = -1$ if $x \in \mathbb{T}_-$, and $\operatorname{sign}(\mathbb{0}) = 0$.

While the tropical addition of signed elements may not be well defined, we can extend the multiplication over $x, y \in \mathbb{T}_\pm$, by defining $x \odot y$ as the unique element of $\mathbb{T}_\pm$ with modulus $|x| \odot |y|$ and sign $(\operatorname{sign}(x) \times \operatorname{sign}(y))$. For instance, $(\ominus 3) \odot 4 = \ominus 7$, and $(\ominus 2) \odot (\ominus 4) = 6$. The exponentiation $x^{\odot k}$ is generalized to the case $x \in \mathbb{T}_\pm$ as well. For any $x \in \mathbb{T}_\pm$, we use the notation $\ominus x$ as a shorthand for the operation $(\ominus \mathbb{1}) \odot x$. The *positive* and *negative parts* x^+ and x^- of an element $x \in \mathbb{T}_\pm$ are defined by $(x^+, x^-) := (x, \mathbb{0})$ if $x \in \mathbb{T}_+$, $(\mathbb{0}, \ominus x)$ if $x \in \mathbb{T}_-$, and $(\mathbb{0}, \mathbb{0})$ if $x = \mathbb{0}$. We extend this notation to vectors and matrices entrywise.

A matrix $M \in \mathbb{T}_\pm^{n \times n}$ is said to be *generic* if the following maximum

$$\max\{|M_{1\sigma(1)}| \odot \ldots \odot |M_{n\sigma(n)}| \mid \sigma \in S_n\}$$

is not equal to $\mathbb{0}$, and is attained by only one permutation σ in the symmetric group S_n. A matrix $A \in \mathbb{T}_\pm^{m \times n}$ is *strongly non-degenerate* if all of its submatrices are generic. In particular, the coefficients of A are not null (tropically).

Generalized Puiseux Series. Tropical arithmetic can be intuitively illustrated by the arithmetic over asymptotic orders of magnitude. For instance, if we denote by $\Theta(t^a)$ the equivalence class of real functions of t which belong to some interval $[Kt^a, K't^a]$ when $t \to +\infty$ $(0 < K \leq K')$, we have $\Theta(t^a) + \Theta(t^b) = \Theta(t^{\max(a,b)})$, and $\Theta(t^a) \times \Theta(t^b) = \Theta(t^{a+b})$. We use generalized Puiseux series as a way to manipulate such orders of magnitude in a formal way.

A *(real) generalized Puiseux series* (or *Puiseux series* for short) is a formal series $\boldsymbol{x}$ in the indeterminate t of the form $x_{\alpha_1} t^{\alpha_1} + x_{\alpha_2} t^{\alpha_2} + \dots$, where $x_{\alpha_i} \in \mathbb{R} \setminus \{0\}$, and the sequence of the α_i is decreasing, and either finite or unbounded. The set of generalized Puiseux series forms a field that we denote $\mathbb{K}$. Given a Puiseux series $\boldsymbol{x}$ as above, the largest exponent α_1 is called the *valuation* of $\boldsymbol{x}$, and is denoted by $\operatorname{val}(\boldsymbol{x})$. By convention, the valuation of the null series $\boldsymbol{x} = 0$ is defined as $\mathbb{0} = -\infty$. Thus the valuation $\operatorname{val}(\cdot)$ maps $\mathbb{K}$ to $\mathbb{T}$.

A Puiseux series $\boldsymbol{x}$ is *positive*, which is denoted by $\boldsymbol{x} > 0$, if the coefficient $x_{\operatorname{val}(x)}$ of the term with largest exponent in $\boldsymbol{x}$ is positive. We denote by $\leq$ the total order over $\mathbb{K}$ defined by $\boldsymbol{x} \leq \boldsymbol{y}$ if $\boldsymbol{x} = \boldsymbol{y}$ or $\boldsymbol{y} - \boldsymbol{x} > 0$. Then, we define the *signed valuation* $\operatorname{sval}(\boldsymbol{x})$ of $\boldsymbol{x}$ as the element of $\mathbb{T}_\pm$ given by $\operatorname{val}(\boldsymbol{x})$ if $\boldsymbol{x} > 0$, $\ominus(\operatorname{val}(\boldsymbol{x}))$ if $\boldsymbol{x} < 0$, and $\mathbb{0}$ if $\boldsymbol{x} = 0$. Given $x \in \mathbb{T}_\pm$, we also denote by $\operatorname{sval}^{-1}(x)$ the set of Puiseux series $\boldsymbol{x}$ such that $\operatorname{sval}(\boldsymbol{x}) = x$. Valuation, signed valuation and its inverse are extended to vectors and matrices coordinate-wise.

As discussed above, the arithmetic operations over $\mathbb{K}$ and $\mathbb{T}$ are related. For instance, for all $\boldsymbol{x}, \boldsymbol{y} \geq 0$, we have $\operatorname{val}(\boldsymbol{x} + \boldsymbol{y}) = \max(\operatorname{val}(\boldsymbol{x}), \operatorname{val}(\boldsymbol{y}))$. Similarly, if $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{K}$, then $\operatorname{sval}(\boldsymbol{x}\boldsymbol{y}) = \operatorname{sval}(\boldsymbol{x}) \odot \operatorname{sval}(\boldsymbol{y})$. More generally, the valuation will be used to transfer "classical" results to the tropical setting. In particular, convex polyhedra and linear programs over $\mathbb{K}$ series essentially behave as over $\mathbb{R}$ ($\mathbb{K}$ is a real-closed field, so Tarski's transfer principle applies). The simplex algorithm can be defined over $\mathbb{K}$ as usual, and the valuation map will allow us to relate it with the tropical simplex algorithm.

General Notations. Given a matrix A of dimension $m \times n$, and two subsets $I \subset [m]$ and $J \subset [n]$, we denote by $A_{I \times J}$ the submatrix formed by the rows and the columns of A respectively indexed by $i \in I$ and $j \in J$. If $J = [n]$, we simply denote this submatrix by A_I. The transpose matrix of A is denoted by $A^\intercal$, and the cardinality of a set I by $|I|$.

2.2 The Tropical Simplex Method

The tropical simplex method, introduced in [12], allows to solve tropical analogues of linear programming problems:

$$\begin{array}{rl} \text{minimize} & c^\intercal \odot x \qquad (x \in \mathbb{T}^n) \\ \text{subject to} & x \geq \mathbb{0},\ A^+ \odot x \oplus b^+ \geq A^- \odot x \oplus b^- \end{array} \qquad \mathrm{LP}(A, b, c)$$

where $A \in \mathbb{T}_\pm^{m \times n}$, $b \in \mathbb{T}_\pm^m$, and $c \in \mathbb{T}_\pm^n$. We denote by $\mathcal{P}$ the tropical polyhedron defined by the constraints of $\mathrm{LP}(A, b, c)$. Note that the inequalities $x \geq \mathbb{0}$ are trivially satisfied by any $x \in \mathbb{T}^n$, hence they are superfluous. However, as we shall see, they are involved in the definition of tropical basic points, which is why we need to keep them.

Similarly to the classical simplex method, the principle of the tropical simplex method is to pivot over the (feasible) tropical basic points, while decreasing the objective function $x \mapsto c^\intercal \odot x$. It handles tropical linear programs which satisfy a certain non-degeneracy assumption. Here, we will make the following sufficient assumption:

Assumption 1. The matrices $\begin{pmatrix} A & b \end{pmatrix}$ and $\begin{pmatrix} A \\ c \end{pmatrix}$ are strongly non-degenerate.

In this setting, a *basis* is a couple (I, J) where $I \subset [m]$, $J \subset [n]$ satisfy $|I| + |J| = n$. It can be shown that the system $A^+_I \odot x \oplus b^+_I = A^-_I \odot x \oplus b^-_I$, $x_J = \mathbb{0}$ admits a unique solution, which is referred to as the *basic point* associated with (I, J). When it belongs to $\mathcal{P}$, it is called a *feasible basic point*, and (I, J) is a *feasible basis*. We often manipulate (I, J) through the disjoint union $I \uplus J$ of I and J.

The execution of the tropical simplex method on $\mathrm{LP}(A, b, c)$ is related with the execution of the classical simplex method on a lifting linear program over Puiseux series. More precisely, a *lift* of $\mathrm{LP}(A, b, c)$ is a linear program over Puiseux series of the form:

$$\begin{array}{ll} \text{minimize} & \boldsymbol{c}^\intercal \boldsymbol{x} \qquad (\boldsymbol{x} \in \mathbb{K}^n) \\ \text{subject to} & \boldsymbol{x} \geq 0,\ \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} \geq 0 \end{array} \tag{$\mathbf{LP}(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c})$}$$

where $\boldsymbol{A} \in \mathrm{sval}^{-1}(A)$, $\boldsymbol{b} \in \mathrm{sval}^{-1}(b)$, and $\boldsymbol{c} \in \mathrm{sval}^{-1}(c)$. Let us denote by $\boldsymbol{\mathcal{P}}$ the convex polyhedron defined by the inequalities of $\mathbf{LP}(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c})$. Then, $\mathcal{P}$ and $\boldsymbol{\mathcal{P}}$ have precisely the same (feasible) bases, and the map $\boldsymbol{x} \mapsto \mathrm{val}(\boldsymbol{x})$ induces a one-to-one correspondence between their basic points [12, Prop. 17]. Besides, if $\bar{\boldsymbol{x}} \in \boldsymbol{\mathcal{P}}$ minimizes the function $\boldsymbol{x} \mapsto \boldsymbol{c}^\intercal \boldsymbol{x}$, then $\mathrm{val}(\bar{\boldsymbol{x}}) \in \mathcal{P}$ minimizes $x \mapsto c^\intercal \odot x$.

Moreover, both simplex methods also iterate over basic points in the same way. Starting from a basic point of basis (I, J), they pivot to an adjacent basic point associated with a basis (I', J') such that $I' \uplus J' = (I \uplus J) \setminus \{k_{\mathrm{out}}\} \cup \{k_{\mathrm{in}}\}$, for some $k_{\mathrm{out}} \in I \uplus J$, $k_{\mathrm{in}} \notin I \uplus J$. The element k_{out} is called the *leaving variable*, and is provided by the pivoting rule. The integer k_{in} is uniquely determined when the problem is non-degenerate. As a consequence, the tropical simplex method traces the image by $\mathrm{val}(\cdot)$ of the path followed by the classical simplex method, provided that they use compatible pivoting rules, *i.e.* at any feasible basis, both rules select the same leaving variable. Recall that, given a tropical basic point with basis (I, J) and a leaving variable k_{out}, the operation of pivoting to the next tropical basic point can be done in time $O(n(m + n))$ [12, Theorem 33].

3 Tropicalizing the Shadow-Vertex Simplex Algorithm

3.1 The Classical Shadow-Vertex Pivoting Rule

Given $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{K}^n$, the shadow-vertex rule aims at solving the following parametric family of linear programs for increasing values of $\boldsymbol{\lambda} \geq 0$:

$$\begin{array}{ll} \text{minimize} & (\boldsymbol{u}^\intercal - \boldsymbol{\lambda} \boldsymbol{v}^\intercal) \boldsymbol{x} \qquad (\boldsymbol{x} \in \mathbb{K}^n) \\ \text{subject to} & \boldsymbol{x} \geq 0,\ \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} \geq 0 \end{array} \tag{3}$$

The vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ are respectively called *objective* and *co-objective* vectors. The input of (3) is supposed to satisfy a genericity property. Here, we will assume that no minor of $\begin{pmatrix} \boldsymbol{A} & \boldsymbol{b} \end{pmatrix}$ and $\begin{pmatrix} \boldsymbol{A}^\intercal & \boldsymbol{u} & \boldsymbol{v} \end{pmatrix}$ is null.

When $\boldsymbol{\lambda}$ is continuously increased from 0, the basic points of $\boldsymbol{\mathcal{P}}$ minimizing the function $\boldsymbol{x} \mapsto (\boldsymbol{u}^\intercal - \boldsymbol{\lambda} \boldsymbol{v}^\intercal)\boldsymbol{x}$ form a sequence $\bar{\boldsymbol{x}}^0, \dots, \bar{\boldsymbol{x}}^p$. The shadow-vertex rule amounts to iterate over this sequence. It relies on the reduced cost vectors w.r.t. the objective and co-objective vectors. Given a basis (I, J), the *reduced cost vector* $\boldsymbol{y}^{(I,J)} \in \mathbb{K}^{I \uplus J}$ w.r.t. the objective vector $\boldsymbol{u}$ is defined as the unique solution $\boldsymbol{y}$ of the system $(\boldsymbol{A}'_{I \uplus J})^\intercal \boldsymbol{y} = \boldsymbol{u}$, where $\boldsymbol{A}' = \left(\begin{smallmatrix}\mathbf{Id} \\ \boldsymbol{A}\end{smallmatrix}\right)$, and $\mathbf{Id}$ is the identity matrix. The reduced cost vector $\boldsymbol{z}^{(I,J)}$ w.r.t. the co-objective vector $\boldsymbol{v}$ can be defined similarly, by replacing $\boldsymbol{u}$ by $\boldsymbol{v}$. Then, at basis (I, J), the shadow-vertex rule selects the leaving variable $k_{\text{out}} \in I \uplus J$ such that $\boldsymbol{y}^{(I,J)}_{k_{\text{out}}} > 0$, $\boldsymbol{z}^{(I,J)}_{k_{\text{out}}} > 0$, and:

$$\boldsymbol{y}^{(I,J)}_{k_{\text{out}}} / \boldsymbol{z}^{(I,J)}_{k_{\text{out}}} = \min\bigl\{ \boldsymbol{y}^{(I,J)}_l / \boldsymbol{z}^{(I,J)}_l \mid l \in I \uplus J,\ \boldsymbol{y}^{(I,J)}_l > 0 \text{ and } \boldsymbol{z}^{(I,J)}_l > 0 \bigr\} \ . \tag{4}$$

Note that k_{out} is unique under the non-degeneracy assumptions. We refer to [16, Chapter 1.3] for a proof of (4). In the following, we will denote by $\rho_{\text{sv}}(\boldsymbol{A}, \boldsymbol{u}, \boldsymbol{v})$ the function which, given a basis (I, J), returns the leaving variable provided by the shadow-vertex rule with objective and co-objective vectors $\boldsymbol{u}$ and $\boldsymbol{v}$.

3.2 Semi-algebraic Pivoting Rules and Their Tropical Counterpart

In this section, we focus on the problem of finding a tropical pivoting rule $\rho^{\text{trop}}_{\text{sv}}$ compatible with the shadow-vertex rule ρ_{sv}. More formally, we aim at defining a function $\rho^{\text{trop}}_{\text{sv}}(A, u, v)$ parametrized by a tropical matrix $A \in \mathbb{T}^{m \times n}_{\pm}$, and objective and co-objective vectors $u, v \in \mathbb{T}^n_{\pm}$, such that:

$$\rho^{\text{trop}}_{\text{sv}}(A, u, v)(I, J) = \rho_{\text{sv}}(\boldsymbol{A}, \boldsymbol{u}, \boldsymbol{v})(I, J) \qquad \text{for any basis } (I, J) \ ,$$

for all $\boldsymbol{A} \in \operatorname{sval}^{-1}(A)$, $\boldsymbol{u} \in \operatorname{sval}^{-1}(u)$, and $\boldsymbol{v} \in \operatorname{sval}^{-1}(v)$. In this case, $\rho^{\text{trop}}_{\text{sv}}$ will be said to be *compatible with* ρ_{sv} *on the instance* (A, u, v).

Tropical Polynomials. The connection we use between the classical and tropical worlds relies on polynomials over generalized Puiseux series.

Let $P \in \mathbb{K}[X_1, \dots, X_p]$ be a multivariate polynomial, and suppose that it is of the form $\sum_{\alpha \in S} \boldsymbol{c}_\alpha X_1^{\alpha_1} \dots X_p^{\alpha_p}$, where $S \subset \mathbb{N}^p$, and $\boldsymbol{c}_\alpha \in \mathbb{K} \setminus \{0\}$ for all $\alpha \in S$. The set S is called the *support* of P. We associate a tropical polynomial $\operatorname{trop}(P) \in \mathbb{T}_{\pm}[X_1, \dots, X_p]$ defined as the following formal function:

$$\operatorname{trop}(P) := \bigoplus_{\alpha \in S} c_\alpha \odot X_1^{\odot \alpha_1} \odot \dots \odot X_p^{\odot \alpha_p} \ ,$$

with $c_\alpha := \operatorname{sval}(\boldsymbol{c}_\alpha)$ for all $\alpha \in S$. Given $x \in \mathbb{T}^p_{\pm}$, we say that the polynomial $\operatorname{trop}(P)$ *vanishes* on x if the following maximum

$$\max\bigl\{ |c_\alpha| \odot |x_1|^{\odot \alpha_1} \odot \dots \odot |x_p|^{\odot \alpha_p} \mid \alpha \in S \bigr\} \tag{5}$$

is reached at least twice, or is equal to $\mathbb{0}$. If P does not vanish on x, we define by $P(x)$ as $c_{\alpha^*} \odot x_1^{\odot \alpha_1^*} \odot \dots \odot x_p^{\odot \alpha_p^*}$, where α^* is the unique element of S reaching the maximum in (5). The following lemma relates the values of P and $\operatorname{trop}(P)$:

Lemma 2. *Let $x \in \mathbb{T}_{\pm}^{p}$, and suppose that* $\operatorname{trop}(P)$ *does not vanish on x. Then, for any $\boldsymbol{x} \in \operatorname{sval}^{-1}(x)$, we have* $\operatorname{sval}(P(\boldsymbol{x})) = \operatorname{trop}(P)(x)$. *In particular, the sign of $P(\boldsymbol{x})$ is equal to the sign of* $\operatorname{trop}(P)(x)$.

Following this, we can define determinants of tropical matrices. Let us define $\operatorname{tdet}_n(X) := \bigoplus_{\sigma \in S_n} \operatorname{tsign}(\sigma) \odot X_{1\sigma(1)} \odot \ldots \odot X_{n\sigma(n)}$, where $\operatorname{tsign}(\sigma) := \mathbb{1}$ if the permutation σ is even, $\ominus\mathbb{1}$ otherwise. The polynomial tdet_n is simply denoted tdet when there is no ambiguity. If $M \in \mathbb{T}_{\pm}^{n \times n}$, the *tropical determinant* of M is defined as $\operatorname{tdet}(M)$ when tdet does not vanish on M. Note that the latter condition is equivalent to the fact that M is generic. In this case, $\operatorname{tdet}(M)$ can be computed in time complexity $O(n^3)$, by solving an assignment problem over the bipartite graph equipped with the weights $|M_{ij}|$ (see *e.g.* [13]).

The Shadow-Vertex Rule as a Semi-Algebraic Rule. We claim that the shadow-vertex rule is a *semi-algebraic rule*, in the sense that the leaving variable returned by $\rho_{\mathrm{sv}}(\boldsymbol{A}, \boldsymbol{u}, \boldsymbol{v})(I, J)$ only depends on the current basis (I, J) and on the signs of finitely many polynomials taken on the matrix $\boldsymbol{M} := \begin{pmatrix} \boldsymbol{A} \\ \boldsymbol{u}^\intercal \\ \boldsymbol{v}^\intercal \end{pmatrix}$. To make the notations simpler, we fix a basis (I, J), and we respectively denote by $\boldsymbol{y}$ and $\boldsymbol{z}$ the reduced cost vectors $\boldsymbol{y}^{(I,J)}$ and $\boldsymbol{z}^{(I,J)}$. We also define $\overline{J} := [n] \setminus J$.

Let us denote by $P_{K \times L}$ the polynomial given by the $(K \times L)$-minor of the matrix $X = (X_{ij})$ of formal variables, for any $K \subset [m+2]$ and $L \subset [n]$ such that $|K| = |L|$. For instance, if $K = \{1, 2\}$ and $L = \{3, 4\}$, $P_{K \times L}$ is given by the determinant of the submatrix $\begin{pmatrix} X_{1,3} & X_{1,4} \\ X_{2,3} & X_{2,4} \end{pmatrix}$, *i.e.* $P_{K \times L} = X_{1,3}X_{2,4} - X_{2,3}X_{1,4}$. For all $l \in I \uplus J$, we define two polynomials Q_l and R_l as follows:

$$\begin{aligned} Q_i &:= P_{(I \setminus \{i\} \cup \{m+1\}) \times \overline{J}} & R_i &:= P_{(I \setminus \{i\} \cup \{m+2\}) \times \overline{J}} & &\text{when } i \in I \ , \\ Q_j &:= P_{(I \cup \{m+1\}) \times (\overline{J} \cup \{j\})} & R_j &:= P_{(I \cup \{m+2\}) \times (\overline{J} \cup \{j\})} & &\text{when } j \in J \ . \end{aligned}$$

With these notations, it can be shown that for all $l \in I \uplus J$,

$$\boldsymbol{y}_l = s_l \, Q_l(\boldsymbol{M}) / P_{I \times \overline{J}}(\boldsymbol{M}) \qquad \boldsymbol{z}_l = s_l \, R_l(\boldsymbol{M}) / P_{I \times \overline{J}}(\boldsymbol{M}) \ , \tag{6}$$

where s_l is a constant in $\{\pm 1\}$ which only depends on the position of l in the ordered set I or J. Thus, the properties $\boldsymbol{y}_l > 0$, $\boldsymbol{z}_l > 0$ can be tested by determining the signs of $Q_l(\boldsymbol{M})$, $R_l(\boldsymbol{M})$ and $P_{I \times \overline{J}}(\boldsymbol{M})$.

Moreover, thanks to (6), we have $\boldsymbol{y}_l / \boldsymbol{z}_l = Q_l(\boldsymbol{M}) / R_l(\boldsymbol{M})$. As a result, the comparison of two ratios $\boldsymbol{y}_k / \boldsymbol{z}_k$ and $\boldsymbol{y}_l / \boldsymbol{z}_l$ involved in the shadow-vertex rule can be made by evaluating the sign of a polynomial of the form $T_{kl} := Q_k R_l - Q_l R_k$ on the matrix $\boldsymbol{M}$. This completes the proof of our claim.

Tropical Shadow-Vertex Rule. Following the previous discussion, we can express $\rho_{\mathrm{sv}}(\boldsymbol{A}, \boldsymbol{u}, \boldsymbol{v})$ as a function defined in terms of the signs of some minors $\det(\boldsymbol{M}_{K \times L})$, and of the signs of the $T_{kl}(\boldsymbol{M})$ $(k \neq l)$. Given tropical entries (A, u, v), we simply define $\rho_{\mathrm{sv}}^{\mathrm{trop}}(A, u, v)$ as the same function, in which the minors of $\boldsymbol{M}$ have been substituted by the corresponding tropical minors of the matrix $M := \begin{pmatrix} A \\ u^\intercal \\ v^\intercal \end{pmatrix}$, and the $T_{kl}(\boldsymbol{M})$ have been replaced by $\operatorname{trop}(T_{kl})(M)$.

In more details, the function $\rho_{\text{sv}}^{\text{trop}}(A, u, v)(I, J)$ returns the unique element $k_{\text{out}} \in L$ such that

$$\operatorname{sign}(\operatorname{trop}(T_{k_{\text{out}} l}(M)) = -s_{k_{\text{out}}} s_l \qquad \text{for all } l \in L \setminus \{k_{\text{out}}\} \ , \tag{7}$$

where L is the set of the elements $l \in I \uplus J$ such that $\operatorname{sign}(\operatorname{trop}(Q_l)(M)) = \operatorname{sign}(\operatorname{trop}(R_l)(M)) = s_l \operatorname{sign}(\operatorname{tdet}(M_{I \times \overline{J}}))$. The latter condition is the tropical counterpart of the conditions $\boldsymbol{y}_l, \boldsymbol{z}_l > 0$ in the definition of ρ_{sv}. Equation (7) is the analog of $\boldsymbol{y}_{k_{\text{out}}}/\boldsymbol{z}_{k_{\text{out}}} < \boldsymbol{y}_l/\boldsymbol{z}_l$ for all $l \in L$, $l \neq k_{\text{out}}$.

The main result of this section is the following:

Theorem 3. *If the matrix* $\begin{pmatrix} A \\ u^\intercal \\ v^\intercal \end{pmatrix}$ *is strongly non-degenerate, then* $\rho_{\text{sv}}^{\text{trop}}$ *is compatible with* ρ_{sv} *on the instance* (A, u, v)*. Moreover, for all bases* (I, J)*, the leaving variable returned by* $\rho_{\text{sv}}^{\text{trop}}(A, u, v)(I, J)$ *can be computed in time* $O(n^4)$*.*

In the rest of the section, we sketch the main arguments which allow to prove Theorem 3.

By assumption, the matrix M is strongly non-degenerate, so that the sign of every tropical minor $\operatorname{tdet}(M_{K \times L})$ coincides with the sign of the corresponding minor of $\boldsymbol{M}$ by Lemma 2. As discussed earlier, each tropical minor can be computed in time $O(n^3)$, and in total, the set L can be determined in time $O(n^4)$. It now remains to examine the case of the polynomials $\operatorname{trop}(T_{kl})$, and to show in particular that they do not vanish on M. Without loss of generality, we restrict to the case $k, l \in I$.

First observe that the coefficients in T_{kl} are integers. Hence, as elements of the field $\mathbb{K}$, they are constant Puiseux series, with valuation $0 = \mathbb{1}$. Therefore, the tropical polynomial $\operatorname{trop}(T_{kl})$ is of the form $\bigoplus_{\alpha \in S} c_\alpha \odot X^{\odot \alpha}$, where S is the support of T_{kl}, and $c_\alpha \in \{\mathbb{1}, \ominus \mathbb{1}\}$ for all $\alpha \in S$ (we use the notation $X^{\odot \alpha}$ as a shorthand of $\bigodot_{ij} X_{ij}^{\odot \alpha_{ij}}$). In particular, as $M_{ij} \neq \mathbb{0}$ for all (i, j) (thanks to the strong non-degeneracy of M), we have $|c_\alpha| \odot |M|^{\odot \alpha} = \sum_{ij} \alpha_{ij} |M_{ij}|$.

As a consequence, it can be shown that $\operatorname{trop}(T_{kl})$ does not vanish on M if, and only if, the following classical linear programming problem admits a unique solution α^*:

$$\begin{array}{rl} \text{maximize} & \sum_{ij} |M_{ij}| \alpha_{ij} \\ \text{subject to} & \alpha \in \operatorname{New}(T_{kl}) \end{array} \tag{8}$$

Here, $\operatorname{New}(T_{kl}) \subset \mathbb{R}^{(m+2) \times n}$ is the *Newton polytope* of the polynomial T_{kl}, defined as the convex hull of its support S. In this case, the sign of $\operatorname{trop}(T_{kl})(M)$ is given as the sign of the term $c_{\alpha^*} \odot M^{\odot \alpha^*}$. We claim that (8) can be solved in time $O(n^3)$, and that the optimal solution α^* is unique.

Indeed, by Plücker quadratic relations (see for instance [17, Chapter 3]), we know that $T_{kl} = P_{I \times \overline{J}} P_{(I \setminus \{k,l\} \cup \{m+1, m+2\}) \times \overline{J}}$. This implies that the Newton polytope of T_{kl} consists in the Minkowski sum of the two polytopes $\varDelta_1 := \operatorname{New}(P_{I \times \overline{J}})$ and $\varDelta_2 := \operatorname{New}(P_{(I \setminus \{k,l\} \cup \{m+1, m+2\}) \times \overline{J}})$. As a result, Problem (8) can be decomposed into the following two linear programs:

$$\begin{array}{rl} \text{maximize} & \sum_{ij} |M_{ij}| \alpha_{ij} \\ \text{subject to} & \alpha \in \operatorname{New}(\varDelta_i) \end{array} \qquad \text{for } i \in \{1, 2\} \tag{9}$$

```
1: procedure TropPCBC($A, b$)
2:     $u := (\epsilon, \epsilon^{\odot 2}, \dots, \epsilon^{\odot n})^{\intercal}$
3:     $\bar{x} := (\mathbb{0}, \dots, \mathbb{0})^{\intercal}$
4:     for $k = 1$ to $m$ do
5:         Starting from $\bar{x}$, iterate over the tropical basic points and edges of $\mathcal{P}^{(k-1)}$ using the tropical rule $\rho_{\mathrm{sv}}^{\mathrm{trop}}(A_{[k-1]}, u, (A_k)^{\intercal})$, until finding a point $\bar{x}' \in \mathcal{P}^{(k-1)}$ such that $A_k^+ \odot \bar{x}' \oplus b_k^+ \geq A_k^- \odot \bar{x}' \oplus b_k^-$.
6:         if there is no such point $\bar{x}'$ then return "Empty"
7:         else $\bar{x} := \bar{x}'$
8:     done
9:     return "Non-empty"
10: end
```

Fig. 3. Tropicalization of the PCBC algorithm

The polytopes New(Δ_i) are bipartite perfect matching polytopes. Consequently, the two problems in (9) correspond to optimal assignment problems, and can be solved in $O(n^3)$. Besides, they both admit a unique solution thanks to the genericity condition on M. This easily proves our claim.

To summarize, the compatibility of $\rho_{\mathrm{sv}}^{\mathrm{trop}}$ and ρ_{sv} follows from Lemma 2. The output k_{out} of $\rho_{\mathrm{sv}}^{\mathrm{trop}}(A, u, v)(I, J)$ can be computed by determining the smallest element of the set L according to the abstract ordering relation $\prec$ defined by $k \prec l \iff \operatorname{sign}(\operatorname{trop}(T_{kl})(M)) = -s_k s_l$. Every comparison has time complexity $O(n^3)$, and so the result can be obtained in time $O(n^4)$.

4 Average-Case Complexity of Mean Payoff Games

As an application of the results of Section 3, we deduce an average-case complexity result on mean payoff games. The algorithm involved in this result is given in Figure 3. It is a straightforward transposition to the tropical setting of the *parametric constraint-by-constraint algorithm* due to Adler, Karp and Shamir [14].

Given $A \in \mathbb{T}_{\pm}^{m \times n}$, $b \in \mathbb{T}_{\pm}^{m}$, we denote by $\mathcal{P}^{(k)}$ the tropical polyhedron defined by the first $n + k$ inequalities of the system $x \geq \mathbb{0}$, $A^+ \odot x \oplus b^+ \geq A^- \odot x \oplus b^-$. The principle of the algorithm TropPCBC is to determine by induction on $k = 1, \dots, m$ whether $\mathcal{P}^{(k)}$ is empty. The inductive step (Line 5) uses the tropical shadow-vertex simplex algorithm in order to find a feasible basic point $\bar{x}'$ of $\mathcal{P}^{(k)}$ (if any), starting from a basic point $\bar{x}$ of $\mathcal{P}^{(k-1)}$. The objective vector is fixed to $u = (\epsilon, \epsilon^{\odot 2}, \dots, \epsilon^{\odot n})^{\intercal}$, where ϵ is small enough[1], and at the k-th iteration, the co-objective vector is $A_k^{\intercal}$.

Adler *et al.* prove that their parametric constraint-by-constraint algorithm visits $O(\min(m^2, n^2))$ basic points on average, assuming that the inequalities in the system $\boldsymbol{x} \geq 0$, $\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} \geq 0$ can be flipped with probability 1/2. By the first part of Theorem 3, the same result holds in the tropical case. We translate this

[1] Even if ϵ needs to be small, it can be shown that we can choose it so as its size is polynomial in the size of the entries A_{ij}, b_j.

result in terms of mean payoff games. The probability distribution is expressed over the payments matrices A, B, and must satisfy the following requirements:

Assumption 4. (i) for all $i \in [m]$ (resp. $j \in [n]$), the distribution of the matrices A, B is invariant by the exchange of the i-th row (resp. j-th column) of A and B.

(ii) almost surely, A_{ij} and B_{ij} are distinct and not equal to $\mathbb{0}$ for all $i \in [m]$, $j \in [n]$. In this case, we introduce the signed matrix $W = (W_{ij}) \in \mathbb{T}_{\pm}^{m \times n}$, defined by $W_{ij} := A_{ij}$ if $A_{ij} > B_{ij}$, and $\ominus B_{ij}$ if $A_{ij} < B_{ij}$.

(iii) almost surely, the matrix W is strongly non-degenerate.

Condition (i) handles discrete distributions (see Figure 2) as well as continuous ones. In particular, if the distribution of the payment matrices admits a density function f, Condition (i) can be expressed as the invariance of f by flip operations on its arguments. For instance, if $m = 1$ and $n = 2$, the flip invariance holds if, and only if, for almost all a_{ij}, b_{ij}, $f(a_{1,1}, a_{1,2}, b_{1,1}, b_{1,2}) = f(b_{1,1}, b_{1,2}, a_{1,1}, a_{1,2}) = f(b_{1,1}, a_{1,2}, a_{1,1}, b_{1,2}) = f(a_{1,1}, b_{1,2}, b_{1,1}, a_{1,2})$. The requirements $A_{ij}, B_{ij} \neq \mathbb{0}$ for all i, j in Condition (ii) ensure that the flip operations always provide games in which the two players have at least one action to play from every position. The matrix W can be intuitively thought of as a tropical subtraction "$A \ominus B$", and the conditions $A_{ij} \neq B_{ij}$ ensure that W is well defined. Condition (iii) is the analog of the non-degeneracy assumption used in [14] to establish the average-case complexity bound. We point out that the set of matrices A, B which do not satisfy the requirements stated in Conditions (ii) and (iii) has measure zero. As a consequence, these two conditions do not impose important restrictions on the distribution of A, B, and they can rather be understood as genericity conditions.

We can show that TropPCBC$(W_{[m]\times[n-1]}, W_{[m]\times\{n\}})$ allows to determine whether the initial state n is winning in the game with matrices A, B. As every iteration of the tropical shadow-vertex simplex algorithm has a polynomial time complexity (second part of Theorem 3), and the number of visited basic points is polynomial on average, this yields the following theorem:

Theorem 5. *Under a distribution satisfying Assumption 4,* TropPCBC *determines in polynomial time on average whether the initial state $j \in [n]$ is winning for Player Max in the mean payoff game with payment matrices A, B.*

5 Conclusion

We have defined a tropical analogue of the shadow-vertex simplex algorithm, and shown that every iteration has polynomial time complexity. As a corollary, we have established a polynomial-time average-case result on mean payoff games, based on the analysis of Adler, Karp and Shamir of the classical shadow-vertex algorithm. The main restriction of the model is the flip invariance property. It is an open question whether the tropical approach can be applied with other probabilistic models. In particular, it would be interesting to transfer smoothed complexity results, *e.g.* [18], to the tropical setting. The results of Section 3 also suggest that there is a general method to tropicalize any semi-algebraic pivoting rule, based on the characterization of the Newton polytopes involved. This will be addressed in a future work.

Acknowledgments. The authors thank the anonymous reviewers for their helpful comments which contributed to improve the presentation of the paper.

References

1. Zwick, U., Paterson, M.: The complexity of mean payoff games on graphs. Theoretical Computer Science 158(1-2), 343–359 (1996)
2. Ehrenfeucht, A., Mycielski, J.: Positional strategies for mean payoff games. International Journal of Game Theory 8(2), 109–113 (1979)
3. Gurvich, V.A., Karzanov, A.V., Khachivan, L.G.: Cyclic games and an algorithm to find minimax cycle means in directed graphs. USSR Computational Mathematics and Mathematical Physics 28(5), 85–91 (1988)
4. Jurdziński, M.: Deciding the winner in parity games is in UP$\cap$co-UP. Information Processing Letters 68(3), 119–124 (1998)
5. Gaubert, S., Gunawardena, J.: The duality theorem for min-max functions. C. R. Acad. Sci. Paris 326(Série I), 43–48 (1998)
6. Vöge, J., Jurdziński, M.: A discrete strategy improvement algorithm for solving parity games. In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000. LNCS, vol. 1855, pp. 202–215. Springer, Heidelberg (2000)
7. Björklund, H., Vorobyov, S.: A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. Discrete Appl. Math. 155, 210–229 (2007)
8. Jurdziński, M., Paterson, M., Zwick, U.: A deterministic subexponential algorithm for solving parity games. SIAM Journal on Computing 38(4), 1519–1532 (2008)
9. Friedmann, O.: An exponential lower bound for the parity game strategy improvement algorithm as we know it. In: LICS, pp. 145–156. IEEE (August 2009)
10. Brim, L., Chaloupka, J., Doyen, L., Gentilini, R., Raskin, J.: Faster algorithms for mean-payoff games. Formal Methods in System Design 38(2), 97–118 (2011)
11. Akian, M., Gaubert, S., Guterman, A.: Tropical polyhedra are equivalent to mean payoff games. Int. J. Algebr. Comput. 22(1), 125001 (2012)
12. Allamigeon, X., Benchimol, P., Gaubert, S., Joswig, M.: Tropicalizing the simplex algorithm. E-print arXiv:1308.0454 (2013) (submitted)
13. Allamigeon, X., Benchimol, P., Gaubert, S., Joswig, M.: Combinatorial simplex algorithms can solve mean payoff games. E-print arXiv:1309.5925 (submitted, 2014)
14. Adler, I., Karp, R.M., Shamir, R.: A simplex variant solving an $m \times d$ linear program in $O(\min(m^2, d^2))$ expected number of pivot steps. Journal of Complexity 3(4), 372–387 (1987)
15. Roth, A., Balcan, M.F., Kalai, A., Mansour, Y.: On the equilibria of alternating move games. In: Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 805–816. SIAM, Philadelphia (2010)
16. Borgwardt, K.H.: The simplex method: a probabilistic analysis. Algorithms and Combinatorics, vol. 1. Springer (1987)
17. Gelfand, I.M., Kapranov, M.M., Zelevinsky, A.V.: Discriminants, Resultants, and Multidimensional Determinants. Mathematics: Theory & Applications. Birkhäuser, Boston (1994)
18. Spielman, D.A., Teng, S.H.: Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. J. ACM 51(3), 385–463 (2004)

Tighter Relations between Sensitivity and Other Complexity Measures

Andris Ambainis[1,⋆], Mohammad Bavarian[2,⋆⋆], Yihan Gao[3], Jieming Mao[4], Xiaoming Sun[5,⋆⋆⋆], and Song Zuo[6]

[1] University of Latvia, Riga, Latvia
[2] Massachusetts Institute of Technology, Cambridge, MA, USA
[3] University of Illinois at Urbana-Champaign, IL, USA
[4] Princeton University, NJ, USA
[5] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[6] Tsinghua University, Beijing, China

Abstract. The sensitivity conjecture of Nisan and Szegedy [12] asks whether the maximum sensitivity of a Boolean function is polynomially related to the other major complexity measures of Boolean functions.[1] Despite major advances in analysis of Boolean functions in the past decade, the problem remains wide open with no positive result toward the conjecture since the work of Kenyon and Kutin from 2004 [11].

In this work, we prove tighter upper bounds for various complexity measures in terms of sensitivity. More precisely, we show that $\deg(f)^{1-o(1)} = O(2^{s(f)})$ and $C(f) \leq 2^{s(f)-1}s(f)$; these in turn imply various corollaries regarding the relation between sensitvity and other complexity measures, such as block sensitivity, via known results. The gap between sensitivity and other complexity measures remains exponential but these results are the first improvement for this difficult problem that has been achieved in a decade.

1 Introduction

Sensitivity conjecture is a well-known and challenging open problem in the study of complexity measures of Boolean functions. As explaind in detail in various works, the conjecture has many equivalent (or morally equivallent) formulations. Although in this work we shall be mostly concerned with the original formulation of the conjecture in terms of complexity measures of Boolean functions, let us begin by stating the combinatorial formulations of the problem— as this formulation perhaps has the benefit of being more immediately accessible. In the

⋆ Supported by the European Commission under the project QALGO (Grant No. 600700) and the ERC Advanced Grant MQC (Grant No. 320731).

⋆⋆ Supported by NSF STC Award 0939370.

⋆⋆⋆ Supported in part by the National Natural Science Foundation of China Grant 61170062, 61222202.

[1] This includes deterministic, randomized or quantum query complexity, block sensitivity or Fourier degree which are all known to be polynomially related to one another.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 101–113, 2014.

language of extremal combinatorics, the problem is about a certain conjectured Ramsey-type phenomenon over the Hamming cube as follows: does there exists a $\delta > 0$ such that any unbalanced two-coloring of vertices of hypercube $\{0,1\}^n$ contains a verctex $x \in \{0,1\}^n$ such that x has $\geq n^{\delta}$ neighbors in the same color class as x?

In the original form of Nisan and Szegedy [12], the conjecture takes the following form:

Conjecture 1 (sensitivity conjecture). There exists a constant $d \in \mathbb{R}^+$ such that for any Boolean function $f : \{-1,1\}^n \to \{-1,1\}$ we have

$$bs(f) = O(s(f)^d),$$

where $s(f)$ and $bs(f)$ denote the sensitivity and the block sensitivity (defined in Section 2) of the function f .

We shall note that the equivalence between these two seemingly different problems, first observed by Gotsman and Linial [8], is not at all that difficult. Moreover, the equivalence is very direct, and there is almost no cost in parameters for switching from one setting to the other. Thus, one may choose to work in whichever setting one finds more convenient. As such, we shall work exclusively in the complexity theoretic framework, though we shall note that our arguements in Section 4 is inspired and is a refinement of an arguement of Chung et al. from [6] which takes place in the combinatorial setting.

We shall make one final remark about the formulation of the problem before moving on to the discussion of previous works and our results. In the statement of the conjecture, we can replace the block sensitivity by several other widely used complexity measure of Boolean functions (such as deterministic and randomized query complexity, certificate complexity, Fourier degree, etc.) which are all polynomially related to block sensitivity (and to each other), as shown by Nisan and Szegedy [12].

Background. As mentioned above, through the work of various researchers by now many different equivalent forms of the sensitivity conjecture are available. Fortunately, almost all of these different formulations and various approaches to the conjecture are discussed in the recent survey of P. Hatami et al. [10] (see also the blogpost of Aaronson [1] which played an important role in the recent surge of attention to the problem). We refer to these works for a more detailed exposition of the background and the prior works. We briefly recall some of the more immediately relevant facts:

The best known upper bound on block sensitivity is

$$bs(f) \leq (\frac{e}{\sqrt{2\pi}})e^{s(f)}\sqrt{s(f)}, \tag{1}$$

given by Kenyon and Kutin [11]. In the other direction, the first progress on the lower bound was made by Rubinstein [13] who gave the first quadratic separation for block sensitivity and sensitivity by constructing a Boolean function f with $bs(f) = \frac{1}{2}s(f)^2$. Currently, the best lower bound is due to Ambainis and Sun who in [3] exhibited a function with $bs(f) = \frac{2}{3}s(f)^2 - \frac{1}{2}s(f)$.

Results. Our first result in this paper is the following estimate regarding the relation between the maximum sensitivity and Fourier degree of a Boolean function.

Theorem 1. *Let $f : \{0,1\}^n \to \{-1,1\}$ be a Boolean function. Then*

$$\deg(f)^{1-o(1)} \leq s(f)2^{s(f)},$$

where $o(1)$ denotes a term that vanishes as $\deg(f) \to \infty$.

The proof of the above theorem is a mixture of techniques from Fourier analysis and combinatorics. The argument is partly inspired by the arguments in the paper of Chung et al. [6] which recently played an important role in solving a question about the query complexity of restrictions of parity function in [2].

For sensitivity versus certificate complexity, we can prove a somewhat stronger theorem which has direct consequences for sensitivity versus block sensitivity problem (which is the original formulation of sensitivity conjecture by Nisan and Szegedy [12]).

Theorem 2. *For any Boolean function f,*

$$C_1(f) \leq 2^{s_0(f)-1}s_1(f), \quad C_0(f) \leq 2^{s_1(f)-1}s_0(f). \tag{2}$$

Here $C_0(f)$ and $C_1(f)$ denote the 0-certificate complexity and 1-certificate complexity of f. These notions – along with the rest of the background material on complexity measures of Boolean functions – are presented in Section 2.

Using the known relations between various complexity measures of Boolean functions, we can derive several consequences from the above result.

Corollary 1. *For any Boolean function f,*

$$bs(f) \leq C(f) \leq 2^{s(f)-1}s(f).$$

Combining Theorem 2 and some previous results, we can also give another upper bound for block sensitivity.

Corollary 2. *For any Boolean function f,*

$$bs(f) \leq \min\{2^{s_0(f)}, 2^{s_1(f)}\}s_1(f)s_0(f). \tag{3}$$

Hence, we see that our Theorems 1 and 2 and their corollaries show an improved exponent in relation between sensitivity and various complexity measures of Boolean functions compared to the previous best bound shown in equation (1). Beside being the first positive result toward the sensitivity conjecture since the work of Kenyon and Kutin from 2004, we believe our results have the merit of introducing new ideas and techniques which could be valuable elesewhere as well as in the future works on this fundamental conjecture.

Although the bounds obtained in Theorems 1 and Theorem 2 look quite similar, the theorems do not follow one from another by using the known relations between certificate complexity and Fourier degree. On the contrary, the two theorems are obtained by using rather different techniques. However, we shall note that despite their differences both proofs of Theorem 1 and 2 crucially rely on the small set expansion properties of Boolean hypercube. It is plausible that better analytic estimates along the lines of [7] could be useful to slightly improve our results— though a significant improvement is likely to require new ideas.

Organization. In Section 2 we recall some basic definitions and concepts relevant to this work. In Section 3, we prove Theorem 1 and in Section 4, we prove Theorem 2 and its corollaries. Both Sections 3 and 4 are self-contained and can be read in any order.

2 Preliminaries

In this paper, we work with total Boolean functions over the hypercube and their measures of complexity. We assume some basic familiarity with the complexity measures of Boolean functions (as described in the survey [5]). For completeness, we briefly recall some basic definitions.

We work with the usual graph structure on the hypercube by connecting $x, y \in \{0,1\}^n$ if and only if x, y differ in a single coordinate. We always denote by $\log(\cdot)$ the logarithm with the base 2.

Definition 1. *The pointwise sensitivity $s(f,x)$ of a function f on input x is defined as the number of bits on which the function is sensitive, i.e.*

$$s(f,x) = \left|\{i \in [n] | f(x) \neq f(x^{(i)})\}\right|,$$

where $x^{(i)}$ is obtained by flipping the i-th bit of x. We define the total sensitivity by

$$s(f) = \max\left\{s(f,x) | x \in \{0,1\}^n\right\},$$

and the 0-sensitivity and 1-sensitivity by

$$s_0(f) = \max\left\{s(f,x) | x \in \{0,1\}^n, f(x) = 0\right\}, \quad s_1(f) = \max\left\{s(f,x) | x \in \{0,1\}^n, f(x) = 1\right\}.$$

Block sensitivity is another important complexity measure which is obtained by relaxing the requirement that we have to only flip single coordinates by allowing flipping disjoint blocks. More formally block sensitivity is defined as follows:

Definition 2. *The pointwise block sensitivity $bs(f,x)$ of f on input x is defined as maximum number of pairwise disjoint subsets $B_1, ..., B_k$ of $[n]$ such that $f(x) \neq f(x^{(B_i)})$ for all $i \in [k]$. Here $x^{(B_i)}$ is obtained by flipping all the bits $\{x_j | j \in B_i\}$ of x. Define the block sensitivity of f as*

$$bs(f) = \max\left\{bs(f,x) | x \in \{0,1\}^n\right\},$$

and the 0-block sensitivity and 1-block sensitivity, analogously to Definition 1, by

$$bs_0(f) = \max\left\{bs(f,x) | x \in \{0,1\}^n, f(x) = 0\right\}, \quad bs_1(f) = \max\left\{bs(f,x) | x \in \{0,1\}^n, f(x) = 1\right\}.$$

The certificate complexity is another very useful complexity measure with a more non-deterministic type of definition. It is defined as follows:

Definition 3. *The certificate complexity $C(f,x)$ of f on input x is defined as the minimum length of a partial assignment of x such that f is constant on this restriction. Define the certificate complexity of f by*

$$C(f) = \max\big\{C(f,x) | x \in \{0,1\}^n\big\},$$

and the 0-certificate and 1-certificate by

$$C_0(f) = \max\big\{C(f,x) | x \in \{0,1\}^n, f(x) = 0\big\}, \quad C_1(f) = \max\big\{C(f,x) | x \in \{0,1\}^n, f(x) = 1\big\}.$$

Another important notion for us is Fourier degree. It is also polynomially related to block-sensitivity and certificate complexity. To define Fourier degree, recall that any function $f : \{0,1\}^n \to \mathbb{C}$ can be expanded in terms of Fourier characters as follows

$$f(x) = \sum_{S \subseteq [n]} \hat{f}(S)\, \chi_S(x)\,,$$

where $\chi_S(x) = (-1)^{\sum_{i \in S} x_i}$.

Definition 1. *Let $f : \{-1,1\}^n \to \mathbb{R}$ and let $\hat{f}(\cdot)$ denote its Fourier transform. We define Fourier degree of f by*

$$\deg(f) = \max_{\hat{f}(S) \neq 0} |S|\,.$$

Finally, we mention an important and well-known combinatorial result over the hypercube, usually attributed to Harper [9].

Lemma 1 (Hamming Cube Isoperimetry [9]). *Assume $\emptyset \neq A \subseteq \{0,1\}^n$. Let d be the average degree of vertices of A with graph structure on A induced from the natural Hamming graph of $\{0,1\}^n$. Then we have*

$$|A| \geq 2^d\,.$$

The above lemma is quite easy to prove by induction. For a detailed proof which covers the application to combinatorics, we recommend consulting the book by Bollobás [4].

The above theorem implies that if $|A|$ is small, the average degree d must also be relatively small. In this case, the ratio between the number of the edges leaving the set A and the total number of incident edges to A, which is equal to $1 - d/n$, is relatively large. This justifies the alternative name given to the above theorem as the "small set expansion" property of the Hamming cube.

In Section 4, we need an equivalent formulation of discrete isoperimetric inequality, Lemma 1, which will be a more convenient for our purposes there.

Lemma 2. *For any $A \subseteq \{0,1\}^n$, the edges between A and $\bar{A} = \{0,1\}^n \setminus A$ is lower bounded by*

$$|E(A,\bar{A})| \geq |A|(n - \log_2|A|).$$

3 Sensitivity versus Degree

In this section, we shall prove Theorem 1. Let $f : \{0,1\}^n \to \{-1,1\}$ be a Boolean function. To prove Theorem 1, the key idea is to count the following objects.

Definition 4. *An (l,r) S-triple consists of a point $x \in \{0,1\}^n$ and two sets $L \subseteq R \subseteq [n]$ with $|L| = l$ and $|R| = r$ such that $f(x) \neq f(x^i)$ for any $i \in L$.*

In our application, the two parameters $l \leq r$ are chosen as follows: $l = c \log r$, for some $c > 0$ an appropriately chosen constant, and r will be a slowly growing function of n which will be asymptotically $\log\log n$. The upper bound on the number of S-triples is easy to establish.

Lemma 3. *The number of (l,r) S-triples is bounded by*

$$2^n \frac{s(f)^l\, n^{r-l}}{l!(r-l)!}.$$

Proof. We can assume $s(f) \geq l$ as otherwise the number of S-triples is zero. Consider any $x \in \{0,1\}^n$. The number of S-triples involving x is bound by $\max_{1 \leq q \leq s(f)} \binom{q}{l}\binom{n-l}{r-l}$. This is clearly bounded by $\frac{s(f)^l\, n^{r-l}}{l!(r-l)!}$ which implies the above lemma. □

The main part of proving Theorem 1 is to prove a lower bound on the number of S-triples which coupled with the above lemma gives the desired lower bound on $s(f)$. The key idea here is study of restriction of function f to subcubes of dimension r. To be able to carry out our argument we will need a few definition regarding restrictions of functions over the discrete cube.

Restrictions of Boolean functions.

Definition 5. *Given $z \in \{0,1,*\}^n$ and $R \subseteq [n]$, we call them a* compatible pair *if $R = \{i \in [n] : z(i) = *\}$. Each $z \in \{0,1,*\}^n$ naturally corresponds to $|R|$-dimensional subcube $Q_z \subseteq \{0,1\}^n$ defined as follows:*

$$Q_z = \{y \in \{0,1\}^n \,:\, z_i \neq * \Rightarrow y_i = z_i\},$$

i.e. Q_z is constructed by freezing the coordinates of y in $[n] \setminus R$ according to z, and letting the rest of coordinates y_i for $i \in R$ to be free.

Let $f : \{0,1\}^n \to \mathbb{R}$. Given a compatible pair $z \in \{0,1,*\}^n$ and $R = \{i \in [n] : z(i) = *\}$ we obtain a restriction function $f|_z$ given by restricting f to Q_z

Definition 6. *Given $z \in \{0,1,*\}^n$ and $x \in \{0,1\}^n$ (here x is not necessarily in Q_z), define $y = (x \downarrow z)$ to be projection of x to Q_z given by $y_i = z_i$ for any $i \in [n]$ such that $z(i) \neq *$ and $y_i = x_i$ for all the other $i \in [n]$. We define*

$$f|_z(x) = f(x \downarrow z)\,.$$

Notice that $f|_z(x)$ is a function over whole $\{0,1\}^n$ though its value only depends on R the coordinates which $z(i) = *$. Given the above definition one can easily infer the Fourier expansion of the restriction function $f|_z(\cdot)$ from that of f as follows.

$$(f|_z)(x) = \sum_{S \subseteq R} \chi_S(x) \sum_{U \cap R = S} \hat{f}(U)\chi_{U \setminus S}(z) \,.$$

We need the following lemma regarding the degree of restrictions of a function.

Lemma 4. *Let $f : \{-1,1\}^n \to \{-1,1\}$ be function of degree n. Let $R \subseteq [n]$. Then there exist $z \in \{-1,1,*\}^n$ compatible with R such that $(f|_z)$ is also full-degree $|R|$.*

Proof. The coefficient of the highest monomial in Fourier expansion of $(f|_z)$ is given by

$$\sum_{R \subseteq U} \hat{f}(U)\chi_{U \setminus R}(z).$$

Now we calculate the expectation of the square of this value for a random z compatible with R.

$$\mathop{\mathbf{E}}_{z}\left(\sum_{R \subseteq U} \hat{f}(U)\chi_{U \setminus R}(z)\right)^2 = \sum_{R \subseteq U} \hat{f}(U)^2 \geq \hat{f}([n])^2 > 0$$

where for the last inequality we used the fact that f is full-degree. □

The importance of the above lemma is that it allows us to use *induction*:[2] Fix some $R \subseteq [n]$. By the lemma above, there exists $z \in \{0,1,*\}^n$ compatible with R such that $f|_z$ is full-degree. The importance of existence of z is that Q_z always contain an S-triple which was the object we were interested to count. More precisely, since $f|_z$ is full-degree by induction on the degree in Theorem 1 there exists subset $L \subseteq R$ with $|L| \geq \frac{1}{3}\log|R|$ such that there exist $x \in Q_z$ such that $f|_z(x) \neq f|_z(x^i)$ for every $i \in L$. Taking $l = |L|$ and $r = |R|$ we see that (x, L, R) constitutes an S-triple. We use the existence of z and Harper's lemma 1 to prove that for every R there exists not only one such z but in fact many. This is the key estimate we need to prove our result.

The Main Proof of Sensitivity versus Fourier Degree Estimate

Proof (Theorem 1). Without loss of generality we can assume f is full-degree. If this is not the case, choose $S \subseteq [n]$ with $|S| = \deg(f)$ such that $\hat{f}(S) \neq 0$,

[2] It is worth noting that for our induction we do not need the full strength of Theorem 1; in fact, a weaker bound of (say) $\deg(f) \leq 10^{s(f)}$, which follows from [11] and the know relations between Fourier degree and block sensitivity, here suffices. If we were able to change our methods to exploit the sharper induction hypothesis provided by Theorem 1, this may be useful for achieving some improved estimate.

then set the coordinates outside S arbitrary to get a Boolean function on $|S|$-dimensional hypercube of full-degree. This reduces the problem to the full-degree case as restricting a function can only decrease the sensitivity.

Let $r = \omega(1)$ be a very slowly growing function of n to be specified later. Fix a set $R \subseteq [n]$ with $|R| = r$. By Lemma 4 there exist $z \in \{0, 1, *\}^n$ compatible with R such that the restricted function $(f|_z)$ has degree r. Now by induction $s(f|_z) \geq l$ where $l = \Theta(\log r)$. (we can take $l = \frac{1}{3}\log r$ for concreteness.) This means we can find a point $x \in Q_z$ with l neighbors $x_1, x_2, \ldots, x_l$ such that

$$(f|_z)(x_1) = (f|_z)(x_2) = \ldots = (f|_z)(x_l) \neq (f|_z)(x) .$$

Let $L = \{i_1, i_2, \ldots, i_l\} \subseteq R$ be the direction of the edges $(x, x_1), (x, x_2), \ldots, (x, x_l)$ respectively. Then (x, L, R) constitutes a (l, r) S-triple.

So far for any $R \subseteq [n]$ we have shown the existence of one such S-triple. Now we show there are many such triples. Consider Z_R which is the set of all $z \in \{0, 1, *\}^n$ compatible with R. Notice that Z_R can be naturally associated with a $(n - r)$-hypercube with $z_1, z_2 \in Z_R$ said to be *neighbors* in direction $j \in [n] \setminus R$ if $z_1(i) = z_2(i)$ for $i \in [n] \setminus \{j\}$ and $z_1(j) \neq z_2(j)$. (Clearly $z_1(j) \neq *$ and $z_2(j) \neq *$ as both z_1 and z_2 are compatible with R.)

We call a $\widetilde{z} \in Z_R$ *good* if

$$(f|_{\widetilde{z}})(x_1) = (f|_{\widetilde{z}})(x_2) = \ldots = (f|_{\widetilde{z}})(x_l) \neq (f|_{\widetilde{z}})(x) .$$

Let A be the set of all good $\widetilde{z}$ in Z_R. Notice that if $\widetilde{z}$ is good, $((x \downarrow \widetilde{z}), L, R)$ constitutes an S-triple. We have shown so far that $z \in A$ so A is non-empty. Now we prove all elements of A have high degree when seen as a subset of $(n - r)$-hypercube. Indeed, notice that for any $\bar{z} \in Z_R$ and any $\bar{x}$, there are at most $s(f)$ directions $j \in [n] \setminus R$ such that $(f|_{\bar{z}})\left(\bar{x}^{(j)}\right) \neq (f|_{\bar{z}})\left(\bar{x}\right)$. Applying the same reasoning to all $x, x_1, x_2, \ldots, x_l$, we see that for any $z \in A$ there is at least $n - r - s(f)(l+1)$ neighbors of z in A. Now applying our isoperimetric inequality (Lemma 1) to A we see that there are at least $2^{n-r-(l+1)s(f)}$ such special triples for a fixed $R \subseteq [n]$ of size$r$.

On the other hand, the number of such special triples is bounded above by Lemma 3. So we have

$$\binom{n}{r} 2^{n-r-s(f)(l+1)} \leq 2^n \frac{s(f)^l \, n^{r-l}}{l!(r-l)!} .$$

As $r \ll n$ we have $\binom{n}{r} \geq \frac{n^r}{2^r r!}$. Simplifying we see $n^{\frac{l}{l+1}} \leq 4^r \binom{r}{l} s(f) 2^{s(f)}$. Choosing $r \log r = \log n$ and $l = \frac{\log r}{3}$ we get

$$n^{1-O\left(\frac{1}{\log\log n}\right)} \leq s(f) 2^{s(f)} ,$$

which is our desired result. □

4 Sensitivity versus Certificate Complexity

In this section we prove Theorem 2. Actually, we prove a slightly stronger result.

Theorem 3. *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function, then*

$$C_1(f) \leq 2^{s_0(f)-1}s_1(f) - (s_0(f) - 1),\ C_0(f) \leq 2^{s_1(f)-1}s_0(f) - (s_1(f) - 1).^3$$

Proof. By symmetry we only need to prove $C_1(f) \leq 2^{s_0(f)-1}s_1(f) - (s_0(f) - 1)$. Without the loss of generality, we assume $C_1(f) = C(f, 0^n)$, i.e. the 1-certificate complexity is achieved on the input 0^n. We have $f(0^n) = 1$. We assume that the minimal certificate of 0^n consists of $x_1 = 0, x_2 = 0, \ldots, x_m = 0$, where $m = C(f, 0^n) = C_1(f)$.

Let Q_0 be the set of inputs x that satisfies $x_1 = x_2 = \ldots = x_m = 0$. Since $x_1 = 0, x_2 = 0, \ldots, x_m = 0$ is a 1-certificate, we have $\forall\ x \in Q_0$, $f(x) = 1$.

For each $i \in [m]$, let Q_i be the set of inputs x with $x_1 = \ldots = x_{i-1} = x_{i+1} = \ldots = x_m = 0$ and $x_i = 1$. Let S be the total sensitivity of all inputs $x \in \bigcup_{i=1}^m Q_i$. It consists of three parts: sensitivity between Q_i and Q_0 (denoted by S_1), sensitivity inside Q_i (denoted by S_2) and sensitivity between Q_i and other input (denoted by S_3), i.e.

$$S = \sum_{i=1}^{m} \sum_{x \in Q_i} s(f, x) = S_1 + S_2 + S_3. \tag{4}$$

In the following we estimate S_1, S_2 and S_3 separately. We use $A_1, \ldots, A_m$ to denote the set of 0-inputs in $Q_1, \ldots, Q_m$, respectively, i.e. $A_i = \{x \in Q_i | f(x) = 0\}$ $(i \in [m])$. Since $x_1 = \ldots = x_m = 0$ is the minimal certificate, i.e. Q_0 is maximal, thus $A_1, \ldots, A_m$ are all nonempty.

We also need the following lemma which follows from Lemma 2 but can be also proven without using it [14]:

Lemma 5. *For any $i \in [m]$,*

$$|A_i| \geq 2^{n-m-s_0(f)+1}.$$

The sensitivity between Q_i and Q_0 is $|A_i|$. By summing over all possible i we get:

$$S_1 = \sum_{i=1}^{m} |A_i|. \tag{5}$$

Sensitivity Inside $Q_1, \ldots, Q_m$: By Lemma 2, for each $i \in [m]$, the number of edges between A_i and $Q_i \setminus A_i$ is bounded by:

$$|E(A_i, Q_i \setminus A_i)| \geq |A_i|(\log_2 |Q_i| - \log_2 |A_i|) = |A_i|(n - m - \log_2 |A_i|).$$

[3] If $s_0(f) = 0$ or $s_1(f) = 0$, then f is constant, hence $s(f) = bs(f) = C(f) = 0$.

Therefore,

$$\begin{aligned} S_2 &= 2\sum_{i=1}^{m} |E(A_i, Q_i \setminus A_i)| \\ &\geq 2\sum_{i=1}^{m} |A_i|(n - m - \log_2 |A_i|). \end{aligned} \tag{6}$$

Sensitivity between Q_i and Other Inputs (i.e. $\{0,1\}^n \setminus \bigcup_{j=0}^{m} Q_j$): For each $1 \leq i < j \leq m$, let $Q_{i,j}$ be the set of inputs (not in Q_0) that are *adjacent* to both Q_i and Q_j, i.e. $Q_{i,j}$ is the set of inputs x that satisfy $x_1 = \ldots x_{i-1} = x_{i+1} = \ldots x_{j-1} = x_{j+1} = \ldots x_m = 0$ and $x_i = x_j = 1$. The sensitivity between Q_i, Q_j and $Q_{i,j}$ is lower bounded by

$$\sum_{x \in Q_0} |f(x + e_i) - f(x + e_j)|.$$

where e_i is the unit vector with the i-th coordinate equal to 1 and all other coordinates equal to 0. Then, $x + e_i$, $x + e_j$ are the neighbors of x in Q_i and Q_j, respectively. Summing over all possible pairs of (i, j) we get

$$\begin{aligned} S_3 &\geq \sum_{1 \leq i < j \leq m} \sum_{x \in Q_0} |f(x + e_i) - f(x + e_j)| \\ &= \sum_{x \in Q_0} \left(\sum_{i=1}^{m} f(x + e_i) \right) \left(m - \sum_{i=1}^{m} f(x + e_i) \right) \\ &= \sum_{x \in Q_0} s(f, x)(m - s(f, x)). \end{aligned} \tag{7}$$

If we combine inequalities (5)-(7), we get

$$\begin{aligned} S &= \sum_{i=1}^{m} \sum_{x \in Q_i} s(f, x) \\ &\geq \sum_{i=1}^{m} |A_i| + 2\sum_{i=1}^{m} |A_i|(n - m - \log_2 |A_i|) + \sum_{x \in Q_0} s(f, x)(m - s(f, x)). \end{aligned} \tag{8}$$

Since $s(f, x)$ is upper bounded by $s_0(f)$ or $s_1(f)$ (depending on whether $f(x) = 0$ or $f(x) = 1$), we have

$$\begin{aligned} \sum_{x \in Q_i} s(f, x) &\leq |A_i| s_0(f) + (|Q_i| - |A_i|) s_1(f) \\ &= |A_i| s_0(f) + (2^{n-m} - |A_i|) s_1(f) \end{aligned}$$

Thus,

$$S = \sum_{i=1}^{m} \sum_{x \in Q_i} s(f, x) \leq \sum_{i=1}^{m} \Big(|A_i| s_0(f) + (2^{n-m} - |A_i|) s_1(f) \Big). \tag{9}$$

We use w to denote the total number of 0-inputs in $Q_1, \ldots, Q_m$. Then,

$$w = \sum_{i=1}^{m} |A_i| = \sum_{x \in Q_0} s(f, x).$$

The inequality (9) can be rewritten as

$$S \le w \cdot s_0(f) + (m \cdot 2^{n-m} - w)s_1(f). \tag{10}$$

Also, $s(f,x) \le s_1(f)$ for each $x \in Q_0$. Thus, the right-hand side of inequality (8) is

$$\begin{aligned}
&\sum_{i=1}^{m} |A_i| + 2\sum_{i=1}^{m} |A_i|(n - m - \log_2 |A_i|) + \sum_{x \in Q_0} s(f,x)(m - s(f,x)) \\
&\ge w + 2\sum_{i=1}^{m} |A_i|(n - m - \log_2 |A_i|) + (m - s_1(f)) \sum_{x \in Q_0} s(f,x) \\
&= w + 2w(n-m) - 2\sum_{i=1}^{m} |A_i| \log_2 |A_i| + (m - s_1(f))w \\
&= w(1 + 2n - m - s_1(f)) - 2\sum_{i=1}^{m} |A_i| \log_2 |A_i|. \qquad (11)
\end{aligned}$$

By combining inequalities (8)-(11) we get

$$w(1 + 2n - m - s_1(f)) - 2\sum_{i=1}^{m} |A_i| \log_2 |A_i| \le w \cdot s_0(f) + (m \cdot 2^{n-m} - w)s_1(f).$$

By rearranging the inequality we get

$$w(1 + 2n - m - s_0(f)) \le 2\sum_{i=1}^{m} |A_i| \log_2 |A_i| + m \cdot 2^{n-m} s_1(f). \tag{12}$$

Since the function $g(x) = x \log_2 x$ is convex and we know that $|A_i| \le |Q_i| = 2^{n-m}$, from Lemma 5 $|A_i| \ge 2^{n-m-s_0(f)+1}$. Therefore,

$$\begin{aligned}
g(|A_i|) &= g\left(\frac{|A_i| - 2^{n-m-s_0(f)+1}}{2^{n-m} - 2^{n-m-s_0(f)+1}} \cdot 2^{n-m} + \frac{2^{n-m} - |A_i|}{2^{n-m} - 2^{n-m-s_0(f)+1}} \cdot 2^{n-m-s_0(f)+1}\right) \\
&\le \frac{|A_i| - 2^{n-m-s_0(f)+1}}{2^{n-m} - 2^{n-m-s_0(f)+1}} \cdot g(2^{n-m}) + \frac{2^{n-m} - |A_i|}{2^{n-m} - 2^{n-m-s_0(f)+1}} \cdot g(2^{n-m-s_0(f)+1}) \\
&= \frac{|A_i| - 2^{n-m-s_0(f)+1}}{2^{n-m} - 2^{n-m-s_0(f)+1}} \cdot 2^{n-m}(n-m) \\
&\quad + \frac{2^{n-m} - |A_i|}{2^{n-m} - 2^{n-m-s_0(f)+1}} \cdot 2^{n-m-s_0(f)+1}(n - m - s_0(f) + 1) \\
&= \frac{|A_i| - 2^{n-m-s_0(f)+1}}{2^{s_0(f)-1} - 1} \cdot 2^{s_0(f)-1}(n-m) + \frac{2^{n-m} - |A_i|}{2^{s_0(f)-1} - 1}(n - m - s_0(f) + 1) \\
&= \left(\frac{|A_i| - 2^{n-m-s_0(f)+1}}{2^{s_0(f)-1} - 1} 2^{s_0(f)-1} + \frac{2^{n-m} - |A_i|}{2^{s_0(f)-1} - 1}\right)(n-m) - \frac{2^{n-m} - |A_i|}{2^{s_0(f)-1} - 1}(s_0(f) - 1) \\
&= |A_i|(n-m) - \frac{2^{n-m} - |A_i|}{2^{s_0(f)-1} - 1}(s_0(f) - 1).
\end{aligned}$$

Hence

$$\begin{aligned}\sum_{i=1}^{m}|A_i|\log_2|A_i| &= \sum_{i=1}^{m} g(|A_i|)\\ &\le \sum_{i=1}^{m}\left(|A_i|(n-m)-\frac{2^{n-m}-|A_i|}{2^{s_0(f)-1}-1}(s_0(f)-1)\right)\\ &= w(n-m+\frac{s_0(f)-1}{2^{s_0(f)-1}-1})-m\cdot 2^{n-m}\frac{s_0(f)-1}{2^{s_0(f)-1}-1}. \quad (13)\end{aligned}$$

By combining inequalities (12) and (13), we get

$$\begin{aligned}&w(1+2n-m-s_0(f))\\ &\le 2\left(w(n-m+\frac{s_0(f)-1}{2^{s_0(f)-1}-1})-m\cdot 2^{n-m}\frac{s_0(f)-1}{2^{s_0(f)-1}-1}\right)+m\cdot 2^{n-m}s_1(f).\end{aligned}$$

It implies that

$$w\left(1+m-s_0(f)-\frac{2(s_0(f)-1)}{2^{s_0(f)-1}-1}\right)\le m\cdot 2^{n-m}\left(s_1(f)-\frac{2(s_0(f)-1)}{2^{s_0(f)-1}-1}\right),$$

Substituting $w=\sum_{i=1}^{m}|A_i|\ge m\cdot 2^{n-m-s_0(f)+1}$, we get

$$m\cdot 2^{n-m-s_0(f)+1}\left(1+m-s_0(f)-\frac{2(s_0(f)-1)}{2^{s_0(f)-1}-1}\right)\le m\cdot 2^{n-m}\left(s_1(f)-\frac{2(s_0(f)-1)}{2^{s_0(f)-1}-1}\right),$$

i.e.

$$1+m-s_0(f)-\frac{2(s_0(f)-1)}{2^{s_0(f)-1}-1}\le 2^{s_0(f)-1}\left(s_1(f)-\frac{2(s_0(f)-1)}{2^{s_0(f)-1}-1}\right),$$

which implies

$$m\le 2^{s_0(f)-1}s_1(f)-s_0(f)+1.$$

□

4.1 Proof of Corollary 2

To prove Corollary 2, we need the following Lemma by Kenyon and Kutin.[4]

Lemma 6. *[11]* $bs_0(f)\le 2(C_1(f)-\frac{1}{2})s_0(f),\ bs_1(f)\le 2(C_0(f)-\frac{1}{2})s_1(f).$

Proof. (of Corollary 2) From Theorem 2 $bs_0(f)\le C_0(f)\le 2^{s_1(f)-1}s_0(f)$. From Corollary 6 we have $bs_0(f)\le 2(C_1(f)-\frac{1}{2})s_0(f)$, together with Theorem 2 we get $bs_0(f)\le 2(2^{s_0(f)-1}s_1(f)-\frac{1}{2})s_0(f)$. Therefore, $bs_0(f)\le \min\{2^{s_1(f)}s_0(f),\ 2^{s_0(f)}s_1(f)s_0(f)\}$. Similarly we can show that $bs_1(f)\le \min\{2^{s_1(f)}s_0(f)s_1(f),\ 2^{s_0(f)}s_1(f)\}$. □

[4] In their original paper there is no "$-\frac{1}{2}$" term, but a careful analysis will provide it.

References

1. Aaronson, S.: My philomath project: Sensitivity versus block-sensitivity, Shtetl Optimized blog (June 13, 2010), http://www.scottaaronson.com/blog/?p=453
2. Aaronson, S., Ambainis, A., Balodis, K., Bavarian, M.: Weak Parity. In: ICALP (to appear, 2014)
3. Ambainis, A., Sun, X.: New separation between $s(f)$ and $bs(f)$. Electronic Colloquium on Computational Complexity (ECCC) 18, 116 (2011)
4. Bollobás, B.: Combinatorics: set systems, hypergraphs, families of vectors and combinatorial probability. Cambridge University Press, Cambridge (1986)
5. Buhrman, H., de Wolf, R.: Complexity measures and decision tree complexity: a survey. Theoretical Computer Science 288(1), 21–43 (2002)
6. Chung, F.R.K., Füredi, Z., Graham, R.L., Seymour, P.: On induced subgraphs of the cube. J. Comb. Theory Ser. A 49 (1988)
7. Falik, D., Samorodnitsky, A.: Edge-isoperimetric inequalities and influences. Combinatorics, Probability & Computing 16(5), 693–712 (2007)
8. Gotsman, C., Linial, N.: The equivalence of two problems on the cube. Journal of Combinatorial Theory, Series A 61(1), 142–146 (1992)
9. Harper, L.: Optimal numberings and isoperimetric problems on graphs. Journal of Combinatorial Theory 1 (1966)
10. Hatami, P., Kulkarni, R., Pankratov, D.: Variations on the Sensitivity Conjecture. Theory of Computing Library, Graduate Surveys (4), 1–27 (2011)
11. Kenyon, C., Kutin, S.: Sensitivity, block sensitivity, and l-block sensitivity of Boolean functions. Information and Computation 189(1), 43–53 (2004)
12. Nisan, N., Szegedy, M.: On the degree of boolean functions as real polynomials. Computational Complexity 4, 301–313 (1994)
13. Rubinstein, D.: Sensitivity vs. Block Sensitivity of Boolean functions. Combinatorica 15(2), 297–299 (1995)
14. Simon, H.U.: A Tight $\Omega(\log\log n)$-Bound on the Time for Parallel Ram's to Compute Nondegenerated Boolean Functions. In: Karpinski, M. (ed.) FCT 1983. LNCS, vol. 158, pp. 439–444. Springer, Heidelberg (1983)

On Hardness of Jumbled Indexing

Amihood Amir[1],*, Timothy M. Chan[2],**, Moshe Lewenstein[3],***, and Noa Lewenstein[4]

[1] Bar-Ilan University and Johns Hopkins University
[2] University of Waterloo
[3] Bar-Ilan University
[4] Netanya College

Abstract. Jumbled indexing is the problem of indexing a text T for queries that ask whether there is a substring of T matching a pattern represented as a Parikh vector, i.e., the vector of frequency counts for each character. Jumbled indexing has garnered a lot of interest in the last four years; for a partial list see [2,6,13,16,17,20,22,24,26,30,35,36]. There is a naive algorithm that preprocesses all answers in $O(n^2|\Sigma|)$ time allowing quick queries afterwards, and there is another naive algorithm that requires no preprocessing but has $O(n \log |\Sigma|)$ query time. Despite a tremendous amount of effort there has been little improvement over these running times.

In this paper we provide good reason for this. We show that, under a 3SUM-hardness assumption, jumbled indexing for alphabets of size $\omega(1)$ requires $\Omega(n^{2-\epsilon})$ preprocessing time or $\Omega(n^{1-\delta})$ query time for any $\epsilon, \delta > 0$. In fact, under a stronger 3SUM-hardness assumption, for any constant alphabet size $r \geq 3$ there exist describable fixed constant ϵ_r and δ_r such that jumbled indexing requires $\Omega(n^{2-\epsilon_r})$ preprocessing time or $\Omega(n^{1-\delta_r})$ query time.

1 Introduction

Equal length strings are said to *jumble-match* if they are commutatively equivalent (sometimes called Abelian equivalent), i.e., if one string can be obtained from the other by permuting its characters. A jumble match can be described using *Parikh vectors* which are vectors maintaining the frequency count of each alphabet character. Two strings jumble-match if they have the same Parikh vector. We also say that a string jumble-matches a Parikh vector ψ if the string's Parikh vector is the same as ψ.

* Supported by NSF grant CCR-09-04581, ISF grant 347/09, and BSF grant 2008217.

** Part of the work of this author was done while visiting the Department of Computer Science and Engineering, Hong Kong University of Science and Technology.

*** This research was done in part while the author was on sabbatical in the U. of Waterloo. The research is supported by BSF grant 2010437 and GIF grant 1147/2011.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 114–125, 2014.

Parikh vectors were introduced in [37] and have been used to analyze grammars [31] and characterize commutative languages [27]. Furthermore, jumbled pattern matching appears in various applications of computational biology, such as SNP discovery [10], analysis of similarities among different protein sequences [28], and automatic pattern discovery in biosequencing applications [21]. It has also been examined in the streaming model [33].

Jumbled pattern matching on its own can easily be solved by using a sliding window in linear time for the alphabet $\{1, \ldots, O(n)\}$, or $O(n \log |\Sigma|)$ time for a general alphabet Σ. In contrast, the exact pattern matching problem can only be solved in linear time via more complex techniques (e.g., see [29,11]). Jumbled pattern matching has also been studied along with other metrics (e.g., see [14,15,1]).

1.1 Jumbled Indexing

Jumbled indexing (JI), currently under very active research, asks whether one can "index" jumbled matching. The goal is to preprocess a given text S efficiently so that when given a Parikh vector ψ one can quickly check whether there exists a substring of S that jumble-matches ψ.

For classical exact matching, text indexing paradigms of linear size and with near linear query time (in the query size) exist since the introduction of suffix trees [40]. Many other efficient text indexing structures have been studied since then, such as suffix array [34]. Other matching problems have also been successfully transformed into efficient indexing paradigms. For example, *parameterized matching* allows parametric symbols that are required to map to characters in a consistent manner. Parameterized matching was introduced by Baker [7,8] for detection of repetitive similar modules in software and has applications for color images [3,5,39] and approximate image search [25]. Parameterized matching can be solved in linear time [4]. In [7] a parameterized suffix tree was introduced. Both the preprocessing and the query times are near-linear (where the latter is linear in the query size). Another example is order-preserving matchings, where two numerical strings match if their order is preserved. Efficient order-preserving matchings were presented in [32] and recently an order-preserving index was introduced [19] that can also be preprocessed in linear time with linear time queries (in the query size). Indexing with errors [18] has proven to be somewhat harder.

Given that jumbled matching can be trivially solved in linear time, for the above-mentioned alphabets, one would expect that jumbled indexing would be a relatively easy problem. However, jumbled indexing is surprisingly difficult.

There are two naive methods to solve jumbled indexing. One is to use the sliding window technique mentioned above for every query that arrives. This can be done in $O(n)$ time if the alphabet is a subset of $[n]$, where n is the text size. Another method is to preprocess all possible answers in advance by computing the Parikh vectors of every substring in $O(n^2|\Sigma|)$ time. Improving upon this has proven to be challenging even for constant-sized alphabets.

In an effort to make progress on JI the simplest version of the problem was considered, that of a binary alphabet. A neat property of a binary alphabet is that a Parikh vector (i,j) appears in text T iff i is between the minimum and maximum number of 1s over all substrings of length $i+j$. This was used in [16] to obtain efficient query time by storing the minimum and maximum values of all possible lengths, yielding an index of $O(n)$ space and $O(1)$ query time. However, the preprocessing still took $O(n^2)$ time.

Burcsi et al. [13] and, independently, Moosa and Rahman [35] succeeded in improving the preprocessing time by a log factor to $O(\frac{n^2}{\log n})$. They achieved this by reducing binary JI to (min,+)-convolution which can be solved in $O(\frac{n^2}{\log n})$ time [12]. Later, Moosa and Rahman [36] improved this to $O(\frac{n^2}{\log^2 n})$ by using the four-Russians trick. Recently, Hermelin et al. [26] reduced the problem to (min,+)-matrix multiplication or all-pairs shortest paths, but a similar reduction has already appeared in an earlier paper by Bremner et al. [12]; with the latest breakthrough by Williams [41] on all-pairs shortest paths, the preprocessing time for binary JI becomes $O(\frac{n^2}{2^{\Omega((\log n/\log\log n)^{0.5})}})$. For the binary case there are also algorithms for run-length encoded strings [6,24] and for an approximate version of the problem [17]. The binary case was also extended to trees [22].

Lately, there has been some progress also for non-binary alphabets. Kociumaka et al. [30] presented a solution for JI for any constant-sized alphabet Σ that uses $O(\frac{n^2\log^2\log n}{\log n})$ preprocessing time and space and answers queries in $O((\frac{\log n}{\log\log n})^{2|\Sigma|-1})$ time. Amir et al. [2] proposed a solution for constant-sized alphabets that preprocesses in $O(n^{1+\epsilon})$ time and answers queries in $\tilde{O}(m^{\frac{1}{\epsilon}})$ time, where m is the sum of the Parikh vector elements. In an even newer paper, Durocher et al. [20] considered alphabet size $|\Sigma| = o((\frac{\log n}{\log\log n})^2)$ and showed how to construct an index in $O(|\Sigma|(\frac{n}{\log_{|\Sigma|} n})^2)$ time and answer queries in $O(n^\epsilon + |\Sigma|)$ time, where $\epsilon > 0$ is an arbitrary small constant. This still leaves us in a sad state of affairs. In all the (exact) solutions mentioned for $|\Sigma| \geq 3$ the time complexity of preprocessing or the time complexity of querying is always within polylogarithmic factors of one of the above two naive algorithms. The question that has troubled the community in these last few years is whether jumbled indexing could be solved with $O(n^{2-\epsilon})$ preprocessing time and $O(n^{1-\delta})$ for some constants $\epsilon, \delta > 0$.

In this paper we show that for alphabets of $\omega(1)$ size this is impossible under a 3SUM-hardness assumption. We further show that for any constant alphabet size $r \geq 3$ there exist describable fixed constants ϵ_r and δ_r such that jumbled indexing requires $\Omega(n^{2-\epsilon_r})$ preprocessing time or $\Omega(n^{1-\delta_r})$ query time under a stronger 3SUM-hardness assumption.

1.2 3SUM

Numerous algorithmic problems have polynomial time upper bounds that we suspect are the best obtainable but proving matching lower bounds is difficult in classical computational models. Recently, a different approach for showing hardness (e.g. [42]) has been to choose an algorithmic problem that seem harder than others, e.g. maximum flow, APSP, edit distance, or 3SUM, and to use them as a hard primitive and reduce them to other problems that we would like to show are hard.

In this paper we use the *3SUM problem* defined as follows.

- **Input:** $x_1, x_2, \dots, x_n$.
- **Output:** yes, if distinct i, j and k exist such that $x_i + x_j = x_k$. No, otherwise.

As far back as the mid 90's there were reductions from 3SUM, especially within the computational geometry community. Gajentaan and Overmars [23] were the first to reduce from 3SUM in order to provide evidence for near-quadratic complexity for computational geometry problems such as minimum-area triangle, finding 3 collinear points, and determining whether n axis-aligned rectangles cover a given rectangle. Others followed and quite a few problems are now known to be 3SUM-hard.

Pătraşcu [38] pointed out that most of the reductions transform the condition $x_i + x_j = x_k$ into some geometric or algebraic condition by common arithmetic, but it is difficult to use 3SUM for reductions to purely combinatorial problems, such as those on graphs or strings. To overcome this he defined *Convolution-3SUM*, a more restricted 3SUM version, which is just as hard as 3SUM in the sense that an $O(n^{2-\epsilon})$-time solution for Convolution-3SUM for some $\epsilon > 0$ would imply an $O(n^{2-\epsilon'})$-time solution for 3SUM for some $\epsilon' > 0$ [38].

The *Convolution-3SUM problem* is defined as follows.

- **Input:** $x_1, \dots, x_n$.
- **Output:** Yes, if there are distinct i and j such that $x_i + x_j = x_{i+j}$. No, otherwise.

By shuffling and changing indices an alternative equivalent output is:

- **Output:** Yes, if there are distinct i and j such that $x_i - x_j = x_{i-j}$. No, otherwise.

We consider these problems in the RAM model with the elements belonging to an integer set $\{-u, \dots, u\}$ as was assumed by others, e.g. [9,38]. It is possible to achieve an algorithm of $O(u \log u)$ time for the 3SUM problem [9] by Fast Fourier transform. This can easily be transformed into an $O(nu \log(nu))$ time algorithm for Convolution-3SUM. Pătraşcu [38] pointed out that the techniques of Baran et al. [9] yield a (randomized) reduction, for the 3SUM problem, from a large domain $\{-u, \dots, u\}$ to the domain of $\{-n^3, \dots, n^3\}$. This reduction can be adapted for Convolution-3SUM from $\{-u, \dots, u\}$ to $\{-n^2, \dots, n^2\}$. The reason is that in 3SUM there are n^3 triples to consider when bounding the number of false positives, but in Convolution-3SUM there are only n^2 triples ($x_i + x_j = x_{i+j}$) to consider.

Hence, Convolution-3SUM for input $\subset \{-u, \dots, u\}$ is hard if $u \geq n^2$ and is easier than $O(n^2)$ for $u \ll n$. Convolution-3SUM for inputs $\subset \{-n, \dots, n\}$ seems (though has not proven) to be as hard as the general case. This leads us to state two hardness assumptions. For both we assume, as in [9,38], the Word RAM model with words of $O(\log n)$ bits.

- **3SUM-hardness Assumption:** Any algorithm for Convolution-3SUM requires $n^{2-o(1)}$ time in expectation to determine whether a set $\{x_1, \dots, x_n\} \subset \{-n^2, \dots, n^2\}$ contains a pair x_i, x_j such that $x_i - x_j = x_{i-j}$.
- **Strong 3SUM-hardness Assumption:** Any algorithm for Convolution-3SUM requires $n^{2-o(1)}$ time in expectation to determine whether a set $\{x_1, \dots, x_n\} \subset \{-n, \dots, n\}$ contains a pair x_i, x_j such that $x_i - x_j = x_{i-j}$.

1.3 Preliminaries and Definitions

Let S be a string of length n over an alphabet $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_{|\Sigma|}\}$. An integer i is a *location* or a *position* in S if $i \in \{1, \dots, |S|\}$. The substring $S[i\,..\,j]$ of S, for any two positions $i \leq j$, is the substring of S that begins at index i and ends at index j. The string generated by a character a repeated r times is shorthanded with a^r.

The *Parikh vector* of a string S is $\psi(S) = (c_1(S), c_2(S), \dots, c_{|\Sigma|}(S))$, where $c_i(S)$ is the count of occurrences of the i-th character of Σ. Two strings (of equal length) S and S' are said to *jumble-match* if they have the same Parikh vector. For a text T and pattern P we say that P *jumble-matches at location* i if the substring $T[i\,..\,i+|P|-1]$ jumble-matches P. *Jumbled pattern matching* refers to the problem where one is given a pattern and text and seeks all locations where the pattern jumble-matches. For a Parikh vector $\psi = (c_1, \dots, c_{|\Sigma|})$, we denote its length with $|\psi|$ which is $\Sigma_{i=1}^{|\Sigma|} c_i$.

Jumbled indexing (JI, for short), also known as *histogram indexing*, *Parikh indexing*, or *permutation indexing*, is defined as follows.

- **Preprocess:** a text S over alphabet Σ.
- **Query:** Given a vector $\psi \in \mathbb{N}^{|\Sigma|}$, decide whether there is a substring S' such that the Parikh vector $\psi(S')$ is equal to ψ.

2 Hardness of Jumbled Indexing

2.1 Outline

We will show that, under the 3SUM-hardness assumption, one cannot improve the running time over the naive methods mentioned in the introduction by any polynomial factors for alphabets of super-constant size; and for alphabets of constant size there are polynomial time lower bounds, dependent on the alphabet size.

To achieve these results we reduce from 3SUM to JI. Naturally, we use Convolution-3SUM, which is more appropriate for problems with structure. A very high-level description of our reduction is as follows. A Convolution-3SUM input is transformed to JI by hashing the input values to much smaller sized values by using mod over a collection of primes. These are then novelly transformed to a string. The queries on the string simulate testing matchings mod primes in parallel. Using mod primes causes several problems, which lead to interesting ideas to overcome these obstacles.

2.2 Setup

Let $x_1, x_2, \ldots, x_n$ be the input of the Convolution-3SUM problem such that each $x_i \in \{-n^2, \ldots, n^2\}$. Under the strong 3SUM-hardness assumption, each $x_i \in \{-n, \ldots, n\}$.

We choose a collection of roughly equal-sized primes $p_1, \ldots, p_k$ (for some choice of k) with their product $p_1 \cdots p_k > n^2$ (or, under the strong 3SUM-hardness assumption, with $p_1 \cdots p_k > n$). It is possible to choose $p_1, \ldots, p_k \in \Theta(n^{2/k})$ (or, for the strong assumption, $p_1, \ldots, p_k \in \Theta(n^{1/k})$) to satisfy this requirement for any given $k \leq \frac{\log n}{\log \log n}$, because of the density of the primes.

The alphabet of JI in the reduction will consist of a character for each prime we choose, plus two more special characters we introduce later. Therefore, the JI alphabet size will be $|\Sigma| = k + 2$.

The lemma below follows directly from properties of mod and will be instrumental in obtaining our result.

Lemma 1. *Let $p_1, \ldots, p_k$ be a set of primes such that $p_1 \cdots p_k > u$ (with $u = n^2$ or $u = n$ depending on the hardness assumption). Let $i > j$. Then $x_i - x_j = x_{i-j} \iff \forall r : (x_i - x_j) \bmod p_r = x_{i-j} \bmod p_r$*

$$\iff \forall r : (x_i \bmod p_r) - (x_j \bmod p_r) \in \begin{cases} (x_{i-j} \bmod p_r) \\ (x_{i-j} \bmod p_r) - p_r \end{cases}$$

2.3 Reduction

In the reduction to the JI instance we will generate an input string S to be preprocessed and a set of n queries $Q_1, \ldots, Q_n$ which we now describe.

JI Input String. We generate an input string S based on the Convolution-3SUM input $x_1, \ldots, x_n$. For every prime p_j we create a character a_j and for each x_i we create a substring

$$S_i = a_1^{EXP(i,1)} a_2^{EXP(i,2)} \cdots a_k^{EXP(i,k)},$$

where

$$EXP(i, j) = (x_{i+1} \bmod p_j) - (x_i \bmod p_j).$$

We note that, for the sake of simplicity, we are cheating since the exponent of a character in a string cannot be negative. We will shortly explain how to fix this.

Finally, we define

$$S = \$\#\ S_1\ \#\$\#\ S_2\ \#\$\#\ \cdots\ \#\$\#\ S_{n-1}\ \#\$,$$

where # and $ are separator characters.

The structure of S is such that substrings beginning and ending within separators #$# have the property that the number of occurrences of each character is reminiscent of the requirements of Lemma 1.

Lemma 2. *Consider the substring of* S, $R_{(j,i)} = \$\#\ S_j\ \#\$\#\ \ldots\ \#\$\#\ S_{i-1}\ \#\$$. *Each character* a_ℓ *has exactly* $(x_i \bmod p_\ell) - (x_j \bmod p_\ell)$ *occurrences in* $R_{(j,i)}$.

Proof. The character a_ℓ has $EXP(j, \ell)$ occurrences in S_j, $EXP(j+1, \ell)$ occurrences in $S_{j+1}, \ldots$, $EXP(i-1, \ell)$ occurrences in S_{i-1}. Hence, we have $\Sigma_{d=j}^{i-1} EXP(d, \ell)$ occurrences of a_ℓ in $R_{(j,i)}$. Then $\Sigma_{d=j}^{i-1} EXP(d, \ell) = \Sigma_{d=j}^{i-1} (x_{d+1} \bmod p_\ell) - (x_d \bmod p_\ell)$, which telescopes to $(x_i \bmod p_\ell) - (x_j \bmod p_\ell)$. □

By combining Lemmas 1 and 2 we can deduce the following.

Corollary 1. *There is a solution* $x_i - x_j = x_{i-j}$ *to the Convolution-3SUM iff the number of occurrences of each character* a_ℓ *in* $R_{(j,i)}$ *is in*

$$\{(x_{i-j} \bmod p_\ell), (x_{i-j} \bmod p_\ell) - p_\ell\}.$$

To fix the problem of the negative exponent we set $D = \max_{i=1}^{k} p_i$ and change $EXP(i, j) = (x_{i+1} \bmod p_j) - (x_i \bmod p_j) + D$. Now, the exponent is not negative, but is still of order $\Theta(n^{2/k})$ (or $\Theta(n^{1/k})$ under the strong 3SUM-hardness assumption), which is the size of each prime. We leave it as an easy exercise to verify that Lemma 2 can be modified so that each character a_ℓ has exactly $(x_i \bmod p_\ell) - (x_j \bmod p_\ell) + D(i-j)$ occurrences in $R_{(j,i)}$ and, in turn, that Corollary 1 can be modified so that $a_\ell \in \{(x_{i-j} \bmod p_\ell) + D(i-j), (x_{i-j} \bmod p_\ell) - p_\ell + D(i-j)\}$.

JI Queries. We generate n queries $\psi_1, \ldots, \psi_n$ for the jumbled indexing instance such that each ψ_L represents x_L, an element of the Convolution-3SUM input. The query ψ_L will imitate a query on the Convolution-3SUM data asking whether there exist i and j such that

(a) $x_L = x_i - x_j$ and
(b) $L = i - j$.

We will also embed the query with data requiring that

(c) any substring that jumble-matches the query ψ must be of the form $R_{(j,i)}$ from Lemma 2.

Obviously, answers to all queries ψ_L will be sufficient to derive a solution to Convolution-3SUM.

To enforce (c) and (b) we use the separators of S, #, and \$. For (c) we require the form of $R_{(j,i)}$ and for (b) we require $L = i - j$, which means that each potential substring $R_{(j,i)}$ should contain exactly L parts S_h.

Observation 1. *Any substring of S that jumble-matches the query ψ, where ψ has $L+1$ for \$ and $2L$ for #, must be of the form $R_{(j,i)}$ (of Lemma 2) and must satisfy $L = i - j$.*

Proof. Let R be a substring of S such that R jumble-matches ψ. Then each set of separators #\$# fully contained in R contributes twice as many #'s than \$'s. Since our query asks for $L+1$ \$'s but only $2L$ #'s, it must be that R begins and ends with a \$ and hence is of the form $R_{(j,i)}$. Moreover, since there are $L+1$ \$'s and R begins and ends with a \$, there must be exactly L parts S_h in R, implying that $L = i - j$. □

It remains to show how to adapt the query in order to enforce (a) $x_L = x_i - x_j$. Here we will use Corollary 1. It is sufficient to find the substrings $R_{(j,i)}$ such the number of occurrences of each character a_ℓ is either $((x_i - x_j) \bmod p_\ell) + DL$ or $((x_i - x_j) \bmod p_\ell) - p_\ell + DL$. However, checking two options (for each a_ℓ) cannot be done with one JI query. So, we split the query ψ_L into 2^k queries $\psi_L^{(1)}, \ldots, \psi_L^{(2^k)}$ for the 2^k different equalities that satisfy Corollary 1.

Hence, we have overall $2^k n$ JI queries. These queries provide a full answer to the Convolution-3SUM problem.

Theorem 2. *Consider the jumbled indexing problem with text size s and alphabet size $r \geq 5$. Then under the 3SUM-hardness assumption, one of the following holds for any fixed $\epsilon > 0$:*

1. *the preprocessing time is $\Omega(s^{2-\frac{4}{r}-\epsilon})$, or*
2. *the query time is $\Omega(s^{1-\frac{2}{r}-\epsilon})$.*

Proof. Without loss of generality, assume that $r \leq \frac{\log s}{\log \log s}$ (otherwise, $s^{\frac{1}{r}} = \tilde{\Theta}(1)$ and we may as well make r equal to $\frac{\log s}{\log \log s}$).

Let $x_1, \ldots, x_n \in \{-n^2, \ldots, n^2\}$ be the input of the Convolution-3SUM problem. We apply the above reduction and generate the string S as described. Denote its length by s. Recall that the alphabet size is $r = |\Sigma| = k+2$, where k is the number of primes (the 2 is for the separators \$ and #). Since each prime $p_i \in \Theta(n^{2/k})$, we have $s = O(kn^{2/k}n) = \tilde{O}(n^{\frac{r}{r-2}})$ for $k+2 = r \in o(\log n)$. In other words, $n = \tilde{\Omega}(s^{1-\frac{2}{r}})$.

Applying the preprocessing and subsequently answering all $2^k n$ defined queries yields a solution to the Convolution-3SUM problem. Letting $P(s)$ and $Q(s)$ be the preprocessing and query time, we then have $P(s) + 2^k n Q(s) \geq \Omega(n^{2-\epsilon})$.

For $k+2 = r \in o(\log n)$ we note that $2^k \in o(n^\epsilon)$. We must thus have $P(s) \geq \Omega(n^{2-\epsilon}) = \Omega(s^{2(1-\frac{2}{r})-O(\epsilon)})$ or $Q(s) \geq \Omega(n^{1-O(\epsilon)}) = \Omega(s^{1-\frac{2}{r}-O(\epsilon)})$. □

Note that for $r \leq 4$, the bound in the above theorem becomes vacuous. We can get somewhat better bounds under the strong 3SUM-hardness assumption.

Theorem 3. *Consider the jumbled indexing problem with text size s and alphabet size $r \geq 4$. Then under the strong 3SUM-hardness assumption, one of the following holds for any fixed $\epsilon > 0$:*

1. *the preprocessing time is $\Omega(s^{2-\frac{2}{r-1}-\epsilon})$, or*
2. *the query time is $\Omega(s^{1-\frac{1}{r-1}-\epsilon})$.*

The proof is the same as in the previous theorem, but with $p_i \in \Theta(n^{1/k})$.

By the same proof and further calculations, we can also get a slightly strengthened lower bound of $\Omega(s^2/2^{O(\sqrt{\log s})})$ preprocessing time or $\Omega(s/2^{O(\sqrt{\log s})})$ query time for alphabet size $r = \Theta(\sqrt{\log s})$, under the assumption that 3SUM has an $\Omega(n^2/2^{O(\sqrt{\log n})})$ lower bound.

2.4 Hardness of JI with Alphabet Size 3

The reduction we have presented contains two separators in the string S. Recall that for every prime we also construct a character. We require that the multiplication of the primes be $> n$ for strong 3SUM-hardness and $> n^2$ for 3SUM-hardness. However, if a prime is of order $\Omega(n)$ then the size of the string S would be $\Omega(n^2)$, too large to gain anything from the reduction. Hence, we need at least two primes for strong 3SUM-Hardness and three primes for 3SUM-hardness. In this section we generate a string which requires only one separator, and for 2 primes p and q of size $\Theta(\sqrt{n})$ this yields a nontrivial result under the strong 3SUM-hardness assumption.

While we construct a different string for JI and need to argue a claim similar to Lemma 2 and Observation 1, the structure of the proof remains the same.

Let a be a character representing prime p, and b be a character that represents prime q, and # be a separator character. Let $D = \max\{p, q\}$. Define

$S_i = (a\#)^{(x_{i+1} \bmod p)-(x_i \bmod p)+D}(b\#)^{(x_{i+1} \bmod q)-(x_i \bmod q)+D}$ and
$S = \#^D a^{2D} \#^D\ S_1\ \#^D a^{2D} \#^D\ S_2\ \#^D a^{2D} \#^D\ \cdots\ \#^D a^{2D} \#^D\ S_{n-1}\ \#^D a^{2D} \#^D$,
where $\#^D a^{2D} \#^D$ is the separator (a has a double role).

Define $R_{(j,i)} = \#^D\ S_j\ \#^D a^{2D} \#^D\ \cdots\ \#^D a^{2D} \#^D\ S_{i-1}\ \#^D$.

It is easy to verify, similar to Lemma 2, that a has exactly $(x_i \bmod p) - (x_j \bmod p) + D(i-j) + 2D(i-j-1)$ occurrences in $R_{(j,i)}$ and b has exactly $(x_i \bmod q) - (x_j \bmod q) + D(i-j)$ occurrences in $R_{(j,i)}$. Hence, as in Corollary 1, there is a solution $x_i - x_j = x_{i-j}$ to Convolution-3SUM iff the number of occurrences of a in $R_{(j,i)}$ is in $\{((x_i - x_j) \bmod p) + D(3i-3j-2), ((x_i - x_j) \bmod p) - p + D(3i-3j-2)\}$ and the number of occurrences of b in $R_{(j,i)}$ is in $\{((x_i - x_j) \bmod p) + D(i-j), ((x_i - x_j) \bmod p) - p + D(i-j)\}$.

The tricky part is to obtain an alternative to Observation 1. The difficulty stems from the fact that a and # appear both in the separator part and in the S_i's.

Observation 4. *Say we have a Parikh vector $\psi = (n_1, n_2, n_1 + n_2 + 4D)$ for $(a, b, \#)$, with $L = (n_1 \text{ div } 3D) + 1$. Then any substring of S which jumble-matches ψ is of the form $R_{(j,i)}$. Moreover $i - j = L$.*

Proof. Define $\Delta = 4D$ the difference between the number of $\#$'s and the number of a's and b's (put together). Let x be a substring of S which jumble-matches ψ. Any S_l or separator $\#^D a^{2D} \#^D$ fully contained in x has a balanced number of $\#$'s and non-$\#$'s and, hence, does not affect Δ. So, at either end there is a part of a separator or an S_l which both together contributes to Δ. It is straightforward to confirm that the only way this is possible is having x begin with $\#^D$, the prefix of a separator, and end with $\#^D$, the suffix of a separator. Hence, x has the required form $R_{(j,i)}$.

We show that $i - j = L = (n_1 \text{ div } 3D) + 1$. We have claimed that the number of a's in $R_{(j,i)}$ is $(x_i \bmod p) - (x_j \bmod p) + D(3i - 3j - 2)$. Hence, since $R_{(j,i)}$ jumble-matches ψ, we have $n_1 = (x_i \bmod p) - (x_j \bmod p) + D + 3D(i - j - 1)$. Since each $(x_i \bmod p) - (x_j \bmod p) + D \in [0, 2D)$ it follows that $(n_1 \text{ div } 3D) = i - j - 1$. □

Finally, for a given L all of our queries have $n_1 = (x_L \bmod p) + D(3L - 2)$ or $n_1 = (x_L \bmod p) - p + D(3L - 2)$ in location a of the Parikh vector. In both cases, $L = (n_1 \text{ div } 3D) + 1$, satisfying the requirement of Observation 4. Hence,

Theorem 5. *Consider the jumbled indexing problem with text size s and alphabet size 3. Under the strong 3SUM-hardness assumption, one of the following holds for any fixed $\epsilon > 0$:*

1. *the preprocessing time is $\Omega(s^{\frac{4}{3}-\epsilon})$, or*
2. *the query time is $\Omega(s^{\frac{2}{3}-\epsilon})$.*

Proof. Note that $p, q \in \Theta(\sqrt{n})$. Hence, S is of length $s = O(n^{\frac{3}{2}})$. Following the same arguments as in Theorem 2 yields the result. □

Epilogue. In a forthcoming work, the second and third author will present new improved algorithms for the jumbled indexing problem for any constant alphabet size $r \geq 2$ that achieves truly sublinear query time and $O(n^{2-\frac{2}{r+O(1)}})$ preprocessing time, thus nearly matching the lower bound in Theorem 3.

References

1. Amir, A., Apostolico, A., Landau, G.M., Satta, G.: Efficient text fingerprinting via Parikh mapping. J. Discrete Algorithms 1(5-6), 409–421 (2003)
2. Amir, A., Butman, A., Porat, E.: On the relationship between histogram indexing and block-mass indexing. In: Philosophical Transactions A (to appear)
3. Amir, A., Church, K.W., Dar, E.: Separable attributes: a technique for solving the sub matrices character count problem. In: SODA, pp. 400–401 (2002)
4. Amir, A., Farach, M., Muthukrishnan, S.: Alphabet dependence in parameterized matching. Inf. Process. Lett. 49(3), 111–115 (1994)

5. Phanendra Babu, G., Mehtre, B.M., Kankanhalli, M.S.: Color indexing for efficient image retrieval. Multimedia Tools and Applications 1(4), 327–348 (1995)
6. Badkobeh, G., Fici, G., Kroon, S., Lipták, Z.: Binary jumbled string matching for highly run-length compressible texts. Inf. Process. Lett. 113(17), 604–608 (2013)
7. Baker, B.S.: Parameterized pattern matching: Algorithms and applications. J. Comput. Syst. Sci. 52(1), 28–42 (1996)
8. Baker, B.S.: Parameterized duplication in strings: Algorithms and an application to software maintenance. SIAM J. Comput. 26(5), 1343–1362 (1997)
9. Baran, I., Demaine, E.D., Pătraşcu, M.: Subquadratic algorithms for 3SUM. In: Dehne, F., López-Ortiz, A., Sack, J.-R. (eds.) WADS 2005. LNCS, vol. 3608, pp. 409–421. Springer, Heidelberg (2005)
10. Böcker, S.: Simulating multiplexed SNP discovery rates using base-specific cleavage and mass spectrometry. Bioinformatics 23(2), 5–12 (2007)
11. Boyer, R.S., Moore, J.S.: A fast string searching algorithm. Commun. ACM 20(10), 762–772 (1977)
12. Bremner, D., Chan, T.M., Demaine, E.D., Erickson, J., Hurtado, F., Iacono, J., Langerman, S., Pătraşcu, M., Taslakian, P.: Necklaces, convolutions, and $X + Y$. Algorithmica 69, 294–314 (2014)
13. Burcsi, P., Cicalese, F., Fici, G., Lipták, Z.: On table arrangements, scrabble freaks, and jumbled pattern matching. In: Boldi, P. (ed.) FUN 2010. LNCS, vol. 6099, pp. 89–101. Springer, Heidelberg (2010)
14. Butman, A., Eres, R., Landau, G.M.: Scaled and permuted string matching. Inf. Process. Lett. 92(6), 293–297 (2004)
15. Butman, A., Lewenstein, N., Munro, I.J.: Permuted scaled matching. In: CPM 2014 (to appear, 2014)
16. Cicalese, F., Fici, G., Lipták, Z.: Searching for jumbled patterns in strings. In: Prague Stringology Conference, pp. 105–117 (2009)
17. Cicalese, F., Laber, E.S., Weimann, O., Yuster, R.: Near linear time construction of an approximate index for all maximum consecutive sub-sums of a sequence. In: Kärkkäinen, J., Stoye, J. (eds.) CPM 2012. LNCS, vol. 7354, pp. 149–158. Springer, Heidelberg (2012)
18. Cole, R., Gottlieb, L.-A., Lewenstein, M.: Dictionary matching and indexing with errors and don't cares. In: STOC, pp. 91–100 (2004)
19. Crochemore, M., Iliopoulos, C.S., Kociumaka, T., Kubica, M., Langiu, A., Pissis, S.P., Radoszewski, J., Rytter, W., Waleń, T.: Order-preserving incomplete suffix trees and order-preserving indexes. In: Kurland, O., Lewenstein, M., Porat, E. (eds.) SPIRE 2013. LNCS, vol. 8214, pp. 84–95. Springer, Heidelberg (2013)
20. Durocher, S., Ian Munro, J., Mondal, D., Thankachan, S.V.: Jumbled pattern matching over large alphabets (2014) (manuscript, personal communication)
21. Eres, R., Landau, G.M., Parida, L.: Permutation pattern discovery in biosequences. Journal of Computational Biology 11(6), 1050–1060 (2004)
22. Gagie, T., Hermelin, D., Landau, G.M., Weimann, O.: Binary jumbled pattern matching on trees and tree-like structures. In: Bodlaender, H.L., Italiano, G.F. (eds.) ESA 2013. LNCS, vol. 8125, pp. 517–528. Springer, Heidelberg (2013)
23. Gajentaan, A., Overmars, M.H.: On a class of $O(n^2)$ problems in computational geometry. Comput. Geom. 5, 165–185 (1995)
24. Giaquinta, E., Grabowski, S.: New algorithms for binary jumbled pattern matching. Inf. Process. Lett. 113(14-16), 538–542 (2013)
25. Hazay, C., Lewenstein, M., Sokol, D.: Approximate parameterized matching. ACM Transactions on Algorithms 3(3) (2007)

26. Hermelin, D., Landau, G.M., Rabinovich, Y., Weimann, O.: Binary jumbled pattern matching via all-pairs shortest paths (2014), http://arxiv.org/abs/1401.2065 (manuscript)
27. Holub, S.: Parikh test sets for commutative languages. ITA 42(3), 525–537 (2008)
28. Huang, X., Ali, H., Sadanandam, A., Singh, R.: SRPVS: a new motif searching algorithm for protein analysis. In: Proceedings of the 2004 IEEE Computational Systems Bioinformatics Conference, CSB 2004, pp. 674–675 (2004)
29. Knuth, D.E., Morris Jr., J.H., Pratt, V.R.: Fast pattern matching in strings. SIAM J. Comput. 6(2), 323–350 (1977)
30. Kociumaka, T., Radoszewski, J., Rytter, W.: Efficient indexes for jumbled pattern matching with constant-sized alphabet. In: Bodlaender, H.L., Italiano, G.F. (eds.) ESA 2013. LNCS, vol. 8125, pp. 625–636. Springer, Heidelberg (2013)
31. Kopczynski, E., Widjaja, A.: Parikh images of grammars: Complexity and applications. In: LICS, pp. 80–89 (2010)
32. Kubica, M., Kulczynski, T., Radoszewski, J., Rytter, W., Walen, T.: A linear time algorithm for consecutive permutation pattern matching. Inf. Process. Lett. 113(12), 430–433 (2013)
33. Lee, L.-K., Lewenstein, M., Zhang, Q.: Parikh matching in the streaming model. In: Calderón-Benavides, L., González-Caro, C., Chávez, E., Ziviani, N. (eds.) SPIRE 2012. LNCS, vol. 7608, pp. 336–341. Springer, Heidelberg (2012)
34. Manber, U., Myers, E.W.: Suffix arrays: A new method for on-line string searches. SIAM J. Comput. 22(5), 935–948 (1993)
35. Moosa, T.M., Rahman, M.S.: Indexing permutations for binary strings. Inf. Process. Lett. 110(18-19), 795–798 (2010)
36. Moosa, T.M., Rahman, M.S.: Sub-quadratic time and linear space data structures for permutation matching in binary strings. J. Discrete Algorithms 10, 5–9 (2012)
37. Parikh, R.: On context-free languages. J. ACM 13(4), 570–581 (1966)
38. Pătraşcu, M.: Towards polynomial lower bounds for dynamic problems. In: STOC, pp. 603–610 (2010)
39. Swain, M.J., Ballard, D.H.: Color indexing. International Journal of Computer Vision 7(1), 11–32 (1991)
40. Weiner, P.: Linear pattern matching algorithms. In: SWAT (FOCS), pp. 1–11 (1973)
41. Williams, R.: Faster all-pairs shortest paths via circuit complexity. In: STOC (to appear, 2014)
42. Williams, V.V., Williams, R.: Subcubic equivalences between path, matrix and triangle problems. In: FOCS, pp. 645–654 (2010)

Morphing Planar Graph Drawings Optimally[⋆]

Patrizio Angelini[1], Giordano Da Lozzo[1], Giuseppe Di Battista[1], Fabrizio Frati[2],
Maurizio Patrignani[1], and Vincenzo Roselli[1]

[1] Dipartimento di Ingegneria, Roma Tre University, Italy
{angelini,dalozzo,gdb,patrigna,roselli}@dia.uniroma3.it
[2] School of Information Technologies, The University of Sydney, Australia
fabrizio.frati@sydney.edu.au

Abstract. We provide an algorithm for computing a planar morph between any two planar straight-line drawings of any n-vertex plane graph in $O(n)$ morphing steps, thus improving upon the previously best known $O(n^2)$ upper bound. Furthermore, we prove that our algorithm is optimal, that is, we show that there exist two planar straight-line drawings Γ_s and Γ_t of an n-vertex plane graph G such that any planar morph between Γ_s and Γ_t requires $\Omega(n)$ morphing steps.

1 Introduction

A *morph* is a continuous transformation between two topologically equivalent geometric objects. The study of morphs is relevant for several areas of computer science, including computer graphics, animation, and modeling. Many of the geometric shapes that are of interest in these contexts can be effectively described by two-dimensional planar graph drawings. Hence, designing algorithms and establishing bounds for morphing planar graph drawings is an important research challenge. We refer the reader to [7–9, 12, 13] for extensive descriptions of the applications of graph drawing morphs.

It has long been known that there always exists a *planar morph* (that is, a morph that preserves planarity at any time instant) transforming any planar straight-line drawing Γ_s of a plane graph G into any other planar straight-line drawing Γ_t of G. The first proof of such a result, published by Cairns in 1944 [5], was "existential", i.e., no guarantee was provided on the complexity of the trajectories of the vertices during the morph. Almost 40 years later, Thomassen proved in [14] that a morph between Γ_s and Γ_t always exists in which vertices follow trajectories of exponential complexity (in the number of vertices of G). In other words, adopting a setting defined by Grünbaum and Shepard [10] which is also the one we consider in this paper, Thomassen proved that a sequence $\Gamma_s = \Gamma_1, \Gamma_2, \ldots, \Gamma_k = \Gamma_t$ of planar straight-line drawings of G exists such that, for $1 \leq i \leq k-1$, the *linear morph* transforming Γ_i into Γ_{i+1} is planar, where a linear morph moves each vertex at constant speed along a straight-line trajectory.

A breakthrough was recently obtained by Alamdari *et al.* by proving that a planar morph between any two planar straight-line drawings of the same n-vertex connected

[⋆] Work partially supported by ESF EuroGIGA GraDR, by the Australian Research Council (grant DE140100708), and by the MIUR project AMANDA "Algorithmics for MAssive and Networked DAta", prot. 2012C4E3KT_001.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 126–137, 2014.

plane graph exists in which each vertex follows a trajectory of polynomial complexity [1]. That is, Alamdari *et al.* showed an algorithm to perform the morph in $O(n^4)$ *morphing steps*, where a morphing step is a linear morph. The $O(n^4)$ bound was shortly afterwards improved to $O(n^2)$ by Angelini *et al.* [3].

In this paper, we provide an algorithm to compute a planar morph with $O(n)$ morphing steps between any two planar straight-line drawings Γ_s and Γ_t of any n-vertex connected plane graph G. Also, we prove that our algorithm is optimal. That is, for every n, there exist two drawings Γ_s and Γ_t of the same n-vertex plane graph (in fact a path) such that any planar morph between Γ_s and Γ_t consists of $\Omega(n)$ morphing steps. To the best of our knowledge, no super-constant lower bound was previously known.

The schema of our algorithm is the same as in [1, 3]. Namely, we morph Γ_s and Γ_t into two drawings Γ_s^x and Γ_t^x in which a certain vertex v can be contracted onto a neighbor x. Such contractions generate two straight-line planar drawings Γ_s' and Γ_t' of a smaller plane graph G'. A morph between Γ_s' and Γ_t' is recursively computed and suitably modified to produce a morph between Γ_s and Γ_t. The main ingredient for our new bound is a drastically improved algorithm to morph Γ_s and Γ_t into Γ_s^x and Γ_t^x. In fact, while the task of making v contractible onto x is accomplished with $O(n)$ morphing steps in [1, 3], we devise and use properties of monotone drawings, level planar drawings, and hierarchical graphs to perform it with $O(1)$ morphing steps.

The idea behind the lower bound is that linear morphs can poorly simulate rotations, that is, a morphing step rotates an edge of an angle whose size is $O(1)$. We then consider two drawings Γ_s and Γ_t of an n-vertex path P, where Γ_s lies on a straight-line, whereas Γ_t has a spiral-like shape, and we prove that in any planar morph between Γ_s and Γ_t there is one edge of P whose total rotation describes an angle whose size is $\Omega(n)$.

Because of space limitations, some proofs are omitted and can be found in [2].

2 Preliminaries

Drawings and Embeddings. A *planar straight-line drawing* of a graph maps each vertex to a distinct point in the plane and each edge to a straight-line segment between its endpoints so that no two edges cross. A planar drawing partitions the plane into topologically connected regions, called *faces*. The bounded faces are *internal*, while the unbounded face is the *outer face*. A planar straight-line drawing is *convex* if each face is delimited by a convex polygon. A planar drawing of a graph determines a circular ordering of the edges incident to each vertex, called *rotation system*. Two drawings of a graph are *equivalent* if they have the same rotation system and the same outer face. A *plane embedding* is an equivalence class of planar drawings. A graph with a plane embedding is called a *plane graph*. A plane graph is *maximal* if no edge can be added to it while maintaining its planarity.

Subgraphs and Connectivity. A *subgraph* $G'(V', E')$ of a graph $G(V, E)$ is a graph such that $V' \subseteq V$ and $E' \subseteq E$; G' is *induced* if, for every $u, v \in V'$, $(u, v) \in E'$ if and only if $(u, v) \in E$. If G is a plane graph, then a subgraph G' of G is regarded as a plane graph whose plane embedding is the one obtained from G by removing all the vertices and edges not in G'.

A graph G is *k-connected* if removing any $k-1$ vertices leaves G connected; a *separating k-set* is a set of k vertices whose removal disconnects G. A 3-cycle in a plane graph G is *separating* if it contains vertices both in its interior and in its exterior. Every separating 3-set in a maximal plane graph G induces a separating 3-cycle.

Monotonicity. An *arc $\boldsymbol{xy}$* is a line segment directed from a point x to a point y; $\boldsymbol{xy}$ is *monotone* with respect to an oriented straight line $\boldsymbol{d}$ if it has a *positive projection* on $\boldsymbol{d}$, i.e., for any two distinct points p and q in this order along $\boldsymbol{xy}$ from x to y, the projection of p on $\boldsymbol{d}$ precedes the projection of q on $\boldsymbol{d}$ according to the orientation of $\boldsymbol{d}$. A path $P = (u_1, \dots, u_n)$ is *$\boldsymbol{d}$-monotone* if the straight-line arc $\boldsymbol{u_i u_{i+1}}$ is monotone with respect to $\boldsymbol{d}$, for $i = 1, \dots, n-1$; a polygon Q is *$\boldsymbol{d}$-monotone* if it contains two vertices s and t such that the two paths between s and t that delimit Q are both $\boldsymbol{d}$-monotone. A path P (a polygon Q) is *monotone* if there exists an oriented straight line $\boldsymbol{d}$ such that P (resp. Q) is $\boldsymbol{d}$-monotone. We show two lemmata about monotone paths and polygons.

Lemma 1. *Let Q be any convex polygon and let* **d** *be any oriented straight line not perpendicular to any straight line through two vertices of Q. Then Q is* **d***-monotone.*

Lemma 2. *Any simple polygon Q with at most* 5 *vertices is monotone.*

Morphing. A *linear morph* between two straight-line planar drawings Γ_1 and Γ_2 of a plane graph G is a continuous transformation from Γ_1 to Γ_2 such that each vertex moves at constant speed along a straight line from its position in Γ_1 to the one in Γ_2. A linear morph between Γ_1 and Γ_2 is denoted by $\langle \Gamma_1, \Gamma_2 \rangle$. A linear morph is *planar* if no crossing or overlap occurs between any two edges or vertices during the transformation. A planar linear morph is also called a *morphing step*. In the remainder of the paper, we will construct *unidirectional* linear morphs, that were defined in [4] as linear morphs in which the straight-line trajectories of the vertices are parallel.

A *morph* $\langle \Gamma_s, \dots, \Gamma_t \rangle$ between two straight-line planar drawings Γ_s and Γ_t of a plane graph G is a finite sequence of morphing steps that transforms Γ_s into Γ_t. A *unidirectional morph* is such that each of its morphing steps is unidirectional.

Let Γ be a planar straight-line drawing of a plane graph G. The *kernel* of a vertex v of G is the open convex region R such that placing v at any point of R while maintaining the position of every other vertex unchanged yields a planar straight-line drawing of G. If a neighbor x of v lies on the boundary of the kernel of v in Γ, we say that v is *x-contractible*. The *contraction of v onto x* in Γ is the operation resulting in: (i) a simple graph $G' = G/(v, x)$ obtained from G by removing v and by replacing each edge (v, w), where $w \neq x$, with an edge (x, w) (if it does not already belong to G); and (ii) a planar straight-line drawing Γ' of G' such that each vertex different from v is mapped to the same point as in Γ. Also, the *uncontraction of v from x into Γ* is the reverse operation of the contraction of v onto x in Γ, i.e., the operation that produces a planar straight-line drawing Γ of G from a planar straight-line drawing Γ' of G'. A vertex v in a plane graph G is *quasi-contractible* if (i) $\deg(v) \leq 5$ and (ii) for any two neighbors u and w of v connected by an edge, cycle (u, v, w) delimits a face of G. We have the following.

Lemma 3. *(Angelini et al. [3]) Every plane graph contains a quasi-contractible vertex.*

In the remainder of the paper, even when not explicitly specified, we will only consider and perform contractions of quasi-contractible vertices.

Let Γ_1 and Γ_2 be two straight-line planar drawings of the same plane graph G. We define a *pseudo-morph* of Γ_1 into Γ_2 inductively, as follows:
(A) a unidirectional morph with m morphing steps of Γ_1 into Γ_2 is a pseudo-morph with m steps of Γ_1 into Γ_2;
(B) a unidirectional morph with m_1 morphing steps of Γ_1 into a straight-line planar drawing Γ_1^x of G, followed by a pseudo-morph with m_2 steps of Γ_1^x into a straight-line planar drawing Γ_2^x of G, followed by a unidirectional morph with m_3 morphing steps of Γ_2^x into Γ_2 is a pseudo-morph of Γ_1 into Γ_2 with $m_1 + m_2 + m_3$ steps; and
(C) let Γ_1' (Γ_2') be the straight-line planar drawing of the plane graph G' obtained by contracting a quasi-contractible vertex v of G onto x in Γ_1 (in Γ_2); the contraction of v onto x, followed by a pseudo-morph with m steps of Γ_1' into Γ_2' and by the uncontraction of v from x into Γ_2 is a pseudo-morph with $m + 2$ steps of Γ_1 into Γ_2.

Pseudo-morphs have two useful and powerful features.

First, it is easy to design an inductive algorithm for constructing a pseudo-morph between any two planar straight-line drawings Γ_1 and Γ_2 of the same n-vertex plane graph G. Namely, consider any quasi-contractible vertex v of G and let x be any neighbor of v. Morph unidirectionally Γ_1 and Γ_2 into two planar straight-line drawings Γ_1^x and Γ_2^x, respectively, in which v is x-contractible. Now contract v onto x in Γ_1^x and in Γ_2^x obtaining two planar straight-line drawings Γ_1' and Γ_2', respectively, of the same $(n-1)$-vertex plane graph G'. Then, the algorithm is completed by inductively computing a pseudo-morph of Γ_1' into Γ_2'.

Second, computing a pseudo-morph between Γ_1 and Γ_2 leads to computing a planar unidirectional morph between Γ_1 and Γ_2, as formalized in Lemma 4. We remark that, although Lemma 4 has never been stated as below, its proof can be directly derived from the results of [1, 3] and, mainly, of Barrera-Cruz *et al.* [4].

Lemma 4. *Let Γ_s and Γ_t be two straight-line planar drawings of a plane graph G. Let $\mathcal{P}$ be a pseudo-morph with m steps transforming Γ_s into Γ_t. It is possible to construct a planar unidirectional morph M with m morphing steps transforming Γ_s into Γ_t.*

Hierarchical Graphs and Level Planarity. A *hierarchical graph* is defined as a tuple $(G, \boldsymbol{d}, L, \gamma)$ where: (i) G is a graph; (ii) $\boldsymbol{d}$ is an oriented straight line in the plane; (iii) L is a set of parallel lines (sometimes called *layers*) that are orthogonal to $\boldsymbol{d}$; the lines in L are assumed to be ordered in the same order as they are intersected by $\boldsymbol{d}$ when traversing such a line according to its orientation; and (iv) γ is a function that maps each vertex of G to a line in L in such a way that, if an edge (u, v) belongs to G, then $\gamma(u) \neq \gamma(v)$. A *level drawing* of $(G, \boldsymbol{d}, L, \gamma)$ (sometimes also called *hierarchical drawing*) maps each vertex v of G to a point on the line $\gamma(v)$ and each edge (u, v) of G such that line $\gamma(u)$ precedes line $\gamma(v)$ in L to an arc $\boldsymbol{uv}$ monotone with respect to $\boldsymbol{d}$. A *hierarchical plane graph* is a hierarchical graph $(G, \boldsymbol{d}, L, \gamma)$ such that G is a plane graph and such that a level planar drawing Γ of $(G, \boldsymbol{d}, L, \gamma)$ exists that "respects" the embedding of G (that is, the rotation system and the outer face of G in Γ are the same as in the plane embedding of G). Given a hierarchical plane graph $(G, \boldsymbol{d}, L, \gamma)$, an *st-face* of G is a face delimited by two paths $(s = u_1, u_2, \ldots, u_k = t)$ and $(s = v_1, v_2, \ldots, v_l = t)$ such that $\gamma(u_i)$

precedes $\gamma(u_{i+1})$ in L, for every $1 \leq i \leq k-1$, and such that $\gamma(v_i)$ precedes $\gamma(v_{i+1})$ in L, for every $1 \leq i \leq l-1$. We say that $(G, \boldsymbol{d}, L, \gamma)$ is a *hierarchical plane st-graph* if every face of G is an st-face. Let Γ be any straight-line level planar drawing of a hierarchical plane graph $(G, \boldsymbol{d}, L, \gamma)$ and let f be a face of G; then, it is easy to argue that f is an st-face if and only if the polygon delimiting f in Γ is $\boldsymbol{d}$-monotone.

In this paper we will use an algorithm by Hong and Nagamochi that constructs convex straight-line level planar drawings of hierarchical plane st-graphs [11]. Here we explicitly formulate a weaker version of their main theorem.[1]

Theorem 1. *(Hong and Nagamochi [11]) Every 3-connected hierarchical plane st-graph $(G, \boldsymbol{d}, L, \gamma)$ admits a convex straight-line level planar drawing.*

Consider any straight-line level planar drawing Γ of a hierarchical plane graph $(G, \boldsymbol{d}, L, \gamma)$. Since each edge (u, v) of G is represented in Γ by a $\boldsymbol{d}$-monotone arc, the fact that (u, v) intersects a line $l_i \in L$ does not depend on the actual drawing Γ, but only on the fact that l_i lies between lines $\gamma(u)$ and $\gamma(v)$ in L. Assume that each line $l_i \in L$ is oriented so that $\boldsymbol{d}$ cuts l_i from the right to the left of l_i. We say that an edge e *precedes* (*follows*) a vertex v on a line l_i in Γ if $\gamma(v) = l_i$, e intersects l_i in a point $p_i(e)$, and $p_i(e)$ precedes (resp. follows) v on l_i when traversing such a line according to its orientation. Also, we say that an edge e *precedes* (*follows*) an edge e' on a line l_i in Γ if e and e' both intersect l_i at points $p_i(e)$ and $p_i(e')$, and $p_i(e)$ precedes (resp. follows) $p_i(e')$ on l_i when traversing such a line according to its orientation.

Now consider two straight-line level planar drawings Γ_1 and Γ_2 of a hierarchical plane graph $(G, \boldsymbol{d}, L, \gamma)$. We say that Γ_1 and Γ_2 are *left-to-right equivalent* if, for any line $l_i \in L$, for any vertex or edge x of G, and for any vertex or edge y of G, we have that x precedes (follows) y on l_i in Γ_1 if and only if x precedes (resp. follows) y on l_i in Γ_2. We are going to make use of the following lemma.

Lemma 5. *Let Γ_1 and Γ_2 be two left-to-right equivalent straight-line level planar drawings of the same hierarchical plane graph $(G, \boldsymbol{d}, L, \gamma)$. Then the linear morph $\langle \Gamma_1, \Gamma_2 \rangle$ transforming Γ_1 into Γ_2 is planar and unidirectional.*

3 A Morphing Algorithm

In this section we describe an algorithm to construct a planar unidirectional morph with $O(n)$ steps between any two straight-line planar drawings Γ_s and Γ_t of the same n-vertex plane graph G. The algorithm relies on two subroutines, called FAST CONVEXIFIER and CONTRACTIBILITY CREATOR, which are described in Sections 3.1 and 3.2, respectively. The algorithm is described in Section 3.3.

[1] We make some remarks. First, the main result in [11] proves that a convex straight-line level planar drawing of $(G, \boldsymbol{d}, L, \gamma)$ exists even if a convex polygon representing the cycle delimiting the outer face of G is arbitrarily prescribed. Second, the result holds for a super-class of the 3-connected planar graphs, namely for all the graphs that admit a convex straight-line drawing [6, 15]. Third, the result assumes that the lines in L are horizontal; however, a suitable rotation of the coordinate axes shows how that assumption is not necessary. Fourth, looking at the figures in [11] one might get the impression that the lines in L need to be equidistant; however, this is nowhere used in their proof, hence the result holds for any set of parallel lines.

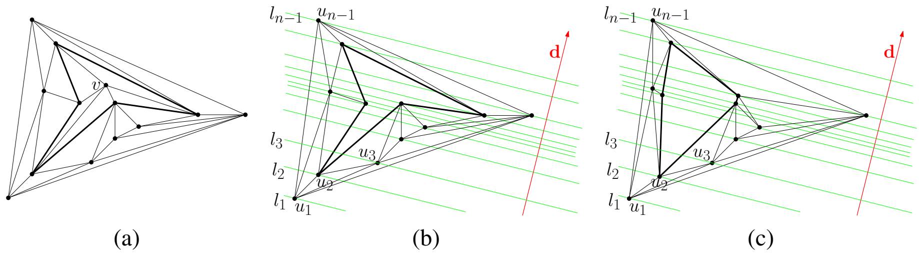


Fig. 1. (a) Straight-line planar drawing Γ of G. (b) Straight-line level planar drawing Γ' of $(G', \boldsymbol{d}, L', \gamma')$. (c) Convex straight-line level planar drawing Γ'_M of $(G', \boldsymbol{d}, L', \gamma')$.

3.1 Fast Convexifier

Consider a straight-line planar drawing Γ of an n-vertex maximal plane graph G, for some $n \geq 4$. Let v be a quasi-contractible internal vertex of G and let C_v be the cycle of G induced by the neighbors of v. See Fig. 1(a). In this section we show an algorithm, that we call FAST CONVEXIFIER, morphing Γ into a straight-line planar drawing Γ_M of G in which C_v is convex with a single unidirectional morphing step.

Let G' be the $(n-1)$-vertex plane graph obtained by removing v and its incident edges from G. Also, let Γ' be the straight-line planar drawing of G' obtained by removing v and its incident edges from Γ. As v is quasi-contractible, we have the following.

Lemma 6. *Graph G' is 3-connected.*

Consider the polygon Q_v representing C_v in Γ and in Γ'. By Lemma 2, Q_v is $\boldsymbol{d}$-monotone, for some oriented straight line $\boldsymbol{d}$. Slightly perturb the slope of $\boldsymbol{d}$ so that no line through two vertices of G in Γ is perpendicular to $\boldsymbol{d}$. If the perturbation is small enough, then Q_v is still $\boldsymbol{d}$-monotone. Denote by $u_1, \dots, u_{n-1}$ the vertices of G' ordered according to their projection on $\boldsymbol{d}$. For $1 \leq i \leq n-1$, denote by l_i the line through u_i orthogonal to $\boldsymbol{d}$. Let $L' = \{l_1, \dots, l_{n-1}\}$; note that the lines in L' are parallel and distinct. Let γ' be the function that maps u_i to l_i, for $1 \leq i \leq n-1$. See Fig. 1(b).

Lemma 7. $(G', \boldsymbol{d}, L', \gamma')$ *is a hierarchical plane st-graph.*

Proof: By construction, Γ' is a straight-line level planar drawing of $(G', \boldsymbol{d}, L', \gamma')$, hence $(G', \boldsymbol{d}, L', \gamma')$ is a hierarchical plane graph. Further, every polygon delimiting a face of G' in Γ' is $\boldsymbol{d}$-monotone. This is true for Q_v by construction and for every other polygon Q_i delimiting a face of G' in Γ' by Lemma 1, given that Q_i is a triangle and hence it is convex. Since every polygon delimiting a face of G' in Γ' is $\boldsymbol{d}$-monotone, every face of G' is an st-face, hence $(G', \boldsymbol{d}, L', \gamma')$ is a hierarchical plane st-graph. □

By Lemmata 6 and 7, $(G', \boldsymbol{d}, L', \gamma')$ is a 3-connected hierarchical plane st-graph. By Theorem 1, a convex straight-line level planar drawing Γ'_M of $(G', \boldsymbol{d}, L', \gamma')$ exists. Denote by Q_v^M the convex polygon representing C_v in Γ'_M. See Fig. 1(c).

Denote by r and s the minimum and the maximum index such that u_r and u_s belong to C_v, respectively. Denote by $l(v)$ the line through v orthogonal to $\boldsymbol{d}$ in Γ. If $l(v)$

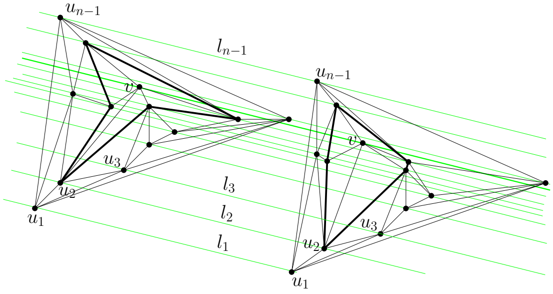


Fig. 2. Morphing Γ into a straight-line planar drawing Γ_M of G in which the polygon Q_v^M representing C_v is convex. The thick line parallel to l_i is $l(v)$.

were contained in the half-plane delimited by l_r and not containing l_s, then v would not lie inside Q_v in Γ, as the projection of every vertex of Q_v on $\boldsymbol{d}$ would follow the projection of v on $\boldsymbol{d}$. Analogously, $l(v)$ is not contained in the half-plane delimited by l_s and not containing l_r. It follows that $l(v)$ is "in-between" l_r and l_s, that is, $l(v)$ lies in the strip defined by l_r and l_s. Construct a straight-line planar drawing Γ_M of G from Γ'_M by placing v on any point at the intersection of $l(v)$ and the interior of Q_v^M. Such an intersection is always non-empty, given that l_r and l_s have non-empty intersection with Q_v^M, given that $l(v)$ is in-between l_r and l_s, and given that Q_v^M is a convex polygon.

Algorithm FAST CONVEXIFIER consists of a single linear morph $\langle \Gamma, \Gamma_M \rangle$ transforming Γ into Γ_M. See Fig. 2. Note that the polygon Q_v^M representing C_v in Γ_M is convex. Also, let γ be the function that maps v to $l(v)$ and u_i to l_i, for $1 \leq i \leq n-1$. We have that Γ and Γ_M are left-to-right equivalent straight-line level planar drawings of $(G, \boldsymbol{d}, L' \cup \{l(v)\}, \gamma)$, hence, by Lemma 5, $\langle \Gamma, \Gamma_M \rangle$ is unidirectional and planar.

3.2 Contractibility Creator

We now describe algorithm CONTRACTIBILITY CREATOR, which receives a straight-line planar drawing Γ of a plane graph G, a quasi-contractible vertex v of G, and a neighbor x of v, and returns a planar unidirectional morph with $O(1)$ morphing steps transforming Γ into a straight-line planar drawing Γ' of G in which v is x-contractible.

Denote by $u_1, \dots, u_k$ the clockwise order of the neighbors of v. If $k = 1$, then v is x-contractible in Γ, hence algorithm CONTRACTIBILITY CREATOR returns $\Gamma' = \Gamma$.

If $k \geq 2$, consider any pair of consecutive neighbors of v, say u_i and u_{i+1} (where $u_{k+1} = u_1$). See Fig. 3(a). If edge (u_i, u_{i+1}) belongs to G, then cycle (u_i, v, u_{i+1}) delimits a face of G, given that v is quasi-contractible. Otherwise, we aim at morphing Γ into a straight-line planar drawing of G where a dummy edge (u_i, u_{i+1}) can be introduced while maintaining planarity and while ensuring that cycle (u_i, v, u_{i+1}) delimits a face of the augmented graph $G \cup \{(u_i, u_{i+1})\}$. (This insertion might not be performed directly in Γ, see Fig. 3(b).) The required morphing is constructed as follows:

(Step 1) We add two dummy vertices r and r', and six dummy edges (r, v), (r, u_i), (r, u_{i+1}), (r', u_i), (r', u_{i+1}), and (r, r') to Γ and G, obtaining a straight-line planar

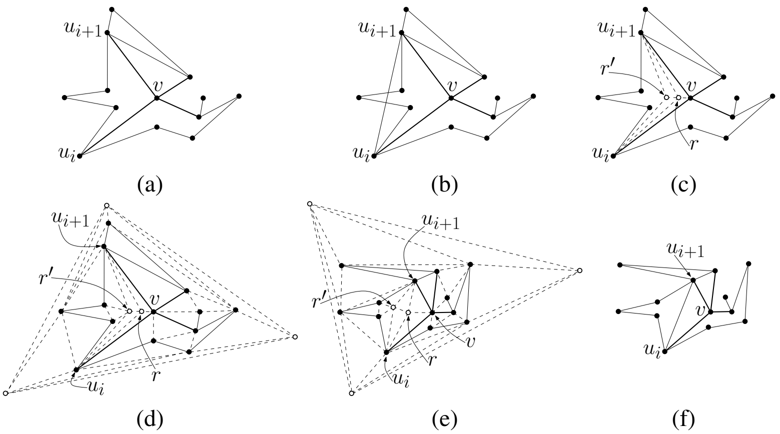


Fig. 3. (a) Drawing Γ of G. (b) Drawing (u_i, u_{i+1}) in Γ might cause a crossing. (c) Drawing Γ^+ of G^+. (d) Drawing Γ^* of G^*. (e) Drawing Γ_M^* of G^*. (f) Drawing Γ_M of $G \cup \{(u_i, u_{i+1})\}$.

drawing Γ^+ of a plane graph G^+, in such a way that Γ^+ is planar and cycles (v, r, u_i), (v, r, u_{i+1}), (r', r, u_i), and (r', r, u_{i+1}) delimit faces of G^+. See Fig. 3(c). **(Step 2)** We add dummy vertices and edges to Γ^+ and G^+, obtaining a straight-line planar drawing Γ^* of a graph G^*, in such a way that Γ^* is planar, that G^* is a maximal planar graph, and that edges (u_i, u_{i+1}) and (r', v) do not belong to G^*. Observe that r is a quasi-contractible vertex of G^*. See Fig. 3(d). **(Step 3)** We apply algorithm FAST CONVEXIFIER to morph Γ^* with one unidirectional morphing step into a straight-line planar drawing Γ_M^* of G^* such that the polygon of the neighbors of r is convex. See Fig. 3(e). **(Step 4)** We remove from Γ_M^* all the dummy vertices and edges that belong to G^* and do not belong to G, and we add edge (u_i, u_{i+1}) to Γ_M^* and G, obtaining a straight-line planar drawing Γ_M of graph $G \cup \{(u_i, u_{i+1})\}$. See Fig. 3(f).

If $k = 2$, then after the above described algorithm is performed, we have that v is x-contractible in $\Gamma' = \Gamma_M$, both if $x = u_1$ or if $x = u_2$, given that (v, u_1, u_2) delimits a face of $G \cup \{(u_1, u_2)\}$. If $3 \leq k \leq 5$, then the above described algorithm is repeated at most k times (namely once for each pair of consecutive neighbors of v that are not adjacent in G), at each time inserting an edge between a distinct pair of consecutive neighbors of v. Eventually, we obtain a straight-line planar drawing Φ of plane graph $G \cup \{(u_1, u_2), (u_2, u_3), (u_3, u_4), (u_4, u_5), (u_5, u_1)\}$ in which v is quasi-contractible. Then we add dummy vertices and edges to Φ, obtaining a straight-line planar drawing Σ of a graph H, in such a way that H is a maximal planar graph and that v is quasi-contractible in Σ. We apply algorithm FAST CONVEXIFIER to morph Σ with one unidirectional morphing step into a straight-line planar drawing Ψ of H such that the polygon of the neighbors of v is convex. Hence, v is contractible onto any of its neighbors in Ψ. Then, we remove the edges of H not in G, obtaining a straight-line planar drawing Γ' of G in which v is contractible onto any of its neighbors; hence, v is x-contractible in Γ'. Finally, observe that Γ' is obtained from Γ in at most $k + 1 \leq 6$ unidirectional morphing steps.

3.3 The Algorithm

We now describe an algorithm to construct a pseudo-morph $\mathcal{P}$ with $O(n)$ steps between any two straight-line planar drawings Γ_s and Γ_t of the same n-vertex plane graph G.

The algorithm works by induction on n. If $n = 1$, then $\mathcal{P}$ consists of a single unidirectional morphing step transforming Γ_s into Γ_t. If $n \geq 2$, then let v be a quasi-contractible vertex of G, which exists by Lemma 3, and let x be any neighbor of v. Let M_s and M_t be the planar unidirectional morphs with $O(1)$ morphing steps produced by algorithm CONTRACTIBILITY CREATOR transforming Γ_s and Γ_t into straight-line planar drawings Γ_s^x and Γ_t^x of G, respectively, such that v is x-contractible both in Γ_s^x and in Γ_t^x. Let G' be the $(n-1)$-vertex plane graph obtained by contracting v onto x in G, and let Γ_s' and Γ_t' be the straight-line planar drawings of G' obtained from Γ_s^x and Γ_t^x, respectively, by contracting v onto x. Further, let $\mathcal{P}'$ be the inductively constructed pseudo-morph between Γ_s' and Γ_t'. Then, pseudo-morph $\mathcal{P}$ is defined as unidirectional morph M_s transforming Γ_s into Γ_s^x, followed by the contraction of v onto x in Γ_s^x, followed by the pseudo-morph $\mathcal{P}'$ between Γ_s' and Γ_t', followed by the uncontraction of v from x into Γ_t^x, followed by the unidirectional morph M_t^{-1} transforming Γ_t^x into Γ_t. Observe that $\mathcal{P}$ has a number of steps which is a constant plus the number of steps of $\mathcal{P}'$. Hence, $\mathcal{P}$ consists of $O(n)$ steps. A unidirectional planar morph M between Γ_s and Γ_t can be constructed with a number of morphing steps equal to the number of steps of $\mathcal{P}$, by Lemma 4. This proves the following:

Theorem 2. *Let Γ_s and Γ_t be any two straight-line planar drawings of the same n-vertex plane graph G. There exists an algorithm to construct a planar unidirectional morph with $O(n)$ morphing steps transforming Γ_s into Γ_t.*

4 A Lower Bound

In this section we show two straight-line planar drawings Γ_s and Γ_t of an n-vertex path $P = (v_1, \dots, v_n)$, and we prove that any planar morph M between Γ_s and Γ_t requires $\Omega(n)$ morphing steps. In order to simplify the description, we consider each edge $e_i = (v_i, v_{i+1})$ as oriented from v_i to v_{i+1}, for $i = 1, \dots, n-1$.

Drawing Γ_s (see Fig. 4(a)) is such that all the vertices of P lie on a horizontal straight line with v_i to the left of v_{i+1}, for each $i = 1, \dots, n-1$. Drawing Γ_t (see Fig. 4(b)) is such that: (a) for each $i = 1, \dots, n-1$ with $i \bmod 3 \equiv 1$, e_i is horizontal with v_i to the left of v_{i+1}; (b) for each $i = 1, \dots, n-1$ with $i \bmod 3 \equiv 2$, e_i is parallel to line $y = \tan(\frac{2\pi}{3})x$ with v_i to the right of v_{i+1}; and (c) for each $i = 1, \dots, n-1$ with $i \bmod 3 \equiv 0$, e_i is parallel to line $y = \tan(-\frac{2\pi}{3})x$ with v_i to the right of v_{i+1}.

Let $M = \langle \Gamma_s = \Gamma_1, \dots, \Gamma_x = \Gamma_t \rangle$ be any planar morph transforming Γ_s into Γ_t.

For $i = 1, \dots, n$ and $j = 1, \dots, x$, we denote by v_i^j the point where vertex v_i is placed in Γ_j and by e_i^j the directed straight-line segment representing edge e_i in Γ_j.

For $1 \leq j \leq x-1$, we define the *rotation* ρ_i^j of e_i around v_i during the morphing step $\langle \Gamma_j, \Gamma_{j+1} \rangle$ as follows (see Figs. 5(a)–(b)). Translate e_i at any time instant of $\langle \Gamma_j, \Gamma_{j+1} \rangle$ so that v_i stays fixed at a point a during the entire morphing step. After this translation, the morph between e_i^j and e_i^{j+1} is a rotation of e_i around a (where e_i might vary its

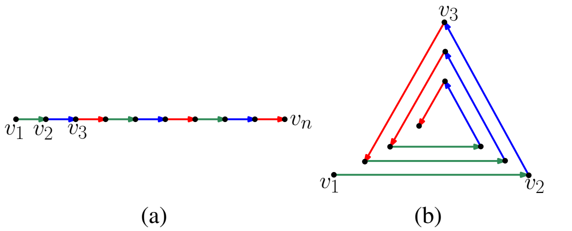


Fig. 4. Drawings Γ_s (a) and Γ_t (b)

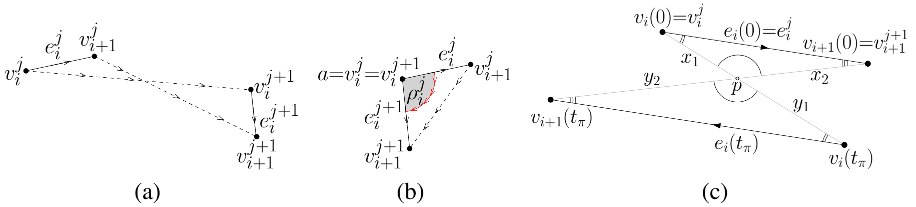


Fig. 5. (a) Morph between e_i^j and e_i^{j+1}. (b) Translation of the positions of e_i during $\langle \Gamma_j, \Gamma_{j+1} \rangle$, resulting in e_i spanning an angle ρ_i^j around v_i. (c) Illustration for the proof of Lemma 8.

length during $\langle \Gamma_j, \Gamma_{j+1} \rangle$) spanning an angle ρ_i^j, where we assume $\rho_i^j > 0$ if the rotation is counter-clockwise, and $\rho_i^j < 0$ otherwise. We have the following.

Lemma 8. *For each $j = 1, \dots, x-1$ and $i = 1, \dots, n-1$, we have $|\rho_i^j| < \pi$.*

Proof: Assume, for a contradiction, that $|\rho_i^j| \geq \pi$, for some $1 \leq j \leq x-1$ and $1 \leq i \leq n-1$. Also assume, w.l.o.g., that the morphing step $\langle \Gamma_j, \Gamma_{j+1} \rangle$ happens between time instants $t = 0$ and $t = 1$. For any $0 \leq t \leq 1$, denote by $v_i(t)$, $v_{i+1}(t)$, $e_i(t)$, and $\rho_i^j(t)$ the position of v_i, the position of v_{i+1}, the drawing of e_i, and the rotation of e_i around v_i at time instant t, respectively. Note that $v_i(0) = v_i^j$, $v_{i+1}(0) = v_{i+1}^j$, $e_i(0) = e_i^j$, $\rho_i^j(0) = 0$, and $\rho_i^j(1) = \rho_i^j$. Since a morph is a continuous transformation and since $|\rho_i^j| \geq \pi$, there exists a time instant t_π with $0 < t_\pi \leq 1$ such that $|\rho_i^j(t_\pi)| = \pi$.

We prove that there exists a time instant t_r with $0 < t_r \leq t_\pi$ in which $v_i(t)$ and $v_{i+1}(t)$ coincide, thus contradicting the assumption that morph $\langle \Gamma_j, \Gamma_{j+1} \rangle$ is planar.

Since $|\rho_i^j(t_\pi)| = \pi$, it follows that $e_i(t_\pi)$ is parallel to $e_i(0)$ and oriented in the opposite way. This easily leads to conclude that t_r exists if $e_i(t_\pi)$ and $e_i(0)$ are aligned. Otherwise, the straight-line segments $\overline{v_i(0)v_i(t_\pi)}$ and $\overline{v_{i+1}(0)v_{i+1}(t_\pi)}$ meet in a point p. Refer to Fig. 5(c). Let $x_1 = |\overline{pv_i(0)}|$, $x_2 = |\overline{pv_{i+1}(0)}|$, $y_1 = |\overline{pv_i(t_\pi)}|$, and $y_2 = |\overline{pv_{i+1}(t_\pi)}|$. By the similarity of triangles $(v_i(0), p, v_{i+1}(0))$ and $(v_i(t_\pi), p, v_{i+1}(t_\pi))$, we have $\frac{x_1}{y_1} = \frac{x_2}{y_2}$ and hence $\frac{x_1}{x_1+y_1} = \frac{x_2}{x_2+y_2}$. Thus, $v_i(\frac{x_1}{x_1+y_1}t_\pi)$ and $v_{i+1}(\frac{x_1}{x_1+y_1}t_\pi)$ are coincident with p. This contradiction proves the lemma. □

For $j = 1, \dots, x-1$, we denote by M_j the subsequence $\langle \Gamma_1, \dots, \Gamma_{j+1} \rangle$ of M; also, for $i = 1, \dots, n-1$, we define the *total rotation* $\rho_i(M_j)$ of edge e_i around v_i during morph M_j as $\rho_i(M_j) = \sum_{m=1}^{j} \rho_i^m$.

We will show in Lemma 10 that there exists an edge e_i, for some $1 \leq i \leq n-1$, whose total rotation $\rho_i(M_{x-1}) = \rho_i(M)$ is $\Omega(n)$. In order to do that, we first analyze the relationship between the total rotation of two consecutive edges of P.

Lemma 9. *For each $j = 1, \dots, x-1$ and for each $i = 1, \dots, n-2$, we have that $|\rho_{i+1}(M_j) - \rho_i(M_j)| < \pi$.*

Proof: Suppose, for a contradiction, that $|\rho_{i+1}(M_j) - \rho_i(M_j)| \geq \pi$ for some $1 \leq j \leq x-1$ and $1 \leq i \leq n-2$. Assume that j is minimal under this hypothesis. Since each vertex moves continuously during M_j, there exists an intermediate drawing Γ^* of P, occurring during morphing step $\langle \Gamma_j, \Gamma_{j+1} \rangle$, such that $|\rho_{i+1}(M^*) - \rho_i(M^*)| = \pi$, where $M^* = \langle \Gamma_1, \dots, \Gamma_j, \Gamma^* \rangle$ is the morph obtained by concatenating M_{j-1} with the morphing step transforming Γ_j into Γ^*. Recall that in Γ_1 edges e_i and e_{i+1} lie on the same straight line and have the same orientation. Then, since $|\rho_{i+1}(M^*) - \rho_i(M^*)| = \pi$, in Γ^* edges e_i and e_{i+1} are parallel and have opposite orientations. Also, since edges e_i and e_{i+1} share vertex v_{i+1}, they lie on the same line. This implies that such edges overlap, contradicting the hypothesis that M^*, M_j, and M are planar. □

We now prove the key lemma for the lower bound.

Lemma 10. *There exists an index i such that $|\rho_i(M)| \in \Omega(n)$.*

Proof: Refer to Fig. 4. For every $1 \leq i \leq n-2$, edges e_i and e_{i+1} form an angle of π radiants in Γ_s, while they form an angle of $\frac{\pi}{3}$ radiants in Γ_t. Hence, $\rho_{i+1}(M) = \rho_i(M) + \frac{2\pi}{3} + 2z_i\pi$, for some $z_i \in \mathbb{Z}$. In order to prove the lemma, it suffices to prove that $z_i = 0$, for every $i = 1, \dots, n-2$. Namely, in this case $\rho_{i+1}(M) = \rho_i(M) + \frac{2\pi}{3}$ for every $1 \leq i \leq n-2$, and hence $\rho_{n-1}(M) = \rho_1(M) + \frac{2\pi}{3}(n-2)$. This implies $|\rho_{n-1}(M) - \rho_1(M)| \in \Omega(n)$, and thus $|\rho_1(M)| \in \Omega(n)$ or $|\rho_{n-1}(M)| \in \Omega(n)$. Assume, for a contradiction, that $z_i \neq 0$, for some $1 \leq i \leq n-2$. If $z_i > 0$, then $\rho_{i+1}(M) \geq \rho_i(M) + \frac{8\pi}{3}$; further, if $z_i < 0$, then $\rho_{i+1}(M) \leq \rho_i(M) - \frac{4\pi}{3}$. Since each of these inequalities contradicts Lemma 9, the lemma follows. □

We are now ready to state the main theorem of this section.

Theorem 3. *There exists two straight-line planar drawings Γ_s and Γ_t of an n-vertex path P such that any planar morph between Γ_s and Γ_t requires $\Omega(n)$ morphing steps.*

Proof: The two drawings Γ_s and Γ_t of path $P = (v_1, \dots, v_n)$ are those illustrated in Fig. 4. By Lemma 10, there exists an edge e_i of P, for some $1 \leq i \leq n-1$, such that $|\sum_{j=1}^{x-1} \rho_i^j| \in \Omega(n)$. Since, by Lemma 8, we have that $|\rho_i^j| < \pi$ for each $j = 1, \dots, x-1$, it follows that $x \in \Omega(n)$. This concludes the proof of the theorem. □

5 Conclusions

In this paper we presented an algorithm to construct a planar morph between two planar straight-line drawings of the same n-vertex plane graph in $O(n)$ morphing steps.

We also proved that this bound is tight (note that our lower bound holds for any morphing algorithm in which the vertex trajectories are polynomial functions of constant degree).

In our opinion, the main challenge in this research area is the one of designing algorithms to construct planar morphs between straight-line planar drawings with good resolution and within polynomial area (or to prove that no such algorithm exists). In fact, the algorithm we presented, as well as other algorithms known at the state of the art [1, 3, 5, 14], construct intermediate drawings in which the ratio between the lengths of the longest and of the shortest edge is exponential. Guaranteeing good resolution and small area seems to be vital for making a morphing algorithm of practical utility.

Finally, we would like to mention an original problem that generalizes the one we solved in this paper and that we repute very interesting. Let Γ_s and Γ_t be two straight-line drawings of the same (possibly non-planar) topological graph G. Does a morphing algorithm exist that morphs Γ_s into Γ_t and that preserves the topology of the drawing at any time instant? A solution to this problem is not known even if we allow the trajectories followed by the vertices to be of arbitrary complexity.

References

1. Alamdari, S., Angelini, P., Chan, T.M., Di Battista, G., Frati, F., Lubiw, A., Patrignani, M., Roselli, V., Singla, S., Wilkinson, B.T.: Morphing planar graph drawings with a polynomial number of steps. In: Khanna, S. (ed.) SODA 2013, pp. 1656–1667. SIAM (2013)
2. Angelini, P., Da Lozzo, G., Di Battista, G., Frati, F., Patrignani, M., Roselli, V.: Morphing planar graph drawings optimally. CoRR abs/1402.4364 (2014)
3. Angelini, P., Frati, F., Patrignani, M., Roselli, V.: Morphing planar graph drawings efficiently. In: Wismath, S., Wolff, A. (eds.) GD 2013. LNCS, vol. 8242, pp. 49–60. Springer, Heidelberg (2013)
4. Barrera-Cruz, F., Haxell, P., Lubiw, A.: Morphing planar graph drawings with unidirectional moves. In: Mexican Conference on Discr. Math. and Comput. Geom. (2013)
5. Cairns, S.S.: Deformations of plane rectilinear complexes. American Math. Monthly 51, 247–252 (1944)
6. Chiba, N., Yamanouchi, T., Nishizeki, T.: Linear algorithms for convex drawings of planar graphs. In: Bondy, J.A., Murty, U.S.R. (eds.) Progress in Graph Theory, pp. 153–173. Academic Press, New York (1984)
7. Erten, C., Kobourov, S.G., Pitta, C.: Intersection-free morphing of planar graphs. In: Liotta, G. (ed.) GD 2003. LNCS, vol. 2912, pp. 320–331. Springer, Heidelberg (2004)
8. Friedrich, C., Eades, P.: Graph drawing in motion. J. Graph Alg. Appl. 6(3), 353–370 (2002)
9. Gotsman, C., Surazhsky, V.: Guaranteed intersection-free polygon morphing. Computers & Graphics 25(1), 67–75 (2001)
10. Grunbaum, B., Shephard, G.: The geometry of planar graphs. Camb. Univ. Pr. (1981)
11. Hong, S.H., Nagamochi, H.: Convex drawings of hierarchical planar graphs and clustered planar graphs. J. Discrete Algorithms 8(3), 282–295 (2010)
12. Surazhsky, V., Gotsman, C.: Controllable morphing of compatible planar triangulations. ACM Trans. Graph 20(4), 203–231 (2001)
13. Surazhsky, V., Gotsman, C.: Intrinsic morphing of compatible triangulations. Internat. J. of Shape Model. 9, 191–201 (2003)
14. Thomassen, C.: Deformations of plane graphs. J. Comb. Th. Ser. B 34(3), 244–257 (1983)
15. Thomassen, C.: Plane representations of graphs. In: Bondy, J.A., Murty, U.S.R. (eds.) Progress in Graph Theory, pp. 43–69. Academic Press, New York (1984)

Incremental Algorithm for Maintaining DFS Tree for Undirected Graphs*

Surender Baswana** and Shahbaz Khan***

Department of CSE, IIT Kanpur
Kanpur, India
{sbaswana,shahbazk}@cse.iitk.ac.in
http://www.cse.iitk.ac.in

Abstract. Depth First Search (DFS) tree is a fundamental data structure for graphs used in solving various algorithmic problems. However, very few results are known for maintaining DFS tree in a dynamic environment - insertion or deletion of edges. The only non-trivial result for this problem is by Franciosa et al. [4]. They showed that, for a directed acyclic graph on n vertices, a DFS tree can be maintained in $O(n)$ amortized time per edge insertion. They stated it as an open problem to maintain a DFS tree dynamically in an undirected graph or general directed graph.

We present the first algorithm for maintaining a DFS tree for an undirected graph under insertion of edges. For processing any arbitrary online sequence of edge insertions, this algorithm takes total $O(n^2)$ time.

Keywords: dynamic·incremental·undirected graph·depth first search.

1 Introduction

Depth First Search (DFS) is a well known graph traversal technique. This technique has been reported to be introduced by Charles Pierre Trémaux, a 19th-century French mathematician who used it for solving mazes. However, it was Tarjan, who in his seminal work [9], demonstrated the power of DFS traversal for solving various fundamental graph problems, namely, topological sorting, connected components, biconnected components, strongly-connected components, etc.

DFS traversal is a recursive algorithm to traverse a graph. This traversal produces a rooted spanning tree (or forest), called DFS tree (forest). Let $G = (V, E)$ be an undirected graph on $n = |V|$ vertices and $m = |E|$ edges. It takes $O(m + n)$ time to perform a DFS traversal and generate its DFS tree (forest).

[*] Full version of this paper is available at http://www.cse.iitk.ac.in/users/sbaswana/Papers-published/Incr-DFS.pdf

[**] This research was partially supported by the Indo-German Max Planck Center for Computer Science (IMPECS).

[***] This research was partially supported by Google India under the Google India PhD Fellowship Award.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 138–149, 2014.

A DFS tree, say T, imposes the following relation on each non-tree edge $(x, y) \in E \backslash T$.

$\mathcal{R}(x, y)$: Either x is an ancestor of y in T or y is an ancestor of x in T.

This elegant relation defined by a DFS tree has played the key role in solving various graph problems. Similar relations exists for the case of DFS tree in directed graphs.

Most of the graph applications in real world deal with graphs that keep changing with time. An algorithmic graph problem is modeled in the dynamic environment as an online sequence of insertion and deletion of edges. The aim is to maintain the solution of the given problem after each edge update using some clever data structure such that the time taken to update the solution after any edge update is much smaller than that of the best static algorithm. A dynamic algorithm is called an incremental algorithm if it supports only insertion of edges.

In spite of the simplicity and elegance of a DFS tree, its parallel and dynamic versions have turned out to be quite challenging. In fact, in the dynamic setting, the ordered DFS problem (where the edges are visited strictly in the order given by the adjacency list of the graph) is shown to be hard by Reif[6,7]. He showed that ordered DFS problem is a P-Complete problem. Milterson et al. [5] later proved that if dynamic version of any non-redundant P-Complete problem is updatable in $t(n)$ time, then every problem in P is updatable in $O(t(n) + \mathbf{polylog}(n))$ time. So it is highly unlikely that any $O(\mathbf{polylog}(n))$ update time algorithm exists for the ordered DFS problem. Though the ordered DFS problem is significant from the perspective of complexity theory, none of the existing algorithmic applications of DFS trees require such restrictions. Hence it is natural to address the problem of maintenance of any DFS tree in a dynamic graph.

For the case of directed acyclic graphs, Franciosa et al. [4] presented an incremental algorithm for maintaining DFS tree in $O(mn)$ total time. This is the only non-trivial result available for the dynamic DFS tree problem. Maintaining DFS tree incrementally for undirected graph (or general directed graph) was stated as an open problem by Franciosa et al. [4]. The following short discussion may help one realize the non-triviality of maintaining a DFS tree incrementally in an undirected graph.

Consider insertion of an edge (x, y). If $\mathcal{R}(x, y)$ holds, then no change in DFS tree is required. Such an edge is called a *back edge*. Otherwise, the relation $\mathcal{R}(x, y)$ does not hold for edge (x, y), and we call such an edge a *cross* edge. See Figure 1 for a better visual description. Let w be the lowest common ancestor of x and y. Let u and v be its children such that $x \in T(u)$ and $y \in T(v)$. Insertion of (x, y) violates the property of a DFS tree as follows.

Let S be the set of visited vertices when the DFS traversal reached w. Since $T(u)$ and $T(v)$ are two disjoint subtrees hanging from w, the vertices of $T(u)$ and $T(v)$ belong to disjoint connected components in the subgraph induced by $V \backslash S$. Insertion of edge (x, y) connects these components such that the vertices of $T(u) \cup T(v)$ have to hang as a single subtree from w. This implies that $T(u)$ will have to be rerooted at x and hung from y (or $T(v)$ has to be rerooted at y and

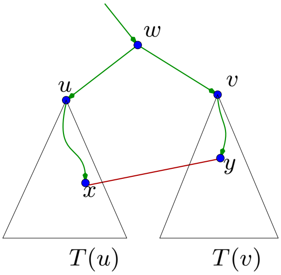


Fig. 1. (x, y) is a cross edge

hung from x). This rerooting will force restructuring of $T(u)$ because, in order to keep it as a DFS subtree, we need to preserve the relation $\mathcal{R}$ for every non-tree edges in $T(u)$. Observe that it is not obvious to perform this restructuring in an efficient manner.

We present the first incremental algorithm for maintaining a DFS tree (or DFS forest if the graph is not connected) in an undirected graph. Our algorithm takes a total of $O(n^2)$ time to process any arbitrary online sequence of edges. Hence the amortized update time per edge insertion is $\Omega(n^2/m)$, which is $O(1)$ for the case $m = \Omega(n^2)$. In addition to the $O(m + n)$ space occupied by the graph, our algorithm uses only $O(m+n)$ extra space. Moreover, excluding the standard data structures for ancestors in a rooted tree [1,2], our algorithm employs very simple data structures. These salient features make this algorithm an ideal candidate for practical applications.

1.1 Related Work

Breadth first search (BFS) is another popular graph traversal algorithm. A BFS tree can be maintained incrementally in $O(mn)$ time [3], improving which is shown to be hard [8].

2 Preliminaries

Given an undirected graph $G = (V, E)$ on $n = |V|$ vertices and $m = |E|$ edges, the following notations will be used throughout the paper.

- T : A DFS tree of G at any particular time.
- r : Root of tree T.
- $par(v)$: Parent of v in T.
- $P(u, v)$: Path between u and v in T.
- $\textsc{level}(v)$: Level of a vertex v in T s.t. $\textsc{level}(r) = 0$ and $\textsc{level}(v) = \textsc{level}(par(v)) + 1$.
- $\textsc{level}(e)$: Level of an edge $e = (x, y)$ in T s.t. $\textsc{level}(e) = \min(\textsc{level}(x), \textsc{level}(y))$.

– $T(x)$: The subtree of T rooted at a vertex x.
– $LCA(u, v)$: The Lowest Common Ancestor of u and v in tree T.
– $LA(u, k)$: The ancestor of u at level k in tree T.

We explicitly maintain the level of each vertex during the algorithm. Since the tree grows from the root downward, a vertex u is said to be at higher level than vertex v if $\textsc{level}(u) < \textsc{level}(v)$. Similar notion is used for edges.

The DFS tree T maintains the following data structure at each stage.

– Each vertex v keeps a pointer to $par(v)$ and $par(r) = r$.
– T is also stored as an adjacency list for traversing $T(v)$ from v.
– Each vertex v keeps a list $\mathcal{B}(v)$ which consists of all those back edges that originate from $T(v)$ and terminate at $par(v)$. This, apparently uncommon and perhaps unintuitive, way of keeping the back edges leads to efficient implementation of the rerooting procedure.

Our algorithm uses the following results for the dynamic version of the Lowest Common Ancestor (LCA) and the Level Ancestors (LA) problems.

Theorem 1 (Cole and Hariharan 2005[2]). *There exists a dynamic data structure for a rooted tree T that uses linear space and can report $LCA(x, y)$ in $O(1)$ time for any two vertices $x, y \in T$. The data structure supports insertion or deletion of any leaf node in $O(1)$ time.*

Theorem 2 (Alstrup and Holm 2000[1]). *There exists a dynamic data structure for a rooted tree T that uses linear space and can report $LA(u, k)$ in $O(1)$ time for any vertex $u \in T$. The data structure supports insertion of any leaf node in $O(1)$ time.*

The data structure for Level Ancestor problem can be easily extended to handle deletion of a leaf node in amortized $O(1)$ time using the standard technique of periodic rebuilding.

If the graph is not connected, the aim would be to maintain a DFS tree for each connected component. However, our algorithm, at each stage, maintains a single DFS tree which stores the entire forest of these DFS trees as follows. We add a dummy vertex s to the graph in the beginning and connect it to all the vertices. We maintain a DFS tree of this augmented graph rooted at s. It can be easily seen that the subtrees rooted at children of s correspond to DFS trees of various connected components of the original graph.

3 Overview of the Algorithm

Our algorithm is based on two principles. The first principle, called *monotonic fall* of vertices, ensures that the level of a vertex may only fall or remain same as the edges are inserted. Consider insertion of a cross edge (x, y) as shown in Figure 1. In order to ensure monotonic fall, the following strategy is used. If $\textsc{level}(y) \le \textsc{level}(x)$, then we reroot $T(v)$ at y and hang it through edge (x, y)

(and vice versa if LEVEL$(y) >$ LEVEL(x)). This strategy surely leads to fall of level of x (or y). However, this rerooting has to be followed by transformation of $T(v)$ into a DFS tree. An obvious, but inefficient way, to do this transformation is to perform a fresh DFS traversal on $T(v)$ from x (as done by Franciosa et al. [4]). We are able to avoid this costly step using our second principle called *minimal restructuring*. Following this principle, only a path of the subtree $T(v)$ is reversed and as a result major portion of the original DFS tree remains intact. Furthermore, this principle also facilitates monotonic fall of all vertices of $T(v)$. The rerooting procedure based on this principle is described and analyzed in the following section.

Our algorithm updates DFS tree upon insertion of any cross edge as follows. Firstly, we carry out rerooting based on the two principles mentioned above. As a result, many back edges now potentially become cross edges. All these edges are collected and virtually (re)-inserted back to the graph, and processed as *fresh* insertions. This simple iterative algorithm, when analyzed in a straightforward manner, has a time complexity $O(mn)$. However, using a more careful analysis, it can be shown that its time complexity is $O(n^{3/2}m^{1/2})$, which is strictly sub-cubic. In order to improve the time complexity further, we process the pool of cross edges in a more structured manner. In particular, we process the highest cross edge first. This leads to our final algorithm that achieves $O(n^2)$ time complexity for any arbitrary sequence of edge insertions.

4 Rerooting a Subtree

Consider insertion of an edge (x, y) which happens to be a cross edge with respect to the DFS tree T. Let w be LCA of x and y, and let u and v be the two children of w such that $x \in T(u)$ and $y \in T(v)$. See Figure 2 for a visual description. As discussed before, updating the DFS tree upon insertion of the cross edge (x, y) entails rerooting of subtree $T(v)$ at y and hanging it from x. We now describe an efficient rerooting procedure for this task based on *minimal restructuring* principle.

The underlying idea of *minimal restructuring* principle is to preserve the current tree structure as much as possible. Consider the path $P(y, v) = \langle z_1(= y), z_2, \ldots, z_k(= v)\rangle$. This path appears from v to y in the rooted tree T. Rerooting reverses this path in T so that it starts at y and terminates at v. In order to see how this reversal affects the DFS structure, let us carefully examine $T(v)$.

The subtree $T(v)$ can be visualized as a collection of disjoint trees joined through the path $P(v, y)$ as follows. Let T_1 denote the subtree $T(y)$ and let T_2 denote the subtree $T(z_2)\backslash T(z_1)$. In general, T_i denote the subtree $T(z_i)\backslash T(z_{i-1})$. Upon reversing the path $P(v, y)$, notice that each subtree remains intact but their ordering gets reversed. Furthermore, level of each subtree T_i surely falls. (see Figure 2). Let us find the consequence of reversing $P(v, y)$ on all those back edges whose at least one endpoint belongs to $T(v)$. Observe that the back edges which originate as well as terminate within the same T_i continue to remain as back edges since tree T_i remains intact. Likewise, any back edge from these

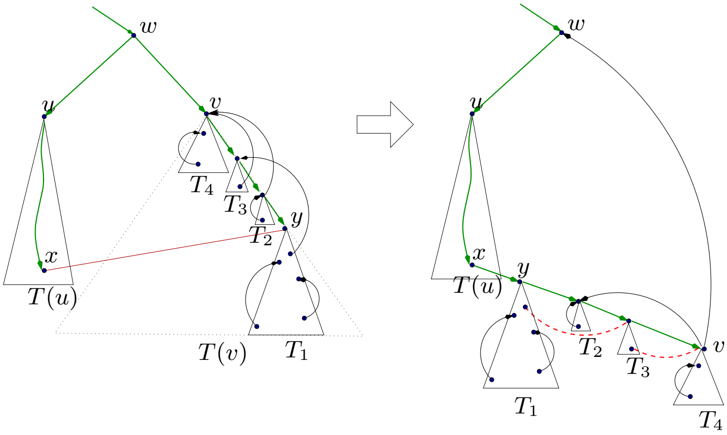


Fig. 2. Rerooting the tree $T(v)$ at y and hanging it from x. Notice that some back edges may become cross edges (shown dotted) due to this rerooting.

subtrees which terminates at any ancestor of v also continue to remain as a back edge. However, the back edges originating in $T(v)$ and terminating on w, which were earlier stored in $\mathcal{B}(v)$, will now have to be stored in $\mathcal{B}(u)$ (recall the definition of $\mathcal{B}$). Also notice that the tree edge (w, v) now becomes a back edge and has to be added to $\mathcal{B}(u)$. The remaining back edges are only those which originate from some T_i and terminate at some $z_j, j > i$. All these edges are present in $\mathcal{B}(z_{j-1})$. Some of these back edges may become cross edges due to the reversal of $P(v, y)$. Their presence violates the DFS property (relation $\mathcal{R}$) of the new tree. We just collect and remove these edges temporarily from the graph. In summary, our rerooting algorithm just does the following: It traverses the path $P(v, y)$ from y to v, collects $\mathcal{B}(z_j)$ for each $1 \leq j < k$, and reverses the path $P(v, y)$. The pseudo code of the rerooting process is described in Procedure `Reroot`.

The following lemma holds based on the above discussion.

Lemma 1. *Tree T at the end of Procedure* `Reroot`*(v, x, y) is a DFS tree for the graph $(V, E \backslash \mathcal{E}_R)$.*

We introduce some terminology to facilitate compact and clean reference to the entities of rerooting procedure. The lower and higher end vertices x and y of the inserted edge (x, y) are called *prime* and *conjugate* vertices respectively. Notice that the restructured subtree now hangs from the *prime* vertex. We define *prime* path as the path from the prime vertex x to u and *conjugate* path as the path from conjugate vertex y to v, where u and v are children of $LCA(x, y)$ s.t. $x \in T(u)$ and $y \in T(v)$.

Each vertex of subtree $T(v)$ suffers a fall in its level due to `Reroot`(v, x, y). We shall now calculate this fall exactly. Let $\Delta = \text{LEVEL}(x) - \text{LEVEL}(y)$. As a result of the rerooting, y has become child of x. Hence it has suffered a fall in

Procedure Reroot(v,x,y): reroots subtree $T(v)$ at vertex y and hangs it through edge (x, y).

```
$\mathcal{E}_R \leftarrow \phi$;
$z \leftarrow y$;
$p \leftarrow x$;
$\mathcal{B}(v) \leftarrow \mathcal{B}(u) \cup \mathcal{B}(v)$;
while $z \neq par(v)$ do
    if $z \neq v$ then $\mathcal{E}_R \leftarrow \mathcal{E}_R \cup \mathcal{B}(z)$;
    $\mathcal{B}(z) \leftarrow \phi$;
    $next \leftarrow par(z)$;
    $par(z) \leftarrow p$;
    $p \leftarrow z$;
    $z \leftarrow next$;
end
Return $\mathcal{E}_R$
```

its level by $\Delta + 1$. Since $T_1 = T(y)$ and T_1 remains intact, so each vertex of T_1 suffers a fall by $\Delta+1$ levels. Consider a vertex z_i which is the root of T_i for some $i > 1$. This vertex was earlier at level $i - 1$ higher than $y(= z_1)$ and now lies at $i - 1$ level below y. Hence overall level of z_i (and hence that of every vertex of T_i) has fallen by $\Delta + 2i - 1$. This leads us to the following lemma.

Lemma 2. *Let Δ be the difference in the levels of prime and conjugate vertices before rerooting. After rerooting, the ith vertex on the conjugate path falls by $\Delta + 2i - 1$ levels.*

Let us analyze the time complexity of the Procedure `Reroot`(v, x, y). It first adds $\mathcal{B}(v)$ to $\mathcal{B}(u)$; this step takes $O(1)$ time since we are uniting two lists. Thereafter, the procedure traverses the conjugate path $P(y, v)$, and collects the edges $\mathcal{B}(z)$ for each $z \in P(y, v)\backslash\{v\}$ in $\mathcal{E}_R$.

Lemma 3. *The time complexity of the Procedure* `Reroot`*(v, x, y) is $O(k+|\mathcal{E}_R|)$, where k is the length of the conjugate path and $\mathcal{E}_R$ is the set of edges returned by the procedure.*

It follows from the rerooting procedure that any back edge getting converted to a cross edge is surely collected in $\mathcal{E}_R$. However, not all the edges of $\mathcal{E}_R$ necessarily become cross edges. In order to understand this subtle point, observe that $\mathcal{E}_R$ contains all those edges which originate from some vertex in T_i and terminate at some $z_j, i < j < k$. Consider any such edge (a, z_j), $a \in T_i$. If $a \neq z_i$ (root of T_i), then surely (a, z_j) has become a cross edge after reversal of $P(v, y)$. But if $a = z_i$, then it still remains a back edge. So we can state the following lemma which will be crucial in our final algorithm.

Lemma 4. *If an edge collected in $\mathcal{E}_R$ is a back edge with respect the modified DFS tree, then both its endpoints must belong to the conjugate path.*

5 Algorithm for Incremental DFS

Consider insertion of an edge (t, z). In order to update the DFS tree, our algorithm maintains a set $\mathcal{E}$ of edges which is initialized as $\{(t, z)\}$. The algorithm then processes the set $\mathcal{E}$ iteratively as follows. In each iteration, an edge (say (x, y)) is extracted from $\mathcal{E}$ using Procedure Choose. If the edge is a back edge, the edge is inserted in the set of back edges $\mathcal{B}$ accordingly and no processing is required. If (x, y) is a cross edge, it is processed as follows. Let w be LCA of x and y, and let v be the child of w such that y is present in subtree $T(v)$. Without loss of generality, let $\text{LEVEL}(x) \geq \text{LEVEL}(y)$. Procedure Reroot$(v, x, y)$ is invoked which reroots subtree $T(v)$ at y and returns a set of edges collected during the procedure. All these edges are extracted from E and added to $\mathcal{E}$. This completes one iteration of the algorithm. The algorithm finishes when $\mathcal{E}$ becomes empty. The correctness of the algorithm follows directly from the following invariant which is maintained throughout the algorithm:

Invariant. T is DFS tree for the subgraph $(V, E \backslash \mathcal{E})$.

Algorithm 1. Processing insertion of an edge (t, z)

```
$\mathcal{E} \leftarrow \{(t, z)\}$ ;            /* $\mathcal{E}$ is a set of edges to be inserted. */
while $\mathcal{E} \neq \phi$ do
    $(x, y) \leftarrow$ Choose($\mathcal{E}$) ;              /* Here LEVEL$(x) \geq$ LEVEL$(y)$. */
    $w \leftarrow$ LCA$(x, y)$;
    $v \leftarrow$ LA$(y,$ LEVEL$(w) + 1)$ ; /* $v$ is $y$'s ancestor & $w$'s child */
    if $w \neq y$ then                          /* $(x, y)$ is a cross edge. */
        $\mathcal{E} \leftarrow \mathcal{E} \cup$ Reroot$(v, x, y)$;
    end
end
```

Procedure Choose($\mathcal{E}$): Chooses and returns an edge from $\mathcal{E}$.

```
Remove an arbitrary edge $(x, y)$ from $\mathcal{E}$.
Return $(x, y)$.
```

5.1 Analysis

The computation cost of collecting and processing each edge $e \in \mathcal{E}$ can be associated with the rerooting event in which it was collected. Thus the computation time spent by the incremental algorithm in processing any sequence of edge insertions is of the order of the time spent in all the rerooting calls invoked. Furthermore, using Lemma 3 we know that the time complexity of a single rerooting call is of the order of the number of edges in $\mathcal{E}_R$ that were collected during that rerooting (the cost of the first term mentioned in Lemma 3 can be associated with the fall of vertices on the conjugate path). Therefore, in order to calculate the time complexity of the algorithm, it suffices to count all the edges collected during various rerooting calls. Note that this count can be much larger than $O(m)$ because an edge can appear multiple times in $\mathcal{E}$ during the algorithm. It follows from Lemma 2 that whenever an edge is collected during a rerooting call, the level of at least one of its endpoints falls. Since level can fall only up to n, it follows that the computation associated with a single edge during the algorithm is of the order of n. Hence, the $O(mn)$ time complexity of the algorithm is immediate. However, using a more careful insight into the rerooting procedure, we shall now show that the time complexity is much better.

Consider any execution of the rerooting procedure. Let $((y =)z_1, z_2, \ldots, z_k(= v))$ be the path that gets reversed during the rerooting process. See Figure 2. The procedure collects the edges $\mathcal{B}(z_i)$ for each $i \leq k$. We shall now *charge* each edge collected to the fall of one of its endpoints. Let τ be a parameter whose value will be fixed later. Consider any edge (a, z_i) that is collected during the rerooting process. Note that level of each of a and z_i has fallen due to the rerooting. If $i \leq \tau$, we charge this edge to the fall of vertex a. In this way, there will be at most τ edges that get charged to the fall of a. If $i > \tau$, we charge this edge to the fall of z_i. It follows from Lemma 2 that z_i falls by at least $2i - 1 > \tau$ levels in this case.

Consider any vertex s in the graph. Over any sequence of edge insertions, if s falls by less than τ levels, we call it a *small* fall; Otherwise we call it a *big* fall for s. It follows that s will be charged τ edges for every small fall. The number of small falls is $O(n)$, so overall cost charged to s due to all its small falls is $O(n\tau)$. On the other hand, s will be charged $O(\deg(s))$ for every big fall. Notice that there will be at most n/τ big falls of s throughout any sequence of edge insertions. So the overall cost charged to all big falls of s will be $O(\deg(s) \cdot n/\tau)$. Hence for all vertices, the total computation charged will be $O(n^2\tau + mn/\tau)$. Fixing $\tau = \sqrt{m/n}$, we can conclude that the overall computation performed during processing of any sequence of m edge insertions by the algorithm is $O(n^{3/2}m^{1/2})$.

Theorem 3. *For an undirected graph G on n vertices, a DFS Tree can be maintained under insertion of any arbitrary sequence of edges with total update time of $O(n^{3/2}\sqrt{m})$.*

It follows from Theorem 3 that even for any sequence of $\Theta(n^2)$ edge insertions, the total update time in maintaining DFS tree is $O(n^{2.5})$ which is strictly subcubic. In fact, with a more structured way of processing the edges of $\mathcal{E}$, we can even achieve a bound of $O(n^2)$. We provide this improved algorithm in the following section.

6 Achieving $O(n^2)$ Update Time

The time complexity of Algorithm 1 is governed by the number of edges in $\mathcal{E}$ that are processed during the algorithm. In order to get an improved algorithm, let us examine $\mathcal{E}$ carefully. An edge from $\mathcal{E}$ can be a cross edge or a back edge. Processing of a cross edge always triggers a rerooting event which, in turn, leads to fall of one or more vertices. Hence, the total number of cross edges processed during the algorithm is $O(n^2)$. All the remaining edges processed in $\mathcal{E}$ during the entire algorithm are back edges. There are two sources of these back edges.

Firstly, some edges added to $\mathcal{E}$ by Procedure Reroot are back edges. Let us analyze their count throughout the algorithm. It follows from Lemma 4 that both endpoints of each such back edge belong to the conjugate path associated with the rerooting call. Notice that ith vertex on the conjugate path falls by at least $2i-1$ levels (see Lemma 2). So, if ℓ is the length of the conjugate path, the total fall in the level of all vertices on the conjugate path is more than $\ell(\ell-1)/2$ which is an upper bound on the number of edges with both endpoints on the conjugate path. Since the total fall in the level of vertices cannot be more than $O(n^2)$, the number of such back edges throughout the algorithm is $O(n^2)$.

Secondly, some edges added to $\mathcal{E}$ by Procedure Reroot are cross edges at the time of their collection, but become back edges before they are processed. This may happen due to rerooting initiated by some other cross edge from $\mathcal{E}$. In order to understand this subtle point, see Figure 3. Here $e_1, e_2, e_3, e_4, e_5 = (x, y)$ are cross edges present in $\mathcal{E}$ at some stage. While we process (x, y), $T(v)$ gets rerooted at y and hangs through edge (x, y). As a result e_1, e_2, e_3, e_4 become back edges.

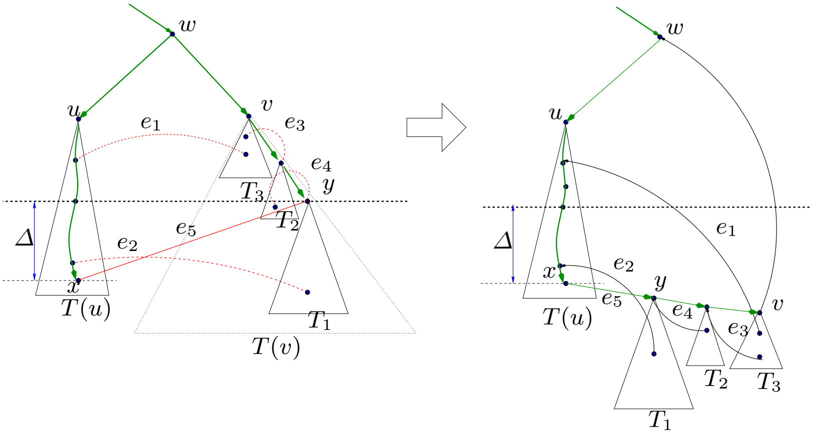


Fig. 3. Some cross edges in $\mathcal{E}$ become back edges due to the rerooting of $T(v)$

In order to bound these cross edges getting transformed into back edges during rerooting of $T(v)$, let us carefully examine one such cross edge. Let v_h and v_l be respectively the higher and lower endpoints of the resulting back edge. The edge (v_h, v_l) was earlier a cross edge, so v_h now has a new descendant v_l. Similarly v_l now has a new ancestor v_h. Note that the descendants of only vertices lying on prime and conjugate paths are changed during rerooting. Also the ancestors of only the vertices in $T(v)$ are changed during rerooting. Hence the following lemma holds true.

Lemma 5. *Cross edges getting converted to back edges as a result of rerooting $T(v)$ are at most $|T(v)|$ times the sum of lengths of prime and conjugate paths.*

Observe that the total sum of fall of vertices of $T(v)$ during the rerooting event is at least $|T(v)| \cdot (\varDelta + 1)$ (see Figure 3). However, the sum of lengths of prime and conjugate paths may be much greater than $\varDelta + 1$. Hence, the number of cross edges getting transformed into back edges is not bounded by the fall of vertices of $T(v)$. This observation, though discouraging, also gives rise to the following insight: If there were no cross edges with level higher than $\textsc{level}(y)$, the number of cross edges converted to back edges will be fewer. This is because the possible higher endpoints of such edges on the prime or conjugate path will be limited. This insight suggests that processing higher cross edges from $\mathcal{E}$ first will be more advantageous than lower ones. Our final algorithm is inspired by this idea.

6.1 The Final Algorithm

Our final algorithm is identical to the first algorithm except that instead of invoking Procedure `Choose` we invoke Procedure `ChooseHigh`. It processes the edges of set $\mathcal{E}$ in non-increasing order of their levels. To extract the highest edge from $\mathcal{E}$, we may use a binary heap on endpoints of these edges taking $O(\log n)$ time per operation. However, this can be improved to $O(1)$ amortized time using a much simpler data structure that exploits the following facts. Firstly, level of a vertex is an integer in $[1, n]$. Secondly, when a rerooting event occurs upon insertion of an edge e, all the edges collected are at a level lower than $\textsc{level}(e)$. Kindly refer to the full version of this paper for details of this data structure.

Procedure ChooseHigh($\mathcal{E}$): Chooses and returns the highest edge from $\mathcal{E}$.

```
    Remove the highest edge $(x, y)$ from $\mathcal{E}$.
    Return $(x, y)$.
```

6.2 Analysis

In order to establish $O(n^2)$ bound on the running time of our final algorithm, it follows from the preceding discussion that we just need to show that the number of cross edges getting converted to back edges throughout the algorithm is $O(n^2)$.

Consider any rerooting event initiated by cross edge (x, y) (refer to Figure 3). Since there is no edge in $\mathcal{E}$ which is at a higher level than $\textsc{level}(y)$, so it follows from Lemma 5 that the cross edges getting converted to back edges during the rerooting event will be of one of the following types only.

- The cross edges with one endpoint in $T(v)$ and another endpoint x or any of $\varDelta$ ancestors of x.
- The cross edges with y as one endpoint and another endpoint anywhere in $T(v) \backslash T(y)$.

Hence the following lemma holds for our final algorithm.

Lemma 6. *During the rerooting event the number of cross edges converted to back edges are at most* $|T(v)| \cdot (\Delta + 1) + |T(v) \backslash T(y)|$.

According to Lemma 2, level of each vertex of $T(v)$ falls by at least $\Delta + 1$. So the first term in Lemma 6 can be clearly associated with the fall of each vertex of $T(v)$. Note that each vertex in $T(v) \backslash T(y)$ becomes a descendant of y and hence falls by at least one extra level (in addition to $\Delta + 1$). This fall by extra one or more levels for vertices of $T(v) \backslash T(y)$ can be associated with the second term mentioned in Lemma 6. Hence the total number of cross edges getting transformed to back edges during the algorithm is of the order of $O(n^2)$. We can thus conclude with the following theorem.

Theorem 4. *For an undirected graph G on n vertices, a DFS Tree can be maintained under insertion of any arbitrary sequence of edges with total update time of $O(n^2)$.*

Remark: The $O(n^2)$ bound of our algorithm is quite tight even for sparse graphs. In fact, we can show the following: There exists a sequence of $\Theta(n)$ edge insertions such that every incremental algorithm for maintaining DFS tree that follows the principle of *monotonic fall* will require $\Omega(n^2)$ time. Kindly refer to the full version of this paper for details.

References

1. Alstrup, S., Holm, J.: Improved algorithms for finding level ancestors in dynamic trees. In: Welzl, E., Montanari, U., Rolim, J.D.P. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 73–84. Springer, Heidelberg (2000)
2. Cole, R., Hariharan, R.: Dynamic lca queries on trees. SIAM J. Comput. 34(4), 894–923 (2005)
3. Even, S., Shiloach, Y.: An on-line edge-deletion problem. J. ACM 28(1), 1–4 (1981)
4. Franciosa, P.G., Gambosi, G., Nanni, U.: The incremental maintenance of a depth-first-search tree in directed acyclic graphs. Inf. Process. Lett. 61(2), 113–120 (1997)
5. Miltersen, P.B., Subramanian, S., Vitter, J.S., Tamassia, R.: Complexity models for incremental computation. Theor. Comput. Sci. 130(1), 203–236 (1994)
6. Reif, J.H.: Depth-first search is inherently sequential. Inf. Process. Lett. 20(5), 229–234 (1985)
7. Reif, J.H.: A topological approach to dynamic graph connectivity. Inf. Process. Lett. 25(1), 65–70 (1987)
8. Roditty, L., Zwick, U.: On dynamic shortest paths problems. In: Albers, S., Radzik, T. (eds.) ESA 2004. LNCS, vol. 3221, pp. 580–591. Springer, Heidelberg (2004)
9. Tarjan, R.E.: Depth-first search and linear graph algorithms. SIAM J. Comput. 1(2), 146–160 (1972)

On the Role of Shared Randomness in Simultaneous Communication

Mohammad Bavarian[1,⋆], Dmitry Gavinsky[2,⋆⋆], and Tsuyoshi Ito[⋆⋆⋆]

[1] Massachusetts Institute of Technology, Cambridge, U.S.A.
[2] Institute of Mathematics, Academy of Sciences, Praha, Czech Republic

Abstract. Suppose two parties who are interested in performing certain distributed computational tasks are given access to a source of correlated random bits ρ. This source of correlated randomness could be quite useful to the parties for solving various distributed computational problems as it enables the parties to act in a correlated manner. In this work, we initiate the study of power of different sources of shared randomness ρ in the setting of communication complexity; we shall do so in the model of simultaneous message passing (SMP) model of communication complexity, and we shall also argue that this model is the appropriate choice among the commonly studied models of two-party communication complexity for the purpose of studying shared randomness as a resource. As such, we introduce a natural measure for the strength of the correlation provided by a bipartite distribution that we call *collision complexity*. We demonstrate that the collision complexity $\mathrm{col}_\rho(n)$ of a bipartite distribution ρ tightly characterises the power of ρ as a resource. We also uncover some surprising phenomenon by showing that even the noisiest shared randomness increases the power of SMP substantially: the *equality* function can be solved very efficiently with virtually any nontrivial shared randomness— whereas without shared randomness the complexity is known to be $\Omega(\sqrt{n})$.

1 Introduction

One of the central aims of complexity theory is to study the power of various resources of computation. By now, the power of many different types of resource – such as time, space, randomness, access to powerful provers, etc. – have been investigated in various models of computation. An important resource in many distributed settings involving multiple parties is the access to a source of correlated random bits; such a resource allows the parties to act in a correlated way

[⋆] This work was completed when the author was an intern at NEC Laboratories America.

[⋆⋆] This work was completed when the author was affiliated with NEC Laboratories America. Partially funded by the grant P202/12/G061 of GA ČR and by RVO: 67985840.

[⋆⋆⋆] This work was completed when the author was affiliated with NEC Laboratories America.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 150–162, 2014.

enabling them to perform better in various tasks. In this work, we would like to quantitatively study the utility of a source of shared randomness ρ to parties interested in solving communication problems.

Having access to correlated bits can allow the parties to reduce the amount of communication significantly in many cases, and in this work we are interested to know how the cost of an optimal solution depends on the source of shared randomness ρ – more specifically, what properties of ρ determine the solution complexity.

1.1 Shared Randomness and Communication Complexity

The central question in communication complexity is how much communication is needed to compute a given function on distributed inputs. In the two-party *simultaneous message passing (SMP)* model, two parties, conventionally called *Alice* and *Bob*, receive separate inputs x and y, respectively. Each of them encodes the input to a message to a third party – the *referee*, who has to output the answer. A *communication task* defines which answer is correct for the given input, and the goal of the players is to produce a correct answer with probability at least 2/3 for each possible input. An optimal communication protocol for the given task satisfies the correctness condition and minimises the total length of the messages sent by Alice and Bob.

A central fact about this model (which makes it appropriate for our study) is that whether Alice and Bob share a random source or they only have access to private randomness can affect significantly the amount of communication needed to compute certain functions. An archetypal example is the equality function on n-bit strings: it can be computed with $O(1)$ bits of communication with public randomness but requires $\Theta(\sqrt{n})$ bits of communication with private randomness [9,1].

Definition 1 (SMP with Shared Distribution ρ). *Let $l \in \mathbb{N}$, and let ρ be a probability distribution on $U \times V$. In the SMP model with shared distribution ρ, Alice receives input x and her part of shared randomness $(u_1, \ldots, u_l) \in U^l$, and Bob receives input y and his part of shared randomness $(v_1, \ldots, v_l) \in V^l$, where the l pairs $(u_1, v_1), \ldots, (u_l, v_l)$ are identically and independently distributed according to ρ. Alice and Bob use their parts of input and their shares of randomness to compute their messages to the referee. Upon receiving the messages from the players, the referee outputs the answer.*

A communication protocol determines the value of l and the actions of all the participants. A protocol is said to solve a communication problem with error probability δ if it guarantees correct answer with probability at least $1 - \delta$ for every allowed input pair. The communication cost of a protocol is the maximum possible total number of bits sent by the players. The communication complexity of a given problem is the minimum cost of a protocol that solves f with error probability 1/3.

It is important to note that SMP is unique among the commonly studied models of communication for being suitable for investigating shared randomness. Namely, SMP is the only model that does not allow direct communication between Alice and Bob. When such communication is allowed, the sender of the first message can locally toss random coins and append the outcomes to the message, and those values can be used in place of shared randomness. It has been shown by Newman [8] that $O(\log n)$ bits of randomness are sufficient for a nearly-optimal protocol for any communication problem, and therefore availability of "free" shared randomness can only save an additive factor of $O(\log n)$ to the communication cost, which is usually viewed as insignificant. On the other hand, as noted above, there are known communication problems whose complexity is $O(1)$ in SMP with shared randomness and $\Omega(\sqrt{n})$ without it.

The main subject of this work are the intermediate cases between the "extremes" of public randomness and private randomness. Namely, we will look at the situation when Alice and Bob have access to unlimited number of (mutually independent) pairs of correlated random variables, where each pair is distributed according to a known bipartite probability distribution ρ. When ρ is uniform over the set $\{00, 11\}$, this corresponds to the usual public randomness, which we call *perfect*. If ρ is something different, can Alice and Bob still use it to reduce communication cost of an optimal protocol, comparing to the case of private randomness? This is the question we answer in this work.

We will see that different choices of the shared distribution ρ give rise to communication models of different power. It is easy to see that given perfect public randomness, Alice and Bob can locally simulate any other bipartite distribution to any precision, which implies that ρ being the uniform distribution over $\{00, 11\}$ is the most powerful type of shared randomness (thus call it *perfect*). We show that many other choices of ρ give rise to communication models that are strictly weaker than perfect randomness but stronger than private randomness.

1.2 Our Results

First, we give an example of an SMP protocol which exploits even the weakest form of correlated randomness. We show that for any bipartite probability distribution ρ other than a product distribution, there is an SMP protocol for the equality function on n-bit strings with shared distribution ρ whose communication cost is a constant independent of n. Since the equality function on n-bits strings requires $\Omega(\sqrt{n})$ bits without shared randomness, this shows that any correlation increases the power of SMP significantly.

Second, we prove that different choices of ρ lead to different power of the SMP model where the players share ρ. That will be done in three steps:

- We will introduce a quantity which we call the *collision complexity of ρ at domain size n* and denote $\mathrm{col}_\rho(n)$, and study its properties.
- We will show that any SMP communication protocol with public randomness of cost at least $\log n$ can be simulated by a protocol with shared distribution ρ at the expense of increasing the cost by a factor of $O(\mathrm{col}_\rho(n))$, where n is the input length.

- We will prove that the above simulation is tight up to a poly-logarithmic factor by constructing a partial Boolean function, whose SMP communication complexity is $O(\log n)$ with public randomness but is $\Omega(\mathrm{col}_\rho(n)/\log n)$ with any shared distribution ρ.

In other words, we will construct an infinite proper hierarchy of SMP models with different types of shared randomness.

Finally, we will study the notion of collision complexity more closely, and relate it to two widely studied measures which quantify how correlated two discrete random variables are: the maximum correlation and the hypercontractivity.

Related Work. The collision complexity introduced in this work quantifies how hard it is to simulate public randomness by imperfect shared randomness in an SMP communication protocol. There are many works which discuss the task of simulating one kind of randomness by another kind of randomness, and particularly close to ours is non-interactive correlation distillation (NICD) [6,7,11]. In the two-party case of NICD, Alice and Bob have an access to an unlimited number of independent copies of ρ just as in our case, and their task is to produce a marginally uniform bit each so that the outputs by Alice and Bob agree with the maximum probability. Although both NICD and the collision complexity are closely related to the maximum correlation and hypercontractivity, their exact relationship is unclear.

Gavinsky, Ito, and Wang [3] study a notion of shared randomness different from the standard perfectly correlated random bits in the multi-party SMP setting, and our proof of Lemma 7 follows the same overall structure as that of Theorem 2 there.

2 Preliminaries

Throughout the paper, the base of logarithm is two unless stated otherwise. For $x, y \in \mathbb{R}^n$, let $x \cdot y$ denote $\sum_{i=1}^{n} x_i y_i$. For bit strings $x, y \in \{0,1\}^n$, let $x \cdot y$ denote $\sum_{i=1}^{n} x_i y_i \bmod 2$.

For a probability distribution ρ on (finite sets) $U \times V$, we write its marginal distributions on U and V as ρ_U and ρ_V, respectively. We denote the uniform distribution over a set X by $\mathcal{U}_X$.

Now we shall discuss the relevant background from communication complexity. We assume some basic familiarity with the subject [5]. For a partial function f from $\{0,1\}^n \times \{0,1\}^n$ to E, we denote by $\mathtt{SMP}(f)$ the SMP communication complexity of f without shared randomness with worst-case error probability at most 1/3.

For $n \geq 1$, let $\mathcal{IP}_n \colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be the inner product function modulo 2. Chor and Goldreich [2] showed that any communication protocol which answers $\mathcal{IP}_n$ with error probability 1/3 on average, even in the two-way communication model, must have communication cost at least $n/2 - o(n)$. The following is an immediate corollary of this.

Corollary 1. *For $n \geq 1$, the communication complexity of $\mathcal{IP}_n$ in the SMP model with public randomness is in $\Omega(n)$.*

We will use the following fact originally proved by Newman [8] and refined by Kushilevitz and Nisan [5, Theorem 3.14]. The statement in [5] assumes that the function takes a Boolean value, but the proof does not use this assumption.

Lemma 1. *Let $0 < \varepsilon < \varepsilon' < 1/2$ be constants. Any SMP protocol with public randomness for a partial function from $\{0,1\}^n \times \{0,1\}^n$ to E with error probability at most ε can be converted to one which uses only $\log n + C$ bits of public randomness with error probability at most ε' without changing the communication cost, where $C > 0$ is a constant which depends only on ε and ε'.*

Maximum Correlation of a Bipartite Probability Distribution. We will make use of the following notion quantifying the amount of correlation between two random variables.

Definition 2 (Maximum Correlation). *The* maximum correlation *of a distribution ρ on $U \times V$, which we denote by $\mathrm{Cor}(\rho)$, is defined as $\mathrm{Cor}(\rho) = \sup_{f,g} \mathbf{E}_{(u,v)\sim\rho}[f(u)g(v)]$, where the supremum is over functions $f\colon U \to \mathbb{R}$ and $g\colon V \to \mathbb{R}$ that satisfy $\mathbf{E}_{(u,v)\sim\rho}[f(u)] = \mathbf{E}_{(u,v)\sim\rho}[g(v)] = 0$ and $\mathbf{E}_{(u,v)\sim\rho}[f(u)^2] = \mathbf{E}_{(u,v)\sim\rho}[g(v)^2] = 1$.*

The following is immediate from the definition.

Lemma 2. *Let ρ be a distribution on $U \times V$. Let $f\colon U \to [0,1]$ and $g\colon V \to [0,1]$, and let $a = \mathbf{E}[f(u)]$ and $b = \mathbf{E}[g(v)]$ where $(u,v) \sim \rho$. Write the variance of f and g as $\mathrm{Var}(f)$ and $\mathrm{Var}(g)$, respectively. Then it holds that $\mathbf{E}_{(u,v)\sim\rho}[f(u)g(v)] \leq ab + \mathrm{Cor}(\rho)\sqrt{\mathrm{Var}(f)\,\mathrm{Var}(g)}$.*

Proof. Note that $\mathbf{E}_{(u,v)\sim\rho}[f(u)g(v)] = ab + \mathbf{E}_{(u,v)\sim\rho}[(f(u)-a)(g(v)-b)]$, and use the fact that $(f-a)/\sqrt{\mathrm{Var}(f)}$ and $(g-b)/\sqrt{\mathrm{Var}(g)}$ have mean 0 and variance 1.

Later we will use an upper bound on the maximum correlation to obtain a lower bound on the collision complexity. It is sometimes easier to deal with the maximum correlation than the collision complexity because the maximum correlation behaves nicely with independent random variables:

Lemma 3 (Witsenhausen [10, Theorem 1]). *For $i \in [n]$, let ρ_i be a probability distribution on $U_i \times V_i$. Then it holds that $\mathrm{Cor}(\rho_1 \otimes \cdots \otimes \rho_n) = \max_{i\in[n]} \mathrm{Cor}(\rho_i)$.*

3 Usefulness of Weak Shared Randomness in Some Settings

In this section, we show an easy but nontrivial use of imperfect shared randomness in an SMP communication protocol for the equality function $\mathcal{EQ}_n\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, defined to be 1 if $x = y$, and 0 otherwise. It is known that $\mathcal{EQ}_n$

has a constant SMP communication complexity with perfect public randomness [1] whereas it has SMP communication complexity $\Theta(\sqrt{n})$ without shared randomness [9,1]. If Alice and Bob share some imperfect randomness, what happens to this communication complexity? The answer is provided by the following theorem:

Theorem 1. *Let ρ be a probability distribution on $U \times V$. If ρ is not a product distribution, then it holds that $\mathtt{SMP}_\rho(\mathcal{EQ}_n) = O(1)$, where the constant depends on ρ but not on n.*

Proof. Because ρ is not a product distribution, there exist subsets $U_1 \subseteq U$ and $V_1 \subseteq V$ such that if (u, v) is distributed according to ρ, it holds that

$$\Pr[u \in U_1 \wedge v \in V_1] \neq \Pr[u \in U_1] \Pr[v \in V_1]. \tag{1}$$

A protocol for solving the equality is as follows. Alice and Bob share 2^n copies of shared randomness; label the 2^n i.i.d. pairs of registers containing the shared randomness as $(u_x, v_x) \sim \rho$ for $x \in \{0,1\}^n$. Alice defines $\alpha = 1$ if $u_x \in U_1$ and $\alpha = 0$ otherwise, and sends α to the referee. Similarly, Bob defines $\beta = 1$ if $v_y \in V_1$ and $\beta = 0$ otherwise, and sends β to the referee. The referee checks whether $\alpha = \beta = 1$ or not. If $x = y$, this happens with probability $\Pr[u \in U_1 \wedge v \in V_1]$, and otherwise it happens with probability $\Pr[u \in U_1] \Pr[v \in V_1]$. Because of inequality (1), these probabilities are different. The referee can tell which is the case with error probability at most 1/3 by repeating this protocol for t times, where $t = O(|\Pr[u \in U_1 \wedge v \in V_1] - \Pr[u \in U_1] \Pr[v \in V_1]|^{-2})$, which is a constant independent of n.

4 Collision Complexity of Bipartite Distributions

We will introduce two ways to quantify the strength of correlation of a bipartite distribution, and prove that they are in fact equivalent.The first one is collision complexity:

Definition 3 (Collision Complexity). *Let ρ be a probability distribution on $U \times V$ and $n \in \mathbb{N}$. A collision protocol for ρ with domain size n is determined by an integer $l \in \mathbb{N}$ and two (possibly randomised) mappings $A\colon U^l \to 2^{[n]}$ and $B\colon V^l \to 2^{[n]}$. The output size of this collision protocol by*

$$\max\{\max_{u \in U^l} |A(u)|, \max_{v \in V^l} |B(v)|\},$$

and the collision probability of this protocol by

$$\min_{i \in [n]} \Pr[i \in A(u_1, \ldots, u_l) \cap B(v_1, \ldots, v_l)],$$

where $(u_j, v_j) \sim \rho$ independently for all $j \in [l]$.

The collision complexity of ρ at domain size n and collision probability p, which we denote by $\operatorname{col}_\rho(n, p)$, is the minimum output size of a collision protocol for ρ with domain size n and collision probability at least p. We write $\operatorname{col}_\rho(n, 1/n)$ also as $\operatorname{col}_\rho(n)$.

It is useful to view this definition operationally. Alice has access to random variables $u_1, \dots, u_l$ and Bob has access to random variables $v_1, \dots, v_l$, where $(u_j, v_j) \sim \rho$ independently for all $j \in [l]$. Alice produces a set $A \subseteq [n]$ using $u_1, \dots, u_l$ and her private randomness, and Bob produces a set $B \subseteq [n]$ using $v_1, \dots, v_l$ and his private randomness. An adversary who knows Alice and Bob's strategy (but does not know the values of random variables) chooses one value $i \in [n]$. Alice and Bob are required to produce sets so that this unknown element i belongs to $A \cap B$ with probability at least p, while minimising the size of sets A and B. This minimum size is the collision complexity $\mathrm{col}_\rho(n, p)$.

Note that without any shared distribution, Alice and Bob can use the birthday paradox to achieve the collision complexity $O(\sqrt{n})$ at collision probability $1/n$; accordingly, the collision complexity of any bipartite distribution at collision probability $1/n$ is $O(\sqrt{n})$. On the other hand, $\mathcal{U}_{\{00,11\}}$ has the lowest possible collision complexity 1 at collision probability $1/n$. In the full version, we will see an example of a bipartite distribution whose collision complexity is strictly between these two extremal cases.

The second measure of correlation is agreement complexity.

Definition 4 (Agreement Complexity). *Let ρ be a probability distribution on $U \times V$.*

An agreement protocol *for ρ is determined by $l \in \mathbb{N}$ and a pair of functions $f\colon U^l \to [0,1]$ and $g\colon V^l \to [0,1]$. The* cost *of this agreement protocol is $\mathbf{E}[f(u_1, \dots, u_l) + g(v_1, \dots, v_l)]$, and the* success probability *of this protocol is $\mathbf{E}[f(u_1, \dots, u_l) g(v_1, \dots, v_l)]$, where $(u_i, v_i) \sim \rho$ independently for all $i \in [l]$.*

The agreement complexity *of ρ at success probability p, denoted by $\mathrm{agr}_\rho(p)$, is the infimum of the cost of an agreement protocol for ρ with success probability at least p.*

Here each of Alice and Bob outputs just one bit instead of a subset of $[n]$. The value $f(u_1, \dots, u_l)$ is interpreted as the probability that Alice outputs bit 1 given her part of shared randomness, and similarly $g(v_1, \dots, v_l)$ is the probability that Bob outputs bit 1 given his part of shared randomness. Their task is to output 1 simultaneously with probability at least p while minimising the sum of the probabilities that each party outputs 1. This minimum (infimum to be precise) is the agreement complexity for ρ with success probability p.

Note the following simple parameter manipulations:

- By repeating a collision protocol m times independently and outputting the union of the results, we can increase the collision probability from p to $1-(1-p)^m$. Therefore, it holds that

$$\mathrm{col}_\rho(n, 1-(1-p)^m) \le m \, \mathrm{col}_\rho(n, p). \tag{2}$$

- Repeat a collision protocol m times independently to obtain $A_1, \dots, A_m \subseteq [n]$ and $B_1, \dots, B_m \subseteq [n]$, and output $A = \{n(i-1)+j : i \in [m], j \in A_j\}$ and $B = \{n(i-1)+j : i \in [m], j \in B_j\}$. This gives a collision protocol with domain size mn, collision probability p, and output size at most $m \, \mathrm{col}_\rho(n, p)$.

Therefore, it holds that

$$\mathrm{col}_\rho(mn, p) \le m\,\mathrm{col}_\rho(n, p). \tag{3}$$

Collision protocols and agreement protocols can be converted to each other in the following sense.

Lemma 4. *For a bipartite distribution ρ, $n \ge 1$, and $0 < p < 1$, it holds that $\mathrm{col}_\rho(n, p) = \Theta(\max\{1, n\,\mathrm{agr}_\rho(p)\})$, where the constant in the asymptotic notation does not depend on ρ, n, or p.*

A proof is given in the full version.

The following lemma states that any protocol in the SMP model with perfect shared randomness can be simulated in the SMP model with shared distribution ρ at the expense of a multiplicative factor which is at most the collision complexity of ρ. A proof is given in the full version.

Lemma 5. *Let ρ be a bipartite distribution, and let f be a partial function from $\{0,1\}^n \times \{0,1\}^n$ to E.*

(i) If there is an SMP protocol for f with public randomness which uses t bits of public randomness, has complexity c, and has error probability at most $\varepsilon < 1/2$, then it holds that $\mathtt{SMP}_\rho(f) \in O(\mathrm{col}_\rho(2^t)(c + t))$, where the constant factor in the O-notation depends on ε and nothing else.

(ii) If $\mathtt{CC}_{\mathtt{perf}} = \mathtt{SMP}_{\mathcal{U}_{\{00,11\}}}(f)$ is the communication complexity of f in the SMP model with perfect public randomness, then it holds that

$$\mathtt{SMP}_\rho(f) = O(\mathrm{col}_\rho(n)\,(\mathtt{CC}_{\mathtt{perf}} + \log n)\,)\,.$$

5 Different Shared Distributions Can Have Different Power

In this section, we will prove the following theorem.

Theorem 2. *There exists a family of partial functions f_n from $\{0,1\}^n \times \{0,1\}^n$ to $\{0,1\}$ such that for any bipartite distribution ρ, it holds that*

$$\mathtt{SMP}_\rho(f_n) = \begin{cases} O\left(\mathrm{col}_\rho\left(\frac{n}{\log n}\right)\log n\right) & (\textit{upper bound}) \\ \Omega\left(\max\left\{\log n, \mathrm{col}_\rho\left(\frac{n}{\log n}\right)\right\}\right) & (\textit{lower bound}) \end{cases},$$

where the constants in the asymptotic notations do not depend on ρ or n.

The family of functions we use in Theorem 2 is defined as follows:

Definition 5 (Gap-Inner-Product). *Let $n, m \in \mathbb{N}$ and $\forall i \in [n] : x_i, y_i \in \{0,1\}^m$. Define the* gap-inner-product function *$\mathcal{GIP}_{n,m}$ as*

$$\mathcal{GIP}_{n,m}((x_1,\ldots,x_n),(y_1,\ldots,y_n)) = \begin{cases} 0, & |\{i \in [n] \colon x_i \cdot y_i = 0\}| \ge 2n/3, \\ 1, & |\{i \in [n] \colon x_i \cdot y_i = 1\}| \ge 2n/3, \\ \perp, & \textit{otherwise.} \end{cases}$$

We will write $\mathcal{GIP}_n$ to address $\mathcal{GIP}_{n,8\log n}$, assuming that n is a power of two in this context.

Proposition 1. *For a bipartite distribution ρ, it holds that $\mathtt{SMP}_\rho(\mathcal{GIP}_n) = O(\mathrm{col}_\rho(n)\log n)$.*

Proof. Note that $\mathcal{GIP}_n$ has a straightforward SMP protocol with public randomness with cost $O(\log n)$ and error probability at most 1/3: Alice and Bob choose a common index $i \in [n]$ uniformly at random, Alice sends x_i to the referee, and Bob sends y_i to the referee. The referee simply answers $x_i \cdot y_i$. Because this protocol uses $\log n$ bits of public randomness, Lemma 5 (i) implies the claim.

Proposition 2. *For a bipartite distribution ρ, it holds that $\mathtt{SMP}_\rho(\mathcal{GIP}_n) = \Omega(\log n)$.*

Proof. Note that $\mathcal{IP}_m$ can be reduced to $\mathcal{GIP}_{n,m}$ by repeating the input vector n times, and in particular $\mathcal{IP}_{8\log n}$ can be reduced to $\mathcal{GIP}_n$. By Corollary 1, this implies the claim.

The remaining task is to prove the following communication lower bound:

Theorem 3. *For any bipartite distribution ρ, it holds that $\mathtt{SMP}_\rho(\mathcal{GIP}_n) = \Omega(\mathrm{col}_\rho(n))$, where the constants in the asymptotic notations do not depend on ρ or n.*

Because the input size of $\mathcal{GIP}_n$ is $8n\log n$ bits, Propositions 1 and 2 and Theorem 3 imply Theorem 2.

Hardness of Random-Access Inner Product

As before, let ρ be a bipartite distribution. Let us see that $\mathtt{SMP}_\rho(\mathcal{GIP}_n) = \Omega(\mathrm{col}_\rho(n))$.

Note that we need a proof technique of carefully balanced strength: on the one hand, it has to be strong enough to capture the hardness of $\mathcal{GIP}_n$; on the other hand, it should not be applicable to the equality function (cf. Theorem 1).

Ignoring some technicalities, our proof will go by considering the behaviour of a hypothetical protocol for $\mathcal{GIP}_n$ under the uniformly random input distribution (ignoring the promise on the input of $\mathcal{GIP}_n$). First we will argue that if the protocol has cost $o(k)$, then for most $i \in [n]$ the referee can guess the value of $x_i \cdot y_i$ correctly with probability at most $1/2 + 1/\Omega(n)$. That will let us use a "hybrid-like" argument to conclude that there exists a set $L \subseteq [n]$ of size at least $3n/4$, such that the referee cannot distinguish with enough confidence the case when $x_i \cdot y_i = 0$ for all $i \in L$ from the case when $x_i \cdot y_i = 1$ for all $i \in L$ (in both the cases, x_i and y_i are uniformly random for $i \notin L$). So, if we define the input distribution μ to be uniform over $\{(x,y)\colon \forall i \in L : x_i = y_i\} \cup \{(x,y)\colon \forall i \in L : x_i \neq y_i\}$, then each element in the support of μ is a valid input to $\mathcal{GIP}_n$, but the protocol under consideration makes a mistake with respect to μ with high probability.

In our analysis we will refer to *pseudo-SMP* model (with or without shared randomness), by which we mean an analogue of the SMP model with the following changes:

– The referee can receive his own portion of input.
– If shared randomness is available, the referee can see both Alice's and Bob's part of the shared distribution.

We consider the following auxiliary communication task for the pseudo-SMP model, which we call the *random-access inner product* problem:

Definition 6 ($\mathcal{RAIP}_{n,m}$). *Let $n, m \in \mathbb{N}$ and $\forall i \in [n] : x_i, y_i \in \{0,1\}^m$. Denote $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$. Then $\forall i \in [n]$: $\mathcal{RAIP}_{n,m}(x, y, i) \stackrel{\text{def}}{=} x_i \cdot y_i$.*

We will write $\mathcal{RAIP}_n$ to address $\mathcal{RAIP}_{n,8\log n}$, in this context always assuming that n is a power of 2.

In the full version, we will prove the following lemma, which states that $\mathcal{RAIP}_n$ is a hard problem.

Lemma 6. *There exist constants $\delta, \lambda > 0$ and a function $\alpha(n) = o(1/n)$ such that for any $n \in \mathbb{N}$, any finite sets U and V, and any pairwise distribution ρ on $U \times V$, the following holds. Let $\mathcal{P}$ be a pseudo-SMP protocol for $\mathcal{RAIP}_n$ that uses shared distribution ρ with cost C. Assume that the input distribution is uniform over all triples $(x, y, i) \in (\{0,1\}^m)^n \times (\{0,1\}^m)^n \times [n]$, and let γ_i be the probability that $\mathcal{P}$ gives the correct answer when $I = i$. If $\lambda(C + \log n) \leq \mathrm{col}_\rho(n)$, then there exists $i \in [n]$ such that $2\gamma_i - 1 \leq \lambda(C + \log n)/(n\,\mathrm{col}_\rho(n)) + \alpha(n)$.*

To understand the statement of this lemma, consider the case where $\mathrm{col}_\rho(n) = \omega(\log n)$. Then the lemma claims that if the cost C of a communication protocol $\mathcal{P}$ for $\mathcal{RAIP}_n$ is too small, i.e., $C = o(\mathrm{col}_\rho(n))$, then there is a coordinate $i \in [n]$ such that $\mathcal{P}$ answers $x_i \cdot y_i$ correctly with probability at most $1/2 + o(1/n)$. The statement of the lemma is more involved because later we will use this lemma with varying choices of ρ, where we need the fact that nothing hidden in the asymptotic notation depends on the choice of ρ.

To prove this lemma, we will consider another auxiliary task, which is the two-player variant of the *bounded-error random access* problem considered by Gavinsky, Kempe, Regev, and de Wolf [4].

Definition 7 (Two-Player Bounded-Error Random Access). *For $n \in \mathbb{N}$, let Alice receive $X \in \{0,1\}^n$, Bob receive $Y \in \{0,1\}^n$, and the referee receive $I \in [n]$ and output $Z \in \{0,1\}^2 \cup \{\bot\}$. We say that the protocol solves $\mathcal{RA}_{n,\delta}$ if for every i, $\Pr_{X,Y \sim \mathcal{U}}[Z = (X_i, Y_i) \mid I = i,\ Z \neq \bot] \geq 1 - \delta$.*

In the proof of Lemma 6, we will first prove that $\mathcal{RA}_{n,\delta}$ is a hard problem by interpreting an efficient protocol for $\mathcal{RA}_{n,\delta}$ as a collision protocol. To prove this, consider any protocol $\mathcal{P}$. Let $X, Y \in \{0,1\}^n$ denote the inputs given to

Alice and Bob in $\mathcal{P}$, respectively. For each $i \in [n]$, we consider whether the conditional entropy of X_i after reading the message from Alice to the referee in $\mathcal{P}$ and the shared randomness between Alice and the referee (allowed by the definition of the pseudo-SMP model) is small or not. If it is small, the referee "knows" the bit X_i with high probability. Then we can prove that if the cost of $\mathcal{P}$ is small, not many locations out of the $2n$ locations of X and Y together are known to the referee, while the bounded-error condition implies that for some i, X_i and Y_i are known to the referee at the same time. Now suppose that Alice outputs the set of the indices i such that X_i is known to the referee, and Bob does an analogous thing. Then the properties above imply that this becomes a good collision protocol.

After proving that $\mathcal{RA}_{n,\delta}$ is a hard problem, we will reduce $\mathcal{RA}_{n,\delta}$ to $\mathcal{RAIP}_n$, establishing that $\mathcal{RAIP}_n$ is also a hard problem.

Hardness of Gapped Inner Product. The hardness of $\mathcal{RAIP}_n$ can be used as an inductive step in the SMP lower bound of $\mathcal{GIP}_n$, in accordance with the idea outlined before.

Lemma 7. *For $m \geq 1$ and $b \in \{0,1\}$, let $\sigma_{m,b}$ be the uniform distribution over $\{(u,v) \in \{0,1\}^m \times \{0,1\}^m \colon u \cdot v = b\}$. Let ρ be a bipartite distribution. Let n be a power of two, and let $m = 8 \log n$. Then it holds that $\mathtt{SMP}_\rho(\mathcal{GIP}_n) \in \Omega(\min\{\mathrm{col}_{\rho\otimes\sigma_{m,0}}(n), \mathrm{col}_{\rho\otimes\sigma_{m,1}}(n)\})$, where the constant factor in the asymptotic notation does not depend on n or ρ.*

Let us understand why $\mathrm{col}_{\rho\otimes\sigma_{m,b}}(n)$ appears in the lower bound instead of just $\mathrm{col}_\rho(n)$. For simplicity, suppose that no distribution is shared (i.e., we only want to establish hardness of $\mathcal{GIP}_n$ for $\mathtt{SMP}$). In that case we can start by applying the corresponding simplified version of Lemma 6 and conclude that if the input distribution is uniform then there exists a coordinate i_1, such that for the protocol under consideration the cases "$x_{i_1} \cdot y_{i_1} = 0$" and "$x_{i_1} \cdot y_{i_1} = 1$" are almost indistinguishable. Recall that our plan was to fix such i_1, then consider the behaviour of the protocol on the rest of coordinates, conditioned upon the value of $x_{i_1} \cdot y_{i_1}$, saying that for some i_2 the value of $x_{i_2} \cdot y_{i_2}$ is (almost) unknown to the protocol, and so on until enough coordinates have been selected in order to satisfy the promise in the definition of $\mathcal{GIP}_n$ by fixing the values of $x_i \cdot y_i = 1$ for those coordinates.

Starting from the second step, the above induction introduces shared randomness to the protocol, because conditioning on the value of $x_{i_1} \cdot y_{i_1}$ introduces correlation between the players. This is why the lower bound is $\mathrm{col}_{\rho\otimes\sigma_{m,b}}(n)$. By using Lemma 6, we can prove that even in the presence of that kind of shared randomness many indices i exist, such that the protocol cannot guess the value of $x_i \cdot y_i$ with enough confidence. The lemma then follows by the hybrid argument. A proof of Lemma 7 is given in the full version.

Removing Additional Shared Randomness Arising from Conditioning. The following two lemmas will be proved in the full version. Lemma 8 uses the basic Fourier analysis of Boolean functions, and Lemma 9 follows from the definitions of the maximum correlation and the agreement complexity by simple calculations.

Lemma 8. *Let $m \geq 2$ and $b \in \{0,1\}$. Let σ be the uniform distribution over $\{(u,v) \in \{0,1\}^m \times \{0,1\}^m \colon u \cdot v = b\}$. Then it holds that* $\mathrm{Cor}(\sigma) \leq 1/2^{m/2-1}$.

Lemma 9. *Let ρ be a bipartite distribution on $U \times V$, and σ be a bipartite distribution on $X \times Y$. Then for p with* $\mathrm{Cor}(\sigma) < p < 1$, *it holds that* $\mathrm{agr}_{\rho\otimes\sigma}(p) \geq \mathrm{agr}_\rho(p - \mathrm{Cor}(\sigma))$.

By using these lemmas, we can replace the bound in Lemma 7 by just $\Omega(\mathrm{col}_\rho(n))$.

Lemma 10. *Let $m \geq 2$ and $b \in \{0,1\}$. Let σ be the uniform distribution over $\{(x,y) \in \{0,1\}^m \times \{0,1\}^m \colon x \cdot y = b\}$. Let $n \leq 2^{m/2-2}$. Then for any bipartite distribution ρ, it holds that* $\mathrm{col}_{\rho\otimes\sigma}(n) = \Omega(\mathrm{col}_\rho(n))$, *where the constant factor does not depend on m, b, n, or ρ.*

Proof. Because $n \leq 2^{m/2-2}$, Lemma 8 implies that $\mathrm{Cor}(\sigma) \leq 1/(2n)$. Then by Lemma 9, it holds that

$$\mathrm{agr}_{\rho\otimes\sigma}\left(\frac{1}{n}\right) \geq \mathrm{agr}_\rho\left(\frac{1}{n} - \mathrm{Cor}(\sigma)\right) \geq \mathrm{agr}_\rho\left(\frac{1}{2n}\right) \geq \frac{\mathrm{agr}_\rho(1/n)}{2}.$$

The lemma follows Lemma 4.

Proof (Theorem 3). Follows from Lemmas 7 and 10.

References

1. Babai, L., Kimmel, P.G.: Randomized simultaneous messages: Solution of a problem of Yao. In: Communication Complexity (1997)
2. Chor, B., Goldreich, O.: Unbiased bits from sources of weak randomness and probabilistic communication complexity. SIAM Journal on Computing 17, 230–261 (1988)
3. Gavinsky, D., Ito, T., Wang, G.: Shared randomness and quantum communication in the multi-party model. In: IEEE 28th Conference on Computational Complexity (CCC), pp. 34–43 (2013)
4. Gavinsky, D., Kempe, J., Regev, O., de Wolf, R.: Bounded-error quantum state identification and exponential separations in communication complexity. SIAM Journal on Computing 39, 1–24 (2009)
5. Kushilevitz, E., Nisan, N.: Communication Complexity. Cambridge University Press (1997)
6. Mossel, E., O'Donnell, R.: Coin flipping from a cosmic source: On error correction of truly random bits. Random Structures and Algorithms 26, 418–436 (2005)
7. Mossel, E., O'Donnell, R., Regev, O., Steif, J.E., Sudakov, B.: Non-interactive correlation distillation, inhomogeneous Markov chains, and the reverse Bonami–Beckner inequality. Israel Journal of Mathematics 154, 299–336 (2006)

8. Newman, I.: Private vs. common random bits in communication complexity. Information Processing Letters 39, 67–71 (1991)
9. Newman, I., Szegedy, M.: Public vs. private coin flips in one round communication games. In: Proceedings of the 28th Symposium on Theory of Computing, pp. 561–570 (1996)
10. Witsenhausen, H.S.: On sequences of pairs of dependent random variables. SIAM Journal on Applied Mathematics 28, 100–113 (1975)
11. Yang, K.: On the (Im)possibility of Non-interactive Correlation Distillation, vol. 382 (2007)

Short PCPs with Projection Queries

Eli Ben-Sasson[*] and Emanuele Viola[**]

Technion — Israel Institute of Technology and Northeastern University
eli@cs.technion.ac.il, viola@ccs.neu.edu

Abstract. We construct a PCP for NTIME(2^n) with constant soundness, 2^npoly(n) proof length, and poly(n) queries where the verifier's computation is simple: the queries are a projection of the input randomness, and the computation on the prover's answers is a 3CNF. The previous upper bound for these two computations was polynomial-size circuits. Composing this verifier with a proof oracle increases the circuit-depth of the latter by 2. Our PCP is a simple variant of the PCP by Ben-Sasson, Goldreich, Harsha, Sudan, and Vadhan (CCC 2005). We also give a more modular exposition of the latter, separating the combinatorial from the algebraic arguments.

If our PCP is taken as a black box, we obtain a more direct proof of the result by Williams, later with Santhanam (CCC 2013) that derandomizing circuits on n bits from a class C in time $2^n/n^{\omega(1)}$ yields that NEXP is not in a related circuit class C'. Our proof yields a tighter connection: C is an And-Or of circuits from C'. Along the way we show that the same lower bound follows if the satisfiability of the And of any 3 circuits from C' can be solved in time $2^n/n^{\omega(1)}$.

1 Introduction

It has long been known that solving satisfiability of circuits, or derandomizing probabilistic circuits implies new circuit lower bounds (for various exponential-time classes), see e.g. [KL80,IKW02]. In [Wil10] Williams gives an interesting instance of this phenomenon, where a non-trivial lower bound against a circuit class C follows from a satisfiability or derandomization algorithm for circuits of a related class C' that runs in time $2^n/n^{\omega(1)}$, where n is the number of variables of circuits in C'. It is an interesting question whether the approach based on satisfiability or the one based on derandomization should be pursed to obtain new circuit lower bounds.

The satisfiability approach – not the derandomization approach – has given non-trivial lower bounds [Wil11]. Moreover this approach has been tightened, by making C' closer to C, in [SW13,JMV13,Oli13], making it plausible that more lower bounds will be obtained. In fact, we will tighten it a bit more in

[*] The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 240258.

[**] Supported by NSF grants CCF-0845003, CCF-1319206

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 163–173, 2014.

this work. However, it is not clear how much this approach can be pushed. Do we believe that the satisfiability of (unrestricted) polynomial-size circuits can be solved faster than brute-force search? Even for seemingly simple problems such as MAX3SAT, no satisfiability algorithm better than brute-force search is known, despite attempts since a decade ago [Wil05]. Note that the MAX3SAT problem – given a 3CNF and an integer ℓ, is there an assignment that satisfies $\geq \ell$ clauses? – corresponds to the restricted class of depth-2 circuits known as MAJ $\circ$ AND$_3$: a Majority on And's on three variables. The lack of progress on MAX3SAT is an obstacle for obtaining new lower bounds from satisfiability.

A priori, the approach based on derandomization should apply more broadly, because most researchers indeed believe that derandomization is possible (and a long line of research has shown that indeed derandomization is possible based on lower bounds). Also, for several classes we have nontrivial derandomization algorithms but not satisfiability ones. For example, for the class mentioned above of MAJ $\circ$ AND$_3$ circuits a derandomization is given in [LVW93,Vio07]. Even when both types of algorithms are available, the speed of the derandomization one often outperforms that of the satisfiability one. For example, the running time for the derandomization of CNF, see [GMR13] for the latest, vastly outperforms that of satisfiability algorithms, cf. [Her11]. For another example, consider the class of poly-size, constant-depth circuits with Or, Not, and Parity gates (AC^0 with parity gates). To our knowledge, the best satisfiability algorithm is the one in [Wil11] which has running time 2^{n-n^ϵ}. By contrast, [FSUV13] derandomize these circuits in time $2^{n-n/\text{poly}\log n}$ (building on available lower bounds).

One advantage of satisfiability over derandomization is that the corresponding connection to lower bounds is simpler and incurs less overhead. To obtain lower bounds from derandomization one relies on probabilistically checkable proofs (PCP), specifically the somewhat intricate work by Ben-Sasson, Goldreich, Harsha, Sudan, and Vadhan [BGH$^+$05]. The intricacy of this work reflects on two aspects of the approach. First, to make it apply to restricted circuit classes such as ACC^0 or TC^0, previous to this work one needed a roundabout argument, provided by Santhanam and Williams [SW13], which actually relies on a subsequent PCP by Mie [Mie09] combining [BGH$^+$05] with Dinur's gap amplification [Din07]. Second, the indirect aspect of the argument is reflected in the overhead in the reduction. For example, to obtain a lower bound against circuits of depth d, one needed a derandomization algorithm for circuits of depth cd for a constant $c > 1$.

1.1 Our Results

In this work we provide a variant of the PCP [BGH$^+$05] where the computation of the verifier is quite modest: Given randomness, the verifier computes its queries just by taking projections of the randomness, and the computation on the prover's answers is a 3CNF. The previous upper bound for these two computations was polynomial-size circuits.

Theorem 1 (Short PCPs with Projection Queries). *Let M be an algorithm running in time $T = T(n) \geq n$ on inputs of the form (x, y) where*

$|x| = n$. Given $x \in \{0,1\}^n$ one can output in time $\mathrm{poly}(n, \log T)$ *circuits* $\mathsf{Query} : \{0,1\}^r \to [2^r]^t$ *for* $t = \mathrm{poly}(r)$ *and* $\mathsf{Dec} : \{0,1\}^t \to \{0,1\}$ *such that:*

- **Proof length.** $2^r \leq T \cdot \mathrm{poly}\log T$,
- **Completeness.** *If there exists y such that $M(x, y)$ accepts then there exists a map $\pi : [2^r] \to \{0,1\}$ such that for any $z \in \{0,1\}^r$ we have* $\mathsf{Dec}(\pi(q_1), \ldots, \pi(q_t)) = 1$ *where* $(q_1, \ldots, q_t) = \mathsf{Query}(z)$,
- **Soundness.** *If no y causes $M(x, y)$ to accept, then for every map $\pi : [2^r] \to \{0,1\}$, at most $1/n^{10}$ fraction of the $z \in \{0,1\}^r$ have* $\mathsf{Dec}(\pi(q_1), \ldots, \pi(q_t)) = 1$ *where* $(q_1, \ldots, q_t) = \mathsf{Query}(z)$,
- **Complexity.** Query *is a projection (a.k.a. 1-local), i.e., each output bit of* Query *is one input bit, the negation of an input bit, or a constant;* Dec *is a 3CNF.*

The polynomial in the soundness item in Theorem 1 can be traded with the number t of queries.

There is a substantial literature that develops PCPs with optimized parameters. One focus of this literature has been to optimize the complexity of Dec. However typically these works do not produce PCPs of length quasilinear in T, and the complexity of Query is not optimized. Both these aspects are critical for our applications.

Remark 1 (Number of queries vs. Query *complexity).* Relaxing the complexity of Query to be a $\mathrm{poly}(r)$-computation allows to reduce the number of queries made to the oracle to a constant, while obtaining constant soundness [Mie09]. It is an interesting open problem to find the lowest complexity obtainable for Query in a PCP statement with proof length quasilinear in T, polylogarithmic verifier running time, and where soundness, alphabet, and number of queries are all constant. In particular, it is not clear to us if it is possible in such a case to have Query be a projection.

Along the way we give a more accessible presentation of [BGH+05]. Our presentation is modular and separates the combinatorial steps (given in Theorem 5) from the algebraic ones.

Taking Theorem 1 as a black box, we eliminate the roundabout argument mentioned before from the result in [SW13] that derandomizing TC^0 circuits on n bits in time $2^n/n^{\omega(1)}$ implies that NEXP is not in TC^0. Also, Theorem 1 is a small variant on [BGH+05], whereas as we mentioned [SW13] needs Mie's PCP [Mie09]. Finally, we also obtain the following alternative argument, which only uses the PCP result in [BGH+05] as a black-box.

The Alternative Argument. Given as a black-box a PCP such as [BGH+05], i.e., with the parameters as in Theorem 1 but where the complexity is replaced by polynomial-size circuits, we can construct a PCP where the verifier has low-complexity but makes adaptive queries to the proof. Specifically, we will rely on the prover to obtain the indices of our queries, and later query the prover at those indices and also verify the prover's computation, again with the help of the prover. This latter verification, as well as the computation Dec on the prover's answers, can be done by a 3CNF via a simple use of the Cook-Levin theorem – cf. Lemma 1.

Again, this alternative argument is sufficient to recover the TC^0 result in [SW13]. However, with Theorem 1 we obtain better parameters. Indeed, we seek very tight connections in the hope they will lead to progress on various challenges in computational lower bounds such as those mapped in [Vio13].

The concurrent work [Wil14] shows that the ability to count the number of satisfying assignments to circuits faster than brute-force search yields lower bounds against related circuits. This connection is used to obtain some new lower bounds. By our work the same lower bounds can be obtained from a satisfiability algorithm (using Theorem 3) or even a derandomization algorithm (using Theorem 2).

Next we state the tighter connections we obtain between derandomization and lower bounds. First we make a definition.

Definition 1. *Let C_n be a set of functions from $\{0,1\}^n$ to $\{0,1\}$. We say that C_n is* efficiently closed under projections *if functions in C_n have a* poly(n)*-size description and given (the description of) a function $f \in C_n$, indexes $i, j \leq n$, and a bit b, we can compute in time* poly(n) *the functions* not$f$*, $f(x_1, \ldots, x_{i-1}, b \text{ XOR } x_j, x_{i+1}, \ldots, x_n)$, and $f(x_1, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_n)$, all of which are in C_n.*

Most of the standard classes have this property. For the theorem, the two occurrences of "poly(n)" above can be relaxed. We also use the notation $\wedge_{\mathrm{poly}(n)} \vee_3 C_{n+O(\log n)}$ to indicates the And of poly(n) Or of 3 functions from $C_{n+O(\log n)}$, all on the same n input bits.

Theorem 2 (Derandomization Implies Lower Bounds, Tightly). *Let C_n be efficiently closed under projections.*

If the acceptance probability of functions of the form $h = \wedge_{\mathrm{poly}(n)} \vee_3 C_{n+O(\log n)}$ can be distinguished from being $= 1$ or $\leq 1/n^{10}$ in Time($2^n/n^{\omega(1)}$), then there is a function f in E^{NP} such that $f_n \notin C_n$.

One can place f in NEXP if we replace $C_{n+O(\log n)}$ with $C_{\mathrm{poly}(n)}$ and reason as in [IKW01,Wil13,Wil11].

The first step of our more modular exposition of [BGH$^+$05] is a reduction to 3SAT that builds on [JMV13] (cf. [BCGT13a]) but achieves incomparable guarantees (Theorem 5). Using that, we can obtain the following connection between satisfiability algorithms and lower bounds.

Theorem 3 (Satisfiability Implies Lower Bounds, Tightly). *Let C_n be efficiently closed under projections.*

If the satisfiability of functions $h = g_1 \wedge g_2 \wedge g_3$, where $g_i \in C_{n+O(\log n)}$ is in Time($2^n/n^{\omega(1)}$), then there is a function f in E^{NP} such that $f_n \notin C_n$.

The overhead to go from a satisfiability algorithm to a lower bound is evident from the theorem. The loss in size is a multiplicative factor $3 + o(1)$.

Previous losses were polynomial [Wil10], or multiplicative by a larger constant [JMV13]. The loss in depth is 2 for circuits with fan-in 2. For unbounded fan-in (or even fan-in 3) circuits with And gates (or threshold) the depth loss is 1. Previous losses were 2 [JMV13,Oli13].

Recall that the best lower bound for an explicit function on n bits is $3n - o(n)$ (non-input) gates [Blu84] (cf. [DK11]). This seems to be the best known even for functions in E^{NP} (note the number of circuits of size $3n$ is superlinear, so one cannot easily diagonalize against them in E^{NP}). By Theorem 3, to obtain a function in E^{NP} of circuit complexity $3n$ one would need to solve satisfiability of a circuit with $3(3n)$ non-input gates and n inputs – ignoring lower-order terms. The Cook-Levin theorem reduces this to a 3SAT instance on $9n + n = 10n$ variables. So one would need to solve 3SAT in deterministic time c^n for any $c < 2^{1/10} = 1.07\ldots$ The current record is $c = 1.33\ldots$ [MTY11], cf. [Her11].

1.2 Techniques

Ideas behind the proof of Theorem 1. We start with the PCP in [BGH+05] and follow its proof closely. There are two computations of the verifier in this PCP that we need to optimize. The first — Query — is taking the input randomness to the queries, which we call *preprocess*. The second — Dec — is the computation on the prover's answers, which we call *postprocess*. We discuss them separately.

Postprocess: It is a common experience in theoretical computer science research, to study at length an intricate proof in the reckless hope of optimizing parameters, only to be surprised by the late realization that a trivial, sweeping argument takes the complex proof as a black-box and gets a pretty good parameter optimization, too.

Lemma 1 (Making Dec a 3CNF). *Suppose Theorem 1 holds except that* Dec *is an unrestricted circuit of size* $\mathrm{poly}(r)$*. Then Theorem 1 holds as stated.*

Proof. By the Cook-Levin theorem we reduce Dec to a 3CNF with $\mathrm{poly}(r)$ variables and terms. The verifier will ask the prover for an additional $\mathrm{poly}(r)$ queries to obtain the satisfying assignments for this 3CNF corresponding to the input randomness. On input z, these queries are of the form (z, i), where i is an $O(\log r)$-bit index to a variable in the 3CNF. The proof contains the values of the variables in the 3CNF that verify the computation on the outputs of the queries that are made by the verifier on input z.

This general technique shows that we can always make the postprocess a 3CNF as long as we allow for $\mathrm{poly}(r)$ queries. Using it, there is no benefit in reducing the number of queries to a constant.

Preprocess: This is in turn comprised of two parts, acting in parallel, known as "algebraic constraint satisfaction" and "low-degree testing", and in this work we offer a clear separation between the two. In the first part we reduce the succinct constraint satisfaction problem (CSP) associated with verifying the M accepts x in T steps, to an algebraic CSP (ACSP) problem, one stated as a question about equality of polynomials. We offer a definition of ACSP that is algebraically cleaner than [BS08] and following works (e.g., [BGH$^+$05,BCGT13b]). In particular, previous definitions included degree bounds on the "assignment polynomial" and involved a "zero-testing" problem. In contrast, we define a satisfying assignment as one that causes a polynomial to vanish, and degree-bounds are dealt with by the separate low-degree testing part, discussed later. We now elaborate on how we obtain efficient preprocessing in each of the two parts.

In the ACSP part, our verifier simply selects a random field element α, generates poly(r) queries to the prover where the ith query is $\alpha + \sigma_i$ where σ_i is fixed and independent of α. Each such query can be verified to be a projection. To reach this simple form of preprocessing we use a modular reduction from the combinatorial succinct CSP captured by Theorem 4 to the succinct ACSP. The mid-point between the combinatorial and algebraic settings is given in Theorem 5. In it we reach a 3CNF formula with $\approx T$ clauses where each clause (i.e., the three variables of the clause and their polarities) can be computed by a simple XOR operation. Since XOR is addition in fields of characteristic 2, irrespective of the basis chosen for them, we get a simpler ACSP than [BS08,BGH$^+$05] albeit one that has a super-constant number of variables.

Turning to the second part, low-degree testing, we use auxiliary information in form of a PCP of Proximity (PCPP) [BSGH$^+$06,DR06]. This part is essentially from [BS08] and regrettably remains an intricate step of the proof. The answers to queries of the verifier in [BS08] can be seen as arranged in the nodes of a tree. The query at each node is indeed a projection. However, the verifier uses part of its input randomness to select a path in this tree, reaching a leaf, then possibly redirects the query to a node higher up in the tree. This computation is more complicated than just a projection. Here we use the following simple, key idea. The path from root to leaf is determined by only $O(\log r)$ of the verifier's r input bits. Additionally, the process of redirecting a query from a leaf to a node elsewhere in the tree is also determined by only $O(\log r)$ input bits. Instead of following the path, we let the verifier query every possible endpoint. This multiplies the number of queries by a factor poly(r), which we can afford. We delegate the task of picking the right query to the postprocess.

One more complication is that each of the two parts needs to be combined with a randomness-efficient hitter to achieve constant soundness. Using e.g. Cayley expanders built from small-bias sets, this step is again just a projection.

Ideas Behind the Proof of Theorem 3. A natural idea is to improve the previous constant-locality result [JMV13] to locality 1. But this may not be possible. Instead, we show how to reduce arbitrary computation to a polynomial number of 3CNF formulae, each of which has locality 1. By enumerating over these 3CNF, and running the satisfiability algorithm on each of them, we get the result. This idea is similar to the one described above to make [BS08] a projection: after reading a logarithmic number of bits, the rest of the computation becomes just a projection.

Open Problems. Improve Theorem 2 to have the same overhead as Theorem 3.

Organization. In §2 we give a variant of the reduction of non-deterministic time to 3SAT given in [JMV13]. Using that and Theorem 1 as a black-box, in §3 we give the proofs of theorems 2 and 3. Due to space restrictions we refer to the full version for the proof of our main Theorem 1.

2 A Combinatorial Reduction to 3SAT

Our starting point is the following result from [JMV13].

Theorem 4 ([JMV13]). *Let M be an algorithm running in time $T = T(n) \geq n$ on inputs of the form (x, y) where $|x| = n$. Given $x \in \{0,1\}^n$ one can output a circuit $D : \{0,1\}^{\ell} \to \{0,1\}^{3v+3}$ in time* $\mathrm{poly}(n, \log T)$ *mapping an index to a clause of a 3CNF ϕ in v-bit variables, for $v = \Theta(\ell)$, such that*

1. *ϕ is satisfiable iff there is $y \in \{0,1\}^T$ such that $M(x, y)$ accepts, and*
2. *For any $r \leq n$ we can have $\ell = \max(\log T, n/r) + O(\log n) + O(\log\log T)$ and each output bit of D is a decision tree of depth $O(\log r)$.*

Note that for $T = 2^n$ and $r = O(1)$ this gives a 3CNF with $T\mathrm{poly}\log T$ clauses such that each clause can be computed from its index by a function with constant locality.

We need an incomparable variant of the latter. We enlarge the locality to $O(\log n)$, but at the same time there are only $O(\log n)$ input bits that affect more than 1 bit. If we fix these bits, the rest of the computation is just a bit-wise xor.

Theorem 5. *Let M be an algorithm running in time $T = T(n) \geq n$ on inputs of the form (x, y) where $|x| = n$. Let $\ell_1 = \log T$. For some $\ell_2 = O(\log\log T) + O(\log n)$ the following is true. Given $x \in \{0,1\}^n$ one can output in time* $\mathrm{poly}(n, \log T)$ *six circuits (of size* $\mathrm{poly}(n, \log T)$*): $S_i : \{0,1\}^{\ell_2} \to \{0,1\}^{\ell_1+\ell_2}, b_i : \{0,1\}^{\ell_2} \to \{0,1\}$, for $i = 1, 2, 3$, such that:*

Let ϕ_x be the 3CNF with $2^{\ell_1+\ell_2} = T\mathrm{poly}(n, \log T)$ clauses (and variables) whose $(\alpha, \beta) \in \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2}$ clause contains variables $V_i = (\alpha, \beta) \oplus S_i(\beta)$ and corresponding sign bits $b_i = b_i(\beta)$, where $i = 1, 2, 3$ and $\oplus$ is bit-wise xor. Then ϕ_x is satisfiable iff there is $y \in \{0,1\}^T$ such that $M(x, y)$ accepts.

Note that in the case $T = 2^{O(n)}$ each output bit of D depends only on $|\beta|+1 = O(\log n)$ bits of the input.

The next proof heavily builds on previous works. We give a sketch that highlights the tiny changes from previous proofs, and to work out parameters. The closest previous proof is [JMV13], to which we also refer for a discussion of other related works.

Proof (Proof sketch). Without loss of generality the algorithm is implemented by a random-access Turing machine running in time $T' = T\mathrm{poly}\log n$ and only using memory cells at indexes $\leq \mathrm{poly}(T)$ (see e.g. [NEU12] for details).

Consider a circuit-sat instance where the circuit first guesses a computation trace consisting of T' configurations of size $O(\log T)$ each; and then the circuit checks its validity and acceptance. The validity check consists of two separate checks. The first is the check of the consistency of the transition function of the machine, assuming that memory reads are correct. The second is the check of the consistency of the memory reads and writes. The trace is valid if and only if both checks pass. These two checks are implemented in a similar fashion; we only describe the second.

Consider a matrix of $r \times T'$ configurations, where $r = \operatorname{poly}\log T$. We use α to index a column in this matrix. (This actually gives $|\alpha| = \ell_1 = \log T' = \log T + O(\log\log T)$, but the low-order summand can be swallowed in ℓ_2.) We use β to index a row, and the gates within the subcircuits discussed next.

The first row is the computation trace mentioned above that the circuit guessed. For every $t = 1, \ldots, T$ we have a $\operatorname{poly}(n, \log T)$-size subcircuit which checks the pair of configurations (C, C') at positions $(1, t)$ and (r, t) in the matrix, i.e., in the same column but at antipodal rows. This subcircuit verifies that either C' accesses the same memory cell of C and has the timestamp of C plus one, or C' accesses a memory cell with index greater than that of the cell accessed by C, or – the wrap-around condition – it does nothing if C' is the configuration with timestamp 0. If all these checks pass then for every t the configuration at position (t, r) is the one that comes next the configuration at position $(t, 1)$ in the order given by memory location accessed breaking ties by timestamp. The subcircuit then verifies consistency of the memory read and write in C and C'. In particular it verifies that cells read for the first time are blank, and that the cells $1, \ldots, n$ read for the first time contain the input x. Note that the latter is possible because our circuits have size $\geq n$ and are built with knowledge of x.

Observe that these subcircuits operate independently on each column, and are identical across columns. Their connections depend only on the row index and an index to one of their gates. By including these two indexes inside β, these connections can be computed in the required format.

It remains to discuss connections across columns, which are needed to move configurations around to put them in the right order. For this we use routing networks such as Beneš', which are a simple composition of butterfly networks. The index of a neighbor in column α is obtained by xoring α with a string (of Hamming weight 1) which only depends on the row, which in turn is part of β. This leads to the desired format for V_i. The implementation of the routing network also needs a simple gadget to swap two configurations depending on a nondeterministic bit. This gadget is the same for every column and row. From this it follows that the V_i and b_i are in the desired format.

3 Lower Bounds from Fast Algorithms

In this section we prove theorems 2 and 3. First we restate the theorem.

Theorem 3 (Satisfiability Implies Lower Bounds, Tightly). *Let C_n be efficiently closed under projections.*

If the satisfiability of functions $h = g_1 \wedge g_2 \wedge g_3$, where $g_i \in C_{n+O(\log n)}$ is in Time($2^n/n^{\omega(1)}$), then there is a function f in E^{NP} such that $f_n \notin C_n$.

For the proof we use Theorem 5, and we *enumerate* over the β in its statement. The key observation is that for any fixed β, the reduction is only computing xor which can be hardwired with no loss in resources. This enumeration is feasible because $|\beta| = O(\log n)$. To get better constants we also work with unary languages.

Proof (Proof of Theorem 3). Suppose that every function in E^{NP} belongs to C_n when restricted to inputs of length n. Let L be a unary language in $\mathrm{NTime}(2^n) \setminus \mathrm{NTime}(o(2^n))$ [Coo73,SFM78,Zák83]. Consider the E^{NP} algorithm that on input $x' \in \{0,1\}^{O(\log n)}$ and $i \leq 2^n \mathrm{poly}(n)$ computes $x = 1^{x'}$, the 3CNF ϕ_x corresponding to L through Theorem 5, computes its first satisfying assignment if one exists, and outputs its ith bit. By assumption, on inputs of length $m = n + O(\log n)$ this function is in C_m. Also, by assumption, if we hardwire x' the resulting function still belongs to C_m. Call this function g_x.

We contradict the assumption on L by showing how to decide it in $\mathrm{Ntime}(o(2^n))$. Let D, S_i, α, β, V_i and b_i be as in Theorem 5. Consider the algorithm that on input $x = 1^n$ guesses g_x. Then it constructs the function g'_x that operates as follows. The input is that of D. Then it connects three copies of g_x to the output variables V_i. Further, the output of the ith copy is negated and then xored with b_i. And finally an And is taken. Call g'_x this new function (which may not belong to any C_n). Note that $g'_x(i) = 1$ iff the ith clause is not satisfied (by satisfying assignment g_x). So by determining the satisfiability of g'_x we can determine if $x \in L$ or not.

The satisfiability algorithm enumerates over all $\mathrm{poly}(n)$ choices for β. For each fixed β, the b_i are determined, and the remaining computation to obtain the V_i is an xor by $S_i(\beta)$. All this can be hardwired into g'_x in time $\mathrm{poly}(m)$, because C_m is efficiently closed under projections. For every i this gives a new function $g_i \in C_m$. There remains to solve the satisfiability of $g_1 \wedge g_2 \wedge g_3$. The latter can be done in time $2^m/m^{\omega(1)}$ by assumption. So overall the running time is $\mathrm{poly}(n, m) 2^m/m^{\omega(1)} = 2^n/n^{\omega(1)} = o(2^n)$.

Theorem 2 (Derandomization Implies Lower Bounds, Tightly). *Let C_n be efficiently closed under projections.*

If the acceptance probability of functions of the form $h = \wedge_{\mathrm{poly}(n)} \vee_3 C_{n+O(\log n)}$ can be distinguished from being $= 1$ or $\leq 1/n^{10}$ in Time($2^n/n^{\omega(1)}$), then there is a function f in E^{NP} such that $f_n \notin C_n$.

Proof (Proof Sketch). Proceed as the proof of Theorem 3, but let ϕ_x be instead of the 3CNF produced by Theorem 5 the constraint satisfaction problem corresponding to our main theorem, 1. As before, we obtain a function g'_x that on input i determines if the ith constraint is satisfied. (To show that the complexity of this function is as desired, we merge the Not gates of the 3CNF corresponding to Dec with the circuits in $C_{n+O(\log n)}$, using the closure of the class.) Thus, approximately determining how many constraints are satisfied amounts to approximately determinining the number of satisfying assignments to g'_x.

Acknowledgements. The first co-author is grateful to the SCIPR-lab (www.scipr-lab.org) members — Alessandro Chiesa, Daniel Genkin, Lior Greenblatt, Shaul Kfir, Michael Riabzev, Gil Timnat, Eran Tromer and Madars Virza — for helpful discussions. The second co-author thanks Ramamohan Paturi for a conversation on MAX3SAT.

References

AGHP92. Alon, N., Goldreich, O., Håstad, J., Peralta, R.: Simple constructions of almost k-wise independent random variables. Random Structures & Algorithms 3(3), 289–304 (1992)

ASE92. Alon, N., Spencer, J.H., Erdős, P.: The Probabilistic Method. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley and Sons, Inc. (1992)

BCGT13a. Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E.: Fast reductions from RAMs to delegatable succinct constraint satisfaction problems. In: ACM Innovations in Theoretical Computer Science Conf. (ITCS), pp. 401–414 (2013)

BCGT13b. Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E.: On the concrete efficiency of probabilistically-checkable proofs. In: ACM Symp. on the Theory of Computing (STOC), pp. 585–594 (2013)

BGH$^+$05. Ben-Sasson, E., Goldreich, O., Harsha, P., Sudan, M., Vadhan, S.P.: Short PCPs verifiable in polylogarithmic time. In: IEEE Conf. on Computational Complexity (CCC), pp. 120–134 (2005)

Blu84. Blum, N.: A boolean function requiring 3n network size. Theoretical Computer Science 28, 337–345 (1984)

BS08. Ben-Sasson, E., Sudan, M.: Short PCPs with polylog query complexity. SIAM J. on Computing 38(2), 551–607 (2008)

BSGH$^+$06. Ben-Sasson, E., Goldreich, O., Harsha, P., Sudan, M., Vadhan, S.P.: Robust PCPs of proximity, shorter PCPs, and applications to coding. SIAM J. on Computing 36(4), 889–974 (2006)

Coo73. Cook, S.A.: A hierarchy for nondeterministic time complexity. J. of Computer and System Sciences 7(4), 343–353 (1973)

Din07. Dinur, I.: The PCP theorem by gap amplification. J. of the ACM 54(3), 12 (2007)

DK11. Demenkov, E., Kulikov, A.S.: An elementary proof of a $3n - o(n)$ lower bound on the circuit complexity of affine dispersers. In: Symp. on Math. Foundations of Computer Science (MFCS), pp. 256–265 (2011)

DR06. Dinur, I., Reingold, O.: Assignment testers: Towards a combinatorial proof of the PCP theorem. SIAM J. on Computing 36(4), 975–1024 (2006)

FSUV13. Fefferman, B., Shaltiel, R., Umans, C., Viola, E.: On beating the hybrid argument. Theory of Computing 9, 809–843 (2013)

GMR13. Gopalan, P., Meka, R., Reingold, O.: DNF sparsification and a faster deterministic counting algorithm. Computational Complexity 22(2), 275–310 (2013)

Her11. Hertli, T.: 3-SAT faster and simpler - unique-SAT bounds for PPSZ hold in general. In: IEEE Symp. on Foundations of Computer Science (FOCS), pp. 277–284 (2011)

IKW01. Impagliazzo, R., Kabanets, V., Wigderson, A.: In search of an easy witness: Exponential time vs. probabilistic polynomial time. In: IEEE Conf. on Computational Complexity (CCC) (2001)

IKW02. Impagliazzo, R., Kabanets, V., Wigderson, A.: In search of an easy witness: exponential time vs. probabilistic polynomial time. J. of Computer and System Sciences 65(4), 672–694 (2002)

JMV13. Jahanjou, H., Miles, E., Viola, E.: Local reductions (2013), `http://www.ccs.neu.edu/home/viola/`

KL80. Karp, R.M., Lipton, R.J.: Some connections between nonuniform and uniform complexity classes. In: ACM Symp. on the Theory of Computing (STOC), pp. 302–309 (1980)

LVW93. Luby, M., Veličković, B., Wigderson, A.: Deterministic approximate counting of depth-2 circuits. In: 2nd Israeli Symposium on Theoretical Computer Science (ISTCS), pp. 18–24 (1993)

Mie09. Mie, T.: Short pcpps verifiable in polylogarithmic time with $o(1)$ queries. Ann. Math. Artif. Intell. 56(3-4), 313–338 (2009)

MTY11. Makino, K., Tamaki, S., Yamamoto, M.: Derandomizing HSSW algorithm for 3-SAT. CoRR, abs/1102.3766 (2011)

NEU12. NEU. From RAM to SAT (2012), `http://www.ccs.neu.edu/home/viola/`

NN90. Naor, J., Naor, M.: Small-bias probability spaces: efficient constructions and applications. In: 22nd ACM Symp. on the Theory of Computing (STOC), pp. 213–223. ACM (1990)

Oli13. Oliveira, I.C.: Algorithms versus circuit lower bounds. CoRR, abs/1309.0249 (2013)

SFM78. Seiferas, J.I., Fischer, M.J., Meyer, A.R.: Separating nondeterministic time complexity classes. J. of the ACM 25(1), 146–167 (1978)

SW13. Santhanam, R., Williams, R.: On medium-uniformity and circuit lower bounds. In: IEEE Conf. on Computational Complexity (CCC) (2013)

Vio07. Viola, E.: Pseudorandom bits for constant-depth circuits with few arbitrary symmetric gates. SIAM J. on Computing 36(5), 1387–1403 (2007)

Vio13. Viola, E.: Challenges in computational lower bounds (2013), `http://www.ccs.neu.edu/home/viola/`

Wil05. Williams, R.: A new algorithm for optimal 2-constraint satisfaction and its implications. Theoretical Computer Science 348(2-3), 357–365 (2005)

Wil10. Williams, R.: Improving exhaustive search implies superpolynomial lower bounds. In: 42nd ACM Symp. on the Theory of Computing (STOC), pp. 231–240 (2010)

Wil11. Williams, R.: Non-uniform ACC circuit lower bounds. In: IEEE Conf. on Computational Complexity (CCC), pp. 115–125 (2011)

Wil13. Williams, R.: Improving exhaustive search implies superpolynomial lower bounds. SIAM J. on Computing 42(3), 1218–1244 (2013)

Wil14. Williams, R.: New algorithms and lower bounds for circuits with linear threshold gates. In: ACM Symp. on the Theory of Computing, STOC (2014)

Zák83. Zák, S.: A turing machine time hierarchy. Theoretical Computer Science 26, 327–333 (1983)

Star Partitions of Perfect Graphs

René van Bevern[1,⋆], Robert Bredereck[1,⋆⋆], Laurent Bulteau[1,⋆⋆⋆],
Jiehua Chen[1,†], Vincent Froese[1,‡], Rolf Niedermeier[1],
and Gerhard J. Woeginger[2,§]

[1] Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Germany
{rene.vanbevern,robert.bredereck,jiehua.chen,vincent.froese}@tu-berlin.de,
l.bulteau@campus.tu-berlin.de
[2] Department of Mathematics and Computer Science, TU Eindhoven,
The Netherlands
gwoegi@win.tue.nl

Abstract. The partition of graphs into nice subgraphs is a central algorithmic problem with strong ties to matching theory. We study the partitioning of undirected graphs into stars, a problem known to be NP-complete even for the case of stars on three vertices. We perform a thorough computational complexity study of the problem on subclasses of perfect graphs and identify several polynomial-time solvable cases, for example, on interval graphs and bipartite permutation graphs, and also NP-hard cases, for example, on grid graphs and chordal graphs.

1 Introduction

We study the computational complexity (tractable versus intractable cases) of the following basic graph problem.

Star Partition
Input: An undirected n-vertex graph $G = (V, E)$ and an integer $s \in \mathbb{N}$.
Question: Can the vertex set V be partitioned into $k := \lceil n/(s+1) \rceil$ disjoint subsets $V_1, V_2, \ldots, V_k$, such that each subgraph $G[V_i]$ contains an s-star (a $K_{1,s}$)?

Two prominent special cases of Star Partition are the case $s = 1$ (finding a perfect matching) and the case $s = 2$ (finding a partition into connected triples). Perfect matchings ($s = 1$), of course, can be found in polynomial time. Partitions into connected triples (the case $s = 2$), however, are hard to find; this problem, denoted P_3-Partition, was proven to be NP-complete by Kirkpatrick and Hell [13].

[⋆] Supported by the DFG, project DAPA (NI 369/12).
[⋆⋆] Supported by the DFG, project PAWS (NI 369/10).
[⋆⋆⋆] Supported by the Alexander von Humboldt Foundation.
[†] Main work done while supported by the Studienstiftung des Deutschen Volkes.
[‡] Supported by the DFG, project DAMM (NI 369/13).
[§] Main work done while supported by the Alexander von Humboldt Foundation.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 174–185, 2014.

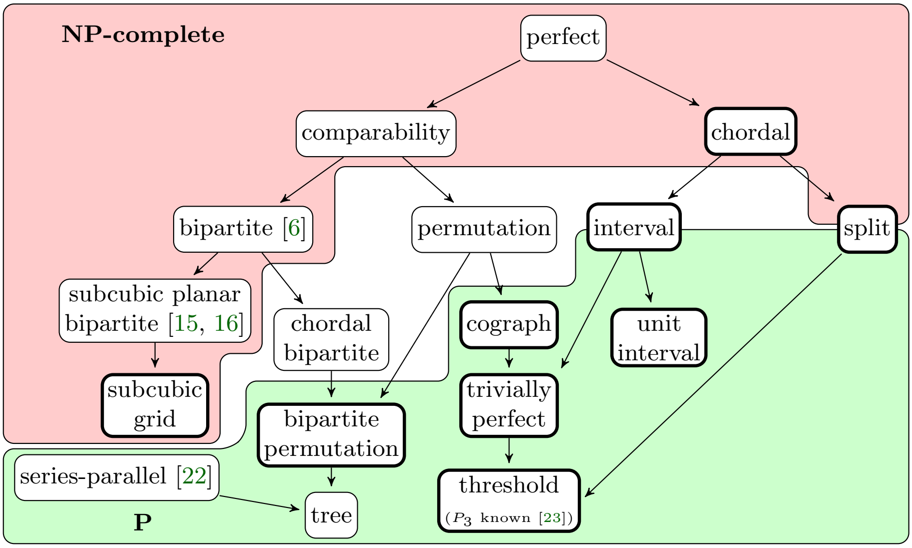


Fig. 1. Complexity classification of Star Partition. Bold borders indicate results of this paper. An arrow from a class A to a class B indicates that A contains B. In most classes, NP-completeness results hold for $s = 2$ (that is, for P_3-Partition). However, on split graphs, Star Partition is polynomial-time solvable for $s \leq 2$, while it is NP-complete for $s \geq 3$. P_3-Partition is solvable on interval graphs in quasilinear time. We are not aware of any result for permutation graphs, chordal bipartite graphs or interval graphs when $s \geq 3$.

Our goal in this paper is to achieve a better understanding of star partitions for certain classes of perfect graphs. We provide a fairly complete classification in terms of polynomial-time solvability versus NP-completeness on the most prominent subclasses of perfect graphs, leaving a few potentially challenging cases open; see Figure 1 for an overview of our results.

Motivation. The literature in algorithmic graph theory is full of packing and partitioning problems. From a more applied point of view, P_3-Packing and P_3-Partition find applications in dividing distributed systems into subsystems [14] as well as in the Test Cover problem arising in bioinformatics [10]. In particular, the application in distributed systems explicitly motivates the consideration of very restricted (perfect) graph classes such as grid-like structures. Star Partition on grid graphs naturally occurs in political redistricting problems [4]. We show that Star Partition remains NP-complete on subcubic grid graphs.

Interval graphs are another famous class of perfect graphs. Here, Star Partition can be considered a team formation problem: Assume that we have a number of agents, each being active during a certain time interval. Our goal is to form teams, all of same size, such that each team contains at least one agent sharing time with every other team member. This specific team member becomes the team leader, since he or she can act as an information hub. Forming such teams is nothing else than solving Star Partition on interval graphs. We present efficient algorithms for Star Partition on unit interval graphs (that is, for the case when all agents are active for the same amount of time) and for P_3-Partition on general interval graphs.

Previous Work. Packing and partitioning problems are central problems in algorithmic graph theory with many applications and with close connections to matching theory [25]. In the case of packing, one wants to maximize the number of graph vertices that are "covered" by vertex-disjoint copies of some fixed pattern graph H. In the case of partitioning, one wants to cover *all* vertices in the graph. We focus on the partitioning problem, which is also called H-Factor in the literature. In this work, we always refer to it as H-Partition. As Kirkpatrick and Hell [13] established the NP-completeness of H-Partition on general graphs for every connected pattern H with at least three vertices, one branch of research has turned to the investigation of classes of specially structured graphs. For instance, on the upside, H-Partition has been shown to be polynomial-time solvable on trees and series-parallel graphs [22] and on graphs of maximum degree two [16]. On the downside, P_k-Partition (for fixed $k \geq 3$) remains NP-complete on planar bipartite graphs [11]; this hardness result generalizes to H-Partition on planar graphs for any outerplanar pattern H with at least three vertices [2]. For every fixed $s \geq 2$, Star Partition is NP-hard on bipartite graphs [6]. Partitioning into triangles, that is, K_3-Partition, is polynomial-time solvable on chordal graphs [9] and linear-time solvable on graphs of maximum degree three [17].

Optimization versions of P_k-Partition, called Min P_k-Partition, have also received considerable interest in the literature. This version asks for a partition of a given graph into a minimum number of paths of length *at most* k. Clearly, all hardness results for P_k-Partition carry over to the minimization version. If k is *part of* the input, then Min P_k-Partition is hard for cographs [20] and chordal bipartite graphs [21]. In fact, Min P_k-Partition is NP-hard even on convex graphs and trivially perfect graphs (also known as quasi-threshold graphs), and hence on interval and chordal graphs [1]. Min P_k-Partition is solvable in polynomial time on trees [24], threshold graphs, cographs (for fixed k) [20] and bipartite permutation graphs [21].

Our Contributions. So far, surprisingly little is known about the complexity of Star Partition for subclasses of perfect graphs. We provide a detailed picture of the complexity landscape of perfect graphs; see Figure 1 for an overview. Let us briefly summarize some of our results.

As a central result, we provide a quasilinear-time algorithm for P_3-Partition, which is Star Partition with $s = 2$, on interval graphs; the complexity of Star Partition for $s \geq 3$ remains open. Furthermore, we develop a polynomial-time algorithm for Star Partition on cographs. Most of our polynomial-time algorithms are simple to describe: they are based on dynamic programming or even on greedy approaches, and hence should work well in implementations. Their correctness proofs, however, are intricate.

On the boundary of NP-hardness, we strengthen a result of Małafiejski and Żyliński [15] and Monnot and Toulouse [16] by showing that P_3-Partition is NP-complete on grid graphs with maximum degree three. Note that in strong contrast to this, K_3-Partition is linear-time solvable on graphs with maximum degree three [17]. Furthermore, we show P_3-Partition to be NP-complete on chordal graphs, while K_3-Partition is known to be polynomial-time solvable in this case [9]. We observe that P_3-Partition is typically not easier than Star Partition for $s \geq 3$. An exception to this rule is provided by the class of split graphs, where P_3-Partition is polynomial-time solvable but Star Partition is NP-complete for any constant value $s \geq 3$. Due to space constraints, most of our proofs are deferred to a full version [3].

Preliminaries. We assume basic familiarity with standard graph classes [5, 12]. Definitions of the graph classes are provided when first studied in this paper. We call the complete bipartite graph $K_{1,s}$ an *s-star* . For a graph $G = (V, E)$, an *s-star partition* is a set of $k := |V|/(s + 1)$ pairwise disjoint vertex subsets $V_1, V_2, \ldots, V_k \subseteq V$ with $\bigcup_{1 \leq i \leq k} V_i = V$ such that each subgraph $G[V_i]$ contains an s-star as a (not necessarily induced) subgraph. We refer to the vertex sets V_i as *stars*, even though the correct description of a star would be *arbitrary $K_{1,s}$-subgraph of $G[V_i]$*. P_3-Partition is the special case of Star Partition with $s = 2$. Without loss of generality, we assume throughout the paper that the input graph G is connected (otherwise, we can solve the partition problem separately for each connected component of G). We denote by $n := |V|$ the number of vertices and by $m := |E|$ the number of edges in a graph $G = (V, E)$.

2 Interval Graphs

In this section, we present algorithms that solve Star Partition on unit interval graphs in linear time and P_3-Partition on interval graphs in quasilinear time.

An *interval graph* is a graph whose vertices one-to-one correspond to intervals on the real line such that there is an edge between two vertices if and only if their representing intervals intersect. In a *unit interval graph*, all representing intervals are open and have the same length.

Star Partition on Unit Interval Graphs

The restricted structure of unit interval graphs allows us to solve Star Partition using a simple greedy approach: repeatedly select the $s + 1$ leftmost intervals to form an s-star and then delete them. If, at some point, the $s+1$ leftmost intervals do not contain an s-star, it can be shown that the graph cannot be partitioned into s-stars. This algorithm yields the following result.

Theorem 1. Star Partition *is $O(n+m)$ time solvable on unit interval graphs.*

Algorithm 1. P_3-partition of an interval graph

```
Input: An interval representation of an interval graph with pairwise distinct
       event points in $\{1, \ldots, 2n\}$.
Output: true if the graph allows for a $P_3$-partition, otherwise false.
$A_0 \leftarrow$ empty token list $\emptyset$;
for $t \leftarrow 1$ to $2n$ do
    if $t = \text{start}(x)$ then $A_t \leftarrow A_{t-1} \oplus (x, x)$ if $t = \text{end}(x)$ then
        if $x \notin A_{t-1}$ then $A_t \leftarrow A_{t-1}$ else if $\|A_{t-1}\| < 3$ then return false
        else
            $(x, y, z) \leftarrow$ lowest three elements of $A_{t-1}$ (intervals ending first);
            $A_t \leftarrow A_{t-1} \ominus (x, y, z)$;
return true;
```

P_3-Partition on Interval Graphs

While it might not come as a surprise that Star Partition can be solved efficiently on unit interval graphs using a greedy strategy, this is far from obvious for general interval graphs. The obstacle here is that two intervals arbitrarily far apart from each other may eventually be required to form a P_3 in the solution. Indeed, the greedy strategy we propose to overcome this obstacle is naive in the sense of allowing wrong choices that can be corrected later. Note that, while we can solve the more general Star Partition in polynomial time on subclasses of interval graphs like unit interval graphs and trivially perfect graphs (see Figure 1), we are not aware of a polynomial-time algorithm for Star Partition with $s \geq 3$ on interval graphs.

Overview of the Algorithm. The algorithm is based on the following analysis of a P_3-partition of an interval graph. Each P_3 contains a *center* and two *leaves* connected to the center via edges called *links*. We associate with each interval two so-called *tokens*. We require that the link between a leaf and a center *consumes* both of the leaf's tokens (such that a leaf can have only one link) and one token of the center (which can thus be linked to two leaves).

The algorithm examines the *event points* (start and end points of intervals) of an interval representation in increasing order. We consider that a link $\{x, y\}$ consumes the tokens of x and y as soon as one of the two intervals ends. Intuitively, a graph is a no-instance if, at some point, an interval with one or two remaining tokens ends, but there are not enough tokens of other adjacent intervals to *create* a link. Note that a link consumes three tokens. A graph is a yes-instance if the number of tokens is always sufficient.

The algorithm works according to the following two rules: when an interval starts, its two tokens are added to a list; when an interval with remaining tokens ends, then three tokens are deleted from this list. Only tokens of the earliest-ending intervals will be deleted (this choice may not directly translate into a "sane" solution, with each link consuming tokens from only two intervals, but it

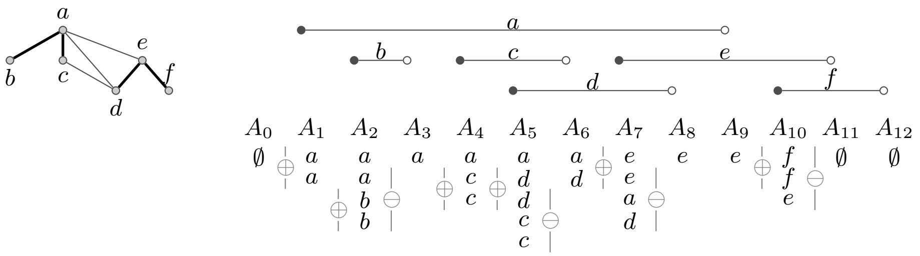

a
b
c
d
e
f
A_0 A_1 A_2 A_3 A_4 A_5 A_6 A_7 A_8 A_9 A_{10} A_{11} A_{12}

Fig. 2. Left: An interval graph with six vertices and a P_3-partition $\mathcal{P}$ (bold). Right: Interval representation of this graph and successive token lists $A_0, \ldots, A_{12}$ computed by Algorithm 1 (additions and deletions are marked with $\oplus$ and $\ominus$).

turns out not to be problem). The algorithm is sketched in Algorithm 1. Figure 2 shows an example instance and the list of tokens maintained by the algorithm. Note that a token of an interval x is simply represented by a copy of interval x itself. We now introduce the necessary formal definitions.

Definitions. We consider a fixed interval graph $G = (V, E)$. We assume that any vertex $u \in V$ represents a right-open interval $u = [\text{start}(u), \text{end}(u)[$ with integer endpoints $\text{start}(u) < \text{end}(u)$. Moreover, without loss of generality, each position in $(1, \ldots, 2n)$ corresponds to exactly one event.

Let $\mathcal{P}$ be a P_3-partition and $P = \{x, y, z\} \in \mathcal{P}$ with $\text{end}(x) < \text{end}(y) < \text{end}(z)$, we write $\text{rank}_{\mathcal{P}}(x) = 1$, $\text{rank}_{\mathcal{P}}(y) = 2$, and $\text{rank}_{\mathcal{P}}(z) = 3$ (we omit the subscript when there is no ambiguity). Moreover, we call the element among $\{y, z\}$ having the earliest start point the *center* of P. The other two elements of P are called *leaves.* Note that the center of P intersects both leaves.

A *token list* Q is a list of intervals $(q_1, \ldots, q_k)$ sorted in decreasing order of their end points ($\text{end}(q_i) \geq \text{end}(q_j)$ for $1 \leq i \leq j \leq k$). To avoid confusion with the left-to-right sequence of event points, we consider the list to be written vertically, with the latest-ending interval on top. We write $\|Q\|$ for the length of Q, $\emptyset$ for the empty token list, and $x \in Q$ if interval x appears in Q. We now define *insertion* $\oplus$, *deletion* $\ominus$, and *comparison* $\preccurlyeq$ of token lists: $Q \oplus (x_1, \ldots, x_l)$ is the token list obtained from Q by inserting intervals $x_1 \ldots, x_l$ so that the list remains sorted. For $x \in Q$, the list $Q \ominus x$ is obtained by deleting one copy of x from Q (otherwise, $Q \ominus x = Q$); and $Q \ominus (x_1, \ldots, x_l) = Q \ominus x_1 \ominus \ldots \ominus x_l$. We write $(q_1, \ldots, q_k) \preccurlyeq (q'_1, \ldots, q'_{k'})$ if $k \leq k'$ and $\forall i \in \{1, \ldots k\}, \text{end}(q_i) \leq \text{end}(q'_i)$.

Let $\mathcal{P}$ be a P_3-partition. We define tokens($\mathcal{P}$) as a tuple of $2n + 1$ token lists $(T_0, T_1, \ldots, T_{2n})$ such that $T_0 := \emptyset$ and for $t > 0$,

- if $t = \text{start}(x)$, then $T_t := T_{t-1} \oplus (x, x)$,
- if $t = \text{end}(x)$, then let $P := \{x, y, z\}$ be the P_3 in $\mathcal{P}$ containing x and
  - if $\text{rank}(x) = 1$, then $T_t := T_{t-1} \ominus (x, x, c)$ where c is the center of P,
  - if $\text{rank}(x) = 2$, then $T_t := T_{t-1} \ominus (x, x, y, y, z, z)$,
  - if $\text{rank}(x) = 3$, then $T_t := T_{t-1}$.

Note that in Figure 2, each token list T_t for $\mathcal{P}$ is equal to the respective A_t, except for $T_6 = (d, d)$ and $T_7 = (e, e, d, d)$.

The following lemmas state that, on the one hand, if there is a P_3-partition, then each token list created by Algorithm 1 is comparable with the corresponding T_t, hence it always contains enough tokens to create the next list, up to A_{2n}, and answer "`true`" in the end. On the other hand, if the algorithm returns "`true`", then it is indeed possible to construct a P_3-partition using (indirectly) the triples of intervals removed from the token list to create the links.

Lemma 1. *If an interval graph G has a P_3-partition $\mathcal{P}$, then for all $0 \leq t \leq 2n$, Algorithm 1 defines set A_t with $T_t \preccurlyeq A_t$ and $\|T_t\| - \|A_t\| \equiv 0 \pmod 3$, where* $\text{tokens}(\mathcal{P}) = (T_0, T_1, \ldots, T_{2n})$.

Lemma 2. *Let G be an interval graph such that Algorithm 1 returns `true` on G. Then G admits a P_3-partition.*

The above lemmas allow us to conclude the correctness of Algorithm 1.

Theorem 2. P_3-Partition *on interval graphs is solvable in $O(n \log n + m)$ time.*

3 Grid Graphs

In this section, we show that P_3-Partition is NP-hard even on grid graphs with maximum degree three, thus strengthening a result of Małafiejski and Żyliński [15] and Monnot and Toulouse [16], who showed that P_3-Partition is NP-complete on planar bipartite graphs of maximum degree three.

A *grid graph* is a graph with a vertex set $V \subseteq \mathbb{N} \times \mathbb{N}$ and edge set $\{\{u, v\} \mid u = (i, j) \in V, v = (k, \ell) \in V, |i - k| + |j - \ell| = 1\}$. That is, its vertices can be given integer coordinates such that every pair of vertices is joined by an edge if and only if their coordinates differ by 1 in exactly one dimension.

To show NP-hardness of P_3-Partition on grid graphs, we exploit the above mentioned result of Małafiejski and Żyliński [15] and Monnot and Toulouse [16] and find a suitable embedding of planar graphs into grid graphs while maintaining the property of a graph having a P_3-partition. This allows us to prove the following.

Theorem 3. P_3-Partition *is NP-hard on grid graphs of maximum degree three.*

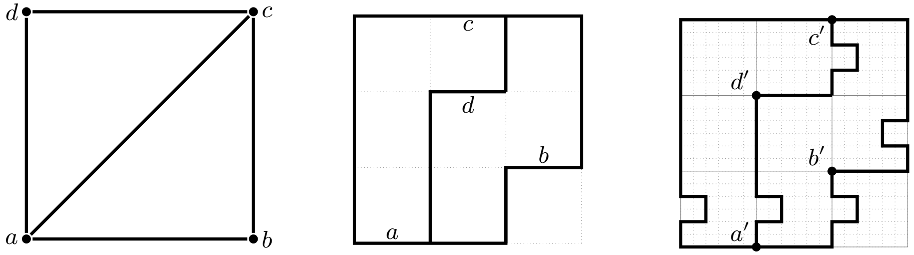
The following observation helps us embed planar graphs into grid graphs, as it allows us to replace edges by paths on $3i$ new vertices for any $i \in \mathbb{N}$.

Observation 1. Let G be a graph, $e = \{v, w\}$ be an edge of G, and G' be the graph obtained by removing the edge e from G and by connecting v and w using a path on three new vertices. Then, G has a P_3-partition if and only if G' has.

We can now prove Theorem 3 by showing that G has a P_3-partition if and only G' has, where G' is the graph obtained from a planar graph G of maximum degree three using the following construction.

Construction 1. Let G be a planar graph of maximum degree three. Using a polynomial-time algorithm of Rosenstiehl and Tarjan [18] we obtain a crossing-free *rectilinear embedding* of G into the plane such that:

1. Each vertex is represented by a horizontal line.
2. Each edge is represented by a vertical line.
3. All lines end at integer coordinates with integers in $O(n)$.
4. If two vertices are joined by an edge, then the vertical line representing this edge ends on the horizontal lines representing the vertices.



(a) A planar graph G

(b) A rectilinear embedding of G

(c) The grid graph G' obtained from G

Fig. 3. Various embeddings of a planar graph. In the rectilinear embedding in Figure 3b, horizontal lines represent vertices of G, while vertical lines represent its edges. In Figure 3c, every intersection of a line with a grid point is a vertex, but only the vertices corresponding to vertices in Figure 3a are shown.

Figure 3b illustrates such an embedding. Without loss of generality, every end point of a line lies on another line. Now, in polynomial time, we obtain a grid graph G' from the rectilinear embedding, as follows:

1. We multiply all coordinates by six (see Figure 3c).
2. Every point in the grid touched by a horizontal line that represents a vertex v of G becomes a vertex in G'. The horizontal path resulting from this horizontal line we denote by $P(v)$.
3. For each vertical line, all its grid points become vertices in G', except for one point that we bypass by adding a bend of five vertices to the vertical line (see Figure 3c).
4. With each vertex v in G, we associate the vertex v' of G' that lies on $P(v)$ and has degree three. There is at most one such vertex. If no such vertex exists, then we arbitrarily associate with v one of the end points of $P(v)$.

4 Bipartite Permutation Graphs

In this section, we show that Star Partition can be solved in $O(n^2)$ time on bipartite permutation graphs. The class of bipartite permutation graphs can be characterized using *strong orderings* of the vertices of a bipartite graph:

Definition 1 (Spinrad et al. [19]). *A* strong ordering $\prec$ *of the vertices of a bipartite graph* $G = (U, W, E)$ *is the union of a total order* $\prec_U$ *of* U *and a total order* $\prec_W$ *of* W*, such that for all* $\{u, w\}$*,* $\{u', w'\}$ *in* E*, where* $u, u' \in U$ *and* $w, w' \in W$*,* $u \prec u'$ *and* $w' \prec w$ *implies that* $\{u, w'\}$ *and* $\{u', w\}$ *are in* E*.*

A graph is a bipartite permutation graph if and only if it is bipartite and there is a strong ordering of its vertices, which can be computed in linear time [19].

Our key to obtain star partitions on bipartite permutation graphs is a structural result that only a certain "normal form" of star partitions has to be searched for. This paves the way to developing a dynamic programming solution exploiting these normal forms. We sketch these structural properties of an s-star partition of bipartite permutation graphs in the following.

Definition 2. *Let* (G, s) *be a* Star Partition *instance, where* $G = (U, W, E)$ *is a bipartite permutation graph,* $\prec$ *is a strong ordering of the vertices, and* $\preccurlyeq$ *is the reflexive closure of* $\prec$*. Assume that* G *admits an* s*-star partition* $\mathcal{P}$*.*

Let $X \in \mathcal{P}$ *form a star. By* $\mathrm{lm}(X)$ *(resp.* $\mathrm{rm}(X)$*), we denote the leftmost (that is, the smallest), resp. the rightmost (that is, the largest) leaf of* X *with respect to* $\prec$*. The* scope *of star* X *is the set* $\mathrm{scope}(X) := \{v \mid x_l \preccurlyeq v \preccurlyeq x_r\}$ *containing all vertices from* $x_l = \mathrm{lm}(X)$ *to* $x_r = \mathrm{rm}(X)$*. The* width *of star* X *is the cardinality of its scope, that is,* $\mathrm{width}(X) := |\,\mathrm{scope}(X)| = r - l + 1$*. The* width *of* $\mathcal{P}$*,* $\mathrm{width}(\mathcal{P})$*, is the sum of* $\mathrm{width}(X)$ *over all* $X \in \mathcal{P}$*.*

Let $e = \{u, w\}$ *and* $e' = \{u', w'\}$ *be two edges. We say that* e *and* e' cross *each other if either (*$u \prec u'$ *and* $w' \prec w$*) or (*$u' \prec u$ *and* $w \prec w'$*). The* edge-crossing number *of two stars* $X, Y \in \mathcal{P}$ *is the number of pairs of crossing edges* e, e' *where* e *is an edge of* X *and* e' *is an edge of* Y*. The edge-crossing number* #edge-crossings($\mathcal{P}$) *of* $\mathcal{P}$ *is the sum of the edge-crossing numbers over all pairs of stars* $X \neq Y \in \mathcal{P}$*.*

We identify the possible configurations of two stars, depending on the relative positions of their leaves and centers, see Figure 4. Among those, the following two configurations are favorable: Given $X, Y \in \mathcal{P}$*, we say that* X *and* Y *are*

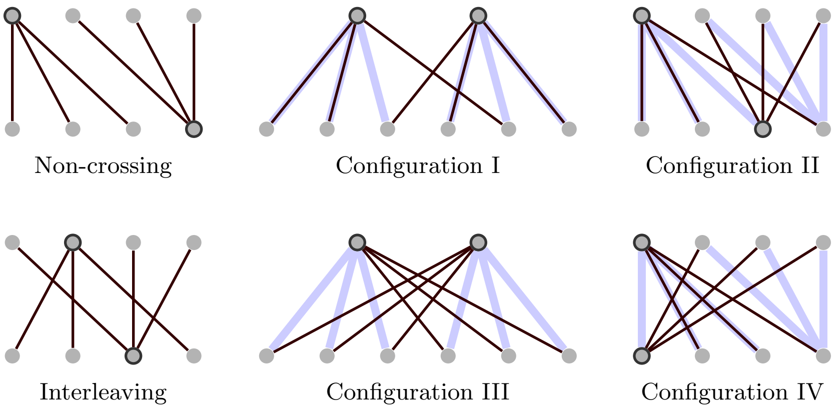
- non-crossing *if their edge-crossing number is zero;*
- interleaving *if* $\mathrm{center}(X) \in \mathrm{scope}(Y)$ *and* $\mathrm{center}(Y) \in \mathrm{scope}(X)$*;*

We say that $\mathcal{P}$ *is* good *if any two stars* $X \neq Y \in \mathcal{P}$ *are either non-crossing or interleaving. We define the* score *of* $\mathcal{P}$ *as the tuple* (width($\mathcal{P}$), #edge-crossings($\mathcal{P}$))*. We use the lexicographical order to compare scores.*

This definition allows us to show a normal form of star partitions in bipartite permutation graphs.

Lemma 3. *Any* s*-star partition of a bipartite permutation graph* G *with minimum score is a good* s*-star partition.*

Corollary 1. *Let* $\mathcal{P}$ *be an* s*-star partition of a bipartite permutation graph* G *with minimum score. Then, for every star* $X \in \mathcal{P}$*, there is at most one* $Y \in \mathcal{P}$ *such that* X *and* Y *are interleaving, and for all* $Z \in \mathcal{P} \setminus \{X, Y\}$*,* X *and* Z *are non-crossing.*



Non-crossing

Configuration I

Configuration II

Interleaving

Configuration III

Configuration IV

Fig. 4. Possible interactions between two stars of a partition. Centers are marked with circled nodes. The four possible configurations of star centers and scopes that are neither non-crossing nor interleaving are labeled I to IV. By Lemma 3, any partition containing one of the configurations I to IV can be edited to reduce the score (see the thick light-color edges).

We now informally describe a dynamic programming algorithm for deciding whether there is a *good* s-star partition. It builds up a solution following the strong ordering of the graph from left to right. A partial solution can be extended in three ways only: either (i) a star is added with the center in U, or (ii) a star is added with the center in W, or (iii) two interleaving stars are added. The algorithm can thus compute, for any given number of centers in U and in W, whether it is possible to partition the leftmost vertices of U and W in one of the three ways (i)–(iii). This algorithm leads to the following result.

Theorem 4. Star Partition *can be solved in $O(n^2)$ time on bipartite permutation graphs.*

5 Further Results

This section briefly summarizes our hardness and tractability results for cographs, split graphs, and chordal graphs.

Cographs. A cograph is a graph that does not contain a P_4 (path on four vertices) as an induced subgraph. Cographs allow for a so-called *cotree* to be computed in linear time [7]. Using a dynamic programming approach on the cotree representation of the cograph, we can solve Star Partition in polynomial time.

Theorem 5. Star Partition *can be solved in $O(kn^2)$ time on cographs.*

Split Graphs. A *split graph* is a graph whose vertices can be partitioned into a clique and an independent set. Remarkably, split graphs are the only graph class where we could show that P_3-PARTITION is solvable in polynomial time, but STAR PARTITION for $k \geq 3$ is NP-hard.

More precisely, we solve P_3-PARTITION on split graphs by reducing it to finding a restricted form of *factor* in an auxiliary graph; herein, a *factor* of a graph G is a spanning subgraph of G (that is, a subgraph containing all vertices). This graph factor problem then can be solved in polynomial time [8].

In contrast, we can show that STAR PARTITION is NP-hard for each $s \geq 3$ by a reduction from EXACT COVER BY s-SETS.

Theorem 6. STAR PARTITION *on split graphs is solvable in $O(m^{2.5})$ time for $s = 2$, but is NP-hard for each $s \geq 3$.*

Chordal graphs. A graph is *chordal* if every induced subgraph containing a cycle of length at least four also contains a *triangle*, that is, a cycle of length three. We show that P_3-PARTITION restricted to chordal graphs is NP-hard by reduction from 3-DIMENSIONAL MATCHING.

Theorem 7. P_3-PARTITION *restricted to chordal graphs is NP-hard.*

For the reduction, we use the construction that Dyer and Frieze [11] provided to show that P_3-PARTITION is NP-complete and observe that we can triangulate the resulting graph while maintaining the correctness of the reduction.

6 Conclusion

We close with three open questions for future research. What is the complexity of STAR PARTITION for $s \geq 2$ on permutation graphs? What is the complexity of STAR PARTITION for $s \geq 3$ on interval graphs? Are there other important graph classes (not necessarily perfect ones) where STAR PARTITION is polynomial-time solvable?

Acknowledgments. We are indebted to three anonymous *ICALP* reviewers whose constructive feedback helped to significantly improve our presentation.

References

[1] Asdre, K., Nikolopoulos, S.D.: NP-completeness results for some problems on subclasses of bipartite and chordal graphs. Theor. Comput. Sci. 381(1-3), 248–259 (2007)

[2] Berman, F., Johnson, D., Leighton, T., Shor, P.W., Snyder, L.: Generalized planar matching. J. Algorithms 11(2), 153–184 (1990)

[3] van Bevern, R., Bredereck, R., Chen, J., Froese, V., Niedermeier, R., Woeginger, G.J.: Star partitions of perfect graphs, TU Berlin (2014a) (manuscript) arXiv:1402.2589 [cs.DM]

[4] van Bevern, R., Bredereck, R., Chen, J., Froese, V., Niedermeier, R., Woeginger, G.J.: Network-based dissolution, TU Berlin (2014b) (manuscript) arXiv:1402.2664 [cs.DM]

[5] Brandstädt, A., Le, V.B., Spinrad, J.P.: Graph Classes: a Survey. In: SIAM Monographs on Discrete Mathematics and Applications, vol. 3. SIAM (1999)

[6] Chalopin, J., Paulusma, D.: Packing bipartite graphs with covers of complete bipartite graphs. Discrete Appl. Math. 168, 40–50 (2014)

[7] Corneil, D., Perl, Y., Stewart, L.: A linear recognition algorithm for cographs. SIAM J. Comput. 14(4), 926–934 (1985)

[8] Cornuéjols, G.: General factors of graphs. J. Combin. Theory Ser. B 45(2), 185–198 (1988)

[9] Dahlhaus, E., Karpinski, M.: Matching and multidimensional matching in chordal and strongly chordal graphs. Discrete Appl. Math. 84(1-3), 79–91 (1998)

[10] De Bontridder, K.M.J., Halldórsson, B.V., Halldórsson, M.M., Hurkens, C.A.J., Lenstra, J.K., Ravi, R., Stougie, L.: Approximation algorithms for the test cover problem. Math. Program. 98(1-3), 477–491 (2003)

[11] Dyer, M.E., Frieze, A.M.: On the complexity of partitioning graphs into connected subgraphs. Discrete Appl. Math. 10(2), 139–153 (1985)

[12] Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs. Annals of Discrete Mathematics. Elsevier, Amsterdam (2004)

[13] Kirkpatrick, D.G., Hell, P.: On the complexity of general graph factor problems. SIAM J. Comput. 12(3), 601–608 (1983)

[14] Kosowski, A., Małafiejski, M., Żyliński, P.: Parallel processing subsystems with redundancy in a distributed environment. In: Wyrzykowski, R., Dongarra, J., Meyer, N., Waśniewski, J. (eds.) PPAM 2005. LNCS, vol. 3911, pp. 1002–1009. Springer, Heidelberg (2006)

[15] Małafiejski, M., Żyliński, P.: Weakly cooperative guards in grids. In: Gervasi, O., Gavrilova, M.L., Kumar, V., Laganá, A., Lee, H.P., Mun, Y., Taniar, D., Tan, C.J.K. (eds.) ICCSA 2005. LNCS, vol. 3480, pp. 647–656. Springer, Heidelberg (2005)

[16] Monnot, J., Toulouse, S.: The path partition problem and related problems in bipartite graphs. Oper. Res. Lett. 35(5), 677–684 (2007)

[17] van Rooij, J.M.M., van Kooten Niekerk, M.E., Bodlaender, H.L.: Partition into triangles on bounded degree graphs. Theory Comput. Syst. 52(4), 687–718 (2013)

[18] Rosenstiehl, P., Tarjan, R.E.: Rectilinear planar layouts and bipolar orientations of planar graphs. Discrete Comput. Geom. 1(1), 343–353 (1986)

[19] Spinrad, J., Brandstädt, A., Stewart, L.: Bipartite permutation graphs. Discrete Appl. Math. 18(3), 279–292 (1987)

[20] Steiner, G.: On the k-path partition problem in cographs. Congressus Numerantium 147, 89–96 (2000)

[21] Steiner, G.: On the k-path partition of graphs. Theor. Comput. Sci. 290(3), 2147–2155 (2003)

[22] Takamizawa, K., Nishizeki, T., Saito, N.: Linear-time computability of combinatorial problems on series-parallel graphs. J. ACM 29(3), 623–641 (1982)

[23] Yan, J.-H., Chen, J.-J., Chang, G.J.: Quasi-threshold graphs. Discrete Appl. Math. 69(3), 247–255 (1996)

[24] Yan, J.-H., Chang, G.J., Hedetniemi, S.M., Hedetniemi, S.T.: k-path partitions in trees. Discrete Appl. Math. 78(1-3), 227–233 (1997)

[25] Yuster, R.: Combinatorial and computational aspects of graph packing and graph decomposition. Computer Science Review 1(1), 12–26 (2007)

Coordination Mechanisms for Selfish Routing over Time on a Tree

Sayan Bhattacharya[1], Janardhan Kulkarni[2,*], and Vahab Mirrokni[3]

[1] Max-Planck Institute for Informatics, Saarbrucken, Germany
[2] Duke University, Durham, USA
[3] Google Inc., New York, USA

Abstract. While selfish routing has been studied extensively, the problem of designing better *coordination mechanisms* for routing over time in general graphs has remained an open problem. In this paper, we focus on tree networks (single source multiple destinations) with the goal of minimizing (weighted) average *sojourn time* of jobs, and provide the first coordination mechanisms with provable price of anarchy for this problem. Interestingly, we achieve our price of anarchy results using simple and strongly local policies such as Shortest Job First and the Smith's Rule (also called HDF). In particular, for the case of unweighted jobs, we design a coordination mechanism with polylogarithmic price of anarchy. For weighted jobs, on the other hand, we show that price of anarchy is a function of the depth of the tree and accompany this result by a lower bound for the price of anarchy for the Smith Rule policy and other common strongly local scheduling policies.

Our price of anarchy results also imply improved approximation algorithms for the underlying optimization problem of routing over a tree. This problem is well motivated from applications of routing in supercomputers and data center networks where average sojourn time is an important metric.

1 Introduction

As a central topic in algorithmic game theory, selfish routing problems have been studied extensively in the context of *congestion games* [20,26,27]. Being a representative class of potential games, network congestion games have served as a foundation for proving price of anarchy results. However they lack an important aspect of real network routing which is the fact that *routing happens over time*, and any realistic model should take this into account. To address this issue, several new models have been proposed to capture the nature of realistic routing over time [10,11,21,19,24,13,12,4]. Amongst these models, the concept of *coordination mechanisms*, first introduced in an influential paper by Christodoulou, Koutsoupias, and Nanavati [10], have been proposed to capture the queueing nature of routing. Coordination mechanisms model the decentralized nature of routing decisions made by machines and the selfish behavior of jobs: they do so by seeking local policies that achieve a good price of anarchy in the resulting equilibria in a corresponding game. While these subjects have attracted a great amount

[*] Supported by NSF Awards CCF-1008065 and IIS-0964560.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 186–197, 2014.

of research, the problem of designing better coordination mechanisms for routing over time has remained a wide open problem, and results have been developed only for very special classes of networks such as parallel edges (corresponding to a multi-processor scheduling problem [10,17,12,4]). In this paper, we focus on tree networks, and provide the first coordination mechanisms with provable price of anarchy.

The significance of our results are two-fold: other than providing approximately optimal coordination mechanisms, our price of anarchy results also imply improved approximation algorithms for the underlying optimization problem of routing over a tree. This problem is well motivated from applications of routing in supercomputers and data centers where average sojourn time is an important QoS metric. Before elaborating the model, we describe the significance of the optimization problem over trees in various applications.

Applications of Optimizing Routing over a Tree: In 1985, Leiserson discovered that a variant of tree topology called *fat tree network* is a *universal network* for efficient network connections [23]. In a typical fat tree topology, each tree edge has a capacity (or bandwidth or processing power) and the edges closer to the root node have higher capacities than those to the leaves. The fat tree topology quickly became the de facto standard for connecting processors within supercomputers. Recently, the efficiency of tree topologies is being rediscovered in the context of data center architectures as data center sizes grow exponentially with the explosion of the cloud-based services. A typical server farm in a data center consists of a set of web severs interconnected by a fat tree topology. See, for example, an important paper on this topic by Al-Fares, Loukissas and Vahdat [1]. Despite such important applications of optimal routing tasks over tree topologies, very little is known from a theoretical standpoint about scheduling jobs over a tree network to minimize their *average sojourn time* which is a fundamental quality of service metric for evaluating the performance of such a system. The sojourn time of a job is defined as the total time the job spends in the network. On the other hand, minimizing the *makespan* of a schedule, defined as the maximum sojourn time over all the jobs, has a vast literature in these settings [14,25,24], including the celebrated result of Leighton, Maggs and Rao [22].

1.1 The Model

We are given a tree $T = (V, E)$ rooted at node $r \in V$. Each edge $e \in E$ in the tree is a machine[1] with *speed* $s(e)$. For the rest of the paper, we use the terms "edge" and "machine" interchangeably. There is a set of jobs J with unique identifiers, which will be used by our policies for breaking ties consistently. Each job $j \in J$ has *weight* w_j and *length* p_j, and its *processing time* on edge e is $p_{je} = p_j/s(e)$. Each edge can process at most one unit of the jobs during a unit time-step, and a job j requires p_{je} time-units of processing on an edge e. At time $t = 0$, all the jobs are located at the root.

A job j can be *served* by only the nodes in the subset $\mathcal{L}_j \subseteq V$. This models, for example, the fact that some servers in a data center may not have the necessary content to satisfy a request. The job j starts at the root and wants to exit the tree through any

[1] The reason that we use the term "machine" to refer to the "edges" is to cope with the scheduling literature. One should think about these machines as a communication link.

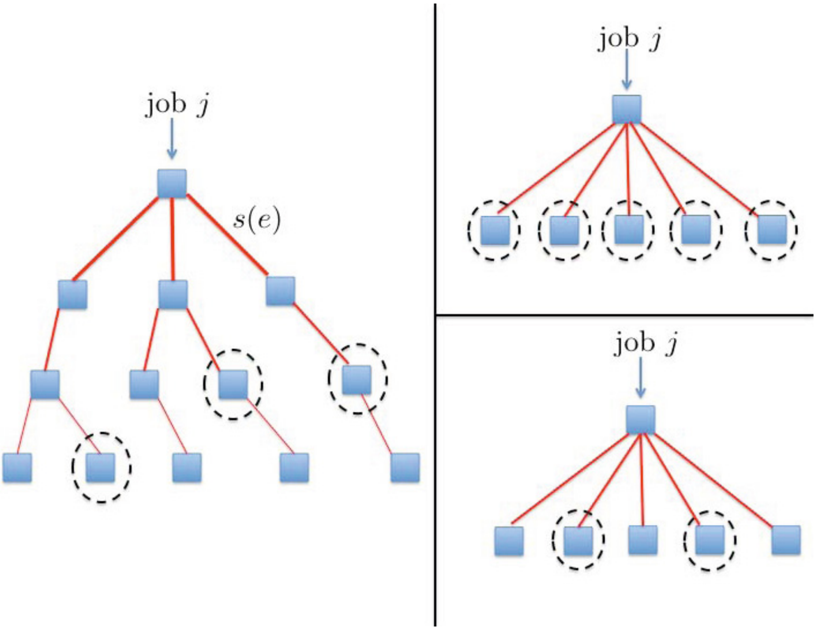


Fig. 1. In our model, each edge e has a speed $s(e)$. Each job j starts at the root and needs to travel to one of the nodes in $\mathcal{L}_j$, shown in dashed circles. To the right, we show how our model generalizes the related machines (top right) and the restricted assignment setting (bottom right).

one of the nodes in $\mathcal{L}_j$, which are called the *destination-nodes* of j. Accordingly, the job j selects a path $i = (e_1, \dots, e_l)$ that begins at the root of the tree and terminates at some node in $\mathcal{L}_j$. Here, e_1 is the first edge on the path (adjacent to the root), and e_l is the last edge. The job can start getting processed on an edge e_k, $k \in \{2, \dots, l\}$, only after it is processed completely by the preceding edge e_{k-1}. The job exits the tree when it gets completely processed on the last edge e_l, and the time at which this event takes place is called the *sojourn time* of the job. The weighted sojourn time of j is equal to its weight w_j times its sojourn time. Note that since all the jobs are at the root node at time $t = 0$, the average sojourn time is equivalent to the average *flow-time* in our context. A reader familiar with the scheduling literature can see that our model is a generalization of the *related machine* and the *restricted assignment* settings (see Figure 1).

The *underlying optimization problem* asks us to route every job j through a root-to-destination path terminating in $\mathcal{L}_j$, and provide a scheduling policy on each edge so as to minimize the sum of their weighted sojourn times. We allow preemption of jobs on an edge. The jobs, however, are selfish agents who cannot be forced to obey the dictate of a centralized authority. Thus, in our model, the machines declare their scheduling policies in advance, and this induces a simultaneous-move game between the jobs. We require that the scheduling policies be *strongly local*, in the sense that the scheduling decision by an edge at any time-instant depends only on the current set of jobs waiting to be processed on that edge, and is independent of the global state of the system.

In this game, the strategy of each job j consists of selecting a path from the root to any one of the destination-nodes in $\mathcal{L}_j$. The vector $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{|J|})$ denotes an outcome (strategy-profile) of the game, where $\boldsymbol{\theta}_j$ is the path selected by the job j. The symbol $(i, \boldsymbol{\theta}_{-j})$ denotes an outcome where the job j selects the path i, and every job $j' \neq j$ selects the path $\boldsymbol{\theta}_{j'}$. The symbol $\text{Cost}_j(\boldsymbol{\theta})$ denotes the cost incurred by the job j under the outcome $\boldsymbol{\theta}$, which is equal to its weighted sojourn time. An outcome $\boldsymbol{\theta}$ is in a (pure) Nash equilibrium iff no job can reduce its cost by unilaterally deviating to another path, i.e., iff $\text{Cost}_j(\boldsymbol{\theta}) \leq \text{Cost}_j(i, \boldsymbol{\theta}_{-j})$ for all jobs j and root-to-destination paths i terminating at a node in $\mathcal{L}_j$.

The *price of anarchy* (PoA) of the game is the worst (maximum) possible ratio between the total cost of the agents in a Nash equilibrium and the optimal objective of the underlying optimization problem. Intuitively, it is a measure of the degradation in the overall system-performance due to the strategic interactions between the jobs.

We want to solve the following problem: Find a set of scheduling policies for the machines so as to minimize the PoA of the resulting game.

1.2 Our Contributions and Techniques

We analyze the PoA of the game induced among the jobs when the machines follow a natural and easy to implement scheduling policy known as Smith's Rule [28] or "Highest Density First" (HDF). Under this policy, at every time-step a machine works on a job j that is available for processing and has maximum "density" w_j/p_j. When all jobs have the same weight, this reduces to the "Shortest Job First" (SJF) scheduling policy. If multiple jobs happen to have the same density, then *all the machines use the same tie-breaking rule* to decide which one of these jobs to consider first. Hence, without loss generality, we will assume throughout the rest of the paper that no two jobs have exactly the same density.

Unweighted Jobs. When the jobs are unweighted and the machines follow SJF policy, we prove that the PoA of the induced game is $O(\log^2(s_{\max}/s_{\min}))$ (Theorem 3). Here, $s_{\max}$ (resp. $s_{\min}$) denotes the speed of the fastest (resp. slowest) machine. Note that this implies constant PoA when all the machines are identical; even in the general case, the PoA is independent of the number of jobs or nodes in the network.

Weighted Jobs. When different jobs have different weights and all the edges follow the Smith Rule (HDF) policy, we prove that the PoA is $O(d^2)$, where d is the depth of the tree (Theorem 5). We complement this result by showing that the PoA must depend polynomially on d (Theorems 4).

Approximation Algorithms. It is known that Smith's Rule (HDF) induces a game with a pure NE [15] which can be computed in polynomial time by greedily adding jobs in the decreasing order of their densities. This implies that our PoA results also lead to approximation algorithms for the underlying optimization problems (both for the weighted and unweighted jobs), the approximation ratio being the same as the PoA of the game.

Our Technique: Both of the PoA upper-bounds in this paper are obtained using the following simple technique: First, we find an LP relaxation for the underlying optimization problem and write down its dual. Next, we consider any arbitrary Nash equilibrium outcome $\boldsymbol{\theta}$, and based on this outcome assign values to the dual variables in such a way that satisfies all the dual constraints, thereby getting a feasible solution to the dual LP. Finally, we show that the objective of this feasible dual solution is at least $1/\alpha$ times the total cost incurred by the jobs under $\boldsymbol{\theta}$. Weak duality implies that the PoA of the game is at most α. Our overall approach is inspired by papers [2,6]. This technique is very powerful and can potentially be applied to bound the PoA in several other settings. Bilò et al. [7] give another application of this technique to analyze PoA.

Apart from the overall idea of using the dual fitting technique to analyze the PoA of the game, writing a linear programming relaxation with small integrality gap turns be a significant challenge for our problem. A direct extension of the time-indexed LP [2] has a huge integrality gap in our setting. We circumvent this difficulty for the case of unweighted jobs by first finding a set of *critical edges* which play a crucial role in how the jobs delay each other. Then, we write a time-indexed LP relaxation taking into account only these edges which brings the integrality gap down. See Section 2 for details.

Related Work. Following the landmark paper of Christodoulou et al. [10] who initiated the study of coordination mechanisms, several papers have been written on the topic for various problems. However, most of these results are for machine scheduling problems, either proving PoA results on the makespan objective function [17,8,4] or recently for the weighted completion times [12,6]. In the context of selfish routing, the multi-machine scheduling problem corresponds to a network of parallel edges and related machine scheduling is a special case of our model where the tree is a star, i.e, a tree of depth one (Figure 1). The only two results that go beyond the scheduling problem are by Hoefer et al [15] and by Christodoulou et al. [11]. The first paper only studies existence and computation of equilibria for various coordination mechanisms and leaves the PoA question as an open problem. The second paper discusses a quite different setting with non-atomic players.

Our work is also related to the literature on the PoA of selfish routing [20,27], and more specifically unsplittable selfish routing [26]. Here we extend the selfish routing model by incorporating a temporal component into the problem formulation. Other attempts to address this issue include [21,19,13,24], but none of these results discuss coordination mechanisms using strongly local (decentralized) policies.

We have recently learned through personal communication that scheduling over tree and line networks are also being considered in the online (and resource augmentation) setting in a recent (ongoing) work by Im et al. [16] and Antoniadis et al. [3]. Finally, our work is related to approximation algorithms of classical optimization problems for minimizing the weighted sum of completion times and flow times [18,9,2,5], none of which present an approximation algorithm for the problem minimizing average completion time in routing over a tree.

2 $O(\log^2(s_{\max}/s_{\min}))$ PoA for Unweighted Jobs

In this section, we assume that all the jobs have unit weights, and bound the PoA of the game where every edge follows the Shortest Job First (SJF) scheduling policy (Theorem 3). We start with a high-level overview of our approach. We say that a job j is *delayed* by another job j' on an edge e at time t iff two conditions are satisfied at the same time: (a) the job j is available for processing on edge e, and (b) instead of j, the edge is processing the job j'. A machine following the SJF policy never sits idle when one or more jobs are waiting in its queue. This ensures that the sojourn time of a job j is exactly equal to the total amount of processing done on j by all the edges, plus the total delay it encounters due to all other jobs. The former quantity is given by $\sum_{e \in i} p_{je}$, where i is the path selected by the job. It is the latter quantity for which it is difficult to get a closed-form expression. We overcome this difficulty by showing that there is a small subset of *critical edges* on any path (Definition 1), and that one job can delay another only on one of these edges. This line of reasoning culminates in a key structural result (Theorem 2) that gives an upper bound on the maximum delay a job j can experience due to any single job $j' \neq j$.

As alluded in Section 1.2, we now confront the task of designing a suitable LP relaxation for the underlying optimization problem. A straightforward extension of the time-indexed LP considered in [2] to our setting leads to a large integrality gap. On the positive side, the time-indexed LP has one nice property: Its dual has variables that can be interpreted as decomposing the total delay incurred by a job across the edges on its path. The duals of other "natural" LP relaxations for our problem do not seem to be amenable to such a nice interpretation. Accordingly, we modify the time-indexed LP relaxation by only taking into account the critical edges of the tree. As the number of critical edges is small, this brings down the integrality gap of the LP. We then manage to fit its dual using Theorem 2.

To proceed with the technical details, let $s_{\max}$ (resp. $s_{\min}$) denote the speed of the fastest (resp. slowest) of a machine, and let $K = \lfloor \log(s_{\max}/s_{\min}) \rfloor$. For ease of exposition, we assume that the speeds of the machines are discretized in powers of two. By standard *time stretching* arguments, it is easy to show that this assumption can lead to at most a factor two loss in the PoA of our coordination mechanism.

Assumption 1. *For all $e \in E$, we have $s(e) = 2^k \cdot s_{\min}$ for some $k \in \{1, 2, \ldots, K\}$.*

Definition 1. *Let $e_{i,k}$ denote the edge of minimum depth on path i that has speed $2^k s_{\min}$. We refer to such an edge as a* critical *edge on that path. We define $E_i = \cup_k \{e_{ik}\}$ to be the set of all critical edges on path i.*

Below, we describe our LP relaxation for the underlying optimization problem. For rest of the paper, we overload the notation $i \in \mathcal{L}_j$ to denote a path i that starts at the root and terminates at a node in $\mathcal{L}_j$.

$$\text{Minimize} \qquad \sum_j \sum_{i \in \mathcal{L}_j} \sum_{e \in E_i} \sum_t x_{ijet} \cdot \left(\frac{t}{p_{je}} \right) + \sum_j \sum_{i \in \mathcal{L}_j} \sum_e \sum_t x_{ijet} \qquad (1)$$

$$\sum_{i \in \mathcal{L}_j} x_{ij} \geq 1 \qquad \forall \text{ jobs } j \tag{2}$$

$$\sum_t \frac{x_{ijet}}{p_{je}} \geq x_{ij} \qquad \forall \text{ jobs } j, \text{ paths } i \in \mathcal{L}_j, \text{ edges } e \in i \tag{3}$$

$$\sum_j \sum_{i \in \mathcal{L}_j : e \in i} x_{ijet} \leq 1 \qquad \forall \text{ edges } e, \text{ times } t \tag{4}$$

$$x_{ijet}, x_{ij} \geq 0 \qquad \forall j, i \in \mathcal{L}_j, e \in i, t \tag{5}$$

In an integral feasible solution of the above linear program, the variable $x_{ij} \in \{0, 1\}$ indicates if the job j takes the path $i \in \mathcal{L}_j$. The variable $x_{ijet} \in \{0, 1\}$ indicates if the job j takes the path $i \in \mathcal{L}_j$ and is being processed on the edge $e \in i$ at time-step t.

Constraint 2 states that every job has to take some path. Constraint 3 states that if a job j takes a path $i \in \mathcal{L}_j$, then it has to get completely processed on all the edges on this path. Finally, constraint 4 states that every edge can process at most one unit of the jobs during one time-step.

The second summation in the LP objective gives the total amount of processing done on all the jobs, which clearly is a lower bound on the sum of their sojourn-times. Now, fix any job j which takes a path $i \in \mathcal{L}_j$, and consider an edge $e \in i$ on this path. The term $\sum_t x_{ijet} \cdot (t/p_{je})$ is known as the *fractional completion time* [2] of the job j on the edge e. This is at most the time at which the edge e finishes processing the job, which, in turn, is at most the sojourn-time of j. We sum this quantity over all the critical edges E_i on path i, and the number of such critical edges is $O(K)$. Thus, the summation $\sum_{i \in \mathcal{L}_j} \sum_{e \in E_i} \sum_t (t/p_{je}) \cdot x_{ijet}$ is $O(K)$ times the sojourn-time of j. Summing over all the jobs, we see that the overall LP objective is $O(K)$ times the objective of the underlying optimization problem.

We now get a new LP by replacing the 1 in the right hand side of constraint 4 with $1/(4K)$. This imposes the condition that a machine can process at most $1/(4K)$ units of the jobs during one time-step, and, by standard scaling arguments, it is easy to show that this increases the LP objective by a factor of $4K$. The dual of this new LP is given by LP (6). By weak duality, we get Lemma 1.

$$\text{Max} \qquad \sum_j y_j - \frac{1}{4K} \sum_{e,t} z_{et} \tag{6}$$

$$y_j \leq \sum_{e \in i} u_{ije} \qquad \forall \text{ jobs } j, \text{ paths } i \in \mathcal{L}_j \tag{7}$$

$$\frac{u_{ije}}{p_{je}} - z_{et} \leq 1 \qquad \forall \text{ jobs } j, i \in \mathcal{L}_j, e \in i \setminus E_i, \text{ times } t \tag{8}$$

$$\frac{u_{ije}}{p_{je}} - z_{et} \leq \frac{t}{p_{je}} + 1 \qquad \forall \text{ jobs } j, i \in \mathcal{L}_j, e \in E_i, \text{ times } t \tag{9}$$

$$y_j, u_{ije}, z_{et} \geq 0 \qquad \forall j, t, i \in \mathcal{L}_j, e \in i \tag{10}$$

Lemma 1. *The objective of any feasible solution to LP (6) is $O(K^2)$ times the optimal objective of the underlying optimization problem, where $K = \lfloor \log(s_{\max}/s_{\min}) \rfloor$.*

For the rest of this section, we will assume that every edge follows SJF scheduling policy, and we will analyze the PoA of the resulting game. The theorem below bounds the maximum delay a job can encounter due to any other job.

Theorem 2. *Suppose that every machine runs SJF scheduling policy. Consider any two jobs j^*, j_1^*, and fix any outcome $\boldsymbol{\theta}$ (not necessarily a Nash Equilibrium) of the induced game. Let $e \in \boldsymbol{\theta}_{j^*} \cap \boldsymbol{\theta}_{j_1^*}$ be an edge with slowest speed that is common to the paths taken by the jobs. Then the total delay of the job j^* due to the job j_1^* is at most $2p_{j^*}/s(e)$.*

Proof (Sketch). Consider the path $\boldsymbol{\theta}_{j^*} = (e_1, \ldots, e_l)$, where e_1 (resp. e_l) is the edge adjacent to (resp. farthest away from) the root. We decompose this path in w segments, for some natural number w, in the following manner. Consider a function $f : [1, w+1] \rightarrow [1, l+1]$ such that $1 = f(1) < f(2) < \cdots < f(w+1) = l+1$. Segment $k \in [1, w]$ corresponds to the sequence of edges $e_{f(k)}, e_{f(k)+1}, \ldots, e_{f(k+1)-1}$. The decomposition satisfies two properties.

- The speeds of the first edges of these segments form a *strictly* decreasing sequence. Hence, Assumption 1 implies that $s(e_{f(1)}) > s(e_{f(2)})/2 > \cdots > s(e_{f(w)})/2^{w-1}$.
- Within each segment, the speed of the first edge is at most the speed of any other edge. Thus, $s(e_{f(k)}) \leq s(e)$ for all $k \in [1, w]$ and $e \in \{e_{f(k)}, \ldots, e_{f(k+1)-1}\}$.

Note that the first edge $e_{f(k)}$ of every segment $k \in [1, w]$ is a critical edge on the path $\boldsymbol{\theta}_{j^*}$. We show that the job j^* can get delayed by other jobs *only* on this set of edges $\{e_{f(k)} : k \in [1, w]\}$. Now, under SJF policy, the job j^* can get delayed by another job j_1^* only if $p_{j_1^*} \leq p_{j^*}$, and, in this case, the delay experienced by j^* due to j_1^* on any edge $e_{f(k)}$ is at most $p_{j_1^*}/s(e_{f(k)}) \leq p_{j^*}/s(e_{f(k)})$. Thus, the total delay incurred by j^* due to j_1^* is upper bounded by the sum $\sum_{k=k^*}^{1} p_{j^*}/s(e_{f(k)})$, where k^* is the segment with the largest index whose first edge also belongs to $\boldsymbol{\theta}_{j_1^*}$. This sum is part of a geometric series with common ratio $1/2$, and is at most $2p_{j^*}/s(e_{e_{f(k^*)}})$. The theorem follows from the fact that the edge $e_{f(k^*)}$ has the slowest speed among all the edges that are common to the paths $\boldsymbol{\theta}_{j^*}$ and $\boldsymbol{\theta}_{j_1^*}$.

In Figure 2, we give an algorithm that takes any Nash equilibrium of the game (under SJF policy), and, depending on this input, sets the variables in LP (6).

Lemma 2. *If the dual variables are assigned values as in Figure 2, then for all jobs j and root-to-leaf paths $i \in \mathcal{L}_j$, we have $Cost_j(i, \boldsymbol{\theta}_{-j}) = \sum_{e \in i} u_{ije}$.*

Proof. Fix the outcome $(i, \boldsymbol{\theta}_{-j})$. Now, the sojourn-time of j equals the amount of time j is processed, *plus* the amount of time it is delayed due to the other jobs. The former quantity is equal to $\sum_{e \in i} p_{je}$, the latter quantity being $\sum_{j' \neq j} \delta_{ij}(j')$. Thus, we get:

$$
\begin{aligned}
\text{Cost}_j(i, \boldsymbol{\theta}_{-j}) = \sum_{e \in i} p_{je} + \sum_{j' \neq j} \delta_{ij}(j') &= \sum_{e \in i} p_{je} + \sum_{e \in E_i} \sum_{j' \in \Gamma_{ije}} \delta_{ij}(j') \qquad (11)\\
&= \sum_{e \in i \setminus E_i} p_{je} + \sum_{e \in E_i} \left(p_{je} + \sum_{j' \in \Gamma_{ije}} \delta_{ij}(j') \right)\\
&= \sum_{e \in i \setminus E_i} u_{ije} + \sum_{e \in E_i} u_{ije} = \sum_{e \in i} u_{ije}
\end{aligned}
$$

```
INPUT: Any outcome $\boldsymbol{\theta}$ of the game (under SJF policy) that is in a Nash equilibrium.

1. Set $y_j \leftarrow \text{Cost}_j(\boldsymbol{\theta})$.
2. Consider an indicator variable $\lambda_{jet} \in \{0, 1\}$ that is set to 1 iff
      (a) the path $\boldsymbol{\theta}_j$ contains the edge $e$ as a critical edge, i.e., $e \in E_{\boldsymbol{\theta}_j}$ and
      (b) the sojourn-time of job $j$ is at least $t$ under the outcome $\boldsymbol{\theta}$.
   Set $z_{et} \leftarrow \sum_j 2 \cdot \lambda_{jet}$.
3. Set the variable $u_{ije}$ as follows.
      (a) If $e \neq E_i$, then $u_{ije} \leftarrow p_{je}$.
      (b) Else if $e \in E_i$, then let $\Gamma_{ije}$ denote the set of all jobs $j' \neq j$ such that $e$ is
          the slowest edge in $i \cap \boldsymbol{\theta}_{j'}$. Let $\delta_{ij}(j')$ be the total delay experienced by $j$ due to
          the job $j'$ under the outcome $(i, \boldsymbol{\theta}_{-j})$. Set $u_{ije} \leftarrow p_{je} + \sum_{j' \in \Gamma_{ije}} \delta_{ij}(j')$.
```

Fig. 2. Setting the dual variables in LP (6) for unweighted jobs

To see why equation 11 holds, define J^+ to be the set of jobs which make positive contributions towards the sum $\sum_{j' \neq j} \delta_{ij}(j')$, i.e., $J^+ = \{j' \neq j \, : \, \delta_{ij}(j') > 0\}$. Each job $j' \in J^+$ has $i \cap \boldsymbol{\theta}_{j'} \neq \emptyset$, and there is a unique critical edge on path i that is also the slowest edge in $i \cap \boldsymbol{\theta}_{j'}$. So each job $j' \in J^+$ is part of *exactly one* of the sets in $\{\Gamma_{ije} : e \in E_i\}$. In other words, $\{\Gamma_{ije} \, : \, e \in E_i\}$ induces a partition of the jobs in J^+.

Lemma 3. *If the dual variables are assigned values as in Figure 2, then they satisfy the constraints 7 and 8 of LP (6).*

Proof. The right hand side of constraint 7, by Lemma 2, is equal to $\text{Cost}_j(i, \boldsymbol{\theta}_{-j})$. The left hand side, by Figure 2, is equal to $\text{Cost}_j(\boldsymbol{\theta})$. The constraint is satisfied as the Nash equilibrium condition dictates that $\text{Cost}_j(\boldsymbol{\theta}) \geq \text{Cost}_j(i, \boldsymbol{\theta}_{-j})$.

In constraint 8, the edge e is not a critical edge on the path i. Hence, by Figure 2, the quantity z_{et} is zero at all times t. Furthermore, the quantity u_{ije} is set to p_{je}, so that the left hand side of the constraint is 1, which is equal to its right hand side.

Lemma 4. *Fix any job j, any path $i \in \mathcal{L}_j$, any critical edge $e \in E_i$, and any job $j' \in \Gamma_{ije}$. Let $\delta_{ij}(j', t)$ be the total delay experienced by j due to the job j' (anywhere in the tree) on or after time t, under the outcome $(i, \boldsymbol{\theta}_{-j})$. We have $\delta_{ij}(j', t) \leq 2\lambda_{j'et} \cdot p_{je}$.*

Proof. The main difficulty in proving the lemma is that $\delta_{ij}(j', t)$ refers to the outcome $(i, \boldsymbol{\theta}_{-j})$, whereas $\lambda_{j'et}$ refers to the outcome $\boldsymbol{\theta}$. So we introduce the quantity $\lambda^*_{j'et}$, which is the exact analogue of $\lambda_{j'et}$ under the outcome $(i, \boldsymbol{\theta}_{-j})$. Note that the edge e already belongs to $\boldsymbol{\theta}_{j'}$ and is a critical edge. Hence, we drop condition (a) used in defining $\lambda_{j'et}$ in Figure 2. We then consider two possible cases.

$$\lambda^*_{j'et} = \begin{cases} 1 & \text{if the sojourn-time of } j' \text{ under } (i, \boldsymbol{\theta}_{-j}) \text{ is at least } t; \\ 0 & \text{otherwise.} \end{cases}$$

By our assumption in the first paragraph of Section 1.2, either $p_j < p_{j'}$ or $p_j > p_{j'}$.

Case 1. Either $\lambda^*_{j'et} = 0$ or $p_j < p_{j'}$. Here, if $\lambda^*_{j'et} = 1$, then the job j' is already *out of the system* by time t under the outcome $(i, \boldsymbol{\theta}_{-j})$. Naturally, we have $\delta_{ij}(j', t) = 0$, and the lemma holds. On the other hand, if $p_j < p_{j'}$, then SJF scheduling policy ensures that the job j never gets delayed by the job j', and we again have $\delta_{ij}(j', t) = 0$.

Case 2. $\lambda^*_{j'et} = 1$ and $p_j > p_{j'}$. In this case, first note that by Theorem 2, $\delta_{ij}(j', t) \leq 2p_{je}$. Since $\lambda^*_{j'et} = 1$, we get:

$$\delta_{ij}(j', t) \leq 2\lambda^*_{j'et} \cdot p_{je} \tag{12}$$

Since $p_j > p_{j'}$, SJF scheduling policy ensures that the processing of j' is not affected if j switches its path. Specifically, the time-steps at which j' is processed by edge e remains unchanged under the two outcomes $(i, \boldsymbol{\theta}_{-j})$ and $\boldsymbol{\theta}$. Thus, we have $\lambda_{j'et} = \lambda^*_{j'et}$, and equation 12 implies that the lemma holds.

Lemma 5. *If the dual variables are assigned values as in Figure 2, then they satisfy all the constraints of LP (6).*

Proof. By Lemma 3, the constraints 7, 8 are already satisfied. We focus on the remaining constraint 9. Fix any job j, any path $i \in \mathcal{L}_j$, and any edge $e \in E_i$. By Lemma 4:

$$\sum_{j' \in \Gamma_{je}} \delta_{ij}(j', t) \leq p_{je} \cdot \sum_{j' \in \Gamma_{je}} 2\lambda_{j'et} \leq p_{je} \cdot z_{et} \tag{13}$$

Under the outcome $(i, \boldsymbol{\theta}_{-j})$, the total delay experienced by j due to the jobs in Γ_{je} till time-step t is, by definition, at most t. This leads to the following inequality.

$$\sum_{j' \in \Gamma_{je}} (\delta_{ij}(j') - \delta_{ij}(j', t)) \leq t \tag{14}$$

From Figure 2, equation 14 and equation 13, we see that constraint 9 is satisfied.

$$u_{ije} = p_{je} + \sum_{j' \in \Gamma_{je}} \delta_{ij}(j') \leq p_{je} + t + \sum_{j' \in \Gamma_{je}} \delta_{ij}(j', t) \leq p_{je} + t + p_{je} \cdot z_{et}.$$

Lemma 6. *If the dual variables are set as in Figure 2, then the objective of LP (6) is at least* $(1/2) \cdot \sum_j \mathit{Cost}_j(\boldsymbol{\theta})$.

Proof. Fix the outcome $\boldsymbol{\theta}$, and focus on any job j with sojourn-time $\text{Cost}_j(\boldsymbol{\theta})$.

(Case 1) $t \leq \text{Cost}_j(\boldsymbol{\theta})$. In this case, Figure 2 implies that the job j contributes 2 to each of the z_{et}'s corresponding to the critical edges in the path $\boldsymbol{\theta}_j$, and it makes zero contribution to the remaining z_{et}'s. Since $\boldsymbol{\theta}_j$ has at most K critical edges, the total contribution of the job to the sum $\sum_e z_{et}$ is at most $2K$.

(Case 2) $t > \text{Cost}_j(\boldsymbol{\theta})$. Here, by Figure 2, the job j contributes 0 to the sum $\sum_e z_{et}$.

Summing over all time-steps, the contribution of any single job j to the sum $\sum_{e,t} z_{et}$ is at most $2K \cdot \text{Cost}_j(\boldsymbol{\theta})$. Next, summing over the contributions from all the jobs, we infer that $\sum_{et} z_{et} \leq (2K) \cdot \sum_j \text{Cost}_j(\boldsymbol{\theta})$. Since $y_j = \text{Cost}_j(\boldsymbol{\theta})$ for all jobs j, we get:

$$\text{LP-objective} = \sum_j y_j - (1/(4K)) \cdot \sum_{e,t} z_{et} \geq (1/2) \cdot \sum_j \text{Cost}_j(\boldsymbol{\theta}).$$

Theorem 3 now follows from Lemma 1, Lemma 5, and Lemma 6.

Theorem 3. *If every machine follows Shortest Job First (SJF) policy and every job has unit weight, then the price of anarchy of the induced game is* $O(\log^2(s_{\max}/s_{\min}))$*, where* $s_{\max}$ *(resp.* $s_{\min}$*) is the maximum (resp. minimum) speed among all the machines.*

Interestingly, we get super-constant upper bound on the PoA because the speeds of the edges may keep on decreasing as we traverse farther along a path starting from the root-node. This is precisely the situation in the real-world fat-tree networks. In contrast, consider an instance where the edges adjacent to the root-node have the slowest speeds. In this instance, the proof of Theorem 2 implies that a job can get delayed by other jobs only on the first edge on its path. Thus, we can write a new time-indexed LP where we take into account the fractional completion times of the jobs only on these edges at depth one. This LP will give a constant approximation to the underlying optimization problem, as a path starting from the root contains exactly one edge of depth one. We can then execute the same analysis as outlined in this section to get constant PoA.

3 $O(d^2)$ PoA for Weighted Jobs

In this section we observe that when jobs can have arbitrarily different weights, the PoA of the game depends polynomially on the depth of the tree.

Theorem 4. *There is an instance of the problem where if every edge follows HDF scheduling policy, then the game induced between the jobs has PoA* $= \Omega(\sqrt{d})$.

On the positive side, we can extend our dual-fitting framework to derive the following upper bound on the PoA.

Theorem 5. *If every machine follows HDF scheduling policy, then the PoA of the game induced between the jobs is at most* $8d^2$*, where* d *is the depth of the tree.*

Ackowledgements. The authors thank Sungjin Im and Kamesh Munagala for several helpful discussions.

References

1. Al-Fares, M., Loukissas, A., Vahdat, A.: A scalable, commodity data center network architecture. In: Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication, SIGCOMM 2008, pp. 63–74. ACM, New York (2008)
2. Anand, S., Garg, N., Kumar, A.: Resource augmentation for weighted flow-time explained by dual fitting. In: Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, pp. 1228–1241. SIAM (2012)
3. Antoniadis, A., Moseley, B., Barcelo, N., Nugent, M., Cole, D., Pruhs, K.: Packet forwarding algorithms in a line network (2014)
4. Azar, Y., Jain, K., Mirrokni, V.: (almost) optimal coordination mechanisms for unrelated machine scheduling. In: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, pp. 323–332. Society for Industrial and Applied Mathematics, Philadelphia (2008)

5. Bansal, N., Kulkarni, J.: Minimizing flow-time on unrelated machines. CoRR, abs/1401.7284 (2014)
6. Bhattacharya, S., Im, S., Kulkarni, J., Munagala, K.: Coordination mechanisms from (almost) all scheduling policies. In: 5th Innovations in Theoretical Computer Science Conference, ITCS 2014 (2014)
7. Bilò, V., Fanelli, A., Moscardelli, L.: On lookahead equilibria in congestion games. In: Chen, Y., Immorlica, N. (eds.) WINE 2013. LNCS, vol. 8289, pp. 54–67. Springer, Heidelberg (2013)
8. Caragiannis, I.: Efficient coordination mechanisms for unrelated machine scheduling. In: SODA, pp. 815–824 (2009)
9. Chekuri, C., Khanna, S.: Approximation algorithms for minimizing average weighted completion time (2004)
10. Christodoulou, G., Koutsoupias, E., Nanavati, A.: Coordination mechanisms. Theor. Comput. Sci. 410(36), 3327–3336 (2009)
11. Christodoulou, G., Mehlhorn, K., Pyrga, E.: Improving the price of anarchy for selfish routing via coordination mechanisms. In: Demetrescu, C., Halldórsson, M.M. (eds.) ESA 2011. LNCS, vol. 6942, pp. 119–130. Springer, Heidelberg (2011)
12. Cole, R., Correa, J.R., Gkatzelis, V., Mirrokni, V., Olver, N.: Inner product spaces for minsum coordination mechanisms. In: Proceedings of the 43rd Annual ACM Symposium on Theory of Computing, STOC 2011, pp. 539–548. ACM Press, New York (2011)
13. Farzad, B., Olver, N., Vetta, A.: A priority-based model of routing. Chicago J. Theor. Comput. Sci. 2008 (2008)
14. Harris, D.G., Srinivasan, A.: Constraint satisfaction, packet routing, and the lovasz local lemma. In: STOC, pp. 685–694 (2013)
15. Hoefer, M., Mirrokni, V.S., Röglin, H., Teng, S.-H.: Competitive routing over time. In: Leonardi, S. (ed.) WINE 2009. LNCS, vol. 5929, pp. 18–29. Springer, Heidelberg (2009)
16. Im, S., Moseley, B.: Scheduling over tree. Personal communication (2014)
17. Immorlica, N., Li, L.E., Mirrokni, V.S., Schulz, A.S.: Coordination mechanisms for selfish scheduling. Theor. Comput. Sci. 410(17), 1589–1598 (2009)
18. Karger, D., Stein, C., Wein, J.: Scheduling algorithms (1997)
19. Koch, R., Nasrabadi, E., Skutella, M.: Continuous and discrete flows over time - a general model based on measure theory. Math. Meth. of OR 73(3), 301–337 (2011)
20. Koutsoupias, E., Papadimitriou, C.H.: Worst-case equilibria. In: STACS, pp. 404–413 (1999)
21. Koutsoupias, E., Papakonstantinopoulou, K.: Contention issues in congestion games. In: ICALP (2), pp. 623–635 (2012)
22. Leighton, F.T., Maggs, B.M., Rao, S.: Universal packet routing algorithms (extended abstract). In: FOCS, pp. 256–269 (1988)
23. Leiserson, C.E.: Fat-trees: Universal networks for hardware-efficient supercomputing. IEEE Trans. Computers 34(10), 892–901 (1985)
24. Peis, B., Skutella, M., Wiese, A.: Packet routing: Complexity and algorithms. In: Bampis, E., Jansen, K. (eds.) WAOA 2009. LNCS, vol. 5893, pp. 217–228. Springer, Heidelberg (2010)
25. Rothvoß, T.: A simpler proof for $O(\text{Congestion}+\text{Dilation})$ packet routing. In: Goemans, M., Correa, J. (eds.) IPCO 2013. LNCS, vol. 7801, pp. 336–348. Springer, Heidelberg (2013)
26. Roughgarden, T.: Intrinsic robustness of the price of anarchy. In: STOC, pp. 513–522 (2009)
27. Roughgarden, T., Tardos, É.: How bad is selfish routing? In: FOCS, pp. 93–102 (2000)
28. Smith, W.: Various optimizers for single-stage production. Naval Research Logistics Quarterly (3), 59–66 (1956)

On Area-Optimal Planar Graph Drawings[⋆]

Therese Biedl

David R. Cheriton School of Computer Science, University of Waterloo,
Waterloo, Ontario N2L 1A2, Canada
biedl@uwaterloo.ca

Abstract. One of the first algorithmic results in graph drawing was how to find a planar straight-line drawing such that vertices are at grid-points with polynomial coordinates. But not until 2007 was it proved that finding such a grid-drawing with optimal area is NP-hard, and the result was only for disconnected graphs.

In this paper, we show that for graphs with bounded treewidth, we can find area-optimal planar straight-line drawings in one of the following two scenarios: (1) when faces have bounded degree and the planar embedding is fixed, or (2) when we want all faces to be drawn convex. We also give NP-hardness results to show that none of these restrictions can be dropped. In particular, finding area-minimal drawings is NP-hard for triangulated graphs minus one edge.

1 Introduction

A planar graph is a graph that can be drawn without crossing in the plane. Naturally one wonders whether such a drawing must use curves, or whether there exists a *planar straight-line drawing*, i.e., a drawing such that vertices are at points, edges are straight-line segments between their endpoints, no edge overlaps a non-incident vertex, and no two edges cross. It was proved multiple times independently that every planar graph has such a drawing [4][12][14].

To increase readability of such a drawing, vertices should be not too close to each other, but the drawing should fit on a small paper or screen. The first objective can be achieved by demanding a *grid drawing*, where all vertices are placed at points with integer coordinates. The second objective can be achieved by minimizing the area of the smallest enclosing box of such a grid-drawing. In 1990, it was shown independently by de Fraysseix, Pach and Pollack [6] and Schnyder [11] that every planar graph has a grid-drawing of area $O(n^2)$.

Numerous papers have since worked on improving the constant factor in this $O(n^2)$-bound; see e.g. [2] and the references therein. However, very few attempts have been made to find drawings with the *optimal* area. Indeed, it was not even shown until 2007 that finding a grid-drawing with optimal area is NP-hard [8]. One of the very few algorithms that achieves optimal area is by Mondal et al. [9]

[⋆] Supported by NSERC and the Ross and Muriel Cheriton Fellowship. Research initiated while participating at Dagstuhl seminar 13421. The author would like to thank Martin Vatshelle; many ideas in this paper were strongly inspired by [1].

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 198–210, 2014.

for triangulated graphs of treewidth 3. The current paper arose out of an attempt to generalize this results to graphs of larger (but bounded) treewidth. We show:

> Minimizing the area of planar grid-drawings is polynomial-time solvable for any planar graph of bounded treewidth and bounded face degrees for which the planar embedding is fixed.

We also prove NP-hardness as soon as any of the three conditions (bounded treewidth, bounded face degree, fixed planar embedding) is dropped. We then turn to *convex drawings*, where we additionally demand that every face (including the outer-face) is drawn as a convex polygon, and show:

> Minimizing the area of convex planar grid-drawings is polynomial-time solvable for any planar graph of bounded treewidth.

We also prove NP-hardness of finding area-optimal convex drawings for graphs where the treewidth is not constant. Due to space restrictions, many details (especially for the NP-hardness reductions) have been omitted.

2 Background

We assume that $G = (V, E)$ is a planar graph, i.e., it can be drawn in the plane without crossing. Any planar drawing of G defines a *rotation system*, i.e., a clockwise order of edges around each vertex. This defines *facial circuits*, which are boundary cycles of the maximal connected regions (*faces*) of the drawing. The *outer-face* is the unbounded face. The drawing also assigns each *angle* (set of two consecutive edges at a vertex) to a face. A *planar embedding* is a rotation system, an angle-face assignment, and one angle fixed as being on the outer-face; this determines a drawing of G up to deformations of the plane.

A graph is *connected* if there exists a path between any two vertices. For a connected graph every face is bounded by one circuit and hence the angle-face-assignment is unique. A graph is *k-connected* if it remains connected after removing any $k-1$ vertices. Any 3-connected planar graph has a unique rotation system (up to reversal of all orders). A planar graph is called *triangulated* if all faces, including the outer-face, are triangles. Such a graph is always 3-connected.

A *planar straight-line grid-drawing* is a mapping Γ of the vertices of G to distinct points with integer coordinates such that if we draw every edge as a straight-line segment, then no edge overlaps a non-incident vertex, and no two edges cross. The drawing is *convex* if all faces, including the outer-face, are drawn as convex polygons. Angles of 180° are allowed (though the results of the paper can easily be generalized to strictly convex drawings). The *width*, *height* and *area* of Γ is the corresponding measure of the minimum axis-aligned box that encloses all points that Γ maps to.

Definition 1. AREAMINIMIZATION *is the following problem: Given a graph G and an integer A, does G have a planar straight-line grid-drawing of area $\leq A$?*

POINTSETEMBEDDABILITY *is the following problem: Given a graph G and a set S of points in $\mathbb{R}^2$, is there a planar straight-line drawing of G where all vertices are on points of S?*

For both problems, we also consider the Convex variant, where we require the planar drawing to be convex.

Treewidth and Sc-decompositions: We will not define the treewidth tw(G) of a graph G, since we do not use it directly. The *pathwidth* pw(G) of a graph G is the smallest k such that G has a vertex order $v_1, \dots, v_n$ where, for any $1 \leq i < n$, at most k vertices in $v_1, \dots, v_i$ have a neighbour in $v_{i+1}, \dots, v_n$. Since tw$(G) \leq$ pw(G), it suffices to prove NP-hardness for bounded pathwidth graphs. A *branch decomposition* of a graph G is a rooted binary tree T with edges of G in 1-1 correspondence with the leaves of T. It will be convenient to assume that the root has only one child. For any arc a of T, the *subgraph* G_a *rooted at* a is the graph formed by all edges at leaves that are below a in T. The *separator* σ_a *at* a is the set of vertices with incident edges in both G_a and $G - G_a$. The *width* of a branch decomposition is $\max_{a \in T} |\sigma_a|$, and the *branchwidth* bw(G) of G is the smallest width of a branch decomposition of G. Since tw$(G) \in \Theta($bw$(G))$ [10], it suffices to give algorithms for bounded branchwidth graphs.

Let G be a planar graph with a fixed rotation system. A *noose* is a sequence $v_0, f_0, v_1, \dots, v_{k-1}, f_{k-1}$ such that for any $0 \leq i < k$, vertices v_i and v_{i+1} both belong to face f_i (addition modulo k), and no vertex repeats. (Faces may repeat.) An *sc-decomposition* is a branch decomposition of G such that for any arc a of the tree T, there exists a noose N_a whose vertices are exactly the separator σ_a. We can picture N_a as a simple closed curve that intersects a planar drawing of G only at vertices in σ_a, contains G_a on one side and $G - G_a$ on the other side. See Fig. 1. Any 2-connected planar graph G with a fixed planar embedding has an sc-decomposition of width bw(G), and it can be found in polynomial time [3].

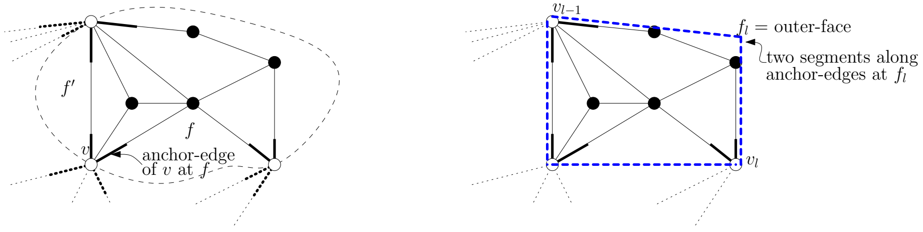


Fig. 1. A noose (dashed) meets the separator vertices (white). Illustrating some concepts for Section 3: Anchor-edges are thick, pole-edges are thick dotted, the noose-polygon is thick dashed.

3 Area-Optimal Drawings

AreaMinimization is NP-hard even for graphs with a fixed planar embedding and constant treewidth [8]. But it becomes polynomial if additionally face-degrees are bounded:

Theorem 1. *Let G be a planar graph with a fixed planar embedding. If G has bounded treewidth and bounded face-degrees, then* AreaMinimization *can be solved in polynomial time.*

Proof. It was shown in [1] that PointSetEmbeddability is polynomial-time solvable for these graphs. For $W = 1, \dots, \lceil\sqrt{A}\rceil$ and $H = \lfloor A/W \rfloor$, solve PointSetEmbeddability for G, using the points of a $W \times H$-grid as S. There exists a drawing of area at most A if and only if we succeed for some S. Clearly this yields a polynomial-time algorithm since $A \leq n^2$.[1]

For convex drawings, the results for ConvexPointSetEmbeddability in [1] are more restrictive than we need them to be. We show in the rest of this section that ConvexPointSetEmbeddability is polynomial-time solvable for a planar graph G of bounded treewidth. This was previously only known for graphs with bounded treewidth, bounded vertex degrees, and where the planar embedding is fixed [1]. So assume that we are given a set S of at least n points in the plane. Fix an arbitrary planar embedding of G (we will later explore all possible ones). The main idea is to do dynamic programming in an sc-decomposition of width $\mathrm{bw}(G)$, where the dynamic programming function fixes the position of separator-vertices, as well as their neighbours at the noose.

Formally, let a be an arc of the sc-decomposition tree T, and let v be a vertex of σ_a. By assumption the noose N_a contains f, v, f' as subsequence, for some faces f, f'. By definition of separator, v has incident edges in both G_a and $G - G_a$. Since N_a contains v only once, the incident edges of v hence form two intervals (in the clockwise order around v): one with edges in G_a and one with edges in $G - G_a$. We call the first and last edge of v in G_a the *anchor-edges of* v, and the first and last edge of v in $G - G_a$ the *pole-edges of* v. (Both terms are "with respect to arc a", but arc a will be clear from the context.) Each anchor-edge and pole-edge belongs to either f or f'. See also Fig. 1. An *anchor/pole* of arc a is a vertex x such that (x, v) is an anchor-edge/pole-edge at some vertex v in σ_a. Let A_a be the set of anchors and poles of a, and let G_a^+ be the graph obtained from G_a by adding to it all pole-edges at vertices in σ_a. The dynamic programming function searches for a drawing of G_a^+, subject to a fixed mapping Γ_a of the vertices of $\sigma_a \cup A_a$ to points in S.

From the locations of $\sigma_a \cup A_a$, we can read a polygon that serves as curve for the noose; we call this the *noose-polygon* $\mathcal{P}_a$. Namely, consider the polygon $\Gamma_a(v_0), \dots, \Gamma_a(v_{k-1})$, where $v_0, f_0, v_1, \dots, v_{k-1}, f_{k-1}$ are the vertices and faces of the noose N_a. If N_a does not include the outer-face, then $\mathcal{P}_a$ is this polygon. If N_a does include the outer-face, say at f_l, then replace line segment $\overline{\Gamma_a(v_l), \Gamma_a(v_{l+1})}$ by two segments along the supporting lines of the anchor-edge of v_l and v_{l+1} at f_l. See Fig. 1(right). In any convex drawing of G, the noose-polygon $\mathcal{P}_a$ does not cross itself, since every edge of it either resides in an interior face (a convex polygon) or is drawn along two supporting lines of edges on the

[1] The precise run-time is $O(n^3 \Delta(t+1) \log(\Delta(t+1)) \cdot |S|^{1.5\Delta(t+1)})$, where Δ is the maximum degree of a face and t is the treewidth.

outer-face (a convex polygon). If all vertices of σ_a are collinear (e.g. if $|\sigma_a| = 2$), then $\mathcal{P}_a$ may overlap itself; we do not consider this a crossing.

One final notation. In what follows, we consider a drawing Γ of a subgraph G_a^+ of G. We say that an angle is *drawn properly* in Γ if either this angle is not a facial angle of G, or if it belongs to an interior face of G and is drawn with at most 180°, or it belongs to the outer-face of G and is drawn with at least 180°. The function to be computed via dynamic programming is now as follows:

Definition 2. *Let a be an arc of the rooted sc-decomposition. Let Γ_a be a mapping from $\sigma_a \cup A_a$ to S. Define $M(a, \Gamma_a)$ to be true if and only if:*

1. *The noose-polygon $\mathcal{P}_a$ defined by Γ_a has no crossings.*
2. *For any anchor α and any pole ρ of a, any curve from $\Gamma_a(\alpha)$ to $\Gamma_a(\rho)$ contains points of $\mathcal{P}_a$. Put differently, the noose-polygon $\mathcal{P}_a$ forms a boundary between the anchors and the poles.*
3. *There exists a planar drawing Γ of G_a^+ on S such that all angles are drawn properly and Γ coincides with Γ_a on $\sigma_a \cup A_a$.*

Observe that computing $M(a, \Gamma_a)$ for all values of a and Γ_a is sufficient to solve ConvexPointSetEmbeddability for a graph with a fixed planar embedding. For $G_r = G$ at the unique arc r below the root, and so $\bigvee_{\Gamma_r} M(r, \Gamma_r)$ is true if and only if G has a drawing on S for which all angles are drawn properly, i.e., G has a convex drawing on S. We explain how to compute $M(a, \Gamma_a)$ by going bottom-up in the tree T of the sc-decomposition.

$M(a, \Gamma_a)$ at a leaf-arc: Assume first that a is an arc incident to a leaf, say the leaf stores edge (v, w). Then G_a^+ consists of (v, w) as well as up to four pole-edges. (The anchor-edges all coincide with (v, w).) Hence all vertices of G_a^+ belong to $\sigma_a \cup A_a$, so the mapping Γ_a determines the drawing of G_a^+. Testing whether $M(a, \Gamma_a)$ is true hence reduces to checking whether the angles are drawn properly or not. Further, to respect the given planar embedding, the clockwise order of pole-edges and (v, w) must be as induced by the rotation scheme. This can all be tested in constant time for one fixed arc a and mapping Γ_a.

$M(a, \Gamma_a)$ at a non-leaf arc: So assume now that arc a is not incident to a leaf. Then the lower end of a is in turn incident to two other arcs a_1 and a_2. We now show how to extract the value for $M(a, \Gamma_a)$ from those of $M(a_1, \Gamma_{a_1})$ and $M(a_2, \Gamma_{a_2})$ for some suitably chosen mappings Γ_{a_1} and Γ_{a_2}.

Recall that Γ_a determines the positions for all points in $\sigma_a \cup A_a$. With this, we can test the first two conditions of Definition 2 directly, and assume from now on that they are satisfied. In particular, the noose-polygon $\mathcal{P}_a$ then has an *anchor-side*, which is the connected component of $\mathbb{R}^2 - \mathcal{P}_a$ that contains the anchors, and the *pole-side*, which is the connected component that contains the poles. Define $\sigma_\times := (\sigma_{a_1} \cup \sigma_{a_2}) - \sigma_a$ and $A_\times := (A_{a_1} \cup A_{a_2}) - A_a$. If we fix any mapping of $\sigma_a \cup \sigma_\times \cup A_a \cup A_\times$ to points in S, then this fixes (for $i = 1, 2$) also a mapping of $\sigma_{a_i} \cup A_{a_i}$ to points in S. One can easily show the following formula:

Lemma 1. *$M(a, \Gamma_a)$ is true if and only if the first two conditions of Definition 2 are true, and there exists a mapping $\Gamma_\times$ from $\sigma_\times \cup A_\times$ to S such that*

- *$M(a_1, \mathit{\Gamma}_{a_1})$ and $M(a_2, \mathit{\Gamma}_{a_2})$ are true, and*
- *the interior of the anchor-side of $\mathcal{P}_{a_1}$ has no points in common with the interior of the anchor-side of $\mathcal{P}_{a_2}$,*

where $\mathit{\Gamma}_{a_i}$ is the mapping from $\sigma_{a_i} \cup A_{a_i}$ to S induced by $\mathit{\Gamma}_a$ and $\mathit{\Gamma}_\times$.

The algorithm for computing $M(a, \mathit{\Gamma}_a)$ is now the obvious: For any choice $\mathit{\Gamma}_\times$ of mapping $\sigma_\times \cup A_\times$ to points of S, compute the induced mappings $\mathit{\Gamma}_{a_i}$ and look up whether $M(a_i, \mathit{\Gamma}_{a_i})$ is true, for $i = 1, 2$. Also compute $\mathcal{P}_{a_i}$, and check whether the anchor-sides are interior-disjoint. Set $M(a, \mathit{\Gamma}_a)$ to be true if and only if we succeed for some choice of $\mathit{\Gamma}_\times$.

Putting it All Together. So to solve ConvexPointSetEmbeddability for a fixed planar embedding, we go bottom-up in the tree T of the sc-decomposition, and at each arc a and each possible mapping $\mathit{\Gamma}_a$ compute $M(a, \mathit{\Gamma}_a)$ as explained above. It remains to analyze the run-time. Computing $M(a, \mathit{\Gamma}_a)$ for an arc a incident to a leaf and a fixed $\mathit{\Gamma}_a$ takes constant time. Doing so for all $\mathit{\Gamma}_a$ takes $O(|S|^5)$ time since there are five vertices to which we must assign points. To compute $M(a, \mathit{\Gamma}_a)$ for an arc a not incident to a leaf and a fixed assignment $\mathit{\Gamma}_a$, we must try all possible mappings $\mathit{\Gamma}_\times$ of points to $\sigma_\times \cup A_\times$. For each of them, we must compute the induced assignments $\mathit{\Gamma}_{a_1}$ and $\mathit{\Gamma}_{a_2}$, look up $M(a_1, \mathit{\Gamma}_{a_1})$ and $M(a_2, \mathit{\Gamma}_{a_1})$, and test whether the noose-polygons are interior-disjoint. This can all be done in $O(n \log n)$ time with suitable data structures. Thus the time to compute $M(a, \cdot)$, for all possible choices of $\mathit{\Gamma}_a$ and $\mathit{\Gamma}_\times$ is

$$O\left(|S|^{|\sigma_a \cup A_a \cup \sigma_\times \cup A_\times|} n \log n\right).$$

Because any separator-vertex contributes at most two anchors and two poles, and any separator-vertex appears in at least two of $\sigma_a, \sigma_{a_1}, \sigma_{a_2}$, one can argue that $|\sigma_a \cup A_a \cup \sigma_\times \cup A_\times| \leq \frac{15}{2}\mathrm{bw}(G)$. Hence the time to compute $M(a, \cdot)$ is $O(|S|^{7.5\mathrm{bw}(G)} n \log n)$, and doing so for all $O(n)$ arcs of the sc-decomposition adds another $O(n)$-factor. So we have:

Theorem 2. *For any planar graph G and any set S of points in $\mathbb{R}^2$, if the planar embedding of G is fixed then we can solve* ConvexPointSetEmbeddability *in $O((|S|^{7.5bw(G)} n^2 \log n)$ time.*

Now we consider the case when the planar embedding is not fixed. If G is 3-connected, then simply try all possible outer-faces for an additional $O(n)$ run-time overhead. So assume from now on that G has cutting pairs.

A graph may have $\mathit{\Omega}(2^n)$ rotation systems that all lead to a convex drawing, so we cannot explore all of them explicitly. Tutte's characterization [13] states that if G has a convex drawing, then any cutting pair must be on the outer-face and have exactly two cut-components (not counting a possible edge between the cutting pair). Therefore rotation systems of convex drawings can differ only by "flipping" (reversing the rotation sub-systems) of a *leaf-component*, i.e., a 3-connected component that is a leaf in the tree of 3-connected components. We use a special branch-decomposition:

Lemma 2. *For any 2-connected graph G, there exists a branch decomposition T of G of width $bw(G)$ such that*

- *for any leaf-component $\mathcal{C}$ there exists an arc $a_{\mathcal{C}}$ in T with $G_{a_{\mathcal{C}}} = \mathcal{C}$,*
- *for any planar embedding of G and any arc a in T, there exists a noose that contains G_a on one side and $G - G_a$ on the other. Furthermore, the (clockwise or counter-clockwise) order of the vertices of the noose is the same regardless of the planar embedding.*

Previously, anchors and poles were vertices defined by the planar embedding. Now we only know which two vertices v_ℓ and $v_{\ell+1}$ are consecutive in the noose at an arc a, but this is sufficient. Change the definition of Γ_a as follows: let Γ'_a be a mapping that assigns five points to each vertex $v_\ell \in \sigma_a$. These five points belong to v_ℓ, the (unknown) anchor at v_ℓ "towards" $v_{\ell+1}$ (i.e., at the (unknown) face which v_ℓ shares with $v_{\ell+1}$,), the pole at v_ℓ twoards $v_{\ell+1}$, and the anchor and pole at v_ℓ "towards" $v_{\ell-1}$. We allow Γ'_a to repeat points, which avoids having to explore explicitly whether these poles/anchors are distinct vertices. Notice that Γ'_a is sufficient to compute the noose-polygon, as long as we pass along the information which consecutive vertices (if any) of the noose belong to the outer-face. Also, Γ'_a determines a drawing of G_a^+, given one of G_a. Hence define $M'(a, \Gamma'_a)$ to be verbatim the same as $M(a, \Gamma_a)$, except that we use Γ'_a and allow all possible planar embeddings in (3).

The computation of $M'(a, \Gamma'_a)$ is nearly the same as the one of $M(a, \Gamma_a)$, except that in the base case we do not check whether the rotation system is respected, and that at any arc of a leaf component we test both possible choices of which face of the noose is the outer-face. Hence $M'(a, \Gamma'_a)$ explores all possible ways of flipping leaf components, and hence implicitly all planar embeddings that could lead to a convex drawing. Thus the run-time is the same as for the fixed planar embedding, except that we need an additional $O(n)$ factor for trying all possible outer-faces (this is needed only if the graph is 3-connected.) Summarizing, we get:

Theorem 3. *For any planar graph G and any set S of points in $\mathbb{R}^2$, we can solve* ConvexPointSetEmbeddability *in $O((|S|^{7.5bw(G)} n^3 \log n)$ time.*

With the same approach as in Theorem 1 (try all choices of S as a $W \times H$-grid for $W \cdot H \leq A$) we hence have:

Corollary 1. *Let G be a planar graph. If G has bounded treewidth, then* ConvexAreaMinimization *can be solved in polynomial time.*

We can use this to give subexponental exact algorithms for area-optimal convex drawings. We are not familiar with any previous results in this area. The obvious brute-force approach (try for any assignment of grid points to the vertices whether it works) yields an algorithm with run-time $O^*((n^2)^n)$, where the $O^*(\cdot)$ notation hides polynomial terms.

Corollary 2. *There exists an algorithm to find a minimum-area convex grid-drawing of a planar graph in $O^*(2^{O(\sqrt{n}\log n)})$ time.*

Proof. Any planar graph has branchwidth $O(\sqrt{n})$. By solving ConvexAreaMinimization for values $A = 1, 2, \dots, n^2$ we can hence find the minimum-area convex grid-drawing in time $O^*((n^2)^{O(\sqrt{n})}) = O^*(2^{O(\sqrt{n}\log n)})$ time.

4 NP-Hardness Results

We now give NP-hardness proofs that show that none of the conditions needed for Theorem 1 and Corollary 1 can be dropped. Our reductions borrow many ideas from [8] and [1].

4.1 Small Treewidth, Small Face-Degrees, Flexible Embedding

Recall that AreaMinimization is polynomial-time solvable if the treewidth and the face-degrees are bounded, and the planar embedding is fixed. We now show that if we allow to choose the planar embedding, the problem becomes NP-hard.

The reduction is from the 3-Partition problem defined as follows: Given $3n$ positive integers $a_1, \dots, a_{3n}$, where $\sum_{i=1}^{3n} a_i = n \cdot B$ and $\frac{1}{4}B < a_i < \frac{1}{2}B$ for all i, is there a partition of $a_1, \dots, a_{3n}$ into n groups of 3 numbers each such that each group sums to B? It is well-known that 3-Partition is strongly NP-hard [7]. Given an instance of 3-Partition, we first define a frame, shown for $n = 3$ and $B = 6$ in Fig. 2. It consists of a $W \times 4$-grid (for $W \geq n(B+1)+2$ odd) with n repetitions of a pattern that leaves a face with $B+3$ points not used by the frame. Above and below this strip are $(W+1)/2$ stacked cycles (not shown fully in Fig. 2); all except the outer-most one are 4-cycles. Set $A := W(2W+4)$.

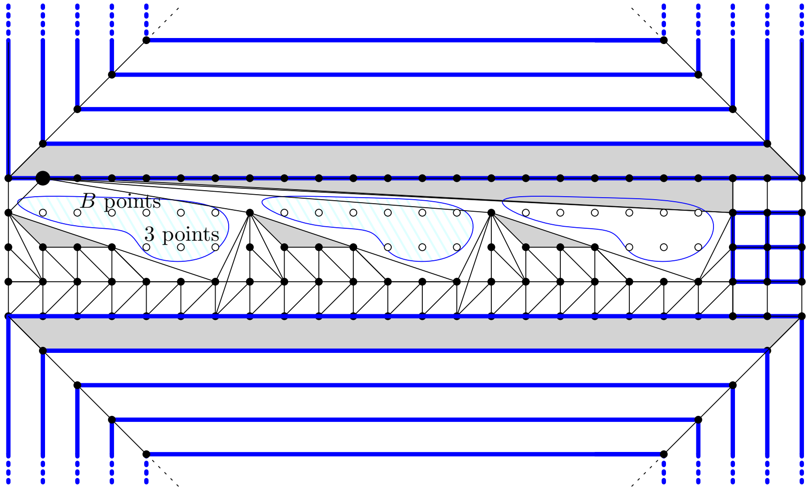


Fig. 2. The frame of the NP-hardness construction. Shaded areas are triangulated; we omit drawing edges in these areas for clarity.

The frame is 3-connected (recall that shaded areas are triangulated) and has a unique rotation system. Any k stacked cycles require a $2k \times 2k$-grid in any planar drawing. Since the frame has two sets of $(W+1)/2$ stacked cycles and a 2×2-grid (shown bold) that are vertex-disjoint, one can argue that any drawing of area A must have the outer-face shown in the figure and the drawing must be (up to rotation) in a $W \times (2W+4)$-grid.

The graph in the middle strip has (once we add the gadgets for the a_i's) *exactly* as many vertices as we can make grid points available to it. So not a single grid point may be "wasted" by not having a vertex on it. One can argue (details are omitted) that this forces the frame to be drawn exactly as shown.

For each $i = 1, \ldots, 3n$, define a path of length a_i. Each vertex on this path is connected to the "fat" vertex of the frame; this is the unique vertex in the graph where the rotation system can be changed. Also add one vertex x_i per index i, which is adjacent to all vertices of the a_i-path. See Fig. 3.

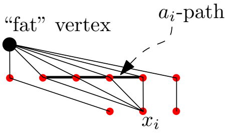


Fig. 3. Encoding $a_1{=}1, a_2{=}4, a_3{=}1$

The frame left n faces with $B+3$ points each. If the 3-partition instance has a solution, then for each group $a_{i_1}, a_{i_2}, a_{i_3}$ that sums to B, we pick one of these faces, place the a_i-paths in the row with B points, and $x_{i_1}, x_{i_2}, x_{i_3}$ in the row below. All edges can be drawn without crossing, and the face that results has degree 17 (12 edges from the three gadgets, and 5 edges from the frame.) Vice versa, if the graph can be drawn in area A, then the frame must be drawn as shown, and so the a_i-gadgets must be split up among the n faces with $B+3$ points each. This gives a partition of the a_i's as desired.

Observe that the middle strip, including the a_i-gadgets, could always be drawn on 5 rows if we allowed an increase in width. It follows that the middle strip has pathwidth at most 5 [5]. Each set of stacked cycles has pathwidth at most 4. By combining their vertex orders, one can hence show that the graph has pathwidth at most 7.

Theorem 4. AreaMinimization *is NP-hard, even for a connected planar graph with pathwidth at most 7, and even if we demand a planar drawing where every face has at most degree 17.*

4.2 3-Connected, and Convex Faces or Small Treewidth

Recall that AreaMinimization is polynomial-time solvable if the treewidth and the face-degrees are bounded, and the rotation system is fixed. We now show NP-hardnes if the condition on treewidth is dropped. The same construction also works for NP-hardness of ConvexAreaMinimization.

Let $a_1, \ldots, a_{3n}$ be an instance of 3-partition. Define $\ell := \max\{12n^2 + 3n, \lceil (B-1)/2 \rceil\}$. We construct a graph that is 3-connected, hence has a unique rotation system. All faces are triangles except one face of degree 4. Fig. 4 shows the *frame* of the graph. The width and height of this drawing is $W := 6\ell + 2n + 6$, and we set $A := W^2$. The frame has three connected components, the *outer frame* as well as two *blobs* that consist of ℓ stacked cycles with one vertex in the middle.

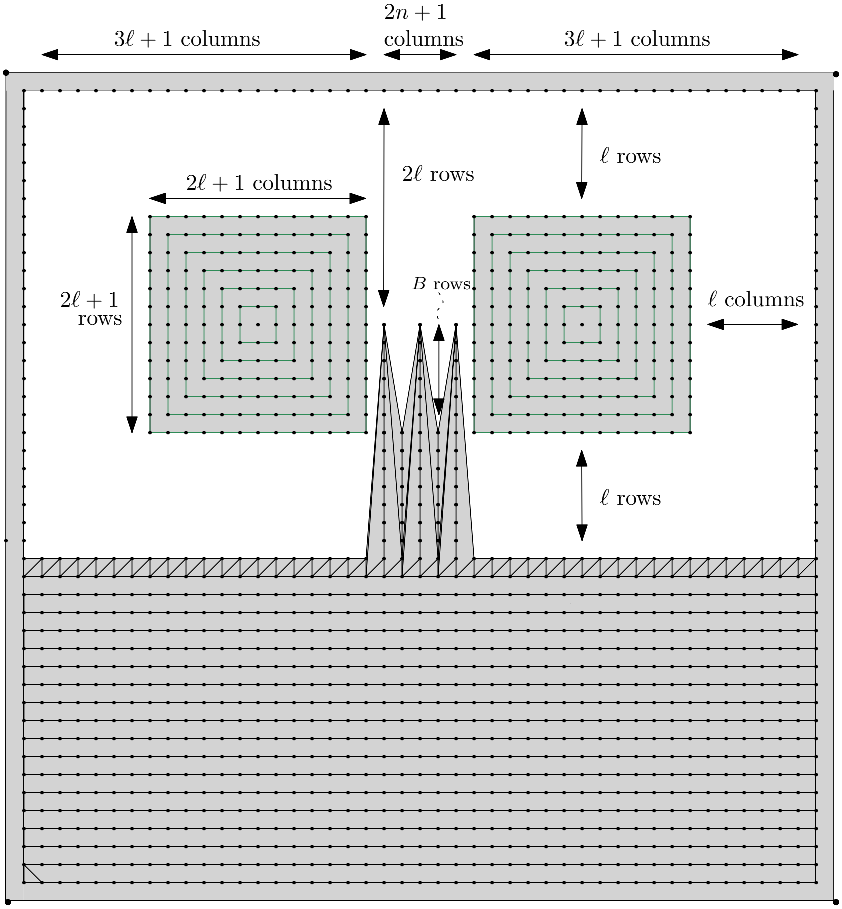


Fig. 4. The NP-hardness reduction for inner triangulated graphs. The frame for $n = 2$ and $B = 6$. The picture uses $\ell = 6$ (while $\ell = 56$ would be correct). Shaded areas are triangulated; we omit showing these edges for clarity.

Any drawing of area $\leq A = W^2$ has at most $(W-1)^2$ grid points that are not on the four extreme grid lines. G will have $(W-1)^2+4$ vertices, so in any drawing with outer-face f it has $(W-1)^2+4-\deg(f)$ vertices that are not on the outer-face, hence cannot be placed on extreme grid lines. It follows that if there exists a drawing of G of area $\leq A$, then it must be on a $W \times W$-grid, and the outer-face of G must be the (unique) face that has degree 4. Furthermore, in such a drawing not a single grid point not on an extreme grid-line may be wasted. Using this, one can argue that the outer frame is drawn exactly as in Fig. 4, up to reflection and rotation. In particular, we have $n+1$ “teeth” that stick into the middle region, each column of a tooth has 2ℓ grid points left (i.e., not used by the outer frame), and each column between two teeth has $2\ell + B$ grid points left. The two blobs are too big to fit into the columns at the teeth, and too big to both fit left of the teeth, so one must be left of the teeth and the other right of the teeth.

Between the outer frame and the blobs we add ℓ stacked cycles called *layers*; the two blobs are inside all these cycles. Each layer must start in the far left (to surround the left blob), go past all teeth to the right (to surround the right blob), and then go back. The length of each layer is set so that it exactly fills the grid points encountered along this path. The ℓ layers hence use up all grid points except for B grid points in each column between two teeth.

There are $\ell \geq 12n^2 + 3n = (3n) \cdot (4n + 1)$ layers. For $i = 1, \ldots, 3n$, insert a path of length a_i in the face between layer $i(4n)$ and layer $i(4n)+1$. These paths can only be placed in the columns between teeth, so a drawing of area A gives an assignments of the a_i's into groups that sum to B each, as desired.

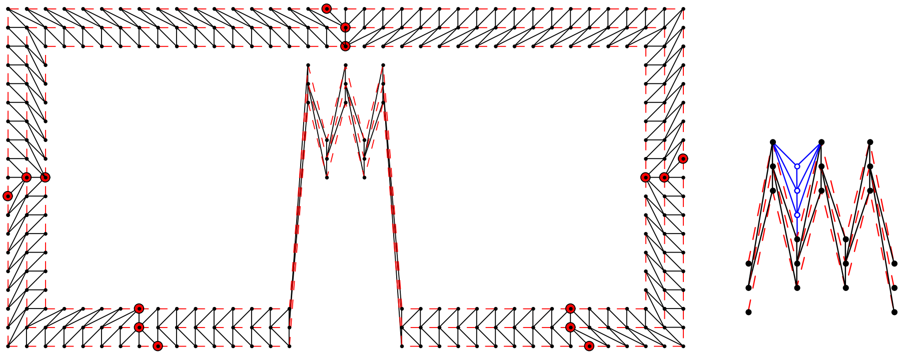


Fig. 5. Triangulating between layers (red, dashed), and how to attach the a_i-path

To make the graph inner triangulated, we connect two consecutive layers with a zig-zag line, except for "collector-points" (thick and red in Fig. 5) that have three or four neighbours on the previous layer, including the previous collector-points. There are two ways to draw these connections; in one way the collector-points are aligned vertically/horizontally, while in the other they are shifted clockwise by one unit. We use this for $2n$ pairs of layers, and then for the next $2n$ pairs of layers use a symmetric construction that allows collector-points to be aligned or to be shifted counter-clockwise by one unit. Finally, when adding the a_i-path, we attach it at the vertex diametrically opposite to the top collector-point, and connect all vertices on the path to the two vertices before/after that attachment-point. Over the course of the $4n$ layers between paths, we can shift collector-points by up to $\pm 2n$ units clockwise or counter-clockwise. With this, we can bring the attachment-point of the a_i-path to any of the n columns between teeth, regardless of where the a_{i-1}-path was. Hence for any solution of 3-partition we can draw G in a $W \times W$-grid, and since faces are triangles or squares, the drawing is convex. We conclude:

Theorem 5. AreaMinimization *and* ConvexAreaMinimization *are NP-hard, even for an internally triangulated 3-connected planar graph for which the outer-face is a 4-cycle.*

By omitting the triangulation edges in shaded areas of Fig. 4, and (roughly speaking) replacing the triangulation between any second pair of layers by three edges only, we can create a variant of this graph that has pathwidth at most 7 and for which any drawing of area A implies a solution to 3-partition.

Theorem 6. AreaMinimization *is NP-hard, even for a 3-connected planar graph of bounded pathwidth.*

5 Conclusion

This paper revisited the problem of drawing planar graphs with optimal area. We showed that finding a convex planar drawing with optimal area is possible in polynomial time for all graphs with bounded treewidth, even if the planar embedding is not fixed. Based on results for point-set embeddability, we also showed that finding an area-optimal planar drawing is polynomial-time solvable if the graph has bounded treewidth, bounded face-degrees, and the planar embedding is fixed.

As for open problems, can one approximate the optimal area, or is the optimization version of AreaMinimization APX-hard? For many other problems, finding a poly-time algorithm for bounded-treewidth graphs was a first step towards approximation algorithms. To do so, it would be helpful to find algorithms that are fixed-parameter tractable in the treewidth, but ours is not. Is Convex-AreaMinimization $W[1]$-hard with respect to the parameter treewidth?

References

1. Biedl, T., Vatshelle, M.: The point-set embeddability problem for plane graphs. International Journal of Computational Geometry and Applications (in press, 2014)
2. Brandenburg, F.: Drawing planar graphs on (8/9)*n^2 area. Electronic Notes in Discrete Mathematics 31, 37–40 (2008)
3. Dorn, F.: Dynamic programming and planarity: Improved tree-decomposition based algorithms. Discrete Applied Mathematics 158(7), 800–808 (2010)
4. Fáry, I.: On straight line representation of planar graphs. Acta Scientiarum Mathematicarum (Szeged) 11(4), 229–233 (1948)
5. Felsner, S., Liotta, G., Wismath, S.: Straight-line drawings on restricted integer grids in two and three dimensions. Journal of Graph Algorithms and Applications 7(4), 335–362 (2003)
6. de Fraysseix, H., Pach, J., Pollack, R.: How to draw a planar graph on a grid. Combinatorica 10, 41–51 (1990)

7. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman (1979)
8. Krug, M., Wagner, D.: Minimizing the area for planar straight-line grid drawings. In: Hong, S.-H., Nishizeki, T., Quan, W. (eds.) GD 2007. LNCS, vol. 4875, pp. 207–212. Springer, Heidelberg (2008)
9. Mondal, D., Nishat, R.I.: Md. S. Rahman, and Md. J. Alam. Minimum-area drawings of plane 3-trees. J. Graph Algorithms Appl. 15(2), 177–204 (2011)
10. Robertson, N., Seymour, P.D.: Graph minors. X. Obstructions in tree-decompositions. J. Combin. Theory Ser. B 52, 153–190 (1991)
11. Schnyder, W.: Embedding planar graphs on the grid. In: ACM-SIAM Symposium on Discrete Algorithms (SODA 1990), pp. 138–148 (1990)
12. Stein, S.: Convex maps, vol. 2, pp. 464–466. American Mathematical Society (1951)
13. Tutte, W.T.: Convex representations of graphs. Proc. London Math. Soc. 10(3), 304–320 (1960)
14. Wagner, K.: Bemerkungen zum Vierfarbenproblem. Jahresbericht der Deutschen Mathematiker-Vereinigung 46, 26–32 (1936)

Shortest Two Disjoint Paths in Polynomial Time

Andreas Björklund[1] and Thore Husfeldt[1,2]

[1] Lund University, Sweden
[2] IT Univeristy of Copenhagen, Denmark

Abstract. Given an undirected graph and two pairs of vertices (s_i, t_i) for $i \in \{1, 2\}$ we show that there is a polynomial time Monte Carlo algorithm that finds disjoint paths of smallest total length joining s_i and t_i for $i \in \{1, 2\}$ respectively, or concludes that there most likely are no such paths at all. Our algorithm applies to both the vertex- and edge-disjoint versions of the problem.

Our algorithm is algebraic and uses permanents over the quotient ring $\mathbf{Z}_4[X]/(X^m)$ in combination with Mulmuley, Vazirani and Vazirani's isolation lemma to detect a solution. We develop a fast algorithm for permanents over said ring by modifying Valiant's 1979 algorithm for the permanent over $\mathbf{Z}_{2^l}$.

1 Introduction

How fast can we find the shortest two vertex-disjoint paths joining two given pairs of vertices in a graph? Figure 1 shows an example instance with an optimal solution of total length $6 + 6 = 12$, where length is the number of edges on the two paths. Note that neither path is a shortest path.

This problem is sometimes called *min-sum* two disjoint paths. We solve it in polynomial time. One would expect an algorithmic graph problem of this type to have been understood a generation ago, but to our best knowledge, no subexponential time algorithm was known [5,7].

By a simple reduction, our algorithm also applies to the edge-disjoint version of the problem, where the two paths may not share an edge (but may share vertices). The complexity of this problem was also open for many decades [6].

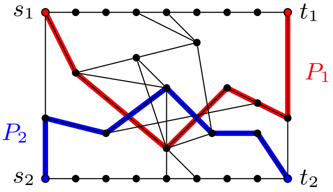


Fig. 1. Two shortest disjoint paths joining the red and blue vertex pairs

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 211–222, 2014.

Our algorithm is algebraic and follows recent activity in algebraic algorithms for some fundamental graph problems developed in the frameworks of fixed-parameter and exponential time algorithms [1,2,8,18,19]. One ingredient is to express a graph problem in terms of a matrix of indeterminates, an idea that goes back to Tutte [16] and Edmonds [4]. The matrix function we compute is a permanent, and we use ideas of Valiant [17] to compute this in a ring of characteristic 4. To work in full generality, our algorithm is randomized, using Mulmuley, Vazirani, and Vazirani's isolation lemma [10].

Related problems. For planar graphs, polynomial-time algorithms for shortest two disjoint paths have been found by Colin de Verdière and Schrijver [3] and Kobayachi and Sommer [7], under certain conditions on the placement of terminals. Our algorithm works for all planar cases, but is much slower.

The problem's *decision* version—decide if two disjoint paths joining given vertex pairs exist, no matter their length—was shown to be polynomial-time computable by Ohtsuki [11], Seymour [12], Shiloah [13], and Thomassen [14]; all published independently in 1980. More recent papers reduce the running time for that problem to near-linear, see Tholey [15] and the references therein. However, no algorithms for finding the *shortest* two disjoint paths seem to follow from these constructions.

It is worth mentioning a different problem that might as well have the same name: Find two disjoint paths of minimal total length from the start nodes $\{s_1, s_2\}$ to the end nodes $\{t_1, t_2\}$. That problem also allows s_1 to be joined with t_2 and s_2 with t_1. It has a solution of total length 10 in the example. That problem is algorithmically much simpler, well understood, and can be solved by standard flow techniques in linear time.

Another nominally related problem, *two disjoint shortest paths*, is to decide if the given endpoints can be joined using two disjoint paths, each of which is a shortest path. (In Figure 1, the answer is "no.") That problem can be solved in polynomial time, as shown by Eilam–Tzoreff [5].

Li, McCormick, and D. Simchi-Levi [9] have shown that the problem of minimizing the maximum of both path lengths (sometimes called *min-max* two disjoint paths) is NP-hard, both in the vertex- and edge-disjoint versions. Also, the problem of finding two disjoint paths, one of which is a shortest path, is NP-hard [5].

1.1 Result

We are given an undirected, loopless, unweighted, and simple input graph G with n vertices and m edges, as well as two pairs $\{s_1, t_1\}$ and $\{s_2, t_2\}$ of *terminal* vertices. We will show how to find two vertex-disjoint paths P_1 and P_2, such that P_i joins s_i and t_i for $i \in \{1, 2\}$ and $|P_1| + |P_2|$ is minimal. We assume that neither edge s_1t_1 or s_2t_2 exists in G; otherwise the problem is solved by finding a shortest path between the other terminal pair using breadth-first search. We consider first the case where there is a unique optimal solution, in which case our algorithm is deterministic and slightly simpler to explain.

Unique solution. We define three matrices over 0, 1, and a formal indeterminate x. First, construct the $n \times n$ matrix A by identifying the vertex set of G with $\{1, \ldots, n\}$ and setting

$$a_{uv} = \begin{cases} x\,, & \text{if } uv \in E\,; \\ 1\,, & \text{if } u = v\,; \\ 0\,, & \text{otherwise}\,. \end{cases} \tag{1}$$

Note that because G is undirected, the matrix is symmetric. It can be viewed as a univariate Tutte matrix.

Then, for given vertices v, w, v', and w' specified below, construct the matrix $A[vw, v'w']$ from a A by putting 0s in the rows corresponding to v and v', then putting 1s in the two entries corresponding to vw and $v'w'$:

$$A[vw, v'w'] = \begin{array}{rl} & \quad\quad\quad w\ \ w' \\ \begin{matrix} v' \\ \\ \\ v \end{matrix} & \begin{bmatrix} 0\text{——}010\text{-}0 \\ \\ \\ 0\text{-}010\text{——}0 \end{bmatrix} \end{array}. \tag{2}$$

We then define the univariate polynomial

$$f = \operatorname{per} A[t_1 s_1, t_2 s_2] + \operatorname{per} A[t_1 s_1, s_2 t_2] - \operatorname{per} A[s_1 s_2, t_1 t_2]\,.$$

Here, the (symbolic) *permanent* of an $n \times n$ matrix $A = (a_{ij})$ is defined as

$$\operatorname{per} A = \sum_{\sigma} \prod_{i} a_{i\sigma(i)}\,, \tag{3}$$

where the summation ranges over all permutations σ of $\{1, \ldots, n\}$. Note that the matrix entries a_{ij}, the permanent $\operatorname{per} A$, and f itself are polynomials in x.

Theorem 1. *The graph G contains a unique pair of disjoint paths of shortest total length k if and only if the lowest-order term in f is $2x^k$.*

This does not seem algorithmically useful, because the definition of f involves several permanents, which usually are hard to compute. Our main algorithmic insight is the following: Because the coefficient we are looking for is 2, it suffices to perform the computation of f in a ring of characteristic larger than 2. We pick the polynomial ring $\mathbf{Z}_4[x]/(x^M)$ of characteristic 4, where $M = \lceil 2n^4 \rceil$. We show in Section 3 that the permanent in this ring can be computed in polynomial time.

We can summarize our algorithm as follows:

Algorithm U. (*Unique shortest two disjoint paths*) *Finds the minimum length of a unique pair of disjoint paths in G joining s_1 and t_1 and joining s_2 and t_2, respectively.*

U1. [Setup.] Construct A as given by (1).

U2. [Compute f.] Construct $A[t_1 s_1, t_2 s_2]$, $A[t_1 s_1, s_2 t_2]$, and $A[s_1 s_2, t_1 t_2]$. Compute f over $\mathbf{Z}_4[x]/(x^M)$ using Algorithm P for the three permanents.

U3. Return the minimum j such that x^j has nonzero coefficient in f.

Non-unique solutions. Theorem 1 does not help us in the general case: If G contains an even number of optimal solutions of total length k, then each contributes $2x^k$ to f, so the coefficient vanishes modulo 4. However, we can use the standard remedy for this type of situation by equipping the edges with random weights and looking for a unique weighted solution.

To be concrete, let

$$W = \{2nm, \ldots, 2nm + 2m - 1\}\,. \tag{4}$$

For each $e \in E$, choose a random integer $w(e) \in W$. Now define A by

$$a_{uv} = \begin{cases} x^{w(uv)}\,, & \text{if } uv \in E\,; \\ 1\,, & \text{if } u = v\,; \\ 0\,, & \text{otherwise}\,. \end{cases} \tag{5}$$

(Note that A is still symmetric.) From here, the construction of the three matrices $A[t_1 s_1, t_2 s_2]$, $A[t_1 s_1, s_2 t_2]$, $A[s_1 s_2, t_1 t_2]$, and the polynomial f is the same as the unique case of Section 1.1.

Theorem 2. *If G contains a unique pair of disjoint paths of shortest total length k, then with probability $\frac{1}{2}$, the lowest-order term in f is $2x^j$ for some j with $k \min W \le j \le k \max W$.*

The algorithm differs from Algorithm U only in the first and last steps:

Algorithm S. (*Shortest two disjoint paths*) *With probability $\frac{1}{2}$, finds the minimum length of two disjoint paths in G joining s_1 and t_1 and joining s_2 and t_2, respectively.*

S1. [Setup] For each $e \in E$, choose integer weight $w(e) \in W$ uniformly at random. Construct A as given by (5).

S2. [Compute f.] (Identical to Step U2.)

S3. Find the minimum j such that x^j has nonzero coefficient in f. Return $\lfloor j/\min W \rfloor$.

As presented here, Algorithm S runs in time $O(n^{11})$. Our main concern is to communicate the existence of a polynomial-time algorithm, so we do not discuss how to reduce the exponent. A witness for an optimal solution can be found using self-reduction, increasing the running time by another linear factor. The error probability can be reduced to 2^{-r} by repeating the algorithm r times. For the edge-disjoint version of the problem, add an edge to each terminal vertex and apply Algorithm S to the line graph of the resulting graph.

1.2 Conclusion

We still have no deterministic polynomial-time algorithm for the two disjoint shortest paths problem, nor do we know how to significantly improve the running time of our algorithm, say to subseptic. For higher $k > 2$, the complexity of the k disjoint shortest paths problem remains unresolved.

Acknowledgements. This research originated in stimulating discussions with Nina Sofia Taslaman. The authors are partially supported by the Swedish Research Council project VR 2012-4730, Exact Exponential-time Algorithms.

2 Proof of Theorems 1 and 2

We will give a combinatorial understanding of f. We subsume the construction of A in the case of unique solutions under the general case by setting $w(e) = 1$ for each edge e.

Construct a weighted directed graph D from G as follows. For each nonterminal vertex v insert the self-loop vv with weight 1. For each edge vw insert the arcs vw and wv, each with weight $w(vw) = w(wv)$. Given two arcs vw and $v'w'$ we define the directed graph $D[vw, v'w']$ as follows: Remove all arcs outgoing from v and v'. Then add the arcs vw and $v'w'$ of weight 1. We call these arcs *forced*; they are the only way to leave v and v'.

Now we can can interpret the matrix $A[vw, v'w']$ from (2) as a weighted adjacency matrix of $D[vw, v'w']$. Each permutation contributing to the permanent corresponds to a directed cycle cover. Recall that a (directed) *cycle cover* is a collection of vertex-disjoint directed cycles that visit every vertex in the graph.

Lemma 1

$$f = \sum_{P_1,P_2} 2x^{w(P_1)+w(P_2)} \operatorname{per} A_{P_1P_2} \, .$$

where the sum is over all undirected disjoint paths P_i joining s_i and t_i for $i \in \{1,2\}$. Here, $A_{P_1P_2}$ denotes the matrix A with all rows and columns corresponding to vertices on P_1 and P_2 removed.

Proof. We consider the contribution of each directed cycle cover to f. Consider a permutation σ. It can be checked that there are 6 cases for how a permutation can pass the 4 terminal vertices in any of the three forced graphs, shown in Fig. 2. (There are many other ways of permuting $\{s_1, t_1, s_2, t_2\}$, but none of the forced graphs contains the necessary arcs.)

Consider first the permutation of Type 1 in the first row, going from s_1 to t_1 and from s_2 to t_2. This corresponds to two disjoint paths in the original graph, so these permutations are what we *want* to count, and indeed they contribute positively to $\operatorname{per} A[t_1s_1, t_2s_2]$.

However, that term also picks up a positive contribution from permutations of Type 2, which we do not want to count. We remedy this problem with the other terms. Each Type 2 permutation also contributes negatively to the third row. To be precise, we can associate each Type 2 permutation contributing to $\operatorname{per} A[t_1s_1, t_2s_2]$ to another permutation contributing negatively to $-\operatorname{per} A[s_1s_2, t_1t_2]$ by changing the forced edges and reverting the path from s_1 to t_2. Since forced edges have weight 1 and all other arcs have the same weight as their reversal, the two permutations contribute the same term with different signs and therefore cancel.

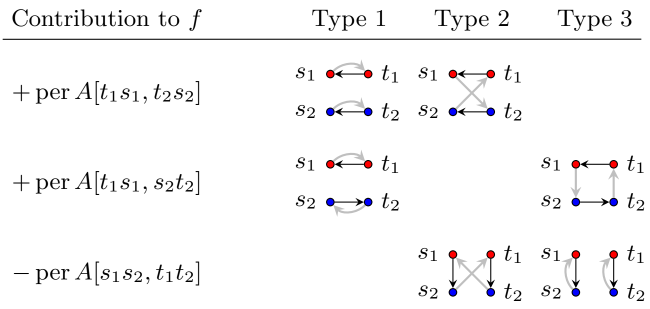

Contribution to f
Type 1
Type 2
Type 3
$+\operatorname{per} A[t_1s_1, t_2s_2]$
$+\operatorname{per} A[t_1s_1, s_2t_2]$
$-\operatorname{per} A[s_1s_2, t_1t_2]$
s_1
t_1
s_2
t_2

Fig. 2. Cycles contributing to f. Gray: directed paths, solid: forced arcs.

The Type 3 permutations cancel in a similar fashion.

The contribution of a permutation of Type 1 consists of the contributions of the edges along the paths P_i joining s_i and t_i, where $i \in \{1, 2\}$. These edges contribute the factor $x^{w(P_1)+w(P_2)}$. The remaining edges avoid the terminal vertices, so their total contribution can be given in terms of the permanent of an induced subgraph of D. Then the total contribution of all permutations in the first and second case is $2x^{w(P_1)+w(P_2)} \operatorname{per} A_{P_1P_2}$. □

Proof (of Theorem 1). Consider $\operatorname{per} A_{P_1P_2}$. The contribution of the identity permutation is exactly 1 because all self-loops are labelled 1. Any other permutation contributes at least the factor x^2. Thus, the term with the smallest exponent is $2x^{w(P_1)+w(P_2)} = 2x^{|P_1|+|P_2|}$, for the shortest two disjoint paths P_1 and P_2. □

For the second theorem, we need the isolation lemma [10]:

Lemma 2. *Let m be a positive integer, W a set of consecutive positive integers, and let $\mathscr{F}$ be a nonempty family of subsets of $\{1, \ldots, m\}$. Suppose each element $x \in \{1, \ldots, m\}$ receives weight $w(x) \in W$ independently and uniformly at random. Define the weight of a set S in $\mathscr{F}$ as $w(S) = \sum_{x \in S} w(x)$. Then, with probability at least $1 - m/|W|$, there is a unique set in $\mathscr{F}$ of minimum weight.*

(The lemma is normally stated for weights of the form $W = \{1, \ldots, |W|\}$, but as observed in [10], the above generalization holds as well.)

Proof (of Theorem 2). Let W be as in (4), and k be the total length of two shortest disjoint paths. Let $\mathscr{F}$ denote the family of edge subsets belonging to P_1 or P_2, for each pair (P_1, P_2) of two shortest disjoint paths.

By Lemma 2, with probability at least $1 - m/2m = \frac{1}{2}$, there is a unique set of edges in $\mathscr{F}$ of minimal weight, corresponding to a pair (P_1, P_2) of paths. Their total weight is at least $k \min W$ and at most $k \max W = k(2mn + 2m - 1) < (k+1)2mn = (k+1) \min W$. In particular, all nonoptimal solutions have even larger weight. As in the proof of Theorem 1, the smallest exponent in f is of the form $2x^{w(P_1)+w(P_2)}$. □

3 Computing the Permanent

We begin with some elementary properties of the permanent in rings. For a ring R we let $\mathrm{M}_n(R)$ denote the set of $n \times n$ matrices over R. Sometimes we write $\mathrm{M}(R) = \bigcup_n \mathrm{M}_n(R)$. For $A \in \mathrm{M}_n(R)$ let A_{ij} denote the matrix in $\mathrm{M}_{n-1}(R)$ that results from deleting the ith row and jth column of A.

Elementary row operations. If A' is constructed from A by exchanging two rows, then $\operatorname{per} A = \operatorname{per} A'$. If A' is constructed from A by multiplying all entries in a single row by $c \in R$, then $\operatorname{per} A' = c \operatorname{per} A$. The third elementary row operation, however, is more complicated:

Lemma 3. *Consider a matrix $A \in M(R)$, ring element $c \in R$ and integers i and j. Let A' be the matrix constructed by adding the cth multiple of row j to row i. Let D be the matrix constructed by replacing row i with row j. Then*

$$\operatorname{per} A' = \operatorname{per} A + c \operatorname{per} D\,.$$

Proof. From the definition,

$$\begin{aligned}\operatorname{per} A' &= \sum_\sigma \prod_k a'_{k\sigma(k)} = \sum_\sigma a'_{i\sigma(i)} \prod_{k\neq i} a_{k\sigma(k)} = \sum_\sigma (a_{i\sigma(i)} + c a_{j\sigma(j)}) \prod_{k\neq i} a_{k\sigma(k)} \\ &= \sum_\sigma a_{i\sigma(i)} \prod_{k\neq i} a_{k\sigma(k)} + \sum_\sigma c a_{j\sigma(j)} \prod_{k\neq i} a_{k\sigma(k)} = \operatorname{per} A + c \operatorname{per} D\,. \qquad \square\end{aligned}$$

In other words, we can get from A to A' at the cost of computing $\operatorname{per} D$. (The good news is that because D has duplicate rows, $\operatorname{per} D$ turns out to be algorithmically inexpensive 'dross' in our algebraic structure.)

Quotient rings. Computation takes place in the two rings $E_l = \mathbf{Z}_l[x]/(x^M)$ for $l \in \{2, 4\}$, where $h = x^M$ and $M = \lceil 2n^4 \rceil > n \max W$ is chosen larger than the degree of the polynomial f. Every element in E_l can be uniquely represented as $[g]_h$, where $g \in \mathbf{Z}_l[x]$ is a polynomial of degree at most $M - 1$ with coefficients in $\mathbf{Z}_l$. We have, *e.g.*, $[2x + 3x^2]_h + [x + x^2]_h = [3x]_h$ and $[x + x^{M-1}]_h \cdot [x]_h = [x^2]_h$ in E_4. Note that the ring elements are formal polynomials, not functions; two polynomials are equal if and only if all their coefficients agree. For instance, $[x]_h$ and $[x^2]_h$ are not equal in E_2. For a ring element $a \in E_l$ and integer $j \in \{0, \ldots, M - 1\}$, we let $[x^j]a$ denote the jth coefficient of the representation of a. Formally, if $a = [g]_h$ and $g = \sum_{j=0}^{M-1} c_j x^j$ then $[x^j]a = c_j \in \mathbf{Z}_l$. We will ignore the distinction between a ring element $[g]_h$ and the polynomial g and regard the elements of E_l as polynomials 'with higher powers chopped off.'

In E_4 we introduce a 'permanent with all coefficients replaced by their parity.' To be precise, define the *parity permanent* $\operatorname{per}_2\colon \mathrm{M}(E_4) \to E_4$ for $A \in \mathrm{M}(E_4)$ for each coefficient by

$$[x^j] \operatorname{per}_2 A = ([x^j] \operatorname{per} A) \bmod 2\,.$$

We will see in Section 3.4 that the parity permanent E_4 is analogous to the (standard) permanent in E_2, which we can compute in polynomial time.

Even and odd polynomials. A polynomial a is *even* if $[x^j]a = 0 \pmod 2$ for each $j \in \{0, \ldots, M-1\}$. Otherwise it is *odd*. In E_2, the zero polynomial is the only even polynomial. Note that the sum of two even polynomials is even, but the sum of two odd polynomials need not be even. A product ab is even if one of its factors is. Let $m(a) = \min\{\, j \colon [x^j]a = 1 \pmod 2 \,\}$ denote the index of its lowest-order odd coefficient of a.

In a product in E_4 with one even factor, the parity permanent can replace the permanent:

Lemma 4. *For $A \in M(E_4)$ and even $a \in E_4$, we have $a \operatorname{per} A = a \operatorname{per}_2 A$.*

Proof. We show that the polynomials on both sides have the same coefficients in $\mathbf{Z}_4$. By the definition of polynomial multiplication,

$$\begin{aligned} [x^j]a \operatorname{per} A &= \sum_{k=0}^{j} [x^k]a \cdot [x^{j-k}] \operatorname{per} A \\ &= \sum_{k=0}^{j} [x^k]a \cdot [x^{j-k}] \operatorname{per}_2 A \pmod 4 = [x^j]a \operatorname{per}_2 A \,, \end{aligned}$$

where the second equality uses $2x = 2(x \bmod 2) \pmod 4$. □

Laplace expansion. We consider the Laplace expansion of the permanent over E_4 in the special case where the first column has only a single odd element.

Lemma 5. *Let $A \in \mathrm{M}_n(E_4)$. Assume a_{i1} is even for all $i > 1$. Then,*

$$\operatorname{per} A = a_{11} \operatorname{per} A_{11} + \sum_{i=2}^{n} a_{i1} \operatorname{per}_2 A_{i1} \qquad (\textit{in } E_4)\,.$$

Proof. In any commutative ring, the permanent satisfies the Laplace expansion,

$$\operatorname{per} A = \sum_{i=1}^{n} a_{i1} \operatorname{per} A_{1j} \,.$$

By Lemma 4, we can use the parity permanent in all terms except for $i = 1$. □

3.1 Overview of Algorithm

Algorithm P. (*Permanent over E_4*) *Given $A \in \mathrm{M}_n(E_4)$ computes* $\operatorname{per} A$ *in polynomial time in n and M.*

It transforms A successively such that $a_{21}, \ldots, a_{n1}$ are all even. This leads to a single recursive call with argument A_{11}, following Lemma 5. For each transformation, a matrix D with duplicate rows is produced, according to Lemma 3. Their contributions are collected in a list L, and subtracted at the end.

P1. [Base case.] If $n = 1$ return a_{11}.

P2. [Initialize.] Let L be the empty list.

P3. [Easy case.] If a_{i1} is even for every $i \in \{1, \dots, n\}$, go to P8.

P4. [Find pivot.] Choose $i \in \{1, \dots, n\}$ such that a_{i1} is odd and $m(a_{i1})$ is minimal. Ties are broken arbitrarily. Exchange rows 1 and i. [Now a_{11} is odd and $m(a_{11})$ minimal.] Set $i = 2$.

P5. [Column done?] If $i = n + 1$ go to P8.

P6. [Make a_{i1} even.] Use Algorithm E to find $c \in E_4$ such that $a_{i1} + ca_{11}$ is even. Let A' and D be as in Lemma 3. Compute $c \operatorname{per} D$ using Lemma 7 and add it to L. Set $A = A'$.

P7. [Next entry in column.] Increment i and return to P5.

P8. [Compute subpermanent.] Compute $p = \operatorname{per} A_{11}$ recursively.

P9. [Return.] Return $a_{11}p + \sum_{i>1} a_{i1} \operatorname{per}_2(A_{i1}) - \sum_{d \in L} d$, using Algorithm Y for the parity permanents.

3.2 Making Odd Polynomials Even

In Step P6 we need to turn an odd polynomial into an even one, which can be done by a simple iterative process. Consider for instance the odd polynomials $a = a_{ij} = x^3 + 2x^5 + 3x^6$ and $b = a_{jj} = x + x^3$ in E_4. If we choose $c = x^2$ then $bc = (x + x^3)x^2 = x^3 + x^5$, so $a + bc = 2x^3 + 3x^5 + 3x^6$. At least $[x^3](a + bc)$ is now even, even though we introduced a new, higher-order, odd coefficient. We add the corresponding higher-order term to c, arriving at $c = x^2 + x^4$. Now we have $a + bc = 2x^3 + 3x^6 + x^7$. Repeating this process, the coefficient of x^j for each $j \in \{0, \dots, M - 1\}$ is eventually made even.

Algorithm E. (*Even polynomial*) *Given $l \in \{2, 4\}$ and odd polynomials $a, b \in E_l$ with $m(a) \geq m(b)$, finds a polynomial $c \in E_l$ such that $a + bc$ is even.*

E1. [Initialize] Set $r = 1$. Set $c_r = 0$, the zero polynomial in E_l.

E2. [Done?] If $a + bc_r$ is even, output c_r and terminate.

E3. Set $c_{r+1} = c_r + x^{m(a+bc_r)-m(b)}$. Increment r. Return to Step E2.

Lemma 6. *Algorithm E runs in polynomial time in M.*

Proof. To see that Step E3 makes progress, set $c = c_r$, $c' = c_{r+1}$ and let $j = m(a+bc)$ denote the index of the lowest-order odd coefficient in $a+bc$. Consider the next polynomial,

$$a + bc' = a + b(c + x^{j-m(b)}) = a + bc + bx^{j-m(b)} .$$

By the definition of polynomial multiplication, its jth coefficient

$$[x^j](bx^{j-m(b)}) = [x^{m(b)}]b \cdot [x^{j-m(b)}]x^{j-m(b)} = [x^{m(b)}]b \cdot 1$$

is odd. Since $[x^j](a + bc)$ is also odd, $[x^j](a + bc')$ is even.

Moreover, for $j' < j$, we can likewise compute

$$[x^{j'}](bx^{j-m(b)}) = [x^{j'-j+m(b)}]b\,,$$

which is even by minimality of $m(b)$. Thus, $bx^{j-m(b)}$ introduces no new odd terms to $a + bc'$ of index smaller than j.

In particular, $m(a+bc_1) < m(a+bc_2) < \cdots$, so Algorithm E terminates after at most M iterations. □

3.3 Duplicate Rows

The elementary row operation in Step P6 produces dross in the form of a matrix with duplicate rows.

Lemma 7. *Let $A \in \mathrm{M}_n(E_4)$ have its first two rows equal. Then*

$$\operatorname{per} A = 2 \sum_{1 \le j < k \le n} a_{1j} a_{2k} \operatorname{per}_2 A_{\{1,j\},\{2,k\}}\,.$$

Proof. Given a permutation σ with $j = \sigma(1)$ and $k = \sigma(2)$, construct the permutation σ' by exchanging these two points: set $\sigma'(1) = k$, $\sigma'(2) = j$, and $\sigma'(i) = \sigma(i)$ for $i \in \{3, \ldots, n\}$. We have $a_{1\sigma(1)} = a_{1j} = a_{2j} = a_{2\sigma'(2)}$ and, similarly, $a_{2\sigma(2)} = a_{1\sigma'(1)}$. Thus,

$$\begin{aligned}\prod_i a_{i\sigma(i)} + \prod_i a_{i\sigma'(i)} &= \big(a_{1\sigma(1)}a_{2\sigma(2)} + a_{1\sigma'(1)}a_{2\sigma'(2)}\big)\prod_{i>2} a_{i\sigma(i)}\\ &= 2a_{1\sigma(1)}a_{2\sigma(2)} \prod_{i>2} a_{i\sigma(i)} = 2\prod_i a_{i\sigma(i)}\,.\end{aligned}$$

In other words,

$$\operatorname{per} A = \sum_{\sigma\colon\, \sigma(1)<\sigma(2)} 2a_{1\sigma(1)}a_{2\sigma(2)} \prod_{i>2} a_{i\sigma(i)} = \sum_{1\le j<k\le n} a_{1j}a_{2k} 2 \operatorname{per} A_{\{1,j\},\{2,k\}}\,,$$

where $A_{P,Q}$ is A without the rows in P and columns in Q. Finally, we replace the permanent with the parity permanent using Lemma 4 with $a = 2$. □

3.4 Computing the Parity Permanent

First we observe that the permanent in E_2 can be computed in polynomial time. We are tempted to argue that since $\mathbf{Z}_2$ is a field, the ring $\mathbf{Z}_2[x]$ is a Euclidean domain, where Gaussian elimination computes the determinant in polynomially many ring operations. Moreover, since the characteristic is 2, the determinant and the permanent are identical. The problem with this argument is that we have little control over the size of the polynomials produced during this process. Instead, we choose to give an explicit algorithm for the permanent in E_2 by simplifying Algorithm P.

First, a matrix with duplicate rows in any ring of characteristic 2 has zero permanent. Thus, we need no analogues of Lemma 3, Algorithm D, or list L. We can remove Step P2 and replace Step P6 by

P′6. [Make $a_{i1} = 0$.] Use algorithm E to find $c \in E_2$ such that $a_{i1} + ca_{11} = 0$. Add the cth multiple of row 1 to row i in A.

Furthermore, since the even polynomials in E_2 are all 0, the only term surviving in the Laplace expansion is $a_{11} \operatorname{per} A$. Thus, Step P9 becomes simply:

P′9. [Return.] Return $a_{11}p$.

It remains to connect the parity permanent in E_4 to the permanent in E_2. Define the map $\phi\colon E_4 \to E_2$ replacing each coefficient by its parity,

$$[x^j]\phi(a) = [x^j]a \bmod 2\,.$$

Lemma 8. *The map ϕ is a ring homomorphism.*

Proof. The unit in both E_4 and E_2 is the constant polynomial 1, and indeed $\phi(1) = 1$. To see that $\phi(a) + \phi(b) = \phi(a + b)$, we consider the jth coefficient on both sides: $[x^j](\phi(a)+\phi(b)) = [x^j]\phi(a)+[x^j]\phi(b) = [x^j]a \bmod 2+[x^j]b \bmod 2 = ([x^j]a + [x^j]b) \bmod 2 = ([x^j](a + b)) \bmod 2 = [x^j](\phi(a) + \phi(b))$. Similarly, to see that $\phi(a)\phi(b) = \phi(ab)$, we expand

$$\begin{aligned}[x^j](\phi(a)\phi(b)) &= \sum_{k=0}^{j}[x^k]\phi(a)[x^{j-k}]\phi(b) = \sum_{k=0}^{j}([x^k]a \bmod 2)([x^{j-k}]b \bmod 2)\\ &= \sum_{k=0}^{j}([x^k]a)([x^{j-k}]b) \bmod 2 = [x^j]ab \bmod 2 = [x^j]\phi(ab)\,. \qquad \square\end{aligned}$$

We extend ϕ to matrices by defining the map $\Phi\colon \mathrm{M}_n(E_4) \to \mathrm{M}_n(E_2)$, where the ijth entry of the matrix $\Phi(A)$ is $\phi(a_{ij})$. Then the following holds in E_2:

Lemma 9. $\phi(\operatorname{per}_2 A) = \operatorname{per} \Phi(A)$.

Proof. From the definition (3) and Lemma 8,

$$\operatorname{per}\Phi(A) = \sum_\sigma \prod_i \phi(a_{i\sigma(i)}) = \phi\Bigl(\sum_\sigma \prod_i a_{i\sigma(i)}\Bigr) = \phi(\operatorname{per} A)\,.$$

Thus, for each $j \in \{0, \dots , M - 1\}$, we have

$$[x^j] \operatorname{per}\Phi(A) = [x^j]\phi(\operatorname{per} A) = ([x^j] \operatorname{per} A) \bmod 2 = [x^j] \operatorname{per}_2 A\,,$$

so the two polynomials have the same coefficients in $\mathbf{Z}_2$. □

Thus we have the following algorithm:

Algorithm Y. (*Parity permanent*) *Given* $A \in \mathrm{M}_n(E_4)$, *compute* $\operatorname{per}_2 A$ *in time polynomial in n and M.*

Y1. [Let $P = \Phi(A)$] Construct $P \in \mathrm{M}_n(E_2)$ such that $[x^j]p_{ij} = [x^j]a_{ij} \bmod 2$.

Y2. [Permanent] Compute $p = \operatorname{per} P$ in E_2 using algorithm P′.

Y3. [Return result] Return the polynomial in E_4 whose jth coefficient is $[x^j]p$.

Thus, in order to compute the expression in Lemma 7 for Step P6, Algorithm Y is called $O(n^2)$ times, once for every $\operatorname{per}_2 A_{\{1,j\},\{2,k\}}$. It is also called in total $O(n^2)$ times in Step P9.

References

1. Björklund, A.: Determinant sums for undirected Hamiltonicity. SIAM J. Comput. 43(1), 280–299 (2014)
2. Björklund, A., Husfeldt, T., Taslaman, N.: Shortest cycle through specified elements. In: Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, pp. 1747–1753. SIAM 2012 (2012)
3. Colin de Verdière, E., Schrijver, A.: Shortest vertex-disjoint two-face paths in planar graphs. ACM T. Algorithms 7(2), 19 (2011)
4. Edmonds, J.: Systems of distinct representatives and linear algebra. J. Res. Nat. Bur. Stand. 71B(4), 241–245 (1967)
5. Eilam–Tzoreff, T.: The disjoint shortest paths problem. Discrete Appl. Math. 85(2), 113–138 (1998)
6. Fenner, T., Lachish, O., Popa, A.: Min-sum 2-paths problems. In: 11th Workshop on Approximation and Online Algorithms, WAOA 2013, Sophia Antipolis, France, September 5-6 (2013)
7. Kobayashi, Y., Sommer, C.: On shortest disjoint paths in planar graphs. Discrete Optim. 7(2), 234–245 (2010)
8. Koutis, I.: Faster algebraic algorithms for path and packing problems. In: Aceto, L., et al. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 575–586. Springer, Heidelberg (2008)
9. Li, C.-L., McCormick, S.T., Simchi-Levi, D.: The complexity of finding two disjoint paths with min-max objective function. Discrete Appl. Math. 26(1), 105–115 (1990)
10. Mulmuley, K., Vazirani, U.V., Vazirani, V.V.: Matching is as easy as matrix inversion. Combinatorica 7(1), 105–113 (1987)
11. Ohtsuki, T.: The two disjoint path problem and wire routing design. In: Graph Theory and Algorithms, Proc. 17th Symposium of Research Institute of Electric Communication, Sendai, Japan, October 24-25, 1980, pp. 207–216. Springer (1980)
12. Seymour, P.D.: Disjoint paths in graphs. Discrete Math. 29, 293–309 (1980)
13. Shiloach, Y.: A polynomial solution to the undirected two paths problem. J. ACM 27, 445–456 (1980)
14. Thomassen, C.: 2-linked graphs. Eur. J. Combin. 1, 371–378 (1980)
15. Tholey, T.: Solving the 2-disjoint paths problem in nearly linear time. Theory Comput. Syst. 39(1), 51–78 (2006)
16. Tutte, W.T.: The factorization of linear graphs. J. London Math. Soc. 22(2), 107–111 (1947)
17. Valiant, L.G.: The complexity of computing the permanent. Theor. Comput. Sci. 8(1), 189–201 (1979)
18. Wahlström, M.: Abusing the Tutte matrix: An algebraic instance compression for the K-set-cycle problem. In: 30th International Symposium on Theoretical Aspects of Computer Science, STACS 2013, Kiel, Germany, February 27-March 2. Schloss Dagstuhl – Leibniz-Zentrum für Informatik LIPIcs, vol. 20, pp. 341–352 (2013)
19. Williams, R.: Finding paths of length k in $O^*(2^k)$ time. Inf. Process. Lett. 109(6), 315–318 (2009)

Listing Triangles

Andreas Björklund[1], Rasmus Pagh[2],
Virginia Vassilevska Williams[3,⋆], and Uri Zwick[4,⋆⋆]

[1] Department of Computer Science, Lund University, Sweden
andreas.bjorklund@cs.lth.se
[2] IT University of Copenhagen, Denmark
pagh@itu.dk
[3] Computer Science Department, Stanford University, USA
virgi@cs.stanford.edu
[4] Blavatnik School of Computer Science, Tel Aviv University, Israel
zwick@tau.ac.il

Abstract. We present new algorithms for listing triangles in dense and sparse graphs. The running time of our algorithm for dense graphs is $\tilde{\mathcal{O}}(n^{\omega} + n^{3(\omega-1)/(5-\omega)} t^{2(3-\omega)/(5-\omega)})$, and the running time of the algorithm for sparse graphs is $\tilde{\mathcal{O}}(m^{2\omega/(\omega+1)} + m^{3(\omega-1)/(\omega+1)} t^{(3-\omega)/(\omega+1)})$, where n is the number of vertices, m is the number of edges, t is the number of triangles to be listed, and $\omega < 2.373$ is the exponent of fast matrix multiplication. With the current bound on ω, the running times of our algorithms are $\tilde{\mathcal{O}}(n^{2.373} + n^{1.568}\, t^{0.478})$ and $\tilde{\mathcal{O}}(m^{1.408} + m^{1.222}\, t^{0.186})$, respectively. We first obtain randomized algorithms with the desired running times and then derandomize them using *sparse recovery* techniques.

If $\omega = 2$, the running times of the algorithms become $\tilde{\mathcal{O}}(n^2 + nt^{2/3})$ and $\tilde{\mathcal{O}}(m^{4/3} + mt^{1/3})$, respectively. In particular, if $\omega = 2$, our algorithm lists m triangles in $\tilde{\mathcal{O}}(m^{4/3})$ time. Pătraşcu (STOC 2010) showed that $\Omega(m^{4/3-o(1)})$ time is required for listing m triangles, unless there exist subquadratic algorithms for 3SUM. We show that unless one can solve quadratic equation systems over a finite field significantly faster than the brute force algorithm, our triangle listing runtime bounds are tight assuming $\omega = 2$, also for graphs with more triangles.

1 Introduction

Algorithmic problems concerning the set of triangles in a graph have recently received much attention, due to applications in various kinds of graph analysis such as the study of social processes [8], community detection [5], and dense subgraph mining [25]. Many of these problems require the listing of all triangles in a graph — see [24,6,4] for a number of examples.

[⋆] Research supported by a Stanford School of Engineering Hoover Fellowship, NSF Grant CCF-1417238 and BSF Grant BSF:2012338.

[⋆⋆] Research supported by BSF grant no. 2012338 and by the The Israeli Centers of Research Excellence (I-CORE) program, (Center No. 4/11).

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 223–234, 2014.

We consider simple, directed or undirected graphs with n vertices and m edges. A dense graph may contain $\Theta(n^3)$ triangles, so in terms of n the worst-case complexity of the trivial cubic time algorithm is optimal. However, most graphs of interest are not dense. In 1978 Itai and Rodeh [13] obtained an algorithm for listing all triangles in $\mathcal{O}\left(m^{3/2}\right)$, which is always an improvement over the naïve $\mathcal{O}\left(n^3\right)$ algorithm. Their algorithm is optimal as a graph with m edges may contain $\Omega\left(m^{3/2}\right)$ triangles.

In this paper we consider *output sensitive* algorithms for triangle listing, which run asymptotically faster when the number t of triangles is small, with *no additional assumptions* on the input graph. (For example, we do not consider running time bounds in terms of graph parameters such as arboricity.) Our approach is to combine known techniques for *counting* the number of triangles, using fast matrix multiplication, with algebraic and combinatorial techniques that allow us to compute the actual triangles. We initially obtain randomized algorithms which we then derandomize using *sparse recovery* techniques.

Since our focus is on constants in the exponents of running time, we use $\tilde{\mathcal{O}}\left(\cdot\right)$ notation to suppress multiplicative factors of size $n^{o(1)}$. For dense graphs, our algorithm runs in $\tilde{\mathcal{O}}\left(n^{\omega} + n^{3(\omega-1)/(5-\omega)}t^{2(3-\omega)/(5-\omega)}\right)$ time, where $\omega < 2.373$ is the exponent of square matrix multiplication [26,18]. For sparse graphs, our algorithm runs in $\tilde{\mathcal{O}}\left(m^{2\omega/(\omega+1)} + m^{3(\omega-1)/(\omega+1)}t^{(3-\omega)/(\omega+1)}\right)$ time. Under the assumption $\omega = 2$ algorithms run in $\tilde{\mathcal{O}}\left(n^3\right)$ and $\tilde{\mathcal{O}}\left(m^{3/2}\right)$ algorithms for *every* possible value of t. Our dense and sparse algorithms are inter-dependent. The dense algorithm performs a sparsifying steps and calls the dense algorithm, while the sparse algorithm performs a densifying step and calls the dense algorithm.

Pătraşcu [23] has shown that listing m triangles in a graph with m edges requires time $\Omega\left(m^{4/3-\varepsilon}\right)$, for every $\varepsilon > 0$, unless there exists an algorithm for 3SUM running in $\mathcal{O}\left(n^{2-\delta}\right)$ time, for some $\delta > 0$. Our algorithm lists m triangles in $\tilde{\mathcal{O}}\left(m^{2\omega/(\omega+1)}\right)$ time. With the current bound $\omega < 2.373$, our algorithm lists m triangles in $\mathcal{O}\left(m^{1.408}\right)$. Interestingly, if $\omega = 2$, the running time becomes $\tilde{\mathcal{O}}\left(m^{4/3}\right)$, essentially matching the conditional lower bound of Pătraşcu [23]. Significant improvements of the exponents in our results are therefore unlikely.

The best previously available algorithms for triangle listing that we are aware of are the $\mathcal{O}\left(m^{3/2}\right)$ algorithm of Itai and Rodeh [13], from which it is also easy to obtain an $\tilde{\mathcal{O}}\left(n^{\omega} + \min(n^3, nt, t^{3/2})\right)$ algorithm, and an $\mathcal{O}\left(t^{1-\omega/3}n^{\omega}\right)$-time algorithm that follows from a reduction by Williams and Williams [27, Corollary G.1] from triangle listing to triangle detection. The running times obtained by our algorithms improve upon both of the aforementioned prior results for all values of t.

1.1 Related Work

Figure 1 compares the results described above, focusing on worst-case time complexity. For completeness we now describe some other related work that is not directly comparable to our results.

Reference	**Time bounds**	**If $\omega = 2$**
Itai and Rodeh [13]	$\tilde{\mathcal{O}}\left(n^{\omega} + \min(n^3, nt, t^{3/2})\right)$ $\mathcal{O}\left(m^{3/2}\right)$	$\tilde{\mathcal{O}}\left(n^2 + \min(nt, t^{3/2})\right)$
Williams and Williams [27]	$\tilde{\mathcal{O}}\left(n^{\omega} t^{1-\omega/3}\right)$	$\tilde{\mathcal{O}}\left(n^2 t^{1/3}\right)$
Pătraşcu [23]	$\tilde{\Omega}(\min(m^{4/3}, n^2, t^{4/3}))$ *	
This paper	$\tilde{\mathcal{O}}\left(n^{\omega} + n^{\frac{3(\omega-1)}{5-\omega}} t^{\frac{2(3-\omega)}{5-\omega}}\right)$ $\tilde{\mathcal{O}}\left(m^{\frac{2\omega}{\omega+1}} + m^{\frac{3(\omega-1)}{\omega+1}} t^{\frac{3-\omega}{\omega+1}}\right)$	$\tilde{\mathcal{O}}\left(n^2 + nt^{2/3}\right)$ $\tilde{\mathcal{O}}\left(m^{4/3} + mt^{1/3}\right)$

Fig. 1. Upper and (conditional) lower bounds for listing t triangles in a graph of n vertices and m edges. The results are stated in terms of the exponent ω of square matrix multiplication, which is known to be below 2.373 [26]. All bounds hold are for worst-case graphs and hold for every choice of $n, m, t \geq 1$. The rightmost column highlights the upper bounds that would result if $\omega = 2$. The lower bound by Pătraşcu marked by * relies on the assumption that 3SUM requires $\tilde{\Omega}(n^2)$ time.

Quite a bit of work has been done on triangle listing algorithms that perform well on real-life graphs. The paper of Schank and Wagner [24] contains a good overview of various algorithms with $\mathcal{O}\left(m^{3/2}\right)$ worst-case running time, and investigates how well these algorithms perform on graphs from various application areas, often running much faster than the worst-case analysis would suggest. One algorithm that is often able to beat the worst-case bound is based on enumerating a set of 2-paths where the degree of the middle vertex is no larger than the degrees of the start and end vertices (this is a simplified version of node-iterator-core from [24]). Recently, Berry et al. [4] gave a theoretical explanation why triangle listing is fast for most graphs, even for graphs with a skewed degree distribution, by studying a class of random graphs.

Recently many authors have studied triangle counting and listing algorithms for massive graphs, using either external memory [10,20] or the MapReduce framework for distributed computation [1]. However, for worst-case graphs these algorithms all use $\Omega(m^{3/2})$ time, even when the number of triangles is zero.

As mentioned above, Pătraşcu [23] showed a link between triangle listing and the time complexity of 3SUM. Jafargholi and Viola [14] further investigated this connection, showing that surprising algorithms for 3SUM would lead to surprising algorithms for triangle listing.

Alon, Yuster, and Zwick [2] show how to efficiently *detect* the presence of small subgraphs in sparse graphs. For triangles they achieve a time bound of $\mathcal{O}\left(m^{2\omega/(\omega+1)}\right)$, and the algorithm even allows *counting* the number of triangles. The algorithm consists of a densification step that enumerates all 2-paths (i.e., paths with two edges) through vertices with degree at most Δ, for a parameter Δ. In this way all triangles that contain a vertex of degree at most Δ are found. The number of triangles within the set of vertices of degree larger than

Δ is found by squaring the adjacency matrix, which for every pair of vertices gives the number of 2-paths that connect them. Summing over all edges we get the number of triangles multiplied by 3.

Many authors have given efficient algorithms for *approximately* counting the number of triangles in a graph, see e.g. [16] and its references. Most of these derive an estimator by some kind of sampling followed by an exact triangle counting algorithm.

1.2 Our Contributions

Our central contribution is a randomized algorithm that lists (with high probability) all triangles in a graph by alternating two procedures:

- **Densifying:** Eliminate vertices of low degrees by enumerating all 2-paths going through them, and
- **Sparsifiying:** Eliminate edges that are part of few triangles by reporting all such triangles using sparse signal recovery techniques.

We can derandomize the algorithm at a cost of a factor $n^{o(1)}$ in the running time by using known *explicit* constructions from the sparse signal recovery literature. Let ω denote the exponent of square matrix multiplication. In section 3 we show:

Theorem 1. *There exists a deterministic algorithm that lists all t triangles in a graph of n vertices in time $\tilde{\mathcal{O}}\left(n^{\omega} + n^{3(\omega-1)/(5-\omega)} t^{2(3-\omega)/(5-\omega)}\right)$.*

With the bound $\omega < 2.373$ [26] we get a time bound of $\mathcal{O}\left(n^{2.373} + n^{1.568}\, t^{0.478}\right)$. In section 3 we also derive the following theorem:

Theorem 2. *There exists a deterministic algorithm that lists all t triangles in a graph of m edges in time $\tilde{\mathcal{O}}\left(m^{2\omega/(\omega+1)} + m^{3(\omega-1)/(\omega+1)} t^{(3-\omega)/(\omega+1)}\right)$.*

Using the bound on ω as above we get: $\mathcal{O}\left(m^{1.408} + m^{1.222}\, t^{0.186}\right)$. In particular, listing m triangles in a graph of m edges can be done in time $\mathcal{O}\left(m^{1.408}\right)$.

We note that if $\omega = 2$ the time complexity for listing m triangles reduces to $\tilde{\mathcal{O}}\left(m^{4/3}\right)$, meeting the conditional lower bound of [23] based on hardness of 3SUM. In section 6 we show that unless another seemingly difficult problem has faster algorithms, namely quadratic systems of equations (QES), our two runtime bounds are tight also for graphs with more triangles.

QES is defined as follows. Let F be a finite field and $|F|$ its number of elements. A quadratic equation system over F^l is a set of k quadratic equations in l variables over F. It is easy to see that QES is NP-complete, as for instance NAESAT easily reduces to it with one equation per clause already over $F = GF(2)$, and it is a polynomial time task to verify a purported solution.

QES is a well-studied problem. The assumption that QES is intractable even on average has been used to design several important cryptosystems (e.g. [17,21]). A faster algorithm for QES would help attack these. Some algorithms that work well in practice have been designed (see e.g. [15,7]), though in the worst case, these do not improve over the exhaustive search $|F|^l \operatorname{poly}(l,k)$ time algorithm. It is a big open problem whether one can obtain a substantial improvement (of the form $|F|^{(1-\varepsilon)l}$ for some $\varepsilon > 0$) over exhaustive search for QES. We show that if one could improve upon our triangle listing algorithms (and $\omega = 2$), then QES does indeed have faster algorithms over any F.

Theorem 3. *Suppose that for some $\epsilon_1 \geq 0$, $\epsilon_2 \geq 0$ with $\epsilon_1 + \epsilon_2 > 0$, there exists an algorithm that lists all t triangles in an m-edge graph in $O(m^{1-\epsilon_1} t^{(1-\epsilon_2)/3})$ time or in an n-vertex graph in $O(n^{1-\epsilon_1} t^{(1-\epsilon_2)2/3})$ time. Then, for any finite field F, there exists an $|F|^{(1-\delta)l} \operatorname{poly}(l,k)$ time algorithm for $\delta > 0$ that solves l-variate quadratic equation systems with k equations over F^l.*

2 Listing Light Triangles

Let Λ be a parameter. We say that an edge is *Λ-light*, or just *light*, if it participates in at most Λ triangles, otherwise, it is said to be *Λ-heavy.* A triangle is *Λ-light* if at least one of the edges participating in it is light, otherwise it is *Λ-heavy.* In this section we describe a simple randomized algorithm for listing all Λ-light triangles with high probability. This algorithm is used as a building block by our algorithms for listing all triangles in dense and sparse graphs.

We include this simple randomized algorithm for completeness. The ideas behind it have been used before, for instance by Gasieniec et al. [9] who, building upon work of Aumann et al. [3] showed how to find k witnesses for Boolean matrix multiplication in $\tilde{\mathcal{O}}\left(n^2 k + n^\omega k^{(3-\omega-\alpha)/(1-\alpha)}\right)$ time. In Section 4 we describe a novel deterministic version of the algorithm described in this section using *sparse recovery* techniques.

Theorem 4. *Let $G = (V, E)$ be a graph on n vertices and let $1 \leq \Lambda \leq n$. Then, all Λ-light triangles in G can be found in $\tilde{\mathcal{O}}\left(n^\omega \Lambda^{3-\omega}\right)$ time, with high probability.*

Proof. We assume, without loss of generality, that $V = [n] = \{1, 2, \ldots, n\}$. Let A be the adjacency matrix of the graph. Let $\bar{A}$ be the matrix A in which all the 1s in the k-th column of A are replaced by k, for $k \in [n]$. Let $S \subseteq V$, let $A[*, S]$ denote the matrix obtained from A by selecting the columns whose indices belong to S. Similarly, let $A[S, *]$ denote the matrix obtained by selecting the rows of A whose indices belong to S. The rectangular Boolean product $A[*, S]A[S, *]$ tells us, for every $i, j \in [n]$, whether there is a path of length 2 from i to j that passes through a vertex of S. If there is only one such 2-path, then the (i, j)-th entry of the product $\bar{A}[*, S]A[S, *]$ identifies the k for which $(i, k), (k, j) \in E$.

Suppose now that $(i, j) \in E$ is a Λ-light edge, and let $T_{i,j} = \{k \in V \mid (i, k), (k, j) \in E\}$ be the set of 'mid-points' of the triangles passing through the edge (i, j). Note that $|T_{i,j}| \leq \Lambda$. Let S be a random subset of V of size n/Λ. Let $k \in T_{i,j}$. The probability that $|S \cap T_{i,j}| = 1$ is at least $\frac{1}{\Lambda}(1 - \frac{1}{\Lambda})^{\Lambda-1} \geq \frac{1}{e\Lambda}$. Thus, if we choose $\mathcal{O}(\Lambda \log n)$ random subsets of size n/Λ, we can, with high probability, identify all light triangles.

As each product $A[*, S]A[S, *]$ and $\bar{A}[*, S]A[S, *]$, where $|S| = n/\Lambda$, can be computed in $\tilde{O}(n^2 (n/\Lambda)^{\omega-2})$ (by decomposing each rectangular matrix product into square matrix products), the $\mathcal{O}(\Lambda \log n)$ products could all be computed in $\tilde{O}(n^{\omega}\Lambda^{3-\omega})$ time. □

It is not difficult to convert the algorithm into a Las Vegas algorithm whose expected running time is $\tilde{O}(n^{\omega}\Lambda^{3-\omega})$. The idea is to check that each reported triangle exists, and check for each edge that the number of triangles reported is correct (by comparing to the number of 2-paths connecting its end points). As pointed out by [9], using fast rectangular matrix multiplication [11,19], one can improve the running time of Theorem 4 to $\tilde{\mathcal{O}}\left(n^{\omega}\Lambda^{(3-\alpha-\omega)/(1-\alpha)} + \Lambda n^2\right) \leq \tilde{\mathcal{O}}\left(\Lambda n^2 + n^{2.373}\Lambda^{0.464}\right)$. Here $\alpha > 0.303$ is the largest constant such that $n \times n^{\alpha}$ by $n^{\alpha} \times n$ matrices can be multiplied in $\tilde{\mathcal{O}}\left(n^2\right)$ time. This implies slight improvements of the time bounds in Theorems 1 and 2.

3 Listing All Triangles

We next describe two algorithms for listing all triangles in dense and sparse graphs that use each other as subroutines. We let `Dense` (n,t) be the algorithm for listing all triangles in a graph on n vertices containing at most t triangles, and use $D(n,t)$ to denote the running time of `Dense` (n,t). Similarly, we let `Sparse` (m,t) be the algorithm for listing all triangles in a graph with m edges (we assume that the graph has no isolated vertices to make m a proper bound on the size of the graph) containing at most t triangles, and let $S(m,t)$ denote the running time of `Sparse` (m,t). We assume that these algorithms receive an upper bound t on the number of triangles in the input graph. This upper bound can be computed before calling our algorithms, either in $\tilde{\mathcal{O}}(n^{\omega})$ time, or in $\mathcal{O}\left(m^{2\omega/(\omega+1)}\right)$ time [2].

`Sparse` (m,t) works as follows. It chooses a parameter Δ depending on m and t. Vertices of degree at most Δ are said to be *low* degree vertices. Vertices of degree greater than Δ are said to be *high* degree. The algorithm starts by finding all triangles that contain a low degree vertex. This can be easily done in $\mathcal{O}(m\Delta)$ time by examining for every edge incident on a low degree vertex x, the length 2-paths formed by taking another edge out of x. Once this is done we can remove all edges incident to low degree vertices. If no edges remain we stop — otherwise, all remaining triangles, i.e., triangles that only include high degree vertices can now be found by a call to `Dense` $(2m/\Delta, t)$, as there are at most $2m/\Delta$ high degree vertices. Thus, ignoring constant factors,

$$S(m,t) \;\leq\; m\Delta + D(2m/\Delta, t) \; . \tag{1}$$

`Dense` (n,t) works as follows. If $n < 3$, it returns no triangles. Otherwise, it chooses a parameter Λ depending on n and t. It then finds all Λ-light triangles in $\tilde{\mathcal{O}}\left(n^{\omega}\Lambda^{3-\omega}\right)$ time by Theorem 4 (or its deterministic version from Section 4). Once this is done we can remove all Λ-light edges. If no edges remain we stop — otherwise, as there can be at most $3t/\Lambda$ Λ-heavy edges, all Λ-heavy triangles can be found by a call to `Sparse` $(3t/\Lambda, t)$. Thus, ignoring $n^{o(1)}$ factors,

$$D(n,t) \;\leq\; n^{\omega}\Lambda^{3-\omega} + S(3t/\Lambda, t) \; . \tag{2}$$

To analyze the running times of the two algorithms we set

$$\Lambda = \lceil \max(3,\ 6n^{-(\omega+1)/(5-\omega)}t^{2/(5-\omega)}) \rceil, \text{ and}$$
$$\Delta = \lceil 2\max(m^{(\omega-1)/(\omega+1)},\ m^{2(\omega-2)/(\omega+1)}t^{(3-\omega)/(\omega+1)}) \rceil.$$

Suppose first that $t \geq m$. Notice that since we never change t and the number of edges never increases over the recursive calls, if we ever have $t \geq m$, we have $t \geq m$ in all subsequent calls. We get

$$\Delta = 2m^{2(\omega-2)/(\omega+1)}t^{(3-\omega)/(\omega+1)},\ m\Delta = 2m^{3(\omega-1)/(\omega+1)}t^{(3-\omega)/(\omega+1)}.$$

Consider the first recursive call to the dense algorithm, and suppose that it was called on n nodes, where we know that $n \leq 2m/\Delta$. We get the following:

$$n \leq 2m/\Delta = m^{(5-\omega)/(\omega+1)}t^{-(3-\omega)/(\omega+1)}.$$

$$n^{(\omega+1)/2} \leq (2m/\Delta)^{(\omega+1)/2} = \left(\frac{m^{(5-\omega)/(\omega+1)}}{t^{(3-\omega)/(\omega+1)}}\right)^{(\omega+1)/2} = t \cdot (m/t)^{(5-\omega)/2}.$$

Thus, $t/n^{(\omega+1)/2} \geq (t/m)^{(5-\omega)/2}$ which is ≥ 1 when $t \geq m$, and since $\Lambda = \max\{3, 6(t/n^{(\omega+1)/2})^{2/(5-\omega)}\}$, we get that

$$\Lambda = 6(t/n^{(\omega+1)/2})^{2/(5-\omega)} \geq 6t/m.$$

We now get:

$$\begin{aligned} n^{\omega}\Lambda^{3-\omega} &= 6^{3-\omega}n^{3(\omega-1)/(5-\omega)}t^{2(3-\omega)/(5-\omega)} \\ &\leq 6^{3-\omega}(2m/\Delta)^{3(\omega-1)/(5-\omega)}t^{2(3-\omega)/(5-\omega)} \\ &= 6^{3-\omega}m^{3(\omega-1)/(\omega+1)}t^{(3-\omega)/(\omega+1)}. \end{aligned} \tag{3}$$

Since $\Lambda \geq 6t/m$, we get that $3t/\Lambda \leq m/2$, and hence $S(3t/\Lambda, t) \leq S(m/2, t)$. By Eq. 1 and Eq. 2 we have

$$\begin{aligned} S(m,t) &\leq m\Delta + n^{\omega}\Lambda^{3-\omega} + S(3t/\Lambda, t) \\ &\leq (2+6^{3-\omega})m^{3(\omega-1)/(\omega+1)}t^{(3-\omega)/(\omega+1)} + S(m/2, t) \\ &\leq \sum_{i=1}^{\lceil \log m \rceil} (2+6^{3-\omega})(m/2^i)^{3(\omega-1)/(\omega+1)}t^{(3-\omega)/(\omega+1)} \\ &\in \mathcal{O}\left(m^{3(\omega-1)/(\omega+1)}t^{(3-\omega)/(\omega+1)}\right). \end{aligned}$$

Next assume $t < m$. We get $\Delta = 2m^{(\omega-1)/(\omega+1)}$. By the above analysis we get that $\Lambda \leq 6(t/m)^{(5-\omega)/2}$, and so when $t < m$, we have $3 \leq \Lambda \leq 6$. We also have $n \leq 2m/\Delta = m^{2/(\omega+1)}$. By Eq. 1 and Eq. 2 we have

$$\begin{aligned} S(m,t) &\leq m\Delta + n^{\omega}\Lambda^{3-\omega} + S(3t/\Lambda, t) \\ &\leq (2+6^{3-\omega})m^{2\omega/(\omega+1)} + S(t,t) \\ &\leq (2+6^{3-\omega})m^{2\omega/(\omega+1)} + \mathcal{O}\left(t^{2\omega/(\omega+1)}\right) \\ &\in \mathcal{O}\left(m^{2\omega/(\omega+1)}\right). \end{aligned}$$

Once we have established the complexity of $\mathtt{Sparse}\,(m,t)$, it is also easy to establish the complexity of $\mathtt{Dense}\,(n,t)$. There are again two cases. If $t \le n^{(\omega+1)/2}$, then $3 \le \Lambda \le 6$ and the running time is

$$D(n,t) \;=\; \mathcal{O}\left(n^{\omega} + S(t,t)\right) = \mathcal{O}\left(n^{\omega}\right)\;.$$

If $t > n^{(\omega+1)/2}$, then $\Lambda = 6n^{-(\omega+1)/(5-\omega)}t^{2/(5-\omega)}$ and then

$$\begin{aligned} D(n,t) \;&\le\; n^{3(\omega-1)/(5-\omega)}t^{2(3-\omega)/(5-\omega)} + S(3t/\Lambda,t) \\ &\le\; (n^{3(\omega-1)}t^{2(3-\omega)})^{1/(5-\omega)} + S((t^{(3-\omega)}n^{(\omega+1)})^{1/(5-\omega)},t) \\ &\le\; (n^{3(\omega-1)}t^{2(3-\omega)})^{1/(5-\omega)} + (t^{3-\omega}n^{\omega+1})^{3(\omega-1)/((\omega+1)(5-\omega))}t^{(3-\omega)/(\omega+1)} \\ &=\; \mathcal{O}\left(n^{3(\omega-1)/(5-\omega)}t^{2(3-\omega)/(5-\omega)}\right)\;, \end{aligned}$$

as required.

4 Deterministic Algorithm

Randomization was only used by the algorithm for listing light triangles.

We now proceed to show how to list all Λ-light triangles. This is achieved by computing, for every light edge, the list of at most Λ 2-paths connecting its vertices. Each such list can be thought of as a vector $x \in \{0,1\}^n$ with at most Λ 1s, corresponding to the connecting nodes. Let P_Λ denote the set of such vectors that we would like to compute.

To this end we make use of a *sparse recovery* matrix T with the following properties, for some function $f(n) = n^{o(1)}$:

- T has $d = \Theta(\Lambda f(n))$ rows.
- The number of non-zero entries in T is at most $nf(n)$.
- For every $x \in P_\Lambda$, we can compute x from Tx in time $\mathcal{O}\left(\Lambda f(n)\right)$.

Random sparse 0-1 matrices are known to have these properties with high probability for $f(n) = (\log n)^{\mathcal{O}(1)}$ (see e.g. [22] for an overview of such constructions), and there also exist explicit, deterministic constructions with $f(n) = n^{o(1)}$ [12]. Let D_i denote the *diagonal* matrix where the jth entry along the diagonal is equal to $T_{i,j}$.

Let A denote the adjacency matrix of the graph. To find all light triangles we compute, for $i = 1,\ldots,d$, the matrix product AD_iA. If D_i has n_i non-zero entries, this reduces to an n-by-n_i times n_i-by-n matrix product. For every $x \in P_\Lambda$ this gives us the vector Tx. Specifically, if x is the set of vertices connecting vertices a and b, $(Tx)_i = (AD_iA)_{a,b}$. This means that we can recover each $x \in P_\Lambda$ in time $\mathcal{O}\left(\Lambda f(n)\right)$.

The matrix product AD_iA can be decomposed into $\mathcal{O}\left((n/n_i)^2\right)$ square matrix products, each taking time $\mathcal{O}\left(n_i^{\omega}\right)$. By choice of T we have $\sum_i n_i \le df(n)$. So the time for computing all d matrix products is bounded by a constant times

$$\sum_{i=1}^{d} n^2 n_i^{\omega-2} \le dn^2\left(\sum_{i=1}^{d} n_i/d\right)^{\omega-2} \le dn^2(nf(n)/d)^{\omega-2} = n^{\omega}\Lambda^{3-\omega}f(n)^{\omega-1}\;.$$

The first inequality uses Jensen's inequality and the fact that $n_i^{\omega-2}$ is concave (since $\omega - 2 \in [0; 1]$). The second inequality uses our bounds on d and $\sum_i n_i$. Similarly to our randomized algorithm, the deterministic algorithm can be improved to have runtime $\tilde{\mathcal{O}}\left(n^{\omega} \Lambda^{(3-\alpha-\omega)/(1-\alpha)} + \Lambda n^2\right)$ using rectangular matrix multiplication.

5 Listing Some Triangles

If a graph contains T triangles and we are only required to list t of them, then an improved running time can be obtained as follows. First assume that the given graph is tripartite by creating three copies of each vertex v, v_I in partition I, v_J in partition J and v_K in partition K. Then each edge (u, v) appears 6 times, once for each pair of copies of u and v in different partitions. Each triangle appears 6 times as well, so it suffices to list $6t$ triangles in this new graph. Suppose now that we want to list t triangles in a tripartite graph with $T > t$ triangles. We design a recursive algorithm as follows. Split I, J, K into 2 parts of $n/2$ nodes each, $I_1, I_2, J_1, J_2, K_1, K_2$. Count the triangles in each of the 8 subgraphs induced by $I_i \cup J_j \cup K_k$, and recurse on the part that has the most triangles. At some point, the number of triangles in the part $I_i \cup J_j \cup K_k$ with most triangles will be $< t$, and at this point we no longer recurse, but use our triangle listing algorithm on the current subgraph G'. We know that when we recursed on G', it had at least t triangles, but since each of the 8 triples of subgraphs of G' have $< t$, then G' has $< 8t$ triangles. Consider now the number of nodes of G'. Suppose that it is $3n/2^j$ for some j, and we have done j recursive steps to find G'. In each step the number of triangles goes down by at most a factor of 8, so G' has at least $T/8^j$ triangles. Yet, G' has $< 8t$ triangles, and hence $8^{j+1} > T/t$, and hence the number of nodes in G' is $O(n/(T/t)^{1/3})$. We thus get a running time of

$$\tilde{\mathcal{O}}\left(n^{\omega} + \left(\left(\frac{t}{T}\right)^{1/3} n\right)^{3(\omega-1)/(5-\omega)} t^{2(3-\omega)/(5-\omega)}\right).$$

Using a similar idea, combined with an approach from [14], we can also get an improvement for sparse graphs (in terms of m).

6 Consequences of Faster Triangle Listing

In this section we prove Theorem 3. We show that if one could improve upon our triangle listing algorithms (and $\omega = 2$), then QES does indeed have faster algorithms over any F.

Let F be a finite field and $q = |F|$ its number of elements. Assume that there is no $q^{(1-\epsilon_3)l}\,\mathrm{poly}(l, k)$ time algorithm for any $\epsilon_3 > 0$ that solves l-variate QES on k equations. Given an instance to QES on l variables with k equations $x'Q_i x + E_i x + S_i = 0$ over F, where Q_i are $l \times l$ matrices, E_i are $1 \times l$ vectors, and S_i are scalars, we will show how one can use triangle listing to solve it.

Much as Pătraşcu [23] did for 3SUM, we use hashing as a filter to find the solutions to QES. We construct h hashed projections of the equations $x'A_ix + B_ix + C_i = 0$ for $i = 1, 2, \ldots, h$ where

$$A_i = \sum_{j=1}^{k} R_i(j)Q_j, \qquad B_i = \sum_{j=1}^{k} R_i(j)E_j, \qquad C_i = \sum_{j=1}^{k} R_i(j)S_j$$

for a random $R_i \in F^k$ (for a vector R we write $R(j)$ to address its jth element). The hashed QES (A, B, C) has the following relations to the original QES:

- Every solution to (Q, E, S) is a solution also to (A, B, C).
- Every non-solution to (Q, E, S) is a solution to (A, B, C) with prob. q^{-h}.

This means that if (Q, E, S) has s solutions, (A, B, C) has at most $2 \cdot q^{l-h} + s$ solutions with probability at least 1/2 by the linearity of expectation and Markov's inequality. We can assume that $s < q^{\epsilon_3 l}$ since if not we can use another algorithm in parallel that simply guesses an assignment and verifies it, which runs in expected time $O(q^l/s)$.

We next construct a graph G that has a triangle for each solution to (A, B, C). Let a be a parameter to be fixed later. The vertex set is the union of three sets:

- V_1 has one vertex labeled (ϕ_1) for each assignment ϕ_1 to the first $l - 2a$ variables, in total q^{l-2a} vertices.
- V_2 has one vertex labeled (ϕ_2, H_2) for each combination of an assignment ϕ_2 to the next a variables $x_{l-2a+1}, \ldots, x_{l-a}$ and a vector H_2 in F^h, in total q^{a+h} vertices.
- V_3 has one vertex labeled (ϕ_3, H_3) for each combination of an assignment ϕ_3 to the last a variables $x_{l-a+1}, \ldots, x_l$ and a vector H_3 in F^h, in total q^{a+h} vertices.

We let 0_k denote the assignment of k variables to the value 0. The edges are:

- (ϕ_1) and (ϕ_2, H_2) has an edge iff the assignments $x = \phi_1\phi_2 0_{l-a}$ and $y = \phi_1 0_{2a}$ to the variables give $x'A_ix + B_iy + C_i = -H_2(i)$, i.e. we consider the contribution where we use all quadratic terms associated with the vertices and the linear term associated with the first one. There are q^{l-a} edges.
- (ϕ_2, H_2) and (ϕ_3, H_3) has an edge iff the assignments $x = 0_{l-2a}\phi_2\phi_3$ and $y = 0_{n-2a}\phi_2 0_a$ to the variables give $x'A_ix + B_iy + C_i = H_2(i) - H_3(i)$. There are q^{2a+h} edges.
- (ϕ_3, H_3) and (ϕ_1) has an edge iff the assignments $x = \phi_1 0_a \phi_3$ and $y = 0_{n-a}\phi_3$ to the variables give $x'A_ix + B_iy + C_i = H_3(i)$. There are q^{2a+h} edges.

A triangle in the graph corresponds to a solution to (A, B, C) since on the left side we count each term exactly once, and on the right hand side H_2 and H_3 are counted twice with opposite signs and cancel.

We can use our triangle listing algorithm on G to solve (Q, E, S): for each found triangle $(\phi_1), (\phi_2, H_2), (\phi_3, H_3)$ we verify if $x = \phi_1\phi_2\phi_3$ is also a solution to (Q, E, S). To arrive at the lower bound, we note that the graph G has

- $q^{l-2a} + 2 \cdot q^{a+h}$ vertices.
- $q^{l-a} + 2 \cdot q^{2a+h}$ edges.
- $2 \cdot q^{l-h} + s < 2 \cdot q^{l-h} + q^{\epsilon_3 l} < 2 \cdot q^{l-h+\epsilon_3 l}$ triangles with probability 1/2.

We set $a = (l-h)/3$ to get $m = 3 \cdot q^{2l/3+h/3}$and $n = 3q^{l/3+2h/3}$. By varying h we can control the number of triangles w.r.t. m and n.

Now assume there is a $O(m^{1-\epsilon_1} t^{(1-\epsilon_2)/3})$ time algorithm for triangle listing for some $t \geq m$. With our bounds on m and t we get

$$\mathcal{O}\left(q^{l-\epsilon_1(2l/3+h/3)-\epsilon_2(l/3-h/3+\epsilon_3 l)+\epsilon_3 l}\right)$$

time. For small enough constant ϵ_3 we get a contradiction of the assumption of non-existence of any $O(q^{(1-\epsilon_3)l})$ time algorithm for QES. If we instead assume a $n^{1-\epsilon_1} t^{(1-\epsilon_2)2/3}$ time algorithm for triangle listing for some t, we get $\mathcal{O}\left(q^{l-\epsilon_1(l/3+2h/3)-\epsilon_2(2l/3-2h/3+\epsilon_3 l)+\epsilon_3 l}\right)$ time, also a contradiction.

Acknowledgement. We send our thanks to Mihai Pătraşcu with whom we first discussed the ideas leading to this paper. We thank Eric Price and Jelani Nelson for information on sparse recovery methods.

References

1. Afrati, F.N., Fotakis, D., Ullman, J.D.: Enumerating subgraph instances using Map-Reduce. In: Proc. IEEE International Conference on Data Engineering (ICDE), pp. 62–73 (2013)
2. Alon, N., Yuster, R., Zwick, U.: Color-coding. J. ACM 42(4), 844–856 (1995)
3. Aumann, Y., Lewenstein, M., Lewenstein, N., Tsur, D.: Finding witnesses by peeling. ACM Transactions on Algorithms 7(2), 24 (2011)
4. Berry, J., Fostvedt, L., Nordman, D., Phillips, C., Seshadhri, C., Wilson, A.: Why do simple algorithms for triangle enumeration work in the real world? In: Proc. Innovations in Theoretical Computer Science (2014)
5. Berry, J.W., Hendrickson, B., LaViolette, R.A., Phillips, C.A.: Tolerating the community detection resolution limit with edge weighting. Phys. Rev. E 83, 056119 (2011)
6. Chu, S., Cheng, J.: Triangle listing in massive networks. ACM Trans. Knowl. Discov. Data 6(4), 17:1–17:32 (2012)
7. Courtois, N., Klimov, A., Patarin, J., Shamir, A.: Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 392–407. Springer, Heidelberg (2000)
8. Welles, B.F., Van Devender, A., Contractor, N.: Is a friend a friend? Investigating the structure of friendship networks in virtual worlds. In: CHI 2010 Extended Abstracts on Human Factors in Computing Systems, pp. 4027–4032. ACM (2010)
9. Gasieniec, L., Kowaluk, M., Lingas, A.: Faster multi-witnesses for boolean matrix multiplication. Inf. Process. Lett. 109(4), 242–247 (2009)
10. Hu, X., Tao, Y., Chung, C.-W.: Massive graph triangulation. In: Proc. of SIGMOD, pp. 325–336. ACM (2013)
11. Huang, X., Pan, V.Y.: Fast rectangular matrix multiplication and applications. J. of Complexity 14(2), 257–299 (1998)

12. Indyk, P.: Explicit constructions for compressed sensing of sparse signals. In: Proc. of 19th SODA, pp. 30–33 (2008)
13. Itai, A., Rodeh, M.: Finding a minimum circuit in a graph. SIAM Journal on Computing 7(4), 413–423 (1978)
14. Jafargholi, Z., Viola, E.: 3SUM, 3XOR, triangles. CoRR, abs/1305.3827 (2013)
15. Kipnis, A., Shamir, A.: Cryptanalysis of the HFE public key cryptosystem by relinearization. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 19–30. Springer, Heidelberg (1999)
16. Kolountzakis, M.N., Miller, G.L., Peng, R., Tsourakakis, C.E.: Efficient triangle counting in large graphs via degree-based vertex partitioning. Internet Mathematics 8(1-2), 161–185 (2012)
17. Landau, S.: Polynomials in the nation's service: using algebra to design the advanced encryption standard. American Mathematical Monthly 111, 89–117 (2004)
18. Le Gall, F.: Powers of tensors and fast matrix multiplication. CoRR, abs:1401.7714 (2014)
19. Gall, F.L.: Faster algorithms for rectangular matrix multiplication. In: Proc. Foundations of Computer Science, pp. 514–523 (2012)
20. Pagh, R., Silvestri, F.: The input/output complexity of triangle enumeration. arXiv preprint arXiv:1312.0723 (2013)
21. Patarin, J.: Cryptoanalysis of the matsumoto and imai public key scheme of eurocrypt'88. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 248–261. Springer, Heidelberg (1995)
22. Porat, E., Strauss, M.J.: Sublinear time, measurement-optimal, sparse recovery for all. In: Proc. of 23rd SODA, pp. 1215–1227. SIAM (2012)
23. Pătraşcu, M.: Towards polynomial lower bounds for dynamic problems. In: Proc. of 42nd STOC, pp. 603–610 (2010)
24. Schank, T., Wagner, D.: Finding, counting and listing all triangles in large graphs, an experimental study. In: Nikoletseas, S.E. (ed.) WEA 2005. LNCS, vol. 3503, pp. 606–609. Springer, Heidelberg (2005)
25. Wang, N., Zhang, J., Tan, K.-L., Tung, A.K.: On triangulation-based dense neighborhood graph discovery. Proc. VLDB Endowment 4(2), 58–68 (2010)
26. Williams, V.V.: Multiplying matrices faster than coppersmith-winograd. In: Proc. of 44nd STOC, pp. 887–898 (2012)
27. Williams, V.V., Williams, R.: Subcubic equivalences between path, matrix and triangle problems. In: Proc. IEEE Foundations of Computer Science (FOCS), pp. 645–654 (2010)

On DNF Approximators for Monotone Boolean Functions

Eric Blais[1,⋆], Johan Håstad[2,⋆⋆], Rocco A. Servedio[3,⋆⋆⋆], and Li-Yang Tan[3,†]

[1] MIT
[2] KTH Royal Institute of Technology
[3] Columbia University

Abstract. We study the complexity of approximating monotone Boolean functions with disjunctive normal form (DNF) formulas, exploring two main directions. First, we construct DNF approximators for arbitrary monotone functions achieving one-sided error: we show that every monotone f can be ε-approximated by a DNF g of size $2^{n-\Omega_\varepsilon(\sqrt{n})}$ satisfying $g(x) \leq f(x)$ for all $x \in \{0,1\}^n$. This is the first non-trivial universal upper bound even for DNF approximators incurring two-sided error.

Next, we study the power of negations in DNF approximators for monotone functions. We exhibit monotone functions for which non-monotone DNFs perform better than monotone ones, giving separations with respect to both DNF size and width. Our results, when taken together with a classical theorem of Quine [1], highlight an interesting contrast between approximation and exact computation in the DNF complexity of monotone functions, and they add to a line of work on the surprising role of negations in monotone complexity [2,3,4].

1 Introduction

Monotone Boolean functions constitute a rich and complex class of functions, and their structural and combinatorial properties have been intensively studied for decades; see e.g. the monograph [5] for an in-depth survey. In complexity theory monotone functions play an especially important role in circuit complexity, where Razborov's celebrated result [2] has led to a significant body of work centered around monotone functions and the circuits that compute them [6,4,7,8,9,10,11,12,13,14,15,16,17].

In this paper we study the circuit complexity of *approximating* monotone functions, focusing on DNF formulas, one of the simplest and most basic types of

[⋆] Supported by a Simons Postdoctoral Fellowship.

[⋆⋆] Supported by ERC Advanced Investigator Grant 226203. Work done while visiting the Simons Institute.

[⋆⋆⋆] Supported by NSF grants CCF-1115703 and CCF-1319788.

[†] Supported by NSF grants CCF-1115703 and CCF-1319788. Part of this research was completed while visiting KTH Royal Institute of Technology, supported by ERC Advanced Investigator Grant 226203.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 235–246, 2014.

circuits. We say that a DNF ε-approximates a function $f : \{0,1\}^n \to \{0,1\}$ if the function g computed by the DNF satisfies $f(x) = g(x)$ on at least a $1-\varepsilon$ fraction of inputs x in $\{0,1\}^n$. Recent works [18,19] have highlighted interesting qualitative and quantitative differences in the landscape of DNF complexity when the formula is only required to approximate f rather than compute it exactly, and while the DNF complexity of exact computation is fairly well-understood, these papers have also pointed to significant gaps in our understanding of seemingly basic questions regarding the DNF complexity of approximate computation.

We continue this study and explore two main directions. In the first direction we seek a non-trivial upper bound on the DNF complexity of approximating an arbitrary monotone function to high accuracy, in the spirit of the positive results of [19]. In the second direction, in the spirit of Razborov's theorem [2] we seek a separation between the relative powers of monotone and non-monotone DNF that approximate monotone functions. As we describe below, our results further illustrate how different DNF complexity can be in the settings of exact versus approximate computation.

Universal bounds on approximability. Recent work of [19] established the first non-trivial universal upper bound on the DNF complexity of approximating an arbitrary Boolean function, achieving logarithmic savings over the worst-case cost of $\Omega(2^n)$ necessary for exact computation:

Theorem 1 of [BT13]. *Every Boolean function can be ε-approximated by a DNF of size $O_\varepsilon(2^n/\log n)$.*

We begin with the simple observation that this result does not say anything meaningful about the approximation of monotone functions. Since the minimal satisfying assignments of a monotone function form a Sperner family, Sperner's classical theorem readily translates into an upper bound on the DNF complexity of *exactly* computing monotone functions that is polynomially stronger:

Fact 1. *Every monotone function can be computed exactly by a DNF of size $\binom{n}{\lceil n/2 \rceil} = \Theta(2^n/\sqrt{n})$.*

This bound is exactly tight by considering the n-variable majority function, and in fact an elementary combinatorial argument establishes that a $1 - o_n(1)$ fraction of monotone functions do actually require DNFs of size $\Omega(2^n/\sqrt{n})$ to compute. Fact 1, taken together with the result of [19], raises a basic qualitative question: are there monotone functions that require DNFs of size $\Omega(2^n/\sqrt{n})$ to approximate, or can every monotone function be approximated by a DNF of size $o(2^n/\sqrt{n})$? Despite the vast literature on monotone functions and Sperner families, this question does not appear to have been explicitly studied before. We answer this question in the first half of the paper, constructing DNF approximators for arbitrary monotone functions that achieve exponential savings over the size necessary for exact computation. Our DNF approximators only make one-sided error, and our construction is based on a new structural decomposition of monotone functions.

Power of negations in approximating monotone functions. In the second half of the paper we turn our attention to the role of *negations* in the DNF complexity of approximating monotone functions. Recall that a circuit is said to be monotone if it does not contain any NOT gates, and non-monotone otherwise. While every monotone function can be computed by a monotone circuit, there is a body of results showing the remarkable fact that for various circuit classes, the optimal circuit computing a monotone function must be non-monotone. The most prominent example is perhaps Razborov's celebrated lower bound [2]:

Razborov's Theorem. *There is a polynomial-time computable monotone function that requires monotone circuits of quasi-polynomial size.*

This separation of monotone NP from monotone P/poly was subsequently improved from quasi-polynomial to exponential by E. Tardos [8]. An analogue of Razborov's result in the setting of bounded-depth circuits was established by Okol'nishnikova, Ajtai, and Gurevich [3,4]:

Okol'nishnikova–Ajtai–Gurevich Theorem. *There is a monotone function in* AC^0 *that is not in monotone* AC^0*.*

For the class of DNFs, however, it is well-known (and straightforward to verify) that the analogue these separations does not hold [1]:

Quine's Theorem. *The optimal DNF, with respect to both size and width, computing a monotone function is monotone as well.*

In the second half of this paper we investigate the question: does Quine's theorem hold for *approximation* by DNFs? In other words, is the optimal DNF approximator for a monotone function monotone as well, or do negations buy us power in the setting of approximation? We show that the answer is the latter, giving separations with respect to both DNF size and width. Our results, taken in contrast with Quine's theorem, highlight an interesting qualitative difference between the DNF complexity of exact and approximate computation. More broadly, we believe that the role of negations in the circuit complexity of approximating monotone functions is a topic of intrinsic interest, and we view our separations as the first steps in its systematic study.

1.1 Our Results

Universal bounds on approximability. Our first result is the construction of DNF approximators for arbitrary monotone Boolean functions that achieve one-sided error:

Theorem 1. *Every monotone function* $f : \{0,1\}^n \to \{0,1\}$ *can be* ε*-approximated by a monotone function* g *of DNF size* $2^{n-\Omega_\varepsilon(\sqrt{n})}$*, satisfying* $g(x) \leq f(x)$ *for all* $x \in \{0,1\}^n$*.*

Prior to our work the only known universal upper bound, even for approximators incurring two-sided error, was the trivial one of $\binom{n}{\lceil n/2 \rceil} = \Theta(2^n/\sqrt{n})$, the size sufficient for exact computation. A standard information-theoretic argument (see [19] for proof) shows that any ε-approximator for a random Boolean function has DNF size $\Omega_\varepsilon(2^n/n)$; Theorem 1 therefore shows that the structure of monotonicity can be leveraged to obtain DNF approximators with complexity exponentially smaller than that required for almost all other functions. Our construction relies on a new structural fact about monotone functions which we believe may be of independent interest:

Lemma 1. *Let $f : \{0,1\}^n \to \{0,1\}$ be a monotone function and $\varepsilon > 0$. There is a function $g = g_1 \vee \cdots \vee g_t$ that ε-approximates f, where $t = O_\varepsilon(1)$ and each g_i is a monotone DNF with terms of width exactly k_i and size at least $(\varepsilon/2)\binom{n}{k_i}$. Furthermore, $g(x) \leq f(x)$ for all $x \in \{0,1\}^n$.*

Since $g(x) \leq f(x)$ for all $x \in \{0,1\}^n$, we say that g is a lower ε-approximator for f. We prove Lemma 1 in Section 2, and with this structural fact in hand, the task of constructing lower approximators for an arbitrary monotone function reduces to that of constructing lower approximators for the g_i's. Since g comprises only a constant number of these g_i's, taking a naive union bound incurs no more than a constant factor in terms of error and DNF size of the overall approximator. Our lower approximators for the g_i's, presented in Section 3, are obtained via a randomized algorithm that constructs an approximating DNF. We complement our positive result with a lower bound showing that Theorem 1 is essentially optimal:

Theorem 2. *Let $g : \{0,1\}^n \to \{0,1\}$ be a $\frac{1}{10}$-approximator for the majority function MAJ_n satisfying $g(x) \leq \mathsf{MAJ}_n(x)$ for all $x \in \{0,1\}^n$. Then g has DNF size $2^{n-O(\sqrt{n}\log n)}$.*

Power of negations in approximating monotone functions. The proof of Quine's classical theorem mentioned in the introduction is simple: given a DNF g that computes a monotone function f, if g contains a term T with a negated variable $\bar{x}_i$, it is easy to check that g still computes the same monotone function f if $\bar{x}_i$ is removed from T. Therefore, by removing all occurrences of negated variables in g, we obtain a monotone DNF h computing the same function f, where the size and width of h are at most those of g.

It is natural to suspect that the same would be true for DNF approximators, that the optimal DNF approximator for a monotone function is always monotone as well; indeed, we note that the universal DNF approximators we construct in Theorem 1 are in fact monotone. To be precise, we consider the following question:

Question 1. *Let f be a monotone function that is ε-approximated by a DNF g of size s (resp. width w). Can f be ε-approximated by a* monotone *DNF h of size s (resp. width w)?*

The simple proof of Quine's theorem does not extend to answer this question in the affirmative. In fact, for all three natural ways of "locally monotonizing" the DNF approximator g — removing $\bar{x}_i$ in T (as is done in the proof of Quine's theorem); replacing $\bar{x}_i$ with x_i in T; and removing T from f entirely — it is possible to construct examples showing that these operations increase the distance of g from f (i.e. worsens the quality of approximation).

In the second half of the paper we resolve Question 1 by showing, perhaps somewhat surprisingly, that the answer is "No" for both complexity measures of DNF size and DNF width. In Section 4 we prove the following two theorems:

Theorem 3 (Separation for DNF Size). *For all sufficiently large n, there exists an n-variable monotone function f and a value $\varepsilon = \varepsilon(n) > 0$ such that f can be ε-approximated by a DNF of size $O(n)$, but any monotone function that ε-approximates f has DNF size $\Omega(n^2)$.*

Theorem 4 (Separation for DNF Width). *For all sufficiently large n, and for all $k = o(n)$, there exists an n-variable monotone function f and a value $\varepsilon = \varepsilon(n) > 0$ such that f can be ε-approximated by a DNF of width $k + \log k$, but any monotone function that ε-approximates f has DNF width at least $2k - 1 - o_n(1)$.*

We view these separations as the first steps in quantifying just how powerful negations can be in the approximation of monotone functions, a question that does not appear to have been explicitly studied before (despite a significant body of results on the power of negations in the computation of monotone functions, as discussed above). We conclude the paper by listing a few interesting questions for future work in this direction.

1.2 Previous Work

The explicit study of the DNF complexity of approximating Boolean functions was initiated by O'Donnell and Wimmer [18]. They showed that DNF size $2^{O_\varepsilon(\sqrt{n})}$ is both necessary and sufficient for ε-approximating the n-variable majority function, and constructed an explicit n-variable monotone function for which any 0.01-approximating DNF must have size $2^{\Omega(n/\log n)}$. As mentioned above, Blais and Tan [19] gave universal upper bounds on DNF size for approximating arbitrary Boolean functions, but [19] does not consider monotone functions.

We also note that the earlier work of Bshouty and Tamon [20], which established Fourier concentration bounds for monotone Boolean functions, implies that every n-variable monotone function is ε-close to a depth-2 circuit of size $2^{O(\sqrt{n}\log(n)/\varepsilon)}$ in which the bottom-level gates are parity gates and the top gate is a threshold gate (with unbounded weights). Recall that while threshold-of-parity circuits can simulate DNF formulas with only a polynomial size increase [21,22], the converse is not true (indeed, even a single parity gate requires exponential DNF size). Thus the results of [20] do not imply the existence of nontrivial DNF approximators for monotone functions.

1.3 Preliminaries

Throughout this paper all probabilities and expectations are with respect to the uniform distribution unless otherwise stated; we will use boldface (*e.g.* $\boldsymbol{x}$ and $\mathbf{X}$) to denote random variables. For strings $x, y \in \{0,1\}^n$ we write $\|x\|$ to denote the Hamming weight $\#\{i \in [n] \colon x_i = 1\}$ of x, and $x \preceq y$ if $x_i \leq y_i$ for all $i \in [n]$, and $x \prec y$ if $x \preceq y$ and $x \neq y$. For $0 \leq k \leq n$, we write $\mathrm{Vol}(n,k) := \sum_{i=0}^{k} \binom{n}{i}$ to denote the volume of the n-dimensional Hamming ball of radius k.

A monotone Boolean function $f : \{0,1\}^n \to \{0,1\}$ is one that satisfies $f(x) \leq f(y)$ whenever $x \preceq y$. A DNF formula is the logical OR of logical ANDs, where we refer to each AND as a *term*. The *size* of a DNF is the number of terms it contains, and the *width* of a DNF is the maximum width of any term. For a term T, we write $|T|$ to denote the width of T, the number of literals occurring in it. For any $x \in \{0,1\}^n$, we write T_x to denote the monotone conjunction that accepts all and only those $y \in \{0,1\}^n$ such that $y \succeq x$. That is, $T_x(y) = 1$ iff $y_i = 1$ for all $i \in [n]$ such that $x_i = 1$. We say that x *defines a minterm in a monotone function* f if T_x is a minterm in the canonical DNF computing f, and we write $\mathrm{minterm}(x, f)$ to denote the indicator for this event. The *canonical DNF* for f is the unique monotone DNF whose terms correspond precisely to the minterms of f.

Let $f, g : \{0,1\}^n \to \{0,1\}$ be Boolean functions and $\varepsilon \in [0,1]$. We say that g is an ε-approximator for f, or that f and g are ε-close, if $\mathbf{Pr}[f(\boldsymbol{x}) \neq g(\boldsymbol{x})] \leq \varepsilon$. We say that g is a lower approximator for f if $g(x) \leq f(x)$ for all $x \in \{0,1\}^n$, and an upper approximator for f if $f(x) \leq g(x)$ for all $x \in \{0,1\}^n$.

Definition 1 (Density). *Let $f : \{0,1\}^n \to \{0,1\}$ and $k \in \{0, 1, \ldots, n\}$. The density of f at level k is defined to be $\mu_k(f) := \mathbf{Pr}_{\|\boldsymbol{x}\|=k}[f(\boldsymbol{x}) = 1]$.*

Fact 2. *Let f be a monotone function. Then $\mu_k(f) \geq \mu_{k-1}(f)$ for all $k \in [n]$.*

Fact 3 (Chernoff Bound). *Let $\mathbf{X} \sim \mathrm{Binomial}(n, 1/2)$. Then for any $0 \leq t \leq \sqrt{n}$, we have $\mathbf{Pr}\left[\mathbf{X} \geq \frac{n}{2} + t\frac{\sqrt{n}}{2}\right] \leq e^{-t^2/2}$ and $\mathbf{Pr}\left[\mathbf{X} \leq \frac{n}{2} - t\frac{\sqrt{n}}{2}\right] \leq e^{-t^2/2}$.*

Fact 4 (Anti-concentration of the Binomial). *For every $\varepsilon \geq 1/\sqrt{n}$ and interval $I \subseteq [0, n]$ of width at most $\varepsilon\sqrt{n}$, we have $\mathbf{Pr}_{\boldsymbol{x} \in \{0,1\}^n}[\|\boldsymbol{x}\| \in I] \leq 2\varepsilon$.*

2 A Regularity Lemma for Monotone DNFs

We begin with a new structural fact about monotone functions, which states that every monotone DNF f is lower approximated by the disjunction g of a constant number of monotone DNFs that are “dense” and “regular.” Here a “regular” DNF is one in which all terms have the same width k, and a “dense” regular DNF is one that contains a constant fraction of the $\binom{n}{k}$ many possible terms of width k. This structural decomposition is useful as it reduces the task of (lower) approximating an arbitrary monotone DNF f to that of (lower) approximating a dense regular one. Since g is the disjunction of only a constant number of dense regular DNFs, taking a naive union bound incurs only a constant factor in terms of error and DNF size of the overall approximator.

Definition 2 (Regular and Dense DNFs). *Let $k \in [n]$. We say that a monotone DNF f is k-regular if all its terms have width exactly k, and regular if it is k-regular for some k. Additionally, we say that f is (ε, k)-regular if it is a k-regular DNF with at least $\varepsilon \binom{n}{k}$ many terms.*

Our structural result (the proof of which is deferred to the full version due to space considerations) says that every monotone function is lower ε-approximated by the disjunction of $O_\varepsilon(1)$ many $(\varepsilon/2, k_i)$-regular DNFs, where each $k_i = (n/2) \pm O(\sqrt{n})$. More precisely:

Lemma 1. *For any $\varepsilon > 0$,every monotone function f is ε-close to the disjunction g of monotone DNFs, $g(x) = g_1(x) \vee \cdots \vee g_t(x)$, where $t \leq 2/\varepsilon$, each g_i is k_i-regular for some $k_i \in \left[(n/2) - \sqrt{n \ln(4/\varepsilon)/2}, (n/2) + \sqrt{n \ln(4/\varepsilon)/2}\right]$, the DNF size of g_i is at least $(\varepsilon/2)\binom{n}{k_i}$ (i.e., $\mu_{k_i}(g_i) \geq \varepsilon/2$), and $g(x) \leq f(x)$ for all $x \in \{0,1\}^n$.*

3 Lower Approximators for Regular DNFs

Given Lemma 1 it suffices to construct lower approximators for regular DNFs:

Proposition 1. *Let f be a regular monotone function. For every $\varepsilon > 0$ there exists a monotone DNF g of size $2^{n-\Omega(\varepsilon\sqrt{n}-\log(n))}$ that is a lower ε-approximator for f.*

Proof (of Theorem 1 assuming Proposition 1). By Lemma 1 every monotone f has a lower $(\varepsilon/2)$-approximator $g(x) = g_1(x) \vee \cdots \vee g_t(x)$ where $t \leq 4/\varepsilon$ and each $g_i(x)$ is a regular monotone function. Next, by Proposition 1 each regular $g_i(x)$ has a lower $(\varepsilon/2t)$-approximator $h_i(x)$ of size $2^{n-\Omega((\varepsilon\sqrt{n}/t)-\log(n))}$. Finally, by the union bound and the triangle inequality, we conclude that $h(x) = h_1(x) \vee \cdots \vee h_t(x)$ is a lower ε-approximator for f of size at most $t \cdot 2^{n-\Omega((\varepsilon\sqrt{n}/t)-\log(n))} = 2^{n-\Omega_\varepsilon(\sqrt{n})}$. □

Proof (of Proposition 1). We may assume that $\varepsilon \geq (C \log n)/\sqrt{n}$ (for some constant $C > 0$ which we will specify below), since otherwise the claimed bound on monotone DNF size is trivial. Let f be a k-regular monotone function for some $k \in [n]$. The minterms of our monotone approximator g will be conjunctions of the form T_y where $y \in f^{-1}(1)$, which guarantees that g will be a lower approximator for f. Furthermore, since $\mathbf{Pr}_{\boldsymbol{x}\in\{0,1\}^n}\left[\|\boldsymbol{x}\| \geq (n/2) + \sqrt{n\ln(3/\varepsilon)/2}\right] \leq \frac{\varepsilon}{3}$, and $\mathbf{Pr}_{\boldsymbol{x}\in\{0,1\}^n}\left[\|\boldsymbol{x}\| \in [k, k+\varepsilon\sqrt{n}/6]\right] \leq \frac{\varepsilon}{3}$, by the Chernoff bound and Fact 4 respectively, it suffices to ensure that the monotone DNF g we construct additionally satisfies:

$$\mathop{\mathbf{Pr}}_{\boldsymbol{x}\in A}\left[g(\boldsymbol{x}) \neq f(\boldsymbol{x})]\right] \leq \frac{\varepsilon}{3},$$

$$A := \left\{x \in \{0,1\}^n \colon \|x\| \in \left[k + \varepsilon\sqrt{n}/6, (n/2) + \sqrt{n\ln(3/\varepsilon)/2}\right]\right\}. \tag{1}$$

Note that if $k + \varepsilon\sqrt{n}/6 > (n/2) + \sqrt{n\ln(3/\varepsilon)/2}$ (*i.e.* the interval in the definition of A is empty) then f is $(2\varepsilon/3)$-close to the constant 0 function and the proposition is trivially true.

For every $\ell \in \{0, 1, \ldots, n-k\}$, we write S_ℓ to denote the 1-inputs of f with Hamming weight exactly $k+\ell$; that is, $S_\ell := \{x \in \{0,1\}^n \colon f(x) = 1 \text{ and } \|x\| = k+\ell\}$. The remainder of this proof will be devoted to showing that for each $\ell \geq \varepsilon\sqrt{n}/6$, there exists a monotone DNF g_ℓ satisfying:

i. The minterms of g_ℓ are of the form T_y for some $y \in S_{\ell/2}$ (and hence $g_\ell \leq f$),
ii. DNF-size$[g_\ell] = O(2^{n-\ell/2}) \leq 2^{n-\Omega(\varepsilon\sqrt{n})}$,
iii. $\mathbf{Pr}_{\boldsymbol{x}\in S_\ell}[g_\ell(\boldsymbol{x}) = 0] \leq \varepsilon/3$.

Indeed, taking g to be the disjunction of all g_ℓ where

$$k + \ell \in \left[k + \varepsilon\sqrt{n}/3, (n/2) + \sqrt{n\ln(3/\varepsilon)/2}\right],$$

we obtain a monotone DNF of size at most $n \cdot 2^{n-\Omega(\varepsilon\sqrt{n})} \leq 2^{n-\Omega(\varepsilon\sqrt{n}-\log(n))}$ satisfying (1), which completes the proof.

Consider a random monotone DNF $\boldsymbol{g}_\ell$ sampled according to the following distribution $\mathcal{D}$: for each $y \in S_{\ell/2}$, independently include T_y as a minterm of $\boldsymbol{g}_\ell$ with probability $p := 2^{-\ell/2}$. By definition, every DNF in the support of this distribution satisfies (i), and so it remains to argue that with positive probability, both (ii) and (iii) are satisfied as well. For (ii), we observe that $\mathbf{E}_{\mathcal{D}}\left[\text{DNF-size}[\boldsymbol{g}_\ell]\right] = p \cdot |S_\ell| < p \cdot 2^n = 2^{n-\ell/2}$, and so by Markov's inequality,

$$\mathop{\mathbf{Pr}}_{\mathcal{D}}\left[\text{DNF-size}[\boldsymbol{g}_\ell] \leq 3 \cdot 2^{n-\ell/2}\right] \geq \frac{2}{3}. \tag{2}$$

For (iii), consider any fixed $x \in S_\ell$. Since f is k-regular, there must exist some $z \in S_0$ such that $z \prec x$, and therefore $\binom{\ell}{\ell/2} = \Theta(2^\ell/\sqrt{\ell})$ many $y \in S_{\ell/2}$ such that $z \prec y \prec x$. By the definition of $\mathcal{D}$, for each such y the term T_y is independently included as a minterm of $\boldsymbol{g}_\ell$ with probability $p = 2^{-\ell/2}$, and so

$$\begin{aligned}\mathop{\mathbf{Pr}}_{\mathcal{D}}[\boldsymbol{g}_\ell(x) = 0] \;&\leq\; (1-p)^{\Theta(2^\ell/\sqrt{\ell})} = \exp\left(-\Omega(2^{\ell/2}/\sqrt{\ell})\right) \\ &< \exp\left(-\Omega(2^{\varepsilon\sqrt{n}/12})/\sqrt{n}\right) \;<\; \frac{\varepsilon}{9},\end{aligned}$$

where we have used $\varepsilon \geq (C\log n)/\sqrt{n}$ for the final inequality. Therefore

$$\mathop{\mathbf{E}}_{\mathcal{D}}\left[\mathop{\mathbf{Pr}}_{\boldsymbol{x}\in S_\ell}[\boldsymbol{g}_\ell(\boldsymbol{x}) = 0]\right] \leq \frac{\varepsilon}{9}, \quad \text{and} \quad \mathop{\mathbf{Pr}}_{\mathcal{D}}\left[\mathop{\mathbf{Pr}}_{\boldsymbol{x}\in S_\ell}[\boldsymbol{g}_\ell(\boldsymbol{x}) = 0] \leq \frac{\varepsilon}{3}\right] \geq \frac{2}{3}. \tag{3}$$

Applying a union bound to the failure probabilities of (2) and (3), we conclude that there is indeed a positive probability that $\boldsymbol{g}_\ell \sim \mathcal{D}$ satisfies all three properties (i), (ii), and (iii), and this completes the proof. □

Our next result, the proof of which we defer to the full version due to space considerations, shows that our upper bound in Theorem 1 is essentially tight.

Theorem 2. *Let $\varepsilon \leq \frac{1}{10}$ and g be an s-term DNF that is a lower ε-approximator for the majority function MAJ_n. Then $s \geq 2^{n-O(\sqrt{n}\log n)}$.*

4 Power of Negations in Monotone Approximation

In this section we present our constructions showing that non-monotone DNFs can asympototically outperform monotone ones in the approximation of monotone functions. Due to space considerations we only prove our separation for DNF size (Theorem 3) in this section; the proof of our separation for DNF width (Theorem 4) is deferred to the full version.

Upper bounds. Given these separations between monotone and non-monotone DNFs, it is natural to explore bounds in the other direction which show that the existence of (non-monotone) DNF approximators implies the existence of monotone DNF approximators of related size, width, and accuracy. We present two results in this direction in the full version.

Theorem 3 (Separation for DNF Size). *Let* $f : \{0,1\}^n \times \{0,1\}^{5n} \to \{0,1\}$ *be the monotone function:*

$$f(x,y) = (x_1 \vee \ldots \vee x_n) \wedge (y_1 \vee \ldots \vee y_{5n}) = \bigvee_{\substack{i\in[n]\\ j\in[5n]}} (x_i \wedge y_j),$$

and $\varepsilon = (2^{n-1}-1)\cdot 2^{-6n}$. *There exists a DNF of size* $6n-1$ *that* ε*-approximates* f, *but any monotone function that* ε*-approximates* f *has DNF size at least* n^2.

Proof. Consider the function $g = g(x,y)$ defined as

$$g = (x_1 \wedge (y_1 \vee \ldots \vee y_{5n})) \vee (\overline{x}_1 \wedge (x_2 \vee \ldots \vee x_n)) \tag{4}$$

This is a non-monotone DNF with $6n-1$ terms that ε-approximates f, since $g(x,y)$ differs from $f(x,y)$ exactly on the $2^{n-1}-1$ inputs satisfying $x_1 = 0$, $y = \mathbf{0}$, and $x_2 \vee \ldots \vee x_n = 1$.

The rest of the proof will be devoted to showing that any monotone function that ε-approximates f has to have more than n^2 terms, asymptotically as many as the canonical DNF for f which has $5n^2$ terms. We will prove the contrapositive: any monotone DNF h with at most n^2 terms differs from f on strictly more than an ε-fraction of inputs.

We group the terms of h into three types: terms with only x-variables, which we call "pure-x"; terms with only y-variables, which we call "pure-y"; and terms with both x- and y-variables, which we call "mixed". We first observe that we may assume that all mixed terms have width exactly two, comprising one x-variable and one y-variable. Indeed, replacing a mixed term $\big(\bigwedge_{i\in S_1} x_i\big) \wedge \big(\bigwedge_{j\in S_2} y_j\big)$, $S_1 \subseteq [n]$ and $S_2 \subseteq [5n]$, in h with $(x_i \wedge y_j)$ for any $i \in S_1$ and $j \in S_2$ yields a DNF h' such that $h'(x,y) \neq h(x,y)$ only on inputs (x,y) such that $h(x,y) = 0$ and $f(x,y) = 1$.

Furthermore, we claim that we may assume all pure-y terms have width greater than $2n$. Indeed, if h contains a term $T(y) = \bigwedge_{i\in S} y_i$ for some $S \subseteq [5n]$ where $|S| \leq 2n$, then $f(x,y) = 0$ and $h(x,y) = 1$ on at least $2^{3n} > \varepsilon \cdot 2^{6n}$ inputs (x,y) satisfying $x = \mathbf{0}$ and $T(y) = 1$.

We proceed by considering two cases, depending on the number of x_i's that occur as a singleton term in h. First suppose at least half of the x_i's occur as a singleton term in h, so there is some $S \subseteq [n]$ where $|S| \geq n/2$ such that if $\mathsf{OR}_S(x) = \bigvee_{i \in S} x_i = 1$ then $h(x, y) = 1$. In this case $f(x, y) = 0$ and $h(x, y) = 1$ on at least $2^n - 2^{n/2} > \varepsilon \cdot 2^{6n}$ inputs satisfying $y = \mathbf{0}$ and $\mathsf{OR}_S(x) = 1$. Finally, suppose less than half of the x_i's occur as singleton terms in h. By our first assumption that all mixed terms have width two (in particular, no mixed term contains more than one x-variable), there must be an x_i that does not occur as a singleton term and participates in at most $2n$ mixed terms (since otherwise h would have more than n^2 terms); without loss of generality suppose x_1 is one such variable. Let $S \subseteq [5n]$ be the set of all $j \in [5n]$ such that $(x_1 \wedge y_j)$ is a mixed term in h, and consider the set of inputs

$$E = \{(x, y) \colon x_1 = 1,\ x_i = 0 \text{ for all } i \geq 2,\ y_j = 0 \text{ for all } j \in S, \text{ and } \|y\| = (3n)/2\}.$$

Note that $f(x, y) = 1$ for all $(x, y) \in E$, and we claim that $h(x, y) = 0$ on these inputs. To see this, consider the restriction h^* of h obtained by setting $x_1 \leftarrow 1$, $x_i \leftarrow 0$ for all $i \geq 2$, and $y_j \leftarrow 0$ for all $j \in S$. Since x_1 does not occur as a singleton term in h, this partial assignment does not satisfy any terms and the canonical DNF for h^* comprises only of pure-y terms. Since the pure-y terms of h have width greater than $2n$ (by our second assumption), the same is true for h^* and so h^* cannot be satisfied by any assignment of weight $(3n)/2$; hence $h(x, y) = h^*(y) = 0$ for all $(x, y) \in E$. Lastly, we check that $|E| \geq \binom{3n}{(3n)/2} = \Theta\big(2^{3n}/\sqrt{3n}\big) > \varepsilon \cdot 2^{6n}$ and this completes the proof. □

Remark 5. *We note that the non-monotone approximator g in (4) is actually computed by a $O(n)$-size decision tree. Recall that every size-s decision tree is a size-s DNF, but not vice versa: there are polynomial-size DNFs that require exponential-size decision trees. Therefore the proof of Theorem 3 in fact establishes a stronger statement: f is a monotone function that can be ε-approximated by a $O(n)$-size decision tree, and yet any monotone function that ε-approximates f has DNF size $\Omega(n^2)$.*

5 Conclusion

Having obtained near-matching upper and lower bounds on the size of universal lower approximators in this paper, the natural next step is to consider *upper* approximators and approximators incurring error on both sides. The task of constructing universal upper approximators appears to be qualitatively different from that of lower approximators, and we are not aware of any construction achieving size better than the trivial one of $O(2^n/\sqrt{n})$ sufficient for exact computation. For approximators incurring two-sided error, our universal lower approximators of size $2^{n - \Omega_\varepsilon(\sqrt{n})}$ represent the current best upper bound. The strongest known lower bound for two-sided approximators is the $2^{\Omega(n/\log n)}$ lower bound of [18]; it would be interesting to find out whether this or the current $2^{n - \Omega_\varepsilon(\sqrt{n})}$ upper bound is closer to the truth.

As for the power of negations in the approximation of monotone functions, we believe that our results in Section 4 suggest a number of interesting avenues for further exploration. We suspect that the separations we presented can be improved, perhaps even to super-polynomial for DNF size and super-constant for DNF width. We remark that in addition to the complexity measures of DNF size and width, the quantitative difference between the accuracy of monotone versus general DNFs is also an aspect in which our separations can be strengthened. In other words, we may view our separations as instantiations of the following general template:

> *There exists a monotone function f and a value $\varepsilon = \varepsilon(n) > 0$ such that f can be ε-approximated by a DNF of size s (resp. width w), but any monotone function that $\varphi(\varepsilon)$-approximates f requires DNF size $\Psi(s)$ (resp. width $\Psi(w)$).*

In Theorems 3 and 4, φ is simply the identity function, but one can consider the possibility of stronger statements where $\varphi(\varepsilon) \gg \varepsilon$.

Beyond DNFs, one may ask quantitatively just how powerful negations can be in the approximation of monotone functions for many other classes of circuits. We conclude by restating an open problem, due to Kalai, on the possibility of strengthening the Okol'nishnikova–Ajtai–Gurevich theorem:

Open Problem 1 ([26]). *Is there a monotone function in AC^0 that cannot be approximated by monotone AC^0?*

References

1. Quine, W.V.O.: Two theorems about truth functions. Bol. Soc. Math. Mexicana 10, 64–70 (1954)
2. Razborov, A.A.: Lower bounds for the monotone complexity of some boolean functions. Soviet Mathematics Doklady 31, 354–357 (1985)
3. Okol'nishnikova, E.: On the influence of negations on the complexity of a realization of monotone Boolean functions by formulas of bounded depth. Metody Diskret. Analiz. 38, 74–80 (1982) (in Russian)
4. Ajtai, M., Grevich, Y.: Monotone versus positive. Journal of the ACM 34(4), 1004–1015 (1987)
5. Korshunov, A.D.: Monotone Boolean functions. Russian Math. Surveys (Uspekhi Mat. Nauk) 58(5), 929–1001 (2003)
6. Alon, N., Boppana, R.: The monotone circuit complexity of Boolean functions. Combinatorica 7, 1–22 (1987)
7. Karchmer, M., Wigderson, A.: Monotone circuits for connectivity require super-logarithmic depth. In: STOC 1988: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, pp. 539–550. ACM, New York (1988)
8. Tardos, É.: The gap between monotone and non-monotone circuit complexity is exponential. Combinatorica 8(1), 141–142 (1988)
9. Raz, R., Wigderson, A.: Monotone circuits for matching require linear depth. In: Proceedings of the 22nd ACM Symposium on Theory of Computing, pp. 287–292 (1990)

10. Karchmer, M., Raz, R., Wigderson, A.: Super-logarithmic depth lower bounds via direct sum in communication coplexity. In: Structure in Complexity Theory Conference, pp. 299–304 (1991)
11. Grigni, M., Sipser, M.: Monotone separation of logarithmic space from logarithmic depth. J. Comput. Syst. Sci. 50(3), 433–437 (1995)
12. Razborov, A., Rudich, S.: Natural proofs. Journal of Computer and System Sciences 55(1), 24–35 (1997)
13. Goldmann, M., Håstad, J.: Monotone circuits for connectivity have depth (log $n)^{2\text{-}o(1)}$. SIAM J. Comput. 27(5), 1283–1294 (1998)
14. Raz, R., McKenzie, P.: Separation of the monotone NC hierarchy. Combinatorica 19(3), 403–435 (1999)
15. Potechin, A.: Bounds on monotone switching networks for directed connectivity. In: Symposium on Foundations of Computer Science (FOCS), pp. 553–562 (2010)
16. Chan, S.M., Potechin, A.: Tight bounds for monotone switching networks via fourier analysis. In: Symposium on Theory of Computing (STOC), pp. 495–504 (2012)
17. Filmus, Y., Pitassi, T., Robere, R., Cook, S.A.: Average case lower bounds for monotone switching networks. In: Symposium on Foundations of Computer Science (FOCS) (2013)
18. O'Donnell, R.T., Wimmer, K.: Approximation by DNF: examples and counterexamples. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 195–206. Springer, Heidelberg (2007)
19. Blais, E., Tan, L.Y.: Approximating Boolean functions with depth-2 circuits. In: Proceedings of the 28th Annual IEEE Conference on Computational Complexity, pp. 74–85 (2013)
20. Bshouty, N., Tamon, C.: On the Fourier spectrum of monotone functions. Journal of the ACM 43(4), 747–770 (1996)
21. Jackson, J.: An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. Journal of Computer and System Sciences 55(3), 414–440 (1997)
22. Krause, M., Pudlák, P.: On the computational power of depth-2 circuits with threshold and modulo gates. Theoretical Computer Science 174(1–2), 137–156 (1997)
23. Friedgut, E.: Boolean functions with low average sensitivity depend on few coordinates. Combinatorica 18(1), 27–36 (1998)
24. Amano, K.: Tight bounds on the average sensitivity of k-CNF. Theory of Computing 7(1), 45–48 (2011)
25. Gopalan, P., Meka, R., Reingold, O.: Dnf sparsification and a faster deterministic counting algorithm. Computational Complexity 22(2), 275–310 (2013)
26. Kalai, G.: Noise stability and threshold circuits. Gil Kalai's, Combinatorics and more, blog (2010), http://gilkalai.wordpress.com/2010/02/10/noise-stability-and-threshold-circuits/

Internal DLA: Efficient Simulation of a Physical Growth Model

(Extended Abstract)

Karl Bringmann[1], Fabian Kuhn[2], Konstantinos Panagiotou[3], Ueli Peter[4], and Henning Thomas[4]

[1] Max Planck Institute for Informatics, Saarbrücken, Germany
[2] Department of Computer Science, University of Freiburg, Germany
[3] Department of Mathematics, LMU München, Germany
[4] Institute of Theoretical Computer Science, ETH Zurich, Zürich, Switzerland

Abstract. The internal diffusion limited aggregation (IDLA) process places n particles on the two dimensional integer grid. The first particle is placed on the origin; every subsequent particle starts at the origin and performs an unbiased random walk until it reaches an unoccupied position.

In this work we study the computational complexity of determining the subset that is generated after n particles have been placed. We develop the first algorithm that provably outperforms the naive step-by-step simulation of all particles. Particularly, our algorithm has a running time of $O(n \log^2 n)$ and a sublinear space requirement of $O(n^{1/2} \log n)$, both in expectation and with high probability. In contrast to some speedups proposed for similar models in the physics community, our algorithm samples from the *exact* distribution.

To simulate a single particle fast we have to develop techniques for combining multiple steps of a random walk to large jumps without hitting a forbidden set of grid points. These techniques might be of independent interest for speeding up other problems based on random walks.

1 Introduction

Internal diffusion limited aggregation (IDLA) is a random process that places n particles on the two-dimensional integer grid $\mathbb{Z}^2$. Let $A(i) \subset \mathbb{Z}^2$ denote the set of occupied grid points after placing i particles. The first particle is placed on the origin, i.e., $A(1) = \{(0,0)\}$. From there on, $A(i+1)$ is constructed from $A(i)$ by adding the first grid point in $\mathbb{Z}^2 \setminus A(i)$ that is reached by a random walk on $\mathbb{Z}^2$ starting at the origin.

Particle diffusion processes are of considerable significance in various branches of science. In fact, the IDLA process was introduced by Meakin and Deutch [8], who used it as a model to describe the dynamics of certain chemical and physical processes like corrosion or the melting of a solid around a source of heat. Since then, the study of the typical properties of $A(n)$, and most prominently

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 247–258, 2014.

its "shape," has been the topic of many works. In particular, numerical simulations in [8] indicated that the surface of $A(n)$ is typically extremely smooth such that the fluctuations from a perfect circle are only of logarithmic order. Proving this rigorously turned out to be a difficult and challenging mathematical problem, which was resolved only recently, after many attempts by several different authors (see e.g. [3,7,2,1]), by Jerison, Levine and Sheffield [6].

In the present paper, we try to understand IDLA from a computational perspective by giving an efficient algorithm for determining the set $A(n)$. This line of research is driven by the pursuit to get efficient algorithmic tools for coping with random walks and by the wish to speed up models from physics, so that one may perform larger experiments. Moreover, understanding such models from a computational perspective might add to their understanding in general.

Using the aforementioned results it is easy to see that a direct simulation of every individual step for determining $A(n)$ is likely to require a total time of $\Omega(n^2)$, i.e., time $\Omega(n)$ per particle. Indeed, since $A(n)$ typically resembles a perfect circle, it has a radius of order $n^{1/2}$. Moreover, the random walk of a particle can be viewed as a combination of two independent one-dimensional random walks, one along the horizontal and one along the vertical axis. Thus, if a particle is placed initially at the origin, one of these two random walks has to travel a distance of order $n^{1/2}$ in some direction in order to escape $A(n)$. A quadratic running time then follows immediately from the well-known fact that a one-dimensional random walk of length ℓ in expectation only deviates $\Theta(\ell^{1/2})$ hops from its initial position.

The computational complexity of determining $A(n)$ was studied by Moore and Machta [9]. Among other results they showed that the simulation of IDLA (given a string of random bits) is complete for the class $\mathcal{CC}$ (even in the case of one particle), which is the subset of $\mathcal{P}$ characterized by circuits that are composed of comparator gates only. Moreover in [4], Friedrich and Levine give an algorithm that samples $A(n)$. They do not provide an analysis of the complexity (and it seems a quite difficult task to do so), but their experiments indicate that it scales like $O(n^{3/2})$, while they inherently use space $\Omega(n)$.

In this paper we develop a time and space-efficient algorithm for determining the set $A(n)$. We present the first algorithm that provably improves upon the "naive" step-by-step simulation of the particles.

Theorem 1. *IDLA can be simulated in $O(n \log^2 n)$ time and $O(n^{1/2} \log n)$ space, both in expectation and with high probability*[1].

Our algorithm simulates all particles consecutively. It crucially uses that the shape of $A(n)$ is almost a perfect circle, as discussed above. Let the *in-circle* be the largest circle centred at the origin that contains only occupied grid points. As long as the current particle $n+1$ is within the in-circle of $A(n)$, the random walk will typically stay in $A(n)$ for many steps. Specifically, if the current distance of the particle to the in-circle of $A(n)$ is d, then typically in the next $\Theta(d^2)$ random walk steps the particle will stay in $A(n)$. We want to utilize this fact by combining

[1] With probability $1 - O(n^{-c})$ for a constant $c > 0$ that can be made arbitrary large.

many steps to a single *jump* of the particle, without simulating all of these steps explicitly. Building on this, we use drift analysis to show that typically $O(\log n)$ such jumps are sufficient to simulate one particle. Intuitively, such a combination of steps to a jump simply amounts to sampling the position of the particle after $T \approx d^2$ steps, which can be done by sampling two binomial random variables $\textsc{Bin}(T, \frac{1}{2})$. However, there is an obstacle to this simple intuition: Within $\Theta(d^2)$ steps we leave $A(n)$ with positive probability, so simply jumping to the outcome of $\Theta(d^2)$ steps necessarily introduces an error. As we want to design an *exact* sampling algorithm, we have to overcome this hurdle.

We present a general framework that utilizes jumps to efficiently simulate IDLA in Section 3. A particular jump procedure is discussed in Section 4.

2 Preliminaries

2.1 Notation

We denote by $\textsc{Bin}(n, p)$ a binomial distribution with parameters n and p and by $\log n$ the natural logarithm of n. For $z = (x, y) \in \mathbb{Z}^2$ we let $|z| = (x^2 + y^2)^{1/2}$ be its 2-norm. For $z \in \mathbb{Z}^2$ and $r > 0$ we define the ball with radius r around z as $B_z(r) := \{w \in \mathbb{Z}^2 \mid |z - w| \leq r\}$. We write $\Gamma(z)$ for the set of grid neighbors of $z \in \mathbb{Z}^2$, and for an arbitrary set $S \subseteq \mathbb{Z}^2$ we write ∂S for the set of all position that can be reached from S, i.e.

$$\partial S := \{z \in \mathbb{Z}^2 \setminus S \mid \Gamma(z) \cap S \neq \emptyset\} \quad \text{and} \quad \bar{S} := S \cup \partial S.$$

Whenever it is clear from the context, which particle we are simulating, we will write A for $A(i)$. For an IDLA shape A let $r_I = r_I(A)$ and $r_O = r_O(A)$ be its in- and outradius (rounded for technical reasons), i.e.,

$$r_I := \Big\lfloor \min_{x \in \mathbb{Z}^2 \setminus A} |x| \Big\rfloor \quad \text{and} \quad r_O := \Big\lfloor \max_{x \in A} |x| \Big\rfloor + 1.$$

Moreover, we say that $B_0(r_I)$ is the *in-* and $B_0(r_O)$ the *out-circle* of A.

2.2 The Shape of IDLA

Recently, Jerison, Levine and Sheffield proved a long open conjecture which stated that $A(n) = B_0(\sqrt{n/\pi}) \pm O(\log n)$ with high probability.

Theorem 2 (Theorem 1 in [6]). *For every $\gamma > 0$ exists a constant $\alpha = \alpha(\gamma) < \infty$ such that for sufficiently large r*

$$\Pr\left[B_0(r - \alpha \log r) \subset A(\lfloor \pi r^2 \rfloor) \subset B_0(r + \alpha \log r)\right] \geq 1 - r^{-\gamma}. \tag{1}$$

Additionally using $r_O \leq n$, this theorem implies that $r_O - r_I = O(\log n)$, both in expectation and with high probability.

2.3 Random Walks on $\mathbb{Z}$ and $\mathbb{Z}^2$

Let $z = z_0, z_1, z_2, \ldots$ be a random walk starting in $z \in \mathbb{Z}^2$. Here we always consider the standard random walk on $\mathbb{Z}^2$ that chooses each adjacent grid point with probability 1/4. We write $\mathrm{RW}_T(z) = z_T$ for the outcome of a random walk of length T starting in z and abbreviate $\mathrm{RW}_T(0) = \mathrm{RW}_T$. Note that $\mathrm{RW}_T(z) \sim z + \mathrm{RW}_T$.

We also reach each adjacent grid point with probability 1/4 by flipping two coins $c_1, c_2 \in \{1, -1\}$ and choosing the next position to be

$$z + c_1 \cdot (1/2, 1/2) + c_2 \cdot (-1/2, 1/2).$$

This yields the following reformulation of a 2-dimensional random walk as a linear combination of two independent 1-dimensional random walks. In particular, the following lemma allows us to quickly sample from RW_T (if one can sample binomial random variables quickly, we refer to the full version of this paper for a thorough discussion of this assumption).

Lemma 1. *Let $z \in \mathbb{Z}^2$ and $T \in \mathbb{N}$. Let S_T be the sum of T independent uniform $\{1, -1\}$ random variables, $S_T \sim 2\,\mathrm{Bin}(T, 1/2) - T$, and let X, Y be independent copies of S_T. Then*

$$\mathrm{RW}_T(z) \sim z + X \cdot (1/2, 1/2) + Y \cdot (1/2, -1/2).$$

Note that our random walks are "bipartite" in the sense that in even timesteps one can reach only the "even" positions of the grid $\{(x, y) \in \mathbb{Z}^2 \mid x + y \equiv 0 \pmod 2\}$, and similarly for odd timesteps. We write $z \equiv_T x$ if z can be reached from x by a walk of length $\ell \in \mathbb{N}$ with $\ell \equiv T \pmod 2$.

The outcome of a one-dimensional random walk of length T has standard deviation $\Theta(\sqrt{T})$. Intuitively, this implies that with at least constant probability the two-dimensional random walk RW_T is further than $\sqrt{T}$ away from the origin. Moreover, in any direction ξ the expected jump length is large.

Lemma 2. *For any $T \in \mathbb{N}$ we have $\Pr[|\mathrm{RW}_T| \geq \sqrt{T}] \geq \Omega(1)$.*

Moreover, let τ be a symmetric stopping time, i.e., for all $z \in \mathbb{Z}^2$ we have $\Pr[\mathrm{RW}_\tau = z] = \Pr[\mathrm{RW}_\tau = z']$ where z' is obtained from z by rotating it by 90°. Then for any $\xi \in \mathbb{R}^2$ with $|\xi| = 1$ we have

$$\mathbb{E}[|\xi \cdot \mathrm{RW}_\tau|] = \Omega(\Pr[|\mathrm{RW}_\tau| \geq \sqrt{T}] \cdot \sqrt{T}).$$

2.4 Drift Analysis

Let Ω be some state space, $Y_k \in \Omega$ ($k \in \mathbb{N}$) a stochastic process and $g : \Omega \to \mathbb{R}_{\geq 0}$ a function on Y_k. Let the hitting time τ be the smallest k such that $g(Y_k) = 0$. We say that $g(Y_k)$ has an additive drift of at least ε if for all $0 \leq k < \tau$

$$\mathbb{E}\left[g(Y_{k+1}) - g(Y_k) \mid Y_k\right] < -\varepsilon. \tag{2}$$

The following theorem bounds the expected hitting time by the inverse of the additive drift.

Theorem 3 ([5]). *In the situation of this section we have $\mathbb{E}[\tau] \leq \frac{g(Y_0)}{\varepsilon}$.*

3 A General Framework

The main idea of our algorithms is to combine many steps of a particle's random walk to a jump as long as the current particle $n+1$ is in the in-circle of $A = A(n)$. In this section, we first formalize the notion of a jump. After that we provide a framework that yields an IDLA simulation algorithm for any given jump procedure. Throughout this paper a *step* refers to a single step in a particle's random walk and a *jump* refers to several steps at once.

3.1 The Concept of a Jump

Ideally, a *jump* does multiple steps of a random walk at once to save the effort of simulating every single step. Jumps should be concatenable to form longer portions of a random walk. More formally, let $z = z_0, z_1, \ldots$ be a random walk starting in z and $\tau = \tau(A, z)$ a stopping time of this random walk. Then $z \mapsto z_\tau$ defines a jump procedure, and the concatenation of two such jumps is again the outcome of a random walk at a certain stopping time. This concatenation property allows us to add up jumps until we finally hit the boundary ∂A. A jump should make at least one single step of the random walk in order to have guaranteed progress, i.e., we require $\tau \geq 1$ (with probability 1). Moreover, in order to have a correct simulation of IDLA, jumps must stop at the latest when the random walk leaves A, since then the particle's simulation is complete. Additionally, all jump procedures considered in this paper are symmetric around z.

There are two important goals for the design of a jump procedure. First, the (expected) *runtime* to compute the outcome of a jump should be as small as possible. In particular, it should be faster than simulating the random walk step-by-step. Second, intuitively a jump should be the combination of as many single steps as possible. This can be formalized by requiring the *expected jumping distance* to be large. The following definition captures this concept of a jump.

Definition 1. *A* jump procedure *is a randomized algorithm J with input (an IDLA structure) $A \subset \mathbb{Z}^2$ and a point $z \in A$ and output $J(A, z) = z_\tau$, where $z = z_0, z_1, \ldots$ is a random walk and $\tau = \tau_J(A, z)$ is any stopping time. We require the jump to make at least one single step of a random walk and to stop at the latest when leaving A for the first time, i.e.,* $\Pr[1 \leq \tau \leq \tau_{\partial A}] = 1$,*where $\tau_{\partial A} = \min\{t \mid z_t \in \partial A\}$ is the hitting time of ∂A. Additionally, J shall be symmetric around z, i.e.,* $\Pr[J(A, z) = z + w] = \Pr[J(A, z) = z - w]$ *for all $w \in \mathbb{Z}^2$.*

We say that J has runtime bound $t_J = t_J(n)$ *if $J(A, z)$ can be computed in time t_J in expectation and with high probability (over the randomness of $A = A(n)$ and $\{z\} = A(n+1) \setminus A(n)$ and the internal randomness of J). Moreover, we define the* expected jumping distance *as*

$$\Delta_J(A, z) := \min_{|\xi|=1} \mathbb{E}[|\xi \cdot (J(A, z) - z)|].$$

When A is clear from the context we also write $J(z)$ for $J(A, z)$.

3.2 From Jumps to IDLA

Any jump procedure can be iterated to find the point where the random walk first leaves the IDLA structure A. Let $z_0 := (0,0)$ and $z_{i+1} := J(A, z_i)$, for every $i = 0, 1, 2, \dots$ as long as z_i is still in A. Moreover, let $\tau^* = \tau^*(J, A) := \min\{i \mid z_i \in \partial A\}$ and $J^* = J^*(A) := z_{\tau^*}$. Note that since J is a randomized algorithm, J^* and τ^* are random variables. Clearly, J^* is distributed exactly as the endpoint of an IDLA particle. This way, any jump procedure gives rise to a simulation algorithm for IDLA.

The following theorem gives an upper bound on the running time of an IDLA simulation with jump procedure J.

Theorem 4. *Let J be a jump procedure with runtime bound t_J. Let Δ_J be its expected jumping distance, $c_J > 0$ some constant, $B_I := B_0(r_I - c_J \log n)$, set*

$$\delta_J(A) := \max_{z \in B_I} \frac{r_O - |z|}{\Delta_J(A, z)}$$

and assume that for some $\bar{\delta}_J = \bar{\delta}_J(n)$ we have $\delta_J(A) \leq \bar{\delta}_J$ in expectation and with high probability (over the randomness of $A = A(n)$). Then we can construct an algorithm for simulating IDLA with runtime

$$O(n \cdot t_J \cdot \log n \cdot (\bar{\delta}_J^2 + \log n))$$

and space usage[2] *$O(n^{1/2} \log n)$, both in expectation and with high probability.*

To see that $O(n^{1/2} \log n)$ bits are sufficient (in expectation) to store $A(n)$, note that by Theorem 2 we have with high probability $B_0(\sqrt{n} - O(\log n)) \subseteq A(n) \subseteq B_0(\sqrt{n} + O(\log n))$, and $B_0(\sqrt{n} + O(\log n)) \setminus B_0(\sqrt{n} - O(\log n))$ contains $O(n^{1/2} \log n)$ grid cells, for each of which we can store whether it is occupied in 1 bit.

In Section 4 we present a jump procedure with $t_J = O(1)$ and $\bar{\delta}_J^2 = O(\log n)$, and thereby provide a proof for Theorem 1.

In order to run efficient IDLA simulations, we need a data structure that has the following properties.

Lemma 3. *We can construct a data structure for A that allows us to*

- *query r_I and r_O in $O(1)$ time,*
- *check $z \in A$ in $O(1)$ time, and*
- *add $z \in \mathbb{Z}^2$ to A.*

Adding the n particles of an IDLA simulation one-by-one to this data structure overall needs $O(n)$ time and $O(n^{1/2} \log n)$ space, both in expectation and with high probability.

In this extended abstract we omit the description of the data structure and the proof of Lemma 3. In the following section, we analyse the expected number of jumps that we need to simulate. Then, Theorem 4 is merely a consequence of Theorem 2, Lemma 3 and Lemma 4 below, and we therefore omit its proof.

[2] Not including the space used by the jump function.

3.3 Number of Jumps

To bound the expected runtime of the simulation of a particle using a jump procedure J, we only have to bound the hitting time τ^* of ∂A. The following lemma provides such a bound.

Lemma 4. *With the notation of Section 3.2 for any IDLA structure A and $k > 0$ we have* $\Pr[\tau^* \geq k\,(\delta_J^2(A)\log n + (r_O - r_I + \log n)^2)] \leq \exp(-\Omega(k))$.

In particular, we have $\mathbb{E}[\tau^*] \leq O\left(\delta_J^2(A)\log n + (r_O - r_I + \log n)^2\right)$.

Proof. Consider again the stochastic process $z_0 = (0,0)$, and $z_{k+1} = J(A, z_k)$ for $k > 0$. Set $\sigma := \sqrt{2}(r_O - r_I + c_J \log n)$. We analyze this process in phases. The process starts in phase 1 and changes to phase 2 the first time it reaches a position $z_k \notin B_I$. For the next σ^2 jumps the process stays in phase 2. After that it returns to phase 1, except if we are again outside B_I, then we directly start another phase 2. This repeats until we hit ∂A. For these phases we prove the following.

(1) Starting phase 1 anywhere in B_I, we stay in this phase for at most $O(\delta_J^2(A)\log n)$ jumps in expectation.
(2) Starting phase 2 anywhere outside B_I, the probability of hitting ∂A before the end of the phase is $\Omega(1)$.

Using Markov's inequality, (1) implies that after at most $O(\delta_J^2(A)\log n)$ jumps we leave phase 1 with probability $\Omega(1)$. Together with (2) we obtain that, wherever we start, within $O(\delta_J^2(A)\log n + \sigma^2)$ jumps we hit ∂A with probability $\Omega(1)$. Hence, within $O(k(\delta_J^2(A)\log n + \sigma^2))$ jumps we hit ∂A with probability $1 - \exp(-\Omega(k))$, yielding both expectation and concentration of the hitting time.

The proof of (2) follows by Lemma 2 and standard calculations and we therefore omit it in this extended abstract.

To show (1) we apply additive drift analysis to prove that the stochastic process $z_0, z_1, \ldots, z_\tau$ (for $z_0 \in B_I$ and $\tau := \min\{k \mid z_k \notin B_I\}$) has an expected hitting time as claimed. In order to apply Theorem 3 we need a suitable distance function $g : \mathbb{Z}^2 \to \mathbb{R}_{\geq 0}$. We let

$$g(z) := \begin{cases} \log(r_O + 2 - |z|), & z \in B_I \\ 0, & z \in \mathbb{Z}^2 \setminus B_I \end{cases}.$$

In the following we will show that g has an additive drift of $\min_{z\in B_I} \frac{(\Delta_J(A,z))^2}{2(r_O+2-|z|)^2}$ for all $0 \leq k < \tau$, i.e., for any $z_k \in B_I$

$$\mathbb{E}\left[g(z_{k+1}) - g(z_k) \mid z_k\right] \leq -\min_{z\in B_I} \frac{(\Delta_J(A,z))^2}{2(r_O + 2 - |z|)^2}\,. \tag{3}$$

Applying Theorem 3 together with $g(z) \leq O(\log n)$ then yields an expected hitting time of $\mathbb{Z}^2 \setminus B_I$ of $O(\delta_J^2(A)\log n)$.

Whenever $z_k \in A$ we know that $z_{k+1} \in \bar{A} \subseteq B_0(r_O + 1)$. In this case we can bound $g(z_{k+1}) \leq \log(r_O + 2 - |z_{k+1}|)$. To shorten notation we let $L(x) := \log(r_O +$

$2 - x)$ for any $x \in \mathbb{R}$ in the remainder of this proof. Hence, the expectation of $g(z_{k+1})$ conditioned on z_k, $z_k \in B_I$, is at most[3]

$$\sum_{x \in \mathbb{Z}^2} \Pr\left[z_{k+1} = x \mid z_k\right] \cdot L(|x|) \leq \sum_{x \in \mathbb{Z}^2} \Pr\left[z_{k+1} = x \mid z_k\right] \cdot L\left(x \frac{z_k}{|z_k|}\right),$$

since the length of the projection of x is bounded by $|x|$ in any direction. Using the transformation $y_x := x - z_k$ and the symmetry of jump procedures we can rewrite this as

$$\begin{aligned} &\sum_{x \in \mathbb{Z}^2} \Pr\left[z_{k+1} = x | z_k\right] \cdot L\left(|z_k| - y_x \frac{z_k}{|z_k|}\right) \\ &= \frac{1}{2} \sum_{x \in \mathbb{Z}^2} \Pr\left[z_{k+1} = x | z_k\right] \cdot \left(L\left(|z_k| - y_x \frac{z_k}{|z_z|}\right) + L\left(|z_k| + y_x \tfrac{z_k}{|z_k|}\right)\right), \end{aligned} \tag{4}$$

where $|y_x \frac{z_k}{|z_k|}| \leq r_O + 1 - |z_k|$ for all x with $\Pr\left[z_{k+1} = x | z_k\right] > 0$.

Now we use the following estimate that holds for any $a, b \in \mathbb{R}$ with $a > 0$ and $|b| \leq a$:

$$\log(a + b) + \log(a - b) \leq 2 \log(a) - \frac{b^2}{a^2}. \tag{5}$$

Combining (4) and (5) yields

$$\begin{aligned} \mathbb{E}\left[g(z_{k+1}) | z_k\right] &\leq \frac{1}{2} \sum_{x \in \mathbb{Z}^2} \Pr\left[z_{k+1} = x | z_k\right] \cdot \left(2L(|z_k|) - \frac{(y_x \cdot z_k / |z_k|)^2}{(r_O + 2 - |z_k|)^2}\right) \\ &= g(z_k) - \frac{\mathbb{E}\left[(y_{z_{k+1}} \cdot z_k / |z_k|)^2 \middle| z_k\right]}{2(r_O + 2 - |z_k|)^2} = g(z_k) - \frac{\mathbb{E}\left[\left|(z_{k+1} - z_k) \frac{z_k}{|z_k|}\right|^2 \middle| z_k\right]}{2(r_O + 2 - |z_k|)^2} \\ &\leq g(z_k) - \frac{\mathbb{E}\left[\left|(z_{k+1} - z_k) \frac{z_k}{|z_k|}\right| \middle| z_k\right]^2}{2(r_O + 2 - |z_k|)^2} \end{aligned} \tag{6}$$

where the last inequality follows from Jensen's inequality. Considering the definition of the expected jumping distance Δ_J (Definition 1) with $\xi = \frac{z_k}{|z_k|}$ we obtain

$$\mathbb{E}\left[g(z_{k+1}) | z_k\right] \leq g(z_k) - \frac{(\Delta_J(A, z_k))^2}{2(r_O + 2 - |z_k|)^2}$$

which proves the drift inequality (3) and, thus, the lemma. □

4 Long Jumps

Consider a particle at position $z \in B_I = B_0(r_I - c_J \log n)$ (for some sufficiently large constant $c_J > 0$) and consider the ball $S := B_z(\sigma)$ with midpoint z and

[3] Here we define the corresponding summand to be 0 whenever the log is undefined.

radius $\sigma := r_I - |z|$, so that S is contained in $B_0(r_I) \subseteq A$. Let $z_0, z_1, \ldots$ be a random walk starting in $z_0 = z$, let $\tau_{\partial S} := \min\{i \mid z_i \in \partial S\}$ be its hitting time of the boundary of S, and similarly let $\tau_{\partial A} := \min\{i \mid z_i \in \partial A\}$. Our procedure will directly jump to $J_{\text{long}}(z) := z_\tau$ with

$$\tau := \min\{\tau_{\partial S}, T\} \quad \text{and} \quad T := \left\lfloor \frac{\sigma^2}{c_J \ln(n/\mathrm{e})} \right\rfloor.$$

Whenever $z \notin B_I$, we simply make one step of the random walk, i.e., $\tau := 1$. This way we make sure that $\tau \geq 1$ (for all $z \in A$). Note that here we use $\tau_{\partial S}$ to ensure $\tau \leq \tau_{\partial S} \leq \tau_{\partial A}$, meaning that we stop at the latest when leaving A. Since τ is a stopping time and J_{long} is symmetric, this is a valid jump procedure according to Definition 1. It is not clear at first sight that J_{long} can be sampled efficiently for all $z \in B_I$. However, we present an algorithm in the next section and prove in Section 4.2 that its expected runtime is constant. Finally, we determine the expected jumping distance of J_{long} in Section 4.3. Overall, we obtain the following result, which together with Theorem 4 proves our main result.

Lemma 5. *The jump procedure J_{long} has runtime bound $t_{J_{\text{long}}} = O(1)$ and for any $z \in B_I$ an expected jumping distance of $\Delta_{J_{\text{long}}}(A, z) = \Omega(\sqrt{T}) = \Omega\left(\frac{r_I(A)-|z|}{\sqrt{\log n}}\right)$. Furthermore, it has a space usage of $O(1)$ memory cells (in expectation and with high probability).*

4.1 An Algorithm for Sampling Long Jumps

Observe that with high probability a random walk of length T starting in z does not leave S. Hence, the minimum of $\tau_{\partial S}$ and T is typically obtained at T. We will design an algorithm that samples the position of z_T (restricted to a certain subset) very efficiently. Additionally, we have to patch this approximate algorithm by a second (slow) algorithm that is executed only with small probability and that compensates for any mistakes we might make by sampling only z_T.

First consider Algorithm 1, which does not yet correctly sample a jump according to the distribution of $J_{\text{long}}(z)$. It simply draws a point $z' = \mathrm{RW}_T(z)$ (by sampling from a binomial random variable, see Lemma 1) and rejects as long as $z' \notin \frac{1}{2}S$ (where $\frac{1}{2}S$ is the ball with midpoint z and radius $\frac{1}{2}\sigma$).

Algorithm 1. Algorithm Long-Jump-Incomplete

```
repeat
    $z' := \mathrm{RW}_T(z)$
until $z' \in \frac{1}{2}S$
return $z'$.
```

For $w \in \mathbb{Z}^2$ let $P_J(w) := \Pr[J_{\text{long}}(z) = w]$ and denote the probability of Algorithm 1 to return w by $P_{\text{Alg}1}(w)$. To patch Algorithm 1 we choose a failure probability p_{fail} (to be fixed later). Then, with probability $1 - p_{\text{fail}}$ we run

Algorithm 1, but with probability p_{fail} we patch the algorithm by exhaustively computing the probabilities $P_J(w)$ and $P_{\text{Alg1}}(w)$ for all $w \in \bar{S}$ and returning $w \in \bar{S}$ with probability $P_{\text{rest}}(w)$, where

$$(1 - p_{\text{fail}}) \cdot P_{\text{Alg1}}(w) + p_{\text{fail}} \cdot P_{\text{rest}}(w) = P_J(w). \tag{7}$$

The above equation ensures that overall we draw $w \in \mathbb{Z}^2$ according to the right probability distribution P_J. The approach is summarized in Algorithm 2.

Algorithm 2. Algorithm Long-Jump-Complete

```
choose $p$ uniformly at random from $[0, 1]$.
if $p < p_{\text{fail}}$ then
    calculate $P_J(w)$ and $P_{\text{Alg1}}(w)$ for all $w \in \bar{S}$
    compute $P_{\text{rest}}(w)$ according to equation (7)
    return $w \in \bar{S}$ drawn according to the distribution $P_{\text{rest}}(w)$
else
    run Algorithm 1
end if
```

This algorithm is correct if p_{fail} can be chosen in such a way that P_{rest} is a probability distribution. The following lemma states for which values of p_{fail} this is the case.

Lemma 6. *The values $P_{\text{rest}}(w)$ for $w \in \bar{S}$ form a probability distribution if we choose $p_{\text{fail}} \geq 28\mathrm{e}^{c_J/2} n^{-\min\{c_J/8, 5c_J/16-1\}}$.*

In this extended abstract we omit the technical proof of Lemma 6. In the remainder of this section we analyze the runtime of our algorithm and prove a lower bound on the expected jump length.

4.2 Runtime of the Algorithm

In the fail compensation part of our algorithm we have to compute P_{Alg1} and P_J exactly. In this section we discuss how to do this efficiently, which yields a bound on the runtime of our algorithm.

Observe that for $P_{\text{RW}}(w) := \Pr[\text{RW}_T(z) = w]$ we have for all $w \in \frac{1}{2}S$ that

$$P_{\text{Alg1}}(w) = P_{\text{RW}}(w) / \sum_{w \in \frac{1}{2}S} P_{\text{RW}}(w).$$

This reduces the calculation of P_{Alg1} to the calculation of $P_{\text{RW}}(w)$ for all $w \in \frac{1}{2}S$. For $w \not\equiv_T z$ we have $P_{\text{RW}}(w) = 0$, so let $w \equiv_T z$. Then we can write $w = x \cdot (1/2, 1/2) + y \cdot (1/2, -1/2)$ with $x, y \in \mathbb{Z}$. With the notation of Lemma 1 we have

$$P_{\text{RW}}(w) = \Pr[X = x] \cdot \Pr[Y = y] = 2^{-T} \binom{T}{\frac{T+x}{2}} \cdot 2^{-T} \binom{T}{\frac{T+y}{2}}.$$

Note that this probability has denominator 4^T, so it can be stored using $O(T)$ bits. Moreover, as $\binom{T}{i}$ can be computed in $O(T)$ multiplications and divisions of a $O(T)$ bit number by a $O(\log T)$ bit number, we can calculate $P_{\mathrm{RW}}(W)$ in time $O(T^2 \log T)$. The total running time for calculating $P_{\mathrm{Alg}1}$ is therefore $O(\sigma^2 T^2 \log T)$ and the occupied space is $O(\sigma^2 T)$.

For computing P_J we use a simple iterative scheme. We recursively define X_w^t for $0 \leq t \leq T$ and $w \in \bar{S}$. For $t = 0$ we set

$$X_w^0 = \begin{cases} 1 & \text{if } w = z, \\ 0 & \text{otherwise,} \end{cases}$$

while for $t > 0$ we set

$$X_w^t = \begin{cases} \sum_{v \in \Gamma(w) \cap S} \frac{1}{4} X_v^{t-1} & \text{if } w \in S, \\ X_w^{t-1} + \sum_{v \in \Gamma(w) \cap S} \frac{1}{4} X_v^{t-1} & \text{if } w \in \partial S. \end{cases}$$

Observe that X_w^T is equal to $P_J(w)$ for every $w \in \bar{S}$, and each probability X_w^t can be stored using $O(T)$ bits. The total running time to calculate P_J is therefore $O(\sigma^2 T^2)$ and the space usage is $O(\sigma^2 T)$ bits.

As the ball S is completely filled with particles, we have $n \geq \sigma^2$. Using $T = \Theta(\sigma^2 / \log n)$ we get a runtime of $O(n^3)$ and a space usage of $O(n^2)$ for computing P_J and $P_{\mathrm{Alg}1}$.

Clearly, Algorithm 1 runs in expected constant time. Moreover, as the probability of $\mathrm{RW}_T \notin \frac{1}{2}S$ is small (smaller than p_{fail}, as chosen in the last section), it even runs in $O(1)$ time with high probability. In total, the expected runtime of our algorithm for sampling long jumps is $O(1 + p_{\mathrm{fail}} \cdot n^3)$, and the probability of having runtime larger than $O(1)$ is at most $O(p_{\mathrm{fail}})$. Hence, for sufficiently large constant c_J, so that Lemma 6 allows us to choose p_{fail} sufficiently small, we obtain a runtime of $t_{J_{\mathrm{long}}} = O(1)$, both in expectation and with high probability. This proves the first part of Lemma 5.

4.3 Expected Jumping Distance

In this section we analyze the expected jumping distance $\Delta_{J_{\mathrm{long}}}(z)$ of long jumps, proving the second part of Lemma 5. Recall that the expected jumping distance at $z \in B_I$ is defined as

$$\Delta_{J_{\mathrm{long}}}(A, z) = \min_{|\xi|=1} \mathbb{E}[|\xi^T (J_{\mathrm{long}}(A, z) - z)|].$$

Since the stopping time τ of J_{long} is symmetric, we can use the second part of Lemma 2 to obtain $\Delta_{J_{\mathrm{long}}}(A, z) = \Omega(\Pr[|J_{\mathrm{long}}(A, z) - z| \geq \sqrt{T}] \cdot \sqrt{T})$. Observe that we have $\Pr[|J_{\mathrm{long}}(A, z) - z| \geq \sqrt{T}] \geq \Pr[|\mathrm{RW}_T| \geq \sqrt{T}]$, where the inequality comes from some walks in $\mathrm{RW}_{\min\{\tau_{\partial S}, T\}}(z)$ ending prematurely (if $\tau_{\partial S} \leq T$). Together with the first part of Lemma 2, this shows $\Delta_{J_{\mathrm{long}}}(A, z) \geq \Omega(\sqrt{T}) = \Omega((r_I(A) - |z|)/\sqrt{\log n})$.

References

1. Asselah, A., Gaudilliere, A.: From logarithmic to subdiffusive polynomial fluctuations for internal DLA and related growth models, arXiv preprint arXiv:1009.2838 (2010)
2. Asselah, A., Gaudilliere, A.: Lower bounds on fluctuations for internal DLA. Probability Theory and Related Fields, 1–15 (2011)
3. Diaconis, P., Fulton, W.: A growth model, a game, an algebra, Lagrange inversion, and characteristic classes. Rend. Sem. Mat. Univ. Politec. Torino 49(1), 95–119 (1991)
4. Friedrich, T., Levine, L.: Fast simulation of large-scale growth models. In: Goldberg, L.A., Jansen, K., Ravi, R., Rolim, J.D.P. (eds.) RANDOM 2011 and APPROX 2011. LNCS, vol. 6845, pp. 555–566. Springer, Heidelberg (2011)
5. He, J., Yao, X.: A study of drift analysis for estimating computation time of evolutionary algorithms. Natural Computing 3(1), 21–35 (2004)
6. Jerison, D., Levine, L., Sheffield, S.: Logarithmic fluctuations for internal DLA. J. Amer. Math. Soc. 25(1), 271–301 (2012)
7. Lawler, G.F., Bramson, M., Griffeath, D.: Internal diffusion limited aggregation. The Annals of Probability, 2117–2140 (1992)
8. Meakin, P., Deutch, J.M.: The formation of surfaces by diffusion limited annihilation. The Journal of Chemical Physics 85, 2320 (1986)
9. Moore, C., Machta, J.: Internal diffusion-limited aggregation: Parallel algorithms and complexity. Journal of Statistical Physics 99, 661–690 (2000)

Lower Bounds for Approximate LDCs[⋆]

Jop Briët[1,⋆⋆], Zeev Dvir[2,⋆⋆⋆], Guangda Hu[3,⋆⋆⋆], and Shubhangi Saraf[4,†]

[1] Courant Institute of Mathematical Sciences, New York University
jop.briet@cims.nyu.edu

[2] Department of Computer Science and Department of Mathematics,
Princeton University
zeev.dvir@gmail.com

[3] Department of Computer Science, Princeton University
guangdah@cs.princeton.edu

[4] Department of Computer Science and Department of Mathematics,
Rutgers University
shubhangi.saraf@gmail.com

Abstract. We study an approximate version of q-query LDCs (Locally Decodable Codes) over the real numbers and prove lower bounds on the encoding length of such codes. A q-query (α, δ)-approximate LDC is a set V of n points in $\mathbb{R}^d$ so that, for each $i \in [d]$ there are $\Omega(\delta n)$ disjoint q-tuples $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_q)$ in V so that $\mathsf{span}(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_q)$ contains a unit vector whose i'th coordinate is at least α. We prove exponential lower bounds of the form $n \geq 2^{\Omega(\alpha\delta\sqrt{d})}$ for the case $q = 2$ and, in some cases, stronger bounds (exponential in d).

1 Introduction

Error Correcting Codes (ECCs) have always played an important part in the development of theoretical computer science. In particular, many of the foundational results of computational complexity rely in some way or another on constructions and analysis of ECCs (e.g., hardness of approximation, hardness-randomness tradeoffs). The study of ECCs from the perspective of complexity theorists sometimes has different a focus than the traditional information theory viewpoint. One such difference is the study of special kinds of codes that are useful for theory (i.e., for proving theorems such as the PCP theorem) but were not studied previously.

One such example are Locally-Decodable-Codes (LDCs) which were formally defined in the seminal work of Katz and Trevisan [KT00] (but were implicit in several prior works [BK95, Lip90, BF90]). These are codes that allow the

[⋆] The full version of this paper is available at http://arxiv.org/abs/1402.6952.

[⋆⋆] Supported by a Rubicon grant from the Netherlands Organisation for Scientific Research (NWO).

[⋆⋆⋆] Research partially supported by NSF grants CCF-0832797, CCF-1217416 and by the Sloan fellowship.

[†] Research supported by NSF grant CCF-1350572.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 259–270, 2014.

receiver of a (possibly corrupted) encoding $y = C(x) \in \{0,1\}^n$ of a message $x \in \{0,1\}^d$ to probabilistically decode w.h.p a single message bit x_i by reading only q positions in y (which might contain at most δn errors). We usually think of q as either a small constant or a very slow growing function of n and of δ as a constant.

The only case of LDCs which is mostly well understood is that of 2-query codes (it is easy to see that 1-query codes do not exist). The Hadamard code $C(x) = (\langle x, a\rangle)_{a\in\{0,1\}^d}$ is a 2-query code with exponential encoding length. In [GKST06, KdW04] it was shown that this is tight, that is, we always have $n \geq 2^{\Omega(\delta d)}$ for 2-query codes. For $q > 2$ there are huge gaps between the known lower and upper bounds. The best known lower bound is $n = \tilde{\Omega}(d^{1+1/(\lceil q/2\rceil-1)})$ for $q > 4$ [Woo07] and $n = \Omega(d^2)$ for $k = 3,4$ [KdW04, Woo12]. The best constructions for $q > 2$ are given by Matching-Vector codes, which were introduced by Yekhanin in [Yek08] and further developed in [Efr09, Rag07, KY09, IS10, CFL$^+$10, DGY11, BET10]. These codes have block-length of roughly $n \leq \exp\exp\left((\log d)^{O(\log\log q/\log q)}(\log\log d)\right)$.

One important sub-case of LDCs is that of *linear codes* (most known constructions are linear as far as we know). That is, the encoding is a linear mapping $C : \mathbb{F}^d \mapsto \mathbb{F}^n$ over some field $\mathbb{F}$. In this case, one can show that w.l.o.g. the decoding is linear as well. More formally, if we let $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n \in \mathbb{F}^d$ be the rows of the generating matrix of C (so that $C(\boldsymbol{x})_i = \langle \boldsymbol{x}, \boldsymbol{v}_i\rangle$) then we have that, for each $i \in [d]$ there must exist a matching M_i of at least $\Omega(\delta n)$ disjoint pairs $\boldsymbol{v}_{j_1}, \boldsymbol{v}_{j_2}$ that span $\boldsymbol{e}_i$ (the i'th standard basis vector). To locally decode x_i one can simply pick a random pair in the matching M_i and calculate:

$$x_i = \langle \boldsymbol{x}, \boldsymbol{e}_i\rangle = a\langle \boldsymbol{x}, \boldsymbol{v}_{j_1}\rangle + b\langle \boldsymbol{x}, \boldsymbol{v}_{j_2}\rangle$$

for some field elements a, b satisfying $a\boldsymbol{v}_{j_1} + b\boldsymbol{v}_{j_2} = \boldsymbol{e}_i$. In [DS05] it was shown that the lower bound of [GKST06] for binary linear codes can be extended to linear codes over any field and so, we know that the Hadamard code cannot be beaten even if we allow for a large alphabet.

In this work we consider a new notion of linear LDCs in which the underlying field is the real numbers and the decoding is 'approximate'. Building on the above characterization of linear codes, we will consider arrangements of points $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n \in \mathbb{R}^d$ in which, for every $i \in [d]$ there are many disjoint pairs that 'almost span' $\boldsymbol{e_i}$ in some concrete way (we give exact definitions below). Overall, our results are negative and show that, even if we allow a very loose notion of approximation, the encoding length is still exponential (either in $\sqrt{d}$ or in d, depending on the model). We prove several theorems for various settings of the parameters, using a wide variety of techniques.

Motivation and related works: Our motivation for studying this problem comes from several directions. Firstly, one could hope to use approximate codes in practice (if these had sufficiently good parameters). As long as the approximation parameter is not too large we could hope to recover some approximation of x_i using the two queries to the code (assuming x_i is some quantity we are interested in and we don't mind some small error). Another motivation comes from trying to understand 3-query codes. Here, even if we restrict our attention to real codes over $\mathbb{R}$, there is still an exponential gap between lower and upper bounds. In a recent work, [DSW14a], a subset of the current authors and Avi Wigderson proved an $n > d^{2+\epsilon}$ lower bound (for some positive ϵ) for a closely related notion of 2-query Locally Correctable Codes (LCCs) over $\mathbb{R}$, improving upon the known quadratic bound. Originally, the proof of [DSW14a] used a reduction from (exact) 3-LCCs over $\mathbb{R}$ to 2-query approximate LDCs (later, a different proof was found). This raises the possibility that, in the future, perhaps approximate codes will find more applications. We are also motivated by connections to well studied questions in combinatorial geometry. In [BDWY12, DSW14b] it was shown that proving lower bounds on LCCs is closely related to questions in the spirit of the Sylvester-Gallai theorem. Here, one tries to take local information about a point configuration (say, many collinear triples) and convert this information to a global bound on the dimension spanned by the points. We can view some of the theorems in this work in this spirit. Approximate versions of Sylveter-Gallai type theorems and LCCs were recently explored in [ADSW14].

1.1 Definitions and Results

We begin with some notations. A *q-matching* M in $[n]$ is defined to be a set of disjoint unordered q-tuples (i.e. disjoint subsets of size q) of $[n]$. We denote by $\boldsymbol{e}_i$ the i'th standard basis vector in $\mathbb{R}^d$. The standard inner product of two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ is given by $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ and the ℓ_2 norm of $\boldsymbol{x} \in \mathbb{R}^d$ is $\|\boldsymbol{x}\|_2 = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle}$.

Definition 1 (weight_i). *For a vector $\boldsymbol{u} \in \mathbb{R}^d$ we define $\mathsf{weight}_i(\boldsymbol{u}) = |\langle \boldsymbol{u}, \boldsymbol{e}_i \rangle| / \|\boldsymbol{u}\|_2$ (i.e., the absolute value of the i'th coordinate of the normalized vector $\boldsymbol{u}/\|\boldsymbol{u}\|_2$). Clearly we have $\sum_{i \in [d]} \mathsf{weight}_i(\boldsymbol{u})^2 = 1$.*

We now state our definition of approximate LDC.

Definition 2 (Approximate LDC). *Let d, n, q be positive integers and $\alpha, \delta \in [0, 1]$ real numbers. A q-query (α, δ)-approximate LDC is a pair (V, M) with*

1. *$V = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$ a multiset of vectors in $\mathbb{R}^d$. The parameter n is the* size *(or* block length*) of the code and the parameter d is the* dimension *(or* message length*) of the code.*
2. *$M = (M_1, \ldots, M_d)$ with each M_i being a q-matching in $[n]$ so that, if $\{j_1, \ldots, j_q\} \in M_i$, then there exists $\boldsymbol{u} \in \mathsf{span}\{\boldsymbol{v}_{j_1}, \ldots, \boldsymbol{v}_{j_q}\}$ with $\mathsf{weight}_i(\boldsymbol{u}) \geq \alpha$.*

The sizes of the matchings M_i must satisfy $|M_1| + |M_2| + \cdots + |M_d| \geq \delta dn$ and the parameter δ is called the density *of the code*[1]*.*

[1] The traditional definition would ask for each M_i to be of size at least δn but our definition is more general, which makes our (negative) results stronger.

Our first theorem gives an exponential bound on the block length of approximate 2-LDCs for any $\alpha > 0$. Notice that the bound gets worse as α approaches $1/\sqrt{d}$, at which point we cannot expect any lower bound to hold (since a single vector $\boldsymbol{u}$ can have $\mathsf{weight}_i(\boldsymbol{u}) \geq 1/\sqrt{d}$ for all $i \in [d]$).

Theorem 1. *[General lower bound] A 2-query (α, δ)-approximate LDC of size n and dimension d must satisfy $n \geq 2^{\Omega(\alpha\delta\sqrt{d})}$.*

We could hope to replace the exponential dependence on $\sqrt{d}$ with an exponential dependence on d (as is the case with exact 2-LDCs). In fact, we conjecture that a general bound of the form $n \geq \exp(\delta\alpha^2 d)$ should hold (the quadratic dependence on α is necessary to avoid hitting the $\alpha = 1/\sqrt{d}$ barrier). Currently, we are only able to prove this conjecture when α is sufficiently close to 1. This is stated in the next theorem.

Theorem 2. *[Lower bound for large α] Let $\alpha_0 = \sqrt{1 - 1/(4\pi^2)} \approx 0.987$. A 2-query (α, δ)-approximate LDC of size n, dimension d and $\alpha > \alpha_0$ must satisfy $n \geq 2^{\Omega(\delta d)}$, where the hidden constant in the $\Omega(\cdot)$ depends on $\alpha - \alpha_0$.*

There is another special case where we can get an exponential dependence on d instead of $\sqrt{d}$. It is a natural restriction of the general definition but it requires two new notions (that will be useful in their own right down the road). The first is that of a *simple* code (we will only care about 2-query codes).

Definition 3 (Simple Code). *Let (V, M) be a 2-query (α, δ)-approximate LDC. We say that (V, M) is a* simple *code if, for every $i \in [d]$ and $\{j_1, j_2\} \in M_i$ we have $\mathsf{weight}_i(\boldsymbol{v}_{j_2} - \boldsymbol{v}_{j_1}) \geq \alpha$.*

In other words, a simple code is an arrangement of points in $\mathbb{R}^d$ so that, for any $i \in [d]$ there are $\approx \delta n$ (on average) disjoint pairs of points that 'point' in a direction that has projection at least α on the i'th axis. An example of such an arrangement is the boolean cube $\{0,1\}^d \subset \mathbb{R}^d$ (all zero/one vectors), where M_i consists of all $n/2$ pairs that differ only in the i'th entry (so $\alpha = 1$).

Another feature of the hypercube is that all the distances between pairs in $M_1, \ldots, M_d$ are equal (they all equal one), motivating the following definition.

Definition 4 (c-bounded). *Let $c \geq 1$ and let (V, M) be a 2-query (α, δ)-approximate LDC. We say that (V, M) is c-*bounded *if, for every $i \in [d]$ and $\{j_1, j_2\} \in M_i$ we have $\|\boldsymbol{v}_{j_2} - \boldsymbol{v}_{j_1}\| \in [1, c]$.*

The fact that the hypercube is both c-bounded (with $c = 1$) and simple motivates the study of structures that satisfy these two conditions. In particular, we ask whether there exists a point arrangement in $\mathbb{R}^d$ which is 'roughly' like the hypercube but has far fewer than 2^d points. Here, the notion of 'roughly' is captured by allowing pairwise distance to be 'close' to 1 and the differences between adjacent vertices to be only somewhat axis parallel. The following theorem shows that such configurations do not exist (that is, you cannot beat the hypercube by much).

Theorem 3. *[Lower bound for simple c-bounded LDCs] A 2-query c-bounded simple (α, δ)-approximate LDC of size n and dimension d must satisfy $n \geq 2^{\Omega(\alpha^2\delta^2 d/(\log c)^2)}$.*

Finally, we consider arbitrary q-query approximate codes and observe that the lower bound proof of [KT00] can be made to work also for approximate LDCs (with some additional work). This gives the following theorem.

Theorem 4. *Let $q \geq 1$ be an integer constant. A q-query (α, δ)-approximate LDC of size n and dimension d must satisfy $n \geq \Omega((\alpha^2\delta^{1/q}d)^{\frac{q}{q-1}})$.*

1.2 Techniques

We briefly outline the techniques that appear in the proofs of our theorems.

Simple codes: An important ingredient in the proofs of Theorem 1 and 2 is a general reduction from any approximate 2-LDC to a simple code. The reduction follows by first normalizing the lengths of all vectors and then observing that if some linear combination $a\boldsymbol{v}_{j_1} + b\boldsymbol{v}_{j_2}$ has large weight_i then either one of the vectors $\boldsymbol{v}_{j_1}, \boldsymbol{v}_{j_2}$ has large weight_i or the coefficients a, b are close to $1, -1$. We can thus throw away all pairs in the matching M_i in which one of the vectors has large weight_i and get a simple code (we do not throw away too many pairs since each vector has only a few large coordinates). Since this reduction does not preserve c-boundedness, we can unfortunately not use it to argue that Theorem 3 works for non-simple c-bounded codes.

Proof of general bound: The proof of Theorem 1 (for simple codes w.l.o.g.) is via a recursive partitioning argument. In each step we pick a random $i \in [d]$ and partition V into two sets using a random shift of a hyperplane orthogonal to $\boldsymbol{e}_i$. We analyze the expected number of edges (pairs in some M_i) cut in this process and show that this is bounded by $O(\sqrt{d}/\alpha) \cdot \min\{|S|, |\bar{S}|\}$ with $S, \bar{S}$ representing the two parts of the cut. The same inequality holds also when partitioning any subset $V' \subset V$ and so we can proceed recursively and obtain a bound of $O((\sqrt{d}/\alpha)n \log_2 n)$ on the total number of edges. Since this number is at least δdn the theorem follows. This proof is inspired by the one appearing in [GKST06] for exact (simple) 2-LDCs.

Proof of bound for large α: Here we rely on a recent work of [KROW12] which gives a (randomized) tiling of $\mathbb{R}^d$ with cells that have volume 1 and surface area $O(\sqrt{d})$ (same as a sphere up to a constant). This result gives a randomized rounding algorithm that we can leverage towards 'rounding' our approximate code to an exact code (very roughly speaking) when α is large. This step is then combined with a random partitioning argument as in the proof of Theorem 1.

Proof for simple c-bounded codes: For this setting we use the LDC to construct a function F from $\mathbb{R}^d$ to the space of complex $n \times n$ matrices given by: $F(\boldsymbol{x}) = \left(e^{-i\langle \boldsymbol{x}, \boldsymbol{v}_s - \boldsymbol{v}_t\rangle}\right)_{s,t=1}^{n}$. The crux of the proof applies an inequality relating the trace norms of the first level (matrix) Fourier-coefficients of a matrix-valued function to its average trace norm. The crucial observation is that the norms of the first level Fourier coefficients of the above defined F can be lower bounded using the LDC property. The result then follows by combining this with the trivial upper bound on the average norm of F. This proof loosely follows an argument of [BARdW08] used for binary (non linear) LDCs and is inspired by work of [BNR12] linking LDCs to geometry of Banach spaces.

Organization. We describe our reduction from general to simple 2-query codes in Section 2. In Section 3 we prove the bound for general codes (Theorem 1). In Section 4 we prove the bound for α close to 1 (Theorem 2). The proofs of the lower bound for c-bounded codes (Theorem 3) and the lower bound for general q-query approximate codes (Theorem 4) are available in the full version of this paper.

2 Simple Codes

In this section we prove the following theorem showing that any 2-query approximate LDC can be transformed into a simple code with similar parameters.

Theorem 5. *If there exists a 2-query (α, δ)-approximate LDC of size n and dimension d, then, for any integer $k > 1/\alpha^2$, there exists a simple 2-query (α', δ')-approximate LDC of size n' and dimension d, where $\alpha' \geq \sqrt{\alpha^2 - 1/k}$, $\delta' \geq \delta - k/d$ and $n' \leq 2n$.*

The complete proof is given in the full version of this paper. We now give a short sketch of the main idea. Suppose that we have a pair of unit vectors $\boldsymbol{u}, \boldsymbol{w} \in \mathbb{R}^d$ with $\mathsf{weight}_i(a\boldsymbol{u}+b\boldsymbol{w}) \geq \alpha$. It will be convenient to think of α as being close to one (the proof will work for any α). So, after normalizing the coefficients a, b we have that the unit vector $v = a\boldsymbol{u} + b\boldsymbol{w}$ is close to $\boldsymbol{e}_i$. We separate into two cases. In the first case, both $\boldsymbol{u}$ and $\boldsymbol{w}$ are almost orthogonal to $\boldsymbol{e}_i$. In this case, we must have that $\boldsymbol{u} - \boldsymbol{w}$ 'points' in the direction of $\boldsymbol{e}_i$ (see diagrams in the complete proof) and so we don't really need the coefficients a, b. In the other case, at least one of $\boldsymbol{u}, \boldsymbol{w}$ have significant inner product with $\boldsymbol{e}_i$. Notice, however, that, for each fixed $\boldsymbol{u}$, this can only happen with a small number of $\boldsymbol{e}_i$'s when $i \in [n]$. These 'bad' pairs can be removed from the matchings without causing a big decrease in their average size.

It will be convenient to use the following corollary of Theorem 5 in which we set $k = \lceil 2/\alpha^2 \rceil$.

Corollary 1. *Suppose $d \geq 6/\alpha^2\delta$. If there exists a 2-query (α, δ)-approximate LDC of size n and dimension d, then there exists a simple 2-query (α', δ')-approximate LDC of size n' and dimension d, where $\alpha' \geq \alpha/\sqrt{2}$, $\delta' \geq \delta/2$ and $n' \leq 2n$.*

3 Lower Bound for General Simple Codes

We associate with a simple code $C = (V, M)$ a labeled graph G_C on vertex set V with edges given by all pairs in $M_1, \ldots, M_d$. We label each edge in M_i with the label i and allow for parallel edges (with different labels). We refer to the label of an edge e as the *direction* of the edge and denote it by $\mathrm{dir}(e) \in [d]$. The proof will follow by analyzing cuts in the graph G_C, which we assume contains at least δdn edges.

For $S \subseteq V$, let $\mathrm{Edge}(S)$ be the set of edges of G_C with both end points in S. We say that (S_1, S_2) is a *cut* if $S_1 \cup S_2 = S$ and $S_1 \cap S_2 = \emptyset$. The cut is *non-trivial* if $S_1, S_2 \neq \emptyset$. We use $\mathrm{Edge}(S_1, S_2)$ to denote the set of edges with one endpoint in S_1 and the other in S_2.

The next lemma of [GKST06, Appendix] relates the sizes of cuts in the graph with the total number of edges (the lemma holds for any graph). We include its proof in the Appendix for completeness.

Lemma 1. *Suppose that for every $S \subseteq V$ with $|S| \geq 2$, there exists a non-trivial cut (S_1, S_2) satisfying $|\mathrm{Edge}(S_1, S_2)| \leq c \cdot \min\{|S_1|, |S_2|\}$, then G_C has at most $\frac{c}{2}|V| \log_2 |V|$ edges.*

We now proceed to prove Theorem 1. We will show $n = 2^{\alpha\delta\sqrt{d}}$ for any (α, δ) simple code (the general case will follow using Corollary 1). This will follow by combining the following lemma and Lemma 1.

Lemma 2. *Let $C = (V, M)$ be an (α, δ) simple code and let G_C be the associated graph described above. Then, for any $S \subseteq V$ with $|S| \geq 2$, there exists a non-trivial cut (S_1, S_2) such that*

$$|\mathrm{Edge}(S_1, S_2)| \leq \frac{2\sqrt{d}}{\alpha} \cdot \min\{|S_1|, |S_2|\}.$$

Proof. If S contains no edges, an arbitrary cut will satisfy the requirement. We thus assume that S contains at least one edge. We now analyze the size of a random cut chosen in a specific way.

Assume all points in V are in a (d-dimensional) box of edge length L. We pick a random direction $i \in [d]$ and then pick a plane perpendicular to $\boldsymbol{e}_i$ at a random position intersecting the box. The plane cuts the box into two parts. We define S_1 to be the set of points in one part and S_2 to be the set of points in the other part (the probability of having a point on the hyperplane is zero). We analyze the edges in this cut (S_1, S_2). We say that an edge $e \in \mathrm{Edge}(S_1, S_2)$ is cut in the *right* direction if the plane is perpendicular to the direction of e, i.e. $\mathrm{dir}(e) = i$.

We consider a specific edge. Let $e_0 = \{\boldsymbol{v}_{j_1}, \boldsymbol{v}_{j_2}\}$ with $\{j_1, j_2\} \in M_{i_0}$ be an edge in direction $\mathrm{dir}(e_0) = i_0$ and denote $\boldsymbol{v}_{j_1} - \boldsymbol{v}_{j_2} = (u_1, u_2, \ldots, u_d)$. For each $i' \in [d]$ the probability that e_0 is cut by a plane perpendicular to $\boldsymbol{e}_{i'}$ is

$$\Pr[i = i'] \cdot \Pr[\text{the plane falls between } v_{j_1 i'} \text{ and } v_{j_2 i'}] = |u_{i'}|/(dL).$$

Therefore, by the definition of an approximate code ($|u_{i_0}| \geq \alpha \|\boldsymbol{v}_{j_1} - \boldsymbol{v}_{j_2}\|_2$) and the Cauchy-Schwarz inequality, edge e_0 is cut in the right direction with probability

$$\begin{aligned} |u_{i_0}|/(dL) &\geq \alpha \sqrt{u_1^2 + u_2^2 + \cdots + u_d^2} \Big/ (dL) \geq \frac{\alpha}{\sqrt{d}} \big(|u_1| + |u_2| + \cdots + |u_d|\big)/(dL) \\ &= \frac{\alpha}{\sqrt{d}} \Pr[e_0 \in \mathrm{Edge}(S_1, S_2)]. \end{aligned}$$

Since $\boldsymbol{v}_{j_1} - \boldsymbol{v}_{j_2}$ has at least one non-zero coordinate, $\Pr[e_0 \in \mathrm{Edge}(S_1, S_2)]$ must be strictly positive. It follows that edge e_0 is cut in the right direction with probability strictly greater than $\alpha \Pr[e_0 \in \mathrm{Edge}(S_1, S_2)]/(2\sqrt{d})$. Hence, the expected number of edges that are cut in the right direction is strictly greater than $\alpha \mathbb{E}\left[|\mathrm{Edge}(S_1, S_2)|\right]/(2\sqrt{d})$. There must therefore exist an $i \in [d]$, a plane perpendicular to $\boldsymbol{e}_i$ and a corresponding cut (S_1, S_2) which cuts strictly more than $\alpha|\mathrm{Edge}(S_1, S_2)|/(2\sqrt{d})$ in the right direction. Since this number is non-negative, there must be at least one edge cut in the right direction. This implies that S_1 and S_2 are not empty.

All edges cut in the right direction must have the same direction i. Hence, these edges are disjoint (they form a matching in V), implying that the total number of cut edges is at most $\min\{|S_1|, |S_2|\}$. It follows immediately that

$$|\mathrm{Edge}(S_1, S_2)| \leq \frac{2\sqrt{d}}{\alpha} \min\{|S_1|, |S_2|\}.$$

Therefore the cut (S_1, S_2) satisfies the requirement. □

Now using Lemma 1 we can conclude that $\delta dn \leq (2\sqrt{d}/\alpha)n \cdot \log_2 n$, which gives $n \geq 2^{\alpha\delta\sqrt{d}}$ as required. This completes the proof of Theorem 1. □

4 Lower Bound for Simple Codes with Large α

In this section we prove Theorem 3. By Theorem 5 it is enough to consider simple codes (the general case will follow by applying Theorem 5 with k a sufficiently large constant). We will use the definition and terminology of the graph G_C defined in the last section for simple codes. Hence, we think of pairs in M_i as edges in 'direction' $\mathrm{dir}(e) = i$. We define the *length* of an edge $e = \{\boldsymbol{v}_{j_1}, \boldsymbol{v}_{j_2}\}$ to be $\|\boldsymbol{v}_{j_1} - \boldsymbol{v}_{j_2}\|_2$.

We will use a recent result of [KROW12] concerning a partitioning (or tiling) of $\mathbb{R}^d$. Let $G = \{g\boldsymbol{z} \mid \boldsymbol{z} \in \mathbb{Z}^d\}$ be the set of grid points with grid distance $g \in \mathbb{R}^+$. Suppose we have a cell containing the origin and no other points of G. We can attempt to tile the space by taking all the shifts of this cell by all vectors in G. Clearly, one can do this using square tiles. However, it was an open problem to find the 'most efficient' way of tiling $\mathbb{R}^d$ (in some well defined geometric sense of 'efficient'). [KROW12] gives a randomized algorithm outputting the shape of the cell so that the entire space is fully covered and no two cells overlap (thus, it is a tiling) and each cell corresponds to one grid point. Let $C(\boldsymbol{x}) \in G$ ($\boldsymbol{x} \in \mathbb{R}^d$) denote the grid point in the cell containing $\boldsymbol{x}$ (so we can think of $C(\boldsymbol{x})$ as a 'rounding' of $\boldsymbol{x}$). [KROW12] proved the following[2]:

Theorem 6 ([KROW12]). *There is a randomized algorithm partitioning the whole space $\mathbb{R}^d$ into cells such that*

1. *For every $\boldsymbol{x} \in \mathbb{R}^d$ and $\boldsymbol{s} \in G$, $C(\boldsymbol{x} + \boldsymbol{s}) = C(\boldsymbol{x}) + \boldsymbol{s}$.*
2. *For every two points $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, $\Pr[C(\boldsymbol{x}) \neq C(\boldsymbol{y})] \leq 2\pi\|\boldsymbol{y} - \boldsymbol{x}\|_2/g$.*

Let $\epsilon \in (0, 1)$ and $t \in \mathbb{Z}^+$ be two parameters to be determined later. We partition $\mathbb{R}^+$ into sets $\mathbb{R}^+ = I_0 \cup I_1 \cup \cdots \cup I_{t-1}$, where

$$I_j = \bigcup_{k \in \mathbb{Z}} \left[(1+\epsilon)^{kt+j}, (1+\epsilon)^{kt+j+1}\right).$$

For $I \subseteq \mathbb{R}^+$, we say an edge is *contained* in I if its length falls in I. Without loss of generality we assume I_0 is the one among $\{I_0, I_1, \ldots, I_{t-1}\}$ that contains the most edges. We remove all edges not contained in I_0. The density δ is decreased by a factor of at most t.

Recall that $I_0 = \bigcup_{k \in \mathbb{Z}} \left[(1+\epsilon)^{kt}, (1+\epsilon)^{kt+1}\right)$. We say that the *level* of an edge is k if it is contained in $\left[(1+\epsilon)^{kt}, (1+\epsilon)^{kt+1}\right)$. For an edge e, we use $\mathrm{lev}(e)$ to denote its level. Let $k_{\min}$ and $k_{\max}$ to be the minimum level and the maximum level of all edges respectively.

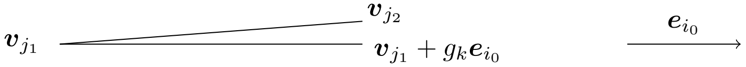
For every integer $k \in [k_{\min}, k_{\max}]$ we use Theorem 6 to generate an (independent) random partition with grid distance

$$g_k = \frac{(1+\epsilon)^{kt} + (1+\epsilon)^{kt+1}}{2\alpha} = \frac{(2+\epsilon)(1+\epsilon)^{kt}}{2\alpha},$$

and use $C_k(\boldsymbol{x})$ to denote the corresponding rounding function.

Consider an edge $e = \{\boldsymbol{v}_{j_1}, \boldsymbol{v}_{j_2}\}$ and say $\mathrm{dir}(e) = i_0$. We assume $\langle \boldsymbol{v}_{j_2} - \boldsymbol{v}_{j_1}, \boldsymbol{e}_{i_0} \rangle > 0$. (Otherwise we switch the order of $\boldsymbol{v}_{j_1}$ and $\boldsymbol{v}_{j_2}$.) We say the edge is *good* if the following properties are satisfied:

1. For $k = \mathrm{lev}(e)$, $C_k(\boldsymbol{v}_{j_1} + g_k \boldsymbol{e}_{i_0}) = C_k(\boldsymbol{v}_{j_2})$. Since $C_k(\boldsymbol{v}_{j_1} + g_k \boldsymbol{e}_{i_0}) = C_k(\boldsymbol{v}_{j_1}) + g_k \boldsymbol{e}_{i_0}$, this means that the two cells containing $\boldsymbol{v}_{j_1}$ and $\boldsymbol{v}_{j_2}$ are adjacent along the direction $\boldsymbol{e}_{i_0}$.
2. For $k > \mathrm{lev}(e)$, $C_k(\boldsymbol{v}_{j_1}) = C_k(\boldsymbol{v}_{j_2})$. In other words, the two ends are in the same cell.



Claim. Every edge is good with probability at least

$$1 - \left(2\pi \sqrt{1 - \alpha^2 + \left(\frac{\alpha\epsilon}{2+\epsilon}\right)^2} + \frac{4\pi\alpha(1+\epsilon)}{(2+\epsilon)\left((1+\epsilon)^t - 1\right)} \right).$$

[2] [KROW12] only considered $g = 1$ but the general result follows by simple scaling.

Proof. We consider the edge $e = \{\boldsymbol{v}_{j_1}, \boldsymbol{v}_{j_2}\}$ with direction i_0, and assume $\langle \boldsymbol{v}_{j_2} - \boldsymbol{v}_{j_1}, \boldsymbol{e}_{i_0} \rangle > 0$. Then $\langle \boldsymbol{v}_{j_2} - \boldsymbol{v}_{j_1}, \boldsymbol{e}_{i_0} \rangle \geq \alpha \|\boldsymbol{v}_{j_2} - \boldsymbol{v}_{j_1}\|_2$ and

$$\frac{\|\boldsymbol{v}_{j_2} - \boldsymbol{v}_{j_1}\|_2}{g_{\mathrm{lev}(e)}} \in \left[\frac{2\alpha}{2+\epsilon}, \frac{2\alpha(1+\epsilon)}{2+\epsilon}\right) = \left[\alpha - \frac{\alpha\epsilon}{2+\epsilon}, \alpha + \frac{\alpha\epsilon}{2+\epsilon}\right),$$

We consider the probability that e is not a good edge.

1. For $k = \mathrm{lev}(e)$, we have

$$\begin{aligned}\Pr\left[C_k(\boldsymbol{v}_{j_1} + g_k \boldsymbol{e}_{i_0}) \neq C_k(\boldsymbol{v}_{j_2})\right] &\leq 2\pi \|\boldsymbol{v}_{j_2} - (\boldsymbol{v}_{j_1} + g_k \boldsymbol{e}_{i_0})\|_2 / g_k \\ &\leq 2\pi/g_k \cdot \sqrt{\|\boldsymbol{v}_{j_2} - \boldsymbol{v}_{j_1}\|_2^2 + g_k^2 - 2\alpha g_k \|\boldsymbol{v}_{j_2} - \boldsymbol{v}_{j_1}\|_2} \\ &= 2\pi \sqrt{\left(\|\boldsymbol{v}_{j_2} - \boldsymbol{v}_{j_1}\|_2/g_k - \alpha\right)^2 + 1 - \alpha^2} \\ &\leq 2\pi \sqrt{1 - \alpha^2 + \left(\frac{\alpha\epsilon}{2+\epsilon}\right)^2}.\end{aligned}$$

2. For $k > \mathrm{lev}(e)$, we have

$$\begin{aligned}\Pr\left[C_k(\boldsymbol{v}_{j_1}) \neq C_k(\boldsymbol{v}_{j_2})\right] &\leq 2\pi \|\boldsymbol{v}_{j_2} - \boldsymbol{v}_{j_1}\|_2 / g_k \\ &\leq 2\pi \cdot \frac{2\alpha(1+\epsilon)}{2+\epsilon} \cdot \frac{g_{\mathrm{lev}(e)}}{g_k} \\ &= \frac{4\pi\alpha(1+\epsilon)}{2+\epsilon} \cdot \frac{1}{(1+\epsilon)^{(k-\mathrm{lev}(e))t}}.\end{aligned}$$

By union bound, the probability that e is not a good edge is at most

$$\begin{aligned}&2\pi \sqrt{1 - \alpha^2 + \left(\frac{\alpha\epsilon}{2+\epsilon}\right)^2} + \sum_{k=\mathrm{lev}(e)+1}^{k_{\max}} \left(\frac{4\pi\alpha(1+\epsilon)}{2+\epsilon} \cdot \frac{1}{(1+\epsilon)^{(k-\mathrm{lev}(e))t}}\right) \\ &< 2\pi \sqrt{1 - \alpha^2 + \left(\frac{\alpha\epsilon}{2+\epsilon}\right)^2} + \frac{4\pi\alpha(1+\epsilon)}{(2+\epsilon)\left((1+\epsilon)^t - 1\right)}.\end{aligned}$$

Thus the claim is proved. □

For any $\alpha > \sqrt{1 - 1/(4\pi^2)}$, we can always pick ϵ sufficiently small and t sufficiently large so that each edge is good with positive probability. For example, if $\alpha = 0.99$, we can take $\epsilon = 0.01$ and $t = 500$, in which case each edge is good with probability at least 0.069. For simplicity, we use $O(\cdot)$ and $\Omega(\cdot)$ to suppress the exact values of constants α, ϵ and t. The above claim tells us every edge is good with probability $\Omega(1)$. By a simple expectation argument, there exists a series of space partitions (for every $k \in [k_{\min}, k_{\max}]$) such that $\Omega(1)$ fraction of all edges are good. We fix these partitions and remove all edges that are not good. In the remaining code the density is $\Omega(\delta)$.

Next, we prove the lower bound $n = 2^{\Omega(\delta d)}$. This follows immediately from the following lemma and Lemma 1.

Lemma 3. *For any $S \subseteq V$ with $|S| \geq 2$, there exists a non-trivial cut (S_1, S_2) such that $|\operatorname{Edge}(S_1, S_2)| \leq \min\{|S_1|, |S_2|\}$.*

Proof. If S contains no edges, an arbitrary partition will satisfy the requirement. Otherwise, we consider the edges in S and pick an edge with the maximum level. Say this edge is $e = \{\boldsymbol{v}_{j_1}, \boldsymbol{v}_{j_2}\}$, and $\operatorname{dir}(e) = i_0$, $\operatorname{lev}(e) = k$. We assume $\langle \boldsymbol{v}_{j_2} - \boldsymbol{v}_{j_1}, \boldsymbol{e}_{i_0} \rangle > 0$. Then since this edge is good, $C_k(\boldsymbol{v}_{j_1})$ and $C_k(\boldsymbol{v}_{j_2})$ are adjacent grid points,

$$C_k(\boldsymbol{v}_{j_1}) + g_k \boldsymbol{e}_{i_0} = C_k(\boldsymbol{v}_{j_1} + g_k \boldsymbol{e}_{i_0}) = C_k(\boldsymbol{v}_{j_2}).$$

For any point $\boldsymbol{v} \in \mathbb{R}^d$ and $i \in [d]$, we use $C_k(\boldsymbol{v})_i$ to denote the i'th coordinate of $C_k(\boldsymbol{v})$. Let $h = \left[C_k(\boldsymbol{v}_{j_1})_{i_0} + C_k(\boldsymbol{v}_{j_2})_{i_0}\right]/2$. We define S_1 and S_2 as follows.

$$S_1 = \{\boldsymbol{v} \in S \mid C_k(\boldsymbol{v})_{i_0} < h\},$$
$$S_2 = \{\boldsymbol{v} \in S \mid C_k(\boldsymbol{v})_{i_0} > h\}.$$

We can see that S_1 and S_2 are not empty because $\boldsymbol{v}_{j_1} \in S_1$ and $\boldsymbol{v}_{j_2} \in S_2$. There is no point $\boldsymbol{v}$ satisfying $C_k(\boldsymbol{v})_{i_0} = h$, because $C_k(\boldsymbol{v})$ is a grid point and h is not a multiple of g_k. Hence (S_1, S_2) is a non-trivial cut of S.

We consider the edges in $\operatorname{Edge}(S_1, S_2)$, and show that every edge in $\operatorname{Edge}(S_1, S_2)$ must have direction i_0. Assume this is not true, and let $e' = \{\boldsymbol{v}'_{j_1}, \boldsymbol{v}'_{j_2}\}$ be such an edge. Say $\operatorname{dir}(e') = i'$ $(i' \neq i_0)$. There are two cases.

1. $\operatorname{lev}(e') = k$. By the first requirement in the definition of good edges,

$$C_k(\boldsymbol{v}'_{j_1}) + g_k \boldsymbol{e}_{i'} = C_k(\boldsymbol{v}'_{j_1} + g_k \boldsymbol{e}_{i'}) = C_k(\boldsymbol{v}'_{j_2}).$$

2. $\operatorname{lev}(e') < k$. By the second requirement in the definition of good edges,

$$C_k(\boldsymbol{v}'_{j_1}) = C_k(\boldsymbol{v}'_{j_2}).$$

In both cases we have $C_k(\boldsymbol{v}'_{j_1})_{i_0} = C_k(\boldsymbol{v}'_{j_2})_{i_0}$. Hence the edge $e' \notin \operatorname{Edge}(S_1, S_2)$.

Therefore all edges in $\operatorname{Edge}(S_1, S_2)$ have direction i_0. Since the edges of the same direction are disjoint, we have $|\operatorname{Edge}(S_1, S_2)| \leq \min\{|S_1|, |S_2|\}$. □

References

[ADSW14] Ai, A., Dvir, Z., Saraf, S., Wigderson, A.: Sylvester-Gallai type theorems for approximate collinearity. Forum of Mathematics, Sigma 4 (2014)

[BARdW08] Ben-Aroya, A., Regev, O., de Wolf, R.: A hypercontractive inequality for matrix-valued functions with applications to quantum computing and LDCs. In: FOCS 2008, pp. 477–486 (2008)

[BDWY12] Barak, B., Dvir, Z., Wigderson, A., Yehudayoff, A.: Fractional Sylvester-Gallai theorems. Proceedings of the National Academy of Sciences (2012)

[BET10] Ben-Aroya, A., Efremenko, K., Ta-Shma, A.: Local list decoding with a constant number of queries. In: FOCS 2010, pp. 715–722 (2010)

[BF90] Beaver, D., Feigenbaum, J.: Hiding instances in multioracle queries. In: Choffrut, C., Lengauer, T. (eds.) STACS 1990. LNCS, vol. 415, pp. 37–48. Springer, Heidelberg (1990)

[BK95] Blum, M., Kannan, S.: Designing programs that check their work. J. ACM 42(1), 269–291 (1995)

[BNR12] Briët, J., Naor, A., Regev, O.: Locally decodable codes and the failure of cotype for projective tensor products. Electronic Research Announcements in Mathematical Sciences (ERA-MS) 19, 120–130 (2012)

[CFL$^+$10] Chee, Y., Feng, T., Ling, S., Wang, H., Zhang, L.: Query-efficient locally decodable codes of subexponential length. ECCC, TR10-173 (2010)

[DGY11] Dvir, Z., Gopalan, P., Yekhanin, S.: Matching vector codes. SIAM J. Comput. 40(4), 1154–1178 (2011)

[DS05] Dvir, Z., Shpilka, A.: Locally decodable codes with 2 queries and polynomial identity testing for depth 3 circuits. In: STOC 2005, pp. 592–601 (2005)

[DSW14a] Dvir, Z., Saraf, S., Wigderson, A.: Breaking the quadratic barrier for 3-LCCs over the reals. In: STOC 2014 (2014)

[DSW14b] Dvir, Z., Saraf, S., Wigderson, A.: Improved rank bounds for design matrices and a new proof of Kelly's theorem. Forum of Mathematics, Sigma 4 (2014)

[Efr09] Efremenko, K.: 3-query locally decodable codes of subexponential length. In: STOC 2009, pp. 39–44 (2009)

[GKST06] Goldreich, O., Karloff, H., Schulman, L.J., Trevisan, L.: Lower bounds for linear locally decodable codes and private information retrieval. Computational Complexity 15(3), 263–296 (2006)

[IS10] Itoh, T., Suzuki, Y.: Improved constructions for query-efficient locally decodable codes of subexponential length. IEICE Transactions on Information and Systems E93-D(2), 263–270 (2010)

[KdW04] Kerenidis, I., de Wolf, R.: Exponential lower bound for 2-query locally decodable codes via a quantum argument. Journal of Computer and System Sciences 69(3), 395–420 (2004)

[KROW12] Kindler, G., Rao, A., O'Donnell, R., Wigdersons, A.: Spherical cubes: optimal foams from computational hardness amplification. Commun. ACM 55(10), 90–97 (2012)

[KT00] Katz, J., Trevisan, L.: On the efficiency of local decoding procedures for error-correcting codes. In: STOC 2000, pp. 80–86 (2000)

[KY09] Kedlaya, K.S., Yekhanin, S.: Locally decodable codes from nice subsets of finite fields and prime factors of Mersenne numbers. SIAM J. Comput. 38(5), 1952–1969 (2009)

[Lip90] Lipton, R.J.: Efficient checking of computations. In: Choffrut, C., Lengauer, T. (eds.) STACS 1990. LNCS, vol. 415, pp. 207–215. Springer, Heidelberg (1990)

[Rag07] Raghavendra, P.: A note on Yekhanin's locally decodable codes. ECCC, TR07-016 (2007)

[Woo07] Woodruff, D.P.: New lower bounds for general locally decodable codes. ECCC, TR07-006 (2007)

[Woo12] Woodruff, D.P.: A quadratic lower bound for three-query linear locally decodable codes over any field. Journal of Computer Science and Technology 27(4), 678–686 (2012)

[Yek08] Yekhanin, S.: Towards 3-query locally decodable codes of subexponential length. Journal of the ACM 55(1), 1–16 (2008)

Holographic Algorithms Beyond Matchgates*

Jin-Yi Cai, Heng Guo, and Tyson Williams

University of Wisconsin–Madison, Madison, WI, USA
{jyc,hguo,tdw}@cs.wisc.edu

Abstract. Holographic algorithms based on matchgates were introduced by Valiant. These algorithms run in polynomial-time and are intrinsically for planar problems. We introduce two new families of holographic algorithms, which work over general, i.e., not necessarily planar, graphs. The two underlying families of constraint functions are of the *affine* and *product* types. These play the role of Kasteleyn's algorithm for counting planar perfect matchings. The new algorithms are obtained by transforming a problem to one of these two families by holographic reductions. We present a polynomial-time algorithm to decide if a given counting problem has a holographic algorithm using these constraint families. When the constraints are symmetric, we give a polynomial-time decision procedure in the size of the succinct presentation of symmetric constraint functions. This procedure shows that the recent dichotomy theorem for Holant problems with symmetric constraints is polynomial-time decidable.

1 Introduction

Recently a number of complexity dichotomy theorems have been obtained for counting problems. Typically, such dichotomy theorems assert that a vast majority of problems expressible within the framework are #P-hard, however an intricate subset manages to escape this fate. They exhibit a great deal of mathematical structure, which leads to a polynomial time algorithm. In recent dichotomy theorems, a pattern has emerged [14,19,21,15,34,23,11,32]. Some of the tractable cases are expressible as "those problems for which there exists a *holographic algorithm*." However, this understanding has been largely restricted to problems where the local constraint functions are symmetric over the Boolean domain. In order to gain a better understanding, we must determine the full extent of holographic algorithms, beyond the symmetric constraints.

Holographic algorithms were first introduced by Valiant [44,43]. They are applicable for any problem that can be expressed as the contraction of a tensor network. Valiant's algorithms have two main ingredients. The first ingredient is to encode computation in planar graphs using matchgates [42,41,9,17,10]. The result of the computation is then obtained by counting the number of perfect matchings in a related planar graph, which can be done in polynomial time by Kasteleyn's (a.k.a. the FKT) algorithm [35,40,36]. The second ingredient is a

* Full version with proofs available at [12].

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 271–282, 2014.

holographic reduction, which is achieved by a choice of linear basis vectors. The computation can be carried out in any basis since the output of the computation is independent of the basis.

In this paper, we introduce two new families of holographic algorithms. These algorithms holographically reduce to problems expressible by either the *affine* type or the *product* type of constraint functions. Both types of problems are tractable over general (i.e. not necessarily planar) graphs [25], so the holographic algorithms are all polynomial time algorithms and work over general graphs. We present a polynomial time algorithm to decide if a given counting problem has a holographic algorithm over general graphs using the affine or product-type constraint functions. Our algorithm also finds a holographic algorithm when one exists. To formally state this result, we briefly introduce some notation.

The counting problems we consider are those expressible as a Holant problem [24,22,20,25]. A Holant problem is defined by a set $\mathcal{F}$ of constraint functions, which we call signatures, and is denoted by Holant($\mathcal{F}$). An instance to Holant($\mathcal{F}$) is a tuple $\Omega = (G, \mathcal{F}, \pi)$ called a signature grid, where $G = (V, E)$ is a graph and π labels each vertex $v \in V$ and its incident edges with some $f_v \in \mathcal{F}$ and its input variables. Here f_v maps $\{0,1\}^{\deg(v)}$ to $\mathbb{C}$. We consider all possible 0-1 edge assignments. An assignment σ to the edges E gives an evaluation $\prod_{v \in V} f_v(\sigma|_{E(v)})$, where $E(v)$ denotes the incident edges of v and $\sigma|_{E(v)}$ denotes the restriction of σ to $E(v)$. The counting problem on the instance Ω is to compute

$$\mathrm{Holant}_{\Omega} = \sum_{\sigma : E \to \{0,1\}} \prod_{v \in V} f_v\left(\sigma|_{E(v)}\right). \tag{1}$$

For example, consider the problem of counting Perfect Matching on G. This problem corresponds to attaching the Exact-One function at every vertex of G. The Exact-One function is an example of a symmetric signature, which are functions that only depend on the Hamming weight of the input. We denote a symmetric signature by $f = [f_0, f_1, \ldots, f_n]$ where f_w is the value of f on inputs of Hamming weight w. For example, $[0, 1, 0, 0]$ is the Exact-One function on three bits. The output is 1 if and only if the input is 001, 010, or 100, and the output is 0 otherwise.

Holant problems contain both counting constraint satisfaction problems and counting graph homomorphisms as special cases. All three classes of problems have received considerable attention, which has resulted in a number of dichotomy theorems (see [38,33,28,2,27,5,30,8] and [4,3,26,1,25,7,13,29,31,14,6]). Despite this success with #CSP and graph homomorphisms, the case with Holant problems is more difficult. A recent dichotomy theorem for Holant problems with symmetric signatures was obtained in [11]. But the general (i.e. not necessarily symmetric) case has a richer and more intricate structure. The same dichotomy for general signatures remains open. Our first main result makes a solid step forward in understanding holographic algorithms based on affine and product-type signatures in this more difficult setting.

Theorem 1. *There is a polynomial time algorithm to decide, given a finite set of signatures $\mathcal{F}$, whether* $\operatorname{Holant}(\mathcal{F})$ *admits a holographic algorithm based on affine or product-type signatures.*

These holographic algorithms for $\operatorname{Holant}(\mathcal{F})$ are all polynomial time in the size of the problem input Ω. The polynomial time decision algorithm of Theorem 1 is on another level; it decides based on any specific set of signatures $\mathcal{F}$ whether the counting problem $\operatorname{Holant}(\mathcal{F})$ defined by $\mathcal{F}$ has such a holographic algorithm.

However, symmetric signatures are an important special case. Because symmetric signatures can be presented exponentially more succinctly, we would like the decision algorithm to be efficient when measured in terms of this succinct presentation. An algorithm for this case needs to be exponentially faster than the one in Theorem 1. In Theorem 2, we present a polynomial time algorithm for the case of symmetric signatures. The increased efficiency is based on several signature invariants under orthogonal transformations.

Theorem 2. *There is a polynomial time algorithm to decide, given a finite set of symmetric signatures $\mathcal{F}$ expressed in the succinct notation, whether* $\operatorname{Holant}(\mathcal{F})$ *admits a holographic algorithm based on affine or product-type signatures.*

A dichotomy theorem classifies every set of signatures as defining either a tractable problem or an intractable problem (e.g. #P-hard). Yet it would be more useful if given a specific set of signatures, one could decide to which case it belongs. This is the decidability problem of a dichotomy theorem. In [11], a dichotomy regarding symmetric complex-weighted signatures for Holant problem was proved. However, the decidability problem was left open. Of the five tractable cases in this dichotomy theorem, three of them are easy to decide, but the remaining two cases are more challenging, which are (1) holographic algorithms using affine signatures and (2) holographic algorithms using product-type signatures. As a consequence of Theorem 2, this decidability is now proved.

Corollary 3. *The dichotomy theorem for symmetric complex-weighted Holant problems in [11] is decidable in polynomial time.*

Previous work on holographic algorithms focused almost exclusively on those with matchgates [44,43,16,19,17,18,32]. (This has led to a misconception in the community that holographic algorithms are always based on matchgates.) The first example of a holographic algorithm using something other than matchgates came in [24]. These holographic algorithms use generalized Fibonacci gates. A symmetric signature $f = [f_0, f_1, \ldots, f_n]$ is a generalized Fibonacci gate of type $\lambda \in \mathbb{C}$ if $f_{k+2} = \lambda f_{k+1} + f_k$ holds for all $k \in \{0, 1, \ldots, n-2\}$. The standard Fibonacci gates are of type $\lambda = 1$, in which case, the entries of the signature satisfy the recurrence relation of the Fibonacci numbers. The generalized Fibonacci gates were immediately put to use in a dichotomy theorem [22]. As it turned out, for nearly all values of λ, the generalized Fibonacci gates are holographically equivalent to product-type signatures. However, generalized Fibonacci gates are symmetric by definition. A main contribution of this paper is to extend the reach of holographic algorithms, other than those based on matchgates, beyond the symmetric case.

The constraint functions we call signatures are essentially tensors. Our central object of study can be rephrased as the orbits of affine and product-type tensors when acted upon by the orthogonal group (and related groups). We show that one can efficiently decide if any such orbit of a given tensor intersects the set of affine or product-type tensors. This result also generalizes to a set of tensors as stated in Theorems 1 and 2. In contrast, this orbit problem with the general linear group acting on two arbitrary tensors is NP-hard [37]. The so-called orbit closure problem has a fundamental importance in the foundation of geometric complexity theory [39].

Our techniques are mainly algebraic. A particularly important insight is that an orthogonal transformation in the standard basis is equivalent to a diagonal transformation in the $\left[\begin{smallmatrix} 1 & 1 \\ i & -i \end{smallmatrix}\right]$ basis, a type of correspondence as in Fourier transform. Since diagonal transformations are much easier to understand, this gives us a great advantage in understanding orbits under orthogonal transformations. Also, the groups of transformations that stabilize the affine and product-type signatures play an important role in our proofs.

2 Preliminaries

The framework of Holant problems is defined for functions mapping any $[q]^k \to \mathbb{F}$ for a finite q and some field $\mathbb{F}$. In this paper, we investigate some of the tractable complex-weighted Boolean Holant problems, that is, all functions are $[2]^k \to \mathbb{C}$. Strictly speaking, for consideration of models of computation, functions take complex algebraic numbers.

A *signature grid* $\Omega = (G, \mathcal{F}, \pi)$ consists of a graph $G = (V, E)$, where each vertex is labeled by a function $f_v \in \mathcal{F}$, and $\pi : V \to \mathcal{F}$ is the labeling. The Holant problem on instance Ω is to evaluate $\text{Holant}_\Omega = \sum_\sigma \prod_{v \in V} f_v(\sigma \mid_{E(v)})$, a sum over all edge assignments $\sigma : E \to \{0, 1\}$. A function f_v can be represented by listing its values in lexicographical order as in a truth table, which is a vector in $\mathbb{C}^{2^{\deg(v)}}$, or as a tensor in $(\mathbb{C}^2)^{\otimes \deg(v)}$. We also use $f_{\mathbf{x}}$ to denote the value $f(\mathbf{x})$, where $\mathbf{x}$ is a binary string. A function $f \in \mathcal{F}$ is also called a *signature*. A symmetric signature f on k Boolean variables can be expressed as $[f_0, f_1, \ldots, f_k]$, where f_w is the value of f on inputs of Hamming weight w. A signature f of arity n is *degenerate* if there exist unary signatures $u_j \in \mathbb{C}^2$ $(1 \le j \le n)$ such that $f = u_1 \otimes \cdots \otimes u_n$. A symmetric degenerate signature has the form $u^{\otimes n}$.

A Holant problem is parametrized by a set of signatures.

Definition 4. *Given a set of signatures $\mathcal{F}$, we define* Holant$(\mathcal{F})$ *as:*
Input: A signature grid $\Omega = (G, \mathcal{F}, \pi)$*;*
Output: Holant$_\Omega$*.*

To introduce the idea of holographic reductions, it is convenient to consider bipartite graphs. For a general graph, we can always transform it into a bipartite graph while preserving the Holant value, as follows. For each edge in the graph, we replace it by a path of length two. (This operation is called the *2-stretch* of the graph and yields the edge-vertex incidence graph.) Each new vertex is assigned the binary EQUALITY signature $(=_2) = [1, 0, 1]$. We use Holant $(\mathcal{F} \mid \mathcal{G})$ to denote the Holant problem on bipartite graphs $H = (U, V, E)$, where each vertex in U or V is assigned a signature in $\mathcal{F}$ or $\mathcal{G}$, respectively. An instance for this bipartite Holant problem is a bipartite signature grid denoted by $\Omega = (H;\ \mathcal{F} \mid \mathcal{G};\ \pi)$. Signatures in $\mathcal{F}$ are considered as row vectors (or covariant tensors); signatures in $\mathcal{G}$ are considered as column vectors (or contravariant tensors).

For a 2-by-2 matrix T and a signature set $\mathcal{F}$, define $T\mathcal{F} = \{g \mid \exists f \in \mathcal{F}$ of arity $n,\ g = T^{\otimes n} f\}$, similarly for $\mathcal{F}T$. Whenever we write $T^{\otimes n} f$ or $T\mathcal{F}$, we view the signatures as column vectors; similarly for $fT^{\otimes n}$ or $\mathcal{F}T$ as row vectors. Let T be an element of $\mathbf{GL}_2(\mathbb{C})$, the group of invertible 2-by-2 complex matrices. The holographic transformation defined by T is the following operation: given a signature grid $\Omega = (H;\ \mathcal{F} \mid \mathcal{G};\ \pi)$, for the same graph H, we get a new grid $\Omega' = (H;\ \mathcal{F}T \mid T^{-1}\mathcal{G};\ \pi')$ by replacing each signature in $\mathcal{F}$ or $\mathcal{G}$ with the corresponding signature in $\mathcal{F}T$ or $T^{-1}\mathcal{G}$.

Theorem 5 (Valiant's Holant Theorem [44]). *If there is a holographic transformation mapping signature grid Ω to Ω', then* $\text{Holant}_{\Omega} = \text{Holant}_{\Omega'}$.

Therefore, an invertible holographic transformation does not change the complexity of the Holant problem in the bipartite setting. Furthermore, there is a particular kind of holographic transformation, the orthogonal transformation, that preserves the binary equality and thus can be used freely in the standard setting. Let $\mathbf{O}_2(\mathbb{C})$ be the group of 2-by-2 complex matrices that are orthogonal. Recall that a matrix T is orthogonal if $TT^{\mathrm{T}} = I$. We also use $\mathbf{SO}_2(\mathbb{C})$ to denote the group of special orthogonal matrices, i.e. the subgroup of $\mathbf{O}_2(\mathbb{C})$ with determinant 1.

The following two signature sets are tractable without a holographic transformation [25].

Definition 6. *A k-ary function $f(x_1, \ldots, x_k)$ is* affine *if it has the form $\lambda \cdot \chi_{Ax=0} \cdot i^{\sum_{j=1}^{n} \langle \mathbf{v}_j, x \rangle}$, where $\lambda \in \mathbb{C}$, $x = (x_1, x_2, \ldots, x_k, 1)^T$, A is a matrix over $\mathbb{F}_2$, $\mathbf{v}_j$ is a vector over $\mathbb{F}_2$, and χ is a 0-1 indicator function such that $\chi_{Ax=0}$ is* 1 *iff* $Ax = 0$. *Note that the dot product $\langle \mathbf{v}_j, x \rangle$ is calculated over $\mathbb{F}_2$, while the summation $\sum_{j=1}^{n}$ on the exponent of $i = \sqrt{-1}$ is evaluated as a sum mod 4 of 0-1 terms. We use $\mathscr{A}$ to denote the set of all affine functions.*

An equivalent way to express the exponent of i is as a quadratic polynomial where all cross terms have an even coefficient.

Definition 7. *A function is of* product type *if it can be expressed as a product of unary functions, binary equality functions* $([1, 0, 1])$, *and binary disequality functions* $([0, 1, 0])$. *We use $\mathscr{P}$ to denote the set of product-type functions.*

The tractable sets $\mathscr{A}$ and $\mathscr{P}$ are still tractable under a suitable holographic transformation. This is captured by the following definition.

Definition 8. *A set $\mathcal{F}$ of signatures is $\mathscr{A}$-transformable (resp. $\mathscr{P}$-transformable) if there exists a holographic transformation T such that $\mathcal{F} \subseteq T\mathscr{A}$ (resp. $\mathcal{F} \subseteq T\mathscr{P}$) and $[1,0,1]T^{\otimes 2} \in \mathscr{A}$ (resp. $[1,0,1]T^{\otimes 2} \in \mathscr{P}$).*

To refine the above definition, we consider the stabilizer group of $\mathscr{A}$, which is $\mathrm{Stab}(\mathscr{A}) = \{T \in \mathbf{GL}_2(\mathbb{C}) \mid T\mathscr{A} \subseteq \mathscr{A}\}$. Technically this set is the left stabilizer group of $\mathscr{A}$, but it turns out that the left and right stabilizer groups of $\mathscr{A}$ coincide. Let $D = \left[\begin{smallmatrix} 1 & 0 \\ 0 & i \end{smallmatrix}\right]$ and $H_2 = \frac{1}{\sqrt{2}}\left[\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right]$. Also let $X = \left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$ and $Z = \frac{1}{\sqrt{2}}\left[\begin{smallmatrix} 1 & 1 \\ i & -i \end{smallmatrix}\right]$. Note that $Z = DH_2$ and that $D^2Z = \frac{1}{\sqrt{2}}\left[\begin{smallmatrix} 1 & 1 \\ -i & i \end{smallmatrix}\right] = ZX$, hence $X = Z^{-1}D^2Z$. It is easy to verify that $D, H_2, X, Z \in \mathrm{Stab}(\mathscr{A})$. In fact, $\mathrm{Stab}(\mathscr{A}) = \mathbb{C}^* \cdot \langle D, H_2 \rangle$, i.e. all nonzero scalar multiples of the group generated by D and H_2. Throughout the paper, we use α to denote $\frac{1+i}{\sqrt{2}} = \sqrt{i} = e^{\frac{\pi i}{4}}$.

Definition 9. *A symmetric signature f of arity n is in, respectively, $\mathscr{A}_1$, or $\mathscr{A}_2$, or $\mathscr{A}_3$ if there exist an $H \in \mathbf{O}_2(\mathbb{C})$ and $c \in \mathbb{C} - \{0\}$ such that f has the form, respectively, $cH^{\otimes n}(\left[\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right]^{\otimes n} + \beta \left[\begin{smallmatrix} 1 \\ -1 \end{smallmatrix}\right]^{\otimes n})$, $cH^{\otimes n}(\left[\begin{smallmatrix} 1 \\ i \end{smallmatrix}\right]^{\otimes n} + \left[\begin{smallmatrix} 1 \\ -i \end{smallmatrix}\right]^{\otimes n})$, or $cH^{\otimes n}(\left[\begin{smallmatrix} 1 \\ \alpha \end{smallmatrix}\right]^{\otimes n} + i^r \left[\begin{smallmatrix} 1 \\ -\alpha \end{smallmatrix}\right]^{\otimes n})$ with $\beta = \alpha^{tn+2r}$, $r \in \{0,1,2,3\}$, and $t \in \{0,1\}$.*

The three sets $\mathscr{A}_1$, $\mathscr{A}_2$, and $\mathscr{A}_3$ capture all symmetric $\mathscr{A}$-transformable signatures. For $i \in \{1,2,3\}$, when such an orthogonal H exists, we say that $f \in \mathscr{A}_i$ with transformation H.

Lemma 10 (Lemma 8.10 in full version of [11]). *Let f be a non-degenerate symmetric signature. Then f is $\mathscr{A}$-transformable iff $f \in \mathscr{A}_1 \cup \mathscr{A}_2 \cup \mathscr{A}_3$.*

We have a similar characterization for $\mathscr{P}$-transformable signatures using the stabilizer group of $\mathscr{P}$, $\mathrm{Stab}(\mathscr{P}) = \{T \in \mathbf{GL}_2(\mathbb{C}) \mid T\mathscr{P} \subseteq \mathscr{P}\}$. The group $\mathrm{Stab}(\mathscr{P})$ is generated by matrices of the form $\left[\begin{smallmatrix} 1 & 0 \\ 0 & \nu \end{smallmatrix}\right]$ for any $\nu \in \mathbb{C}$ and $X = \left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$.

Definition 11. *A symmetric signature f of arity n is in $\mathscr{P}_1$ if there exist an $H \in \mathbf{O}_2(\mathbb{C})$ and a nonzero $c \in \mathbb{C}$ such that $f = cH^{\otimes n}\left(\left[\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right]^{\otimes n} + \beta \left[\begin{smallmatrix} 1 \\ -1 \end{smallmatrix}\right]^{\otimes n}\right)$, where $\beta \neq 0$.*

It is easy to check that $\mathscr{A}_1 \subset \mathscr{P}_1$. We define $\mathscr{P}_2 = \mathscr{A}_2$. Similarly, for $i \in \{1,2\}$, when such an H exists, we say that $f \in \mathscr{P}_i$ with transformation H. The following lemma is similar to Lemma 10.

Lemma 12 (Lemma 8.13 in full version of [11]). *Let f be a non-degenerate symmetric signature. Then f is $\mathscr{P}$-transformable iff $f \in \mathscr{P}_1 \cup \mathscr{P}_2$.*

3 General Signatures

In this section we consider general (i.e. not necessarily symmetric) signatures. Let f be a signature of arity n. It is given as a column vector in $\mathbb{C}^{2^n}$ with bit length $N = O(2^n)$. We denote its entries by $f_{\mathbf{x}} = f(\mathbf{x})$ indexed by $\mathbf{x} \in \{0,1\}^n$.

The entries are from a fixed degree algebraic extension of $\mathbb{Q}$ and we may assume arithmetic operations take unit time.

We begin with $\mathscr{A}$-transformable signatures. Let f be a signature and $H = \left[\begin{smallmatrix} a & b \\ -b & a \end{smallmatrix}\right] \in \mathbf{SO}_2(\mathbb{C})$ where $a^2 + b^2 = 1$. Notice that $v_0 = (1, i)$ and $v_1 = (1, -i)$ are row eigenvectors of H with eigenvalues $a - bi$ and $a + bi$ respectively.

For a vector $\mathbf{u} = (u_1, \ldots, u_n) \in \{0, 1\}^n$ of length n, let $v_{\mathbf{u}} = v_{u_1} \otimes v_{u_2} \otimes \ldots \otimes v_{u_n}$, and let $w(\mathbf{u})$ be the Hamming weight of $\mathbf{u}$. Then for the 2^n-by-2^n matrix $H^{\otimes n}$, $v_{\mathbf{u}}$ is a row eigenvector with eigenvalue $(a-bi)^{n-w(\mathbf{u})}(a+bi)^{w(\mathbf{u})} = (a-bi)^{n-2w(\mathbf{u})} = (a+bi)^{2w(\mathbf{u})-n}$ as $(a+bi)(a-bi) = a^2+b^2 = 1$. Let $Z' = \left[\begin{smallmatrix} 1 & i \\ 1 & -i \end{smallmatrix}\right]$ and $\hat{f} = Z'^{\otimes n} f$. Then $\hat{f}_{\mathbf{u}} = \langle v_{\mathbf{u}}, f \rangle$, as a dot product. The following lemma summarizes the above discussion and is a very important ingredient of this paper. It states that proper orthogonal transformations are diagonal transformations in the $\left[\begin{smallmatrix} 1 & i \\ 1 & -i \end{smallmatrix}\right]$ basis.

Lemma 13. *Suppose f and g are signatures of arity n and let $H = \left[\begin{smallmatrix} a & b \\ -b & a \end{smallmatrix}\right]$ and $T = \left[\begin{smallmatrix} a-bi & 0 \\ 0 & a+bi \end{smallmatrix}\right]$. Then $g = H^{\otimes n} f$ iff $\hat{g} = T^{\otimes n} \hat{f}$.*

With Lemma 13, we characterize signatures that are invariant under $\mathbf{SO}_2(\mathbb{C})$ transformations.

Lemma 14. *Let f be a signature. Then f is invariant under transformations in $\mathbf{SO}_2(\mathbb{C})$ (up to a nonzero constant) iff the support of $\hat{f}$ contains at most one Hamming weight.*

With Lemma 13 and Lemma 14, we are able to give the algorithm for $\mathscr{A}$-transformable signatures.

Theorem 15. *There is a polynomial time algorithm to decide, for any finite set of signatures $\mathcal{F}$, whether $\mathcal{F}$ is $\mathscr{A}$-transformable. If so, at least one transformation can be found.*

The algorithm for $\mathscr{P}$ is also based on Lemma 13. The difference here is that we need to first factor the signatures. We show a unique factorization lemma for signatures in general.

Definition 16. *We call a function f of arity n on variable set* $\mathbf{x}$ reducible *if there exist f_1 and f_2 of arities n_1 and n_2 on variable sets $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively, such that $1 \le n_1, n_2 \le n-1$, $\mathbf{x}_1 \cup \mathbf{x}_2 = \mathbf{x}$, $\mathbf{x}_1 \cap \mathbf{x}_2 = \emptyset$, and $f(\mathbf{x}) = f_1(\mathbf{x}_1) f_2(\mathbf{x}_2)$. Otherwise we call f* irreducible.

If a function f is reducible, then we can factor it into functions of smaller arity. This procedure can be applied recursively and terminates when all components are irreducible. Therefore any function has at least one irreducible factorization. We show that such a factorization is unique for functions that are not identically zero. Furthermore, it can be computed in polynomial time.

Lemma 17. *Let f be a function of arity n on variables* $\mathbf{x}$ *that is not identically zero. Assume there exist irreducible functions f_i and g_j, and two partitions $\{\mathbf{x}_i\}$*

and $\{\mathbf{y}_j\}$ *of* $\mathbf{x}$ *for* $1 \le i \le k$ *and* $1 \le j \le k'$*, such that* $f(\mathbf{x}) = \prod_{i=1}^{k} f_i(\mathbf{x}_i) = \prod_{j=1}^{k'} g_j(\mathbf{y}_j)$*. Then* $k = k'$*, the partitions are the same, and* $\{f_i\}$ *and* $\{g_j\}$ *are the same up to a permutation.*

The factorization algorithm leads to a decision algorithm for membership in $\mathscr{P}$. Combined with Lemma 13, we can obtain the algorithm for $\mathscr{P}$-transformable signatures.

Theorem 18. *There is a polynomial time algorithm to decide, for any finite set of signatures* $\mathcal{F}$*, whether* $\mathcal{F}$ *is* $\mathscr{P}$*-transformable. If so, at least one transformation can be found.*

4 Symmetric Signatures

In this section, we consider the case when the signatures are symmetric. The significant difference is that a symmetric signature of arity n is given by $n+1$ values, instead of 2^n values. This exponentially more succinct representation requires us to find a more efficient algorithm. To begin, we provide efficient algorithms to decide membership each of $\mathscr{A}_1$, $\mathscr{A}_2$, and $\mathscr{A}_3$ for a single signature. If the signature is in one of the sets, then the algorithm also finds at least one corresponding orthogonal transformation. By Lemma 10, this is enough to check if a signature is $\mathscr{A}$-transformable.

We say a signature f satisfies a second order recurrence relation, if for all $0 \le k \le n-2$, there exist $a, b, c \in \mathbb{C}$ not all zero, such that $af_k + bf_{k+1} + cf_{k+2} = 0$. In fact, satisfying a second order recurrence relation with $b^2 - 4ac \neq 0$ is a necessary condition for a signature to be $\mathscr{A}$- or $\mathscr{P}$-transformable. This also implies a tensor decomposition of f. The following definition of the θ function is crucial.

Definition 19. *For a pair of linearly independent vectors* $v_0 = \left[\begin{smallmatrix} a_0 \\ b_0 \end{smallmatrix}\right]$ *and* $v_1 = \left[\begin{smallmatrix} a_1 \\ b_1 \end{smallmatrix}\right]$*, we define* $\theta(v_0, v_1) = \left(\frac{a_0 a_1 + b_0 b_1}{a_1 b_0 - a_0 b_1}\right)^2$*. Furthermore, suppose that a signature* f *of arity* $n \ge 3$ *can be expressed as* $f = v_0^{\otimes n} + v_1^{\otimes n}$*, where* v_0 *and* v_1 *are linearly independent. Then we define* $\theta(f) = \theta(v_0, v_1)$*.*

Intuitively, this formula is the square of the cotangent of the angle from v_0 to v_1. This notion of cotangent is properly extended to the complex domain. By insisting that v_0 and v_1 be linearly independent, we ensure $\theta(v_0, v_1)$ is well-defined. The expression is squared so that $\theta(v_0, v_1) = \theta(v_1, v_0)$. Let $f = v_0^{\otimes n} + v_1^{\otimes n}$ be a non-degenerate signature of arity $n \ge 3$. Since f is non-degenerate, v_0 and v_1 are linearly independent. This expression for f via v_0 and v_1 is unique to up a root of unity. In particular, $\theta(f)$ from Definition 19 is well-defined since every possible expression gives the same value for θ. It is easy to verify that θ is invariant under an orthogonal transformation. Formally, we have the following lemma, which is proved by simple algebra.

Lemma 20. *For two linearly independent vectors v_0, $v_1 \in \mathbb{C}^2$ and $H \in \mathbf{O}_2(\mathbb{C})$, let $\widehat{v_0} = Hv_0$ and $\widehat{v_1} = Hv_1$. Then $\theta(v_0, v_1) = \theta(\widehat{v_0}, \widehat{v_1})$.*

Now we have some necessary conditions for a signature f to be in $\mathscr{A}_1 \cup \mathscr{A}_2 \cup \mathscr{A}_3$. First f must satisfy a second order recurrence relation with $b^2 - 4ac \neq 0$. Then $\theta(f)$ is well defined. It is easy to observe $\theta(f) = 0, -1, -\frac{1}{2}$ for f in $\mathscr{P}_1$, $\mathscr{A}_2$, $\mathscr{A}_3$ respectively. Recall that $\mathscr{A}_1 \subseteq \mathscr{P}_1$ and $\mathscr{A}_2 = \mathscr{P}_2$.

This condition via $\theta(f)$ is not sufficient for f to be $\mathscr{A}$-transformable. For example, if $f = v_0^{\otimes n} + v_1^{\otimes n}$ with $v_0 = [1, i]$ and v_1 is not a multiple of $[1, -i]$, then $\theta(f) = -1$ but f is not in $\mathscr{A}_2 = \mathscr{P}_2$. Nevertheless, this is essentially the only exceptional case. The other cases are handled with some extra conditions on the coefficients, as follows.

Lemma 21. *Let $f = v_0^{\otimes n} + v_1^{\otimes n}$ be a symmetric signature of arity $n \geq 3$, where $v_0 = \left[\begin{smallmatrix} a_0 \\ b_0 \end{smallmatrix}\right]$ and $v_1 = \left[\begin{smallmatrix} a_1 \\ b_1 \end{smallmatrix}\right]$ are linearly independent. Then $f \in \mathscr{A}_1$ iff $\theta(f) = 0$ and there exist an $r \in \{0, 1, 2, 3\}$ and $t \in \{0, 1\}$ such that $a_1^n = \alpha^{tn+2r} b_0^n \neq 0$ or $b_1^n = \alpha^{tn+2r} a_0^n \neq 0$.*

Lemma 22. *Let $f = v_0^{\otimes n} + v_1^{\otimes n}$ be a symmetric signature of arity $n \geq 3$, where $v_0 = \left[\begin{smallmatrix} a_0 \\ b_0 \end{smallmatrix}\right]$ and $v_1 = \left[\begin{smallmatrix} a_1 \\ b_1 \end{smallmatrix}\right]$ are linearly independent. Then $f \in \mathscr{A}_3$ iff there exist an $\varepsilon \in \{1, -1\}$ and $r \in \{0, 1, 2, 3\}$ such that $a_1 \left(\sqrt{2} a_0 + \varepsilon i b_0\right) = b_1 \left(\varepsilon i a_0 - \sqrt{2} b_0\right)$, $a_1^n = i^r \left(\varepsilon i a_0 - \sqrt{2} b_0\right)^n$, and $b_1^n = i^r \left(\sqrt{2} a_0 + \varepsilon i b_0\right)^n$.*

For $\mathscr{A}_2 = \mathscr{P}_2$, we require a stronger condition.

Lemma 23 (Lemma 8.8 in full version of [11]). *Let f be a non-degenerate symmetric signature. Then $f \in \mathscr{A}_2$ iff f is of the form $c \left(\left[\begin{smallmatrix} 1 \\ i \end{smallmatrix}\right]^{\otimes n} + \beta \left[\begin{smallmatrix} 1 \\ -i \end{smallmatrix}\right]^{\otimes n} \right)$ for some $c, \beta \neq 0$.*

To summarize, we have the following lemma.

Lemma 24. *Given a non-degenerate symmetric signature f of arity at least 3, there is a polynomial time algorithm to decide whether $f \in \mathscr{A}_k$ for each $k \in \{1, 2, 3\}$. If so, k is unique and at least one corresponding orthogonal transformation can be found in polynomial time.*

Next we show that if a non-degenerate signature f of arity $n \geq 3$ is in $\mathscr{A}_1$, $\mathscr{A}_2$, or $\mathscr{A}_3$, then for any set $\mathcal{F}$ containing f, there are only $O(n)$ many transformations to be checked to decide whether $\mathcal{F}$ is $\mathscr{A}$-transformable.

Lemma 25. *Let $\mathcal{F}$ be a set of symmetric signatures and suppose $\mathcal{F}$ contains a non-degenerate signature $f \in \mathscr{A}_1$ of arity $n \geq 3$ with $H \in \mathbf{O}_2(\mathbb{C})$. Then $\mathcal{F}$ is $\mathscr{A}$-transformable iff $\mathcal{F}$ is a subset of $H\mathscr{A}$, or $H \left[\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right] \mathscr{A}$, or $H \left[\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right] \left[\begin{smallmatrix} 1 & 0 \\ 0 & \alpha \end{smallmatrix}\right] \mathscr{A}$.*

Lemma 26. *Let $\mathcal{F}$ be a set of symmetric signatures and suppose $\mathcal{F}$ contains a non-degenerate signature $f \in \mathscr{A}_2$ of arity $n \geq 3$. Then there exists a set $\mathcal{H} \subseteq \mathbf{O}_2(\mathbb{C})$ of size $O(n)$ such that $\mathcal{F}$ is $\mathscr{A}$-transformable iff there exists an $H \in \mathcal{H}$ such that $\mathcal{F} \subseteq H\mathscr{A}$. Moreover $\mathcal{H}$ can be computed in polynomial time in the input length of the symmetric signature f.*

Lemma 27. *Let $\mathcal{F}$ be a set of symmetric signatures and suppose $\mathcal{F}$ contains a non-degenerate signature $f \in \mathscr{A}_3$ of arity $n \geq 3$ with $H \in \mathbf{O}_2(\mathbb{C})$. Then $\mathcal{F}$ is $\mathscr{A}$-transformable iff $\mathcal{F} \subseteq H \left[\begin{smallmatrix} 1 & 0 \\ 0 & \alpha \end{smallmatrix}\right] \mathscr{A}$.*

Now we can decide if a finite set of signatures is $\mathscr{A}$-transformable. To avoid trivialities, we assume $\mathcal{F}$ contains a non-degenerate signature of arity at least 3.

Theorem 28. *There is a polynomial time algorithm to decide, for any finite input set $\mathcal{F}$ of symmetric signatures containing a non-degenerate signature f of arity $n \geq 3$, whether $\mathcal{F}$ is $\mathscr{A}$-transformable.*

Now we consider $\mathscr{P}$-transformable signatures. To decide if a single signature is $\mathscr{P}$-transformable, it is equivalent to decide membership in $\mathscr{P}_1 \cup \mathscr{P}_2$ by Lemma 12. The following lemma tells how to decide the membership of $\mathscr{P}_1$.

Lemma 29. *Let $f = v_0^{\otimes n} + v_1^{\otimes n}$ be a symmetric signature of arity $n \geq 3$, where v_0 and v_1 are linearly independent. Then $f \in \mathscr{P}_1$ iff $\theta(f) = 0$.*

Since $\mathscr{A}_2 = \mathscr{P}_2$, deciding membership in $\mathscr{P}_2$ is handled by Lemma 23. Using Lemma 29 and Lemma 23, we can efficiently decide membership in $\mathscr{P}_1 \cup \mathscr{P}_2$.

Lemma 30. *Given a non-degenerate symmetric signature f of arity at least 3, there is a polynomial time algorithm to decide whether $f \in \mathscr{P}_k$ for some $k \in \{1, 2\}$. If so, k is unique and at least one corresponding orthogonal transformation can be found in polynomial time.*

With a signature in $\mathscr{P}_1 \cup \mathscr{P}_2$, we can decide if a set of symmetric signatures is $\mathscr{P}$-transformable.

Lemma 31. *Let $\mathcal{F}$ be a set of symmetric signatures and suppose $\mathcal{F}$ contains a non-degenerate signature $f \in \mathscr{P}_1$ of arity $n \geq 3$ with $H \in \mathbf{O}_2(\mathbb{C})$. Then $\mathcal{F}$ is $\mathscr{P}$-transformable iff $\mathcal{F} \subseteq H \left[\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right] \mathscr{P}$.*

Lemma 32. *Let $\mathcal{F}$ be a set of symmetric signatures and suppose $\mathcal{F}$ contains a non-degenerate signature $f \in \mathscr{P}_2$ of arity $n \geq 3$. Then $\mathcal{F}$ is $\mathscr{P}$-transformable iff all non-degenerate signatures in $\mathcal{F}$ are contained in $\mathscr{P}_2 \cup \{=_2\}$.*

With all these results, we show how to decide if a finite set of signatures is $\mathscr{P}$-transformable. To avoid trivialities, we assume $\mathcal{F}$ contains a non-degenerate signature of arity at least 3.

Theorem 33. *There is a polynomial time algorithm to decide, for any finite input set $\mathcal{F}$ of symmetric signatures containing a non-degenerate signature f of arity $n \geq 3$, whether $\mathcal{F}$ is $\mathscr{P}$-transformable.*

Acknowledgements. All authors were supported by NSF CCF-1217549. The second author was also supported by a Simons Award for Graduate Students in Theoretical Computer Science from the Simons Foundation.

References

1. Bulatov, A., Dyer, M., Goldberg, L.A., Jalsenius, M., Richerby, D.: The complexity of weighted Boolean #CSP with mixed signs. Theor. Comput. Sci. 410(38-40), 3949–3961 (2009)
2. Bulatov, A., Grohe, M.: The complexity of partition functions. Theor. Comput. Sci. 348(2), 148–186 (2005)
3. Bulatov, A.A.: The complexity of the counting constraint satisfaction problem. J. ACM 60(5), 34:1–34:41 (2013)
4. Bulatov, A.A., Dalmau, V.: Towards a dichotomy theorem for the counting constraint satisfaction problem. Inform. and Comput. 205(5), 651–678 (2007)
5. Cai, J.-Y., Chen, X.: A decidable dichotomy theorem on directed graph homomorphisms with non-negative weights. In: FOCS, pp. 437–446. IEEE Computer Society (2010)
6. Cai, J.-Y., Chen, X.: Complexity of counting CSP with complex weights. In: STOC, pp. 909–920. ACM (2012), arXiv:1111.2384
7. Cai, J.-Y., Chen, X., Lu, P.: Non-negatively weighted #CSP: An effective complexity dichotomy. In: IEEE Conference on Computational Complexity, pp. 45–54. IEEE Computer Society (2011)
8. Cai, J.-Y., Chen, X., Lu, P.: Graph homomorphisms with complex values: A dichotomy theorem. SIAM J. Comput. 42(3), 924–1029 (2013)
9. Cai, J.-Y., Choudhary, V., Lu, P.: On the theory of matchgate computations. Theory of Computing Systems 45(1), 108–132 (2009)
10. Cai, J.-Y., Gorenstein, A.: Matchgates revisited. Theory of Computing 10(868), 4001–4030 (2014)
11. Cai, J.-Y., Guo, H., Williams, T.: A complete dichotomy rises from the capture of vanishing signatures (extended abstract). In: STOC, pp. 635–644. ACM (2013), arXiv:1204.6445
12. Cai, J.-Y., Guo, H., Williams, T.: Holographic algorithms beyond matchgates. CoRR, abs/1307.7430 (2013)
13. Cai, J.-Y., Huang, S., Lu, P.: From Holant to #CSP and back: Dichotomy for Holant^c problems. Algorithmica 64(3), 511–533 (2012)
14. Cai, J.-Y., Kowalczyk, M.: Spin systems on k-regular graphs with complex edge functions. Theor. Comput. Sci. 461, 2–16 (2012)
15. Cai, J.-Y., Kowalczyk, M., Williams, T.: Gadgets and anti-gadgets leading to a complexity dichotomy. In: ITCS, pp. 452–467. ACM (2012)
16. Cai, J.-Y., Lu, P.: Holographic algorithms with unsymmetric signatures. In: SODA, pp. 54–63. Society for Industrial and Applied Mathematics (2008)
17. Cai, J.-Y., Lu, P.: Holographic algorithms: From art to science. J. Comput. Syst. Sci. 77(1), 41–61 (2011)
18. Cai, J.-Y., Lu, P.: Signature theory in holographic algorithms. Algorithmica 61(4), 779–816 (2011)
19. Cai, J.-Y., Lu, P., Xia, M.: Holographic algorithms with matchgates capture precisely tractable planar #CSP. In: FOCS, pp. 427–436. IEEE Computer Society (2010)
20. Cai, J.-Y., Lu, P., Xia, M.: Computational complexity of Holant problems. SIAM J. Comput. 40(4), 1101–1132 (2011)
21. Cai, J.-Y., Lu, P., Xia, M.: Dichotomy for Holant* problems of Boolean domain. In: SODA, pp. 1714–1728. SIAM (2011)

22. Cai, J.-Y., Lu, P., Xia, M.: Holographic reduction, interpolation and hardness. Computational Complexity 21(4), 573–604 (2012)
23. Cai, J.-Y., Lu, P., Xia, M.: Dichotomy for Holant* problems with domain size 3. In: SODA, pp. 1278–1295. SIAM (2013)
24. Cai, J.-Y., Lu, P., Xia, M.: Holographic algorithms by Fibonacci gates. Linear Algebra and its Applications 438(2), 690–707 (2013)
25. Cai, J.-Y., Lu, P., Xia, M.: The complexity of complex weighted Boolean #CSP. J. Comput. System Sci. 80(1), 217–236 (2014)
26. Dyer, M., Goldberg, L.A., Jerrum, M.: The complexity of weighted Boolean #CSP. SIAM J. Comput. 38(5), 1970–1986 (2009)
27. Dyer, M., Goldberg, L.A., Paterson, M.: On counting homomorphisms to directed acyclic graphs. J. ACM 54(6) (2007)
28. Dyer, M., Greenhill, C.: The complexity of counting graph homomorphisms. Random Struct. Algorithms 17(3-4), 260–289 (2000)
29. Dyer, M., Richerby, D.: An effective dichotomy for the counting constraint satisfaction problem. SIAM J. Comput. 42(3), 1245–1274 (2013)
30. Goldberg, L.A., Grohe, M., Jerrum, M., Thurley, M.: A complexity dichotomy for partition functions with mixed signs. SIAM J. Comput. 39(7), 3336–3402 (2010)
31. Guo, H., Huang, S., Lu, P., Xia, M.: The complexity of weighted Boolean #CSP modulo k. In: STACS, pp. 249–260. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2011)
32. Guo, H., Williams, T.: The complexity of planar Boolean #CSP with complex weights. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) ICALP 2013, Part I. LNCS, vol. 7965, pp. 516–527. Springer, Heidelberg (2013)
33. Hell, P., Nešetřil, J.: On the complexity of H-coloring. J. Comb. Theory Ser. B 48(1), 92–110 (1990)
34. Huang, S., Lu, P.: A dichotomy for real weighted Holant problems. In: IEEE Conference on Computational Complexity, pp. 96–106. IEEE Computer Society (2012)
35. Kasteleyn, P.W.: The statistics of dimers on a lattice. Physica 27(12), 1209–1225 (1961)
36. Kasteleyn, P.W.: Graph theory and crystal physics. In: Harary, F. (ed.) Graph Theory and Theoretical Physics, pp. 43–110. Academic Press, London (1967)
37. Kayal, N.: Affine projections of polynomials: Extended abstract. In: STOC, pp. 643–662. ACM (2012)
38. Lovász, L.: Operations with structures. Acta Math. Hung. 18(3-4), 321–328 (1967)
39. Mulmuley, K.D., Sohoni, M.: Geometric complexity theory I: An approach to the P vs. NP and related problems. SIAM J. Comput. 31(2), 496–526 (2001)
40. Temperley, H.N.V.: Michael E. Fisher. Dimer problem in statistical mechanics—an exact result. Philosophical Magazine 6(68), 1061–1063 (1961)
41. Valiant, L.G.: Expressiveness of matchgates. Theor. Comput. Sci. 289(1), 457–471 (2002)
42. Valiant, L.G.: Quantum circuits that can be simulated classically in polynomial time. SIAM J. Comput. 31(4), 1229–1254 (2002)
43. Valiant, L.G.: Accidental algorthims. In: FOCS, pp. 509–517. IEEE Computer Society (2006)
44. Valiant, L.G.: Holographic algorithms. SIAM J. Comput. 37(5), 1565–1594 (2008)

Testing Probability Distributions Underlying Aggregated Data[⋆]

Clément Canonne[1] and Ronitt Rubinfeld[2,⋆⋆]

[1] Columbia University
ccanonne@cs.columbia.edu
[2] CSAIL, MIT and the Blavatnik School of Computer Science, Tel Aviv University
ronitt@csail.mit.edu

Abstract. In this paper, we analyze and study a hybrid model for testing and learning probability distributions. Here, in addition to samples, the testing algorithm is provided with one of two different types of oracles to the unknown distribution D over $[n]$. More precisely, we consider both the *dual* and *cumulative dual access models*, in which the algorithm A can both sample from D and respectively, for any $i \in [n]$,

– query the probability mass $D(i)$ *(query access)*; or

– get the total mass of $\{1, \ldots, i\}$, i.e. $\sum_{j=1}^{i} D(j)$ *(cumulative access)*

In these two models, we bypass the strong lower bounds established in both of the previously studied sampling and query oracle settings for a number of problems, giving constant-query algorithms for most of them. Finally, we show that while the testing algorithms can be in most cases strictly more efficient, some tasks remain hard even with this additional power.

1 Introduction

Given data sampled from a population or an experiment, understanding the distribution from which it has been drawn is a fundamental problem in statistics, and one which has been extensively studied for decades. However, it is only rather recently that these questions have been considered when the distribution is over a *very large* domain (see for instance [3,18,21]). In this case, the usual techniques in statistics and learning theory become impractical, motivating the search for better algorithms, in particular by relaxing the goals so that learning is not required. This is useful in many real-world applications where only a particular aspect of the distribution is investigated, such as estimating the entropy or the distance between two distributions. In these examples, as well as many others, one *can* achieve sublinear sample complexity. However, strong lower bounds show that the complexity of these tasks is still daunting, as it has polynomial, and often nearly linear, dependence on the size of the support of the distribution.

[⋆] A full version of this paper is available as ECCC TR14–021 [11].

[⋆⋆] Research supported by NSF grants CCF-1217423 and CCF-1065125.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 283–295, 2014.

To address this difficulty, new lines of research have emerged. One approach is to obtain more efficient algorithms for special classes of distributions. For instance, improved algorithms whose sample complexity is exponentially smaller can be achieved for many tasks, assuming the distribution is monotone, unimodal, or a "k-histogram" [5,20,14]. A different approach applies to general distributions, but gives the algorithm more power in the form of more flexible access to the distribution: as in many applications the data has already been collected and aggregated, it may be reasonable to assume that the testing algorithm can make other limited queries to the probability density function. For example, the algorithm may be provided with query access to the probability density function of the distribution [24], or samples from conditional distributions induced by the original distribution [12,9,10].

1.1 Our Model: Dual and Cumulative Dual Oracles

In this work, we consider the power of two natural oracles. The first is a *dual oracle*, which combines the standard model for distributions and the familiar one commonly assumed for testing Boolean and real-valued functions. In more detail, the testing algorithm is granted access to the unknown distribution D through two independent oracles, one providing samples of the distribution, while the other, on query i in the domain of the distribution, provides the value of the probability density function at i.[1]

Definition 1 (Dual Access Model). *Let D be a fixed distribution over $[n] = \{1, \ldots, n\}$. A* dual oracle *for D is a pair of oracles $(\mathsf{SAMP}_D, \mathsf{EVAL}_D)$ defined as follows: when queried, the* sampling *oracle SAMP_D returns an element $i \in [n]$, where the probability that i is returned is $D(i)$ independently of all previous calls to any oracle; while the* evaluation *oracle EVAL_D takes as input a query element $j \in [n]$, and returns the probability weight $D(j)$ that the distribution puts on j.*

It is worth noting that this type of dual access to a distribution has been considered (under the name *combined oracle*) in [6] and [19], where they address the task of estimating (multiplicatively) the entropy of the distribution, or the f-divergence between two of them (see Sect. 4 for a discussion of their results).

The second oracle that we consider provides samples of the distribution as well as queries to the *cumulative distribution function* (cdf) at any point in the domain[2].

Definition 2 (Cumulative Dual Access Model). *Let D be a fixed distribution over $[n]$. A* cumulative dual oracle *for D is a pair of oracles*

[1] Note that in both definitions, one can decide to disregard the corresponding evaluation oracle, which in effect amounts to falling back to the standard sampling model; moreover, for our domain $[n]$, any EVAL_D query can be simulated by (at most) two queries to a CEVAL_D oracle – in other terms, the cumulative dual model is at least as powerful as the dual one.

[2] We observe that such a cumulative evaluation oracle CEVAL appears in [5] (Sect. 8).

($\mathsf{SAMP}_D, \mathsf{CEVAL}_D$) *defined as follows: the* sampling *oracle* SAMP_D *behaves as before, while the* evaluation *oracle* CEVAL_D *takes as input a query element* $j \in [n]$*, and returns the probability weight that the distribution puts on* $[j]$*, that is* $D([j]) = \sum_{i=1}^{j} D(i)$*.*

1.2 Motivation and Discussion

As a first motivation to this hybrid model, consider the following scenario: There is a huge and freely available dataset, which a computationally-limited party – call it Arthur – needs to process. Albeit all the data is public and Arthur can view any element of his choosing, extracting further information from the dataset (such as the number of occurrences of a particular element) takes too much time. However, a third-party, Merlin, has already spent resources in preprocessing this dataset and is willing to disclose such information – yet at a price. This leaves Arthur with the following question: *how can he get his work done as quickly as possible, while paying as little as possible?* This type of question is captured by our new model, and can be analyzed in this framework. For instance, if the samples are stored in sorted order, implementing either of our oracles becomes possible with only a logarithmic overhead per query. It is worth noting that Google has published their N-gram models, which describe their distribution model on 5-word sequences in the English language. In addition, they have made available the texts on which their model was constructed. Thus, samples of the distribution in addition to query access to probabilities of specific domain elements may be extracted from the Google model.

A second and entirely theoretical motivation for studying distribution testing in these two dual oracle settings arises from attempting to understand the limitations and underlying difficulties of the standard sampling model. Indeed, by circumventing the lower bound, one may get a better grasp on the core issues whence the hardness stemmed in the first place.

A third motivation arises from data privacy, when a curator administers a database of highly sensitive records (e.g, healthcare information, or financial records). Differential privacy [15,17,16] studies mechanisms which allow the curator to release relevant information about its database without without jeopardizing the privacy of the individual records. In particular, mechanisms have been considered that enable the curator to *release* a sanitized approximation $\tilde{D}$ of its database D, which "behaves" essentially the same for all queries of a certain type – such as *counting* or *interval queries*[3] [8]. Specifically, if the user needs to test a property of a database, it is sufficient to test whether the sanitized database has the property, using now both samples and interval (i.e., CEVAL) or counting (EVAL) queries. As long as the tester has some tolerance (in that it accepts databases that are close to having the property), it is then possible to decide whether the true database itself is close to having the property of interest.

[3] A counting query is of the form "how many records in the database satisfy predicate χ?" – or, equivalently, "what is the probability that a random record drawn from the database satisfies χ?".

Finally, a further motivation is the tight connection between the dual access model and the *data-stream model*, as shown by Guha et al. ([19], Theorem 25): more precisely, they prove that any (multiplicative) approximation algorithm for a large class of functions of the distribution (functions that are invariant by relabeling of any two elements of the support) in the dual access model yields a space-efficient, $O(1)$-pass approximation algorithm for the same function in the data-stream model.

1.3 Our Results and Techniques

We focus here on four fundamental and pervasive problems in distribution testing, which are testing *uniformity*, *identity* to a known distribution D^*, *closeness* between two (unknown) distributions D_1, D_2, and finally *entropy and support size*. As usual in the distribution testing literature, the notion of distance we use is the *total variation distance* (or statistical distance), which is essentially the ℓ_1 distance between the probability distributions (see Sect. 2 for the formal definition). Testing closeness is thus the problem of deciding if two distributions are equal or far from each other in total variation distance; while tolerant testing aims at deciding whether they are sufficiently close versus far from each other.

As shown in Table 1, which summarizes our results and compares them to the corresponding bounds for the standard sampling-only (SAMP), evaluation-only (EVAL) and conditional sampling (COND) models, we indeed manage to bypass the aforementioned limitations of the sampling model, and give (often tight) algorithms with sample complexity either constant (with relation to n) or logarithmic, where a polynomial dependence was required in the standard setting. A similar observation holds for the evaluation-only model, where some natural problems require $\Omega(n)$ queries[4] – yet admit constant-query testers in our setting.

Our main finding overall is that *both dual models allow testing algorithms to significantly outperform both* SAMP *and* COND *algorithms*, either with relation to the dependence on n or, for the latter, in $1/\varepsilon$. Examples of such improvements were previously known for the COND model [12,9]. In the models considered in this work, we extend these improvements to further problems and get strenghtened bounds. Further, these new testing algorithms are *significantly simpler*, both conceptually and in their analysis, and can often be made robust to some multiplicative noise in the evaluation oracle. Another key observation is that this new flexibility not only allows us to tell whether two distributions are close or far, but also to efficiently estimate their distance[5].

In more detail, we show that for the problem of testing equivalence between distributions, both models allow one to get rid of any dependence on n, with a

[4] Consider two point distributions D_1, D_2, each of them putting mass 1 on a single random element of $[n]$; with EVAL access only, testing whether $D_1 = D_2$ amounts to finding these points, which requires linearly many queries.

[5] For details on the equivalence between tolerant testing and distance estimation, the reader is referred to [23].

(tight) sample complexity of $\Theta(1/\varepsilon)$. The upper bound is achieved by adapting an EVAL-only algorithm of [24] (for identity testing) to our setting, while the lower bound is obtained by designing a far-from-uniform instance which "defeats" simultaneously both oracles of our models. Turning to tolerant testing of equivalence, we describe algorithms whose sample complexity is again independent of n, in sharp contrast with the $n^{1-o(1)}$ lower bound of the standard sampling model. Moreover, we are able to show that, at least in the Dual access model, our quadratic dependence on ε is optimal. The same notable improvements apply to the query complexity of estimating the support size of the distribution, which becomes constant (with relation to n) in both of our access models – versus quasilinear if one only allows sampling.

As for the task of (additively) estimating the entropy of an arbitrary distribution, we give an algorithm whose sample complexity is only polylogarithmic in n, and show that this is tight in the Dual access model, up to the exponent of the logarithm. Once more, this is to be compared to the $n^{1-o(1)}$ lower bound for the sampling model.

Table 1. Summary of results (last two columns) and comparison with previous work. The bounds with an asterisk are those which, in spite of being for different models, derive from the results of the last two columns.

Problem	SAMP[6]	COND[9,10]	EVAL	Dual	Cumulative
Testing uniformity	$\Theta\left(\frac{\sqrt{n}}{\varepsilon^2}\right)$	$\tilde{O}\left(\frac{1}{\varepsilon^2}\right)$, $\Omega\left(\frac{1}{\varepsilon^2}\right)$	$O\left(\frac{1}{\varepsilon}\right)$ [24], $\Omega\left(\frac{1}{\varepsilon}\right)^*$	$\Theta\left(\frac{1}{\varepsilon}\right)$	$\Theta\left(\frac{1}{\varepsilon}\right)$
Testing $\equiv D^*$	$\tilde{\Theta}\left(\frac{\sqrt{n}}{\varepsilon^2}\right)$	$\tilde{O}\left(\frac{1}{\varepsilon^4}\right)$			
Testing $D_1 \equiv D_2$	$\Theta\left(\frac{N^{2/3}}{\varepsilon^{4/3}}\right)$	$\tilde{O}\left(\frac{\log^5 n}{\varepsilon^4}\right)$	$\Omega(n)$		
Tolerant uniformity	$O\left(\frac{1}{(\varepsilon_2-\varepsilon_1)^2}\frac{n}{\log n}\right)$ $\Omega\left(\frac{n}{\log n}\right)$	$\tilde{O}\left(\frac{1}{(\varepsilon_2-\varepsilon_1)^{20}}\right)$	$\Omega\left(\frac{1}{(\varepsilon_2-\varepsilon_1)^2}\right)^*$	$\Theta\left(\frac{1}{(\varepsilon_2-\varepsilon_1)^2}\right)$	$O\left(\frac{1}{(\varepsilon_2-\varepsilon_1)^2}\right)$
Tolerant D^* Tolerant D_1, D_2	$\Omega\left(\frac{n}{\log n}\right)$				
Estimating entropy to $\pm\Delta$	$\Theta\left(\frac{n}{\log n}\right)$			$O\left(\frac{\log^2\frac{n}{\Delta}}{\Delta^2}\right)$, $\Omega(\log n)$	$O\left(\frac{\log^2\frac{n}{\Delta}}{\Delta^2}\right)$
Estimating support size to $\pm\varepsilon n$	$\Theta\left(\frac{n}{\log n}\right)$			$\Theta\left(\frac{1}{\varepsilon^2}\right)$	$O\left(\frac{1}{\varepsilon^2}\right)$

While it is not clear, looking at these problems, whether the additional flexibility that the Cumulative Dual access grants over the Dual one can *unconditionally* yield strictly more sample-efficient testing algorithms, we do provide a separation between the two models in Sect. 4.2 by showing an exponential improvement in the query complexity for estimating the entropy of a distribution given the promise that the latter is (close to) monotone. This leads us to

[6] Results from [18,4,22,2,28,13,27,26,25].

suspect that for the task of testing monotonicity, under a structural assumption on the distribution, or more generally for properties intrinsically related to the underlying total order of the domain, such a speedup holds. Moreover, we stress the fact that our $\Omega\big(1/(\varepsilon_2 - \varepsilon_1)^2\big)$ lower bound for tolerant identity testing does not apply to the Cumulative Dual setting.

One of the main techniques we use for algorithms in the dual model is a general approach for estimating very efficiently any quantity of the form $\mathbb{E}_{i\sim D}\left[\Phi(i, D(i))\right]$, for any *bounded* function Φ (we note that a similar method was utilized in [6], albeit in a less systematic way). In particular, in light of our lower bounds, this technique is both an intrinsic and defining feature of the Dual model, as it gives essentially tight upper bounds for the problems we consider.

On the other hand, for the task of proving lower bounds, we no longer can take advantage of the systematic characterizations known for the sampling model (see e.g. [1], Sect. 2.4.1). For this reason, we have to rely on reductions from known-to-be-hard problems (such as estimating the bias of a coin), or prove indistinguishability in a *customized* fashion.

1.4 Organization

After the relevant definitions and preliminaries in Sect. 2, we pursue by considering the first three problems of testing equivalence of distributions in Sect. 3, where we describe our testing upper and lower bounds. We then turn to the harder problem of *tolerant* testing. Finally, we tackle in Sect. 4 the task of performing entropy, estimation, and give for the latter matching upper and lower bounds. Due to space constraints, proofs of the theorems and details of the constructions have been omitted from this extended abstract; for the full version, the reader is referred to [11].

2 Preliminaries

We consider discrete probability distributions over the subset of integers $[n] = \{1, \ldots, n\}$. As aforementioned, the notion of distance we use between distributions D_1, D_2 is their *total variation distance*, defined as

$$\mathrm{d_{TV}}(D_1, D_2) \overset{\text{def}}{=} \max_{S\subseteq[n]} \left(D_1(S) - D_2(S)\right) = \frac{1}{2}\sum_{i\in[n]} \left|D_1(i) - D_2(i)\right| .$$

Recall that any property $\mathcal{P}$ can equivalently be seen as the subset of distributions that satisfy it; in particular, the distance $\mathrm{d_{TV}}(D, \mathcal{P})$ from some D to $\mathcal{P}$ is the minimum distance to any distribution in this subset, $\min_{D'\in\mathcal{P}} \mathrm{d_{TV}}(D, D')$. Testing algorithms for distributions over $[n]$ are defined as follows:[7]

[7] Note that, as standard in property testing, the threshold 2/3 is arbitrary: any $1 - \delta$ confidence can be achieved at the cost of a multiplicative factor $\log(1/\delta)$ in the query complexity.

Definition 3. *Fix any property $\mathcal{P}$ of distributions, and let* ORACLE_D *be an oracle providing some type of access to D. A q*-query testing algorithm for $\mathcal{P}$ *is a randomized algorithm $\mathcal{T}$ which takes as input n, $\varepsilon \in (0, 1]$, as well as access to* ORACLE_D*. After making at most $q(\varepsilon, n)$ calls to the oracle, $\mathcal{T}$ outputs either* ACCEPT *or* REJECT*, such that the following holds:*

- *if $D \in \mathcal{P}$, $\mathcal{T}$ outputs* ACCEPT *with probability at least* $2/3$*;*
- *if* $\mathrm{d_{TV}}(D, \mathcal{P}) \geq \varepsilon$*, $\mathcal{T}$ outputs* REJECT *with probability at least* $2/3$*.*

We shall also be interested in *tolerant* testers – roughly, algorithms robust to a relaxation of the first item above:

Definition 4. *Fix property $\mathcal{P}$ and* ORACLE_D *as above. A q*-query tolerant testing algorithm for $\mathcal{P}$ *is a randomized algorithm $\mathcal{T}$ which takes as input n, $0 \leq \varepsilon_1 < \varepsilon_2 \leq 1$, as well as access to* ORACLE_D*. After making at most $q(\varepsilon_1, \varepsilon_2, n)$ calls to the oracle, $\mathcal{T}$ outputs either* ACCEPT *or* REJECT*, such that the following holds:*

- *if* $\mathrm{d_{TV}}(D, \mathcal{P}) \leq \varepsilon_1$*, $\mathcal{T}$ outputs* ACCEPT *with probability at least* $2/3$*;*
- *if* $\mathrm{d_{TV}}(D, \mathcal{P}) \geq \varepsilon_2$*, $\mathcal{T}$ outputs* REJECT *with probability at least* $2/3$*.*

Observe in particular that if $\mathrm{d_{TV}}(D, \mathcal{P}) \in (0, \varepsilon)$ (resp. $\mathrm{d_{TV}}(D, \mathcal{P}) \in (\varepsilon_1, \varepsilon_2)$), the tester's output can be arbitrary. Furthermore, we stress that the two definitions above only deal with the query complexity, and not the running time. However, it is worth noting that while our lower bounds hold even for such computationally unbounded algorithms, all of our upper bounds are achieved by testing algorithms whose running time is polynomial in the number of queries they make.

3 Uniformity and Identity of Distributions

3.1 Testing

In this section, we consider the three following testing problems, each of them a generalization of the previous:

Uniformity testing: given oracle access to D, decide whether $D = \mathcal{U}$ (the uniform distribution on $[n]$) or is ε-far from it;

Identity testing: given oracle access to D and the full description of a fixed D^*, decide whether $D = D^*$ or is ε-far from it;

Closeness testing: given independent oracle accesses to D_1, D_2 (both unknown), decide whether $D_1 = D_2$ or D_1, D_2 are ε-far from each other.

We begin by stating here two results from the literature that transpose straighforwardly in our setting. Observe that since the problem of testing closeness between two unknown distributions D_1, D_2 in particular encompasses the problem of identity testing to a known D^* (and *a fortiori* the problem of uniformity testing), this upper bound automatically applies to these as well.

Theorem 1 (Theorem 24 from [24]). *In the query access model, there exists a tester for identity to a known distribution D^* with query complexity $O\big(\frac{1}{\varepsilon}\big)$.*

Note that the tester given in [24] is neither tolerant nor robust; however, it only uses query access. [9] later adapt this algorithm to give a tester for closeness between two unknown distributions, in a setting which can be seen as a "relaxed" dual access model[8]:

Theorem 2 (Theorem 12 from [9]). *In the dual access model, there exists a tester for closeness between two unknown distributions D_1, D_2 with sample complexity $O\big(\frac{1}{\varepsilon}\big)$.*

It is worth noting that the algorithm in question is conceptually very simple – namely, it consists in drawing samples from both distributions and then querying the respective probability mass both distributions put on them, hoping to detect a violation.

Lower bound. Getting more efficient testing seems unlikely – the dependence on $1/\varepsilon$ being "as good as it gets". The following result formalizes this, showing that indeed both Theorems 1 and 2 are tight, even for the least challenging task of testing uniformity:

Theorem 3 (Lower bound for cumulative dual and dual testers). *Both in the dual and cumulative dual access model, any tester for uniformity must have query complexity $\Omega\big(\frac{1}{\varepsilon}\big)$.*

3.2 Tolerant Testing

In this section, we describe tolerant testing algorithms for the three problems of uniformity, identity and closeness; note that by a standard reduction (see Parnas et al. ([23], Sec. 3.1), this is equivalent to estimating the distance between the corresponding distributions. As hinted in the introduction, our algorithm relies on a general estimation approach that will be illustrated further in Section 4, and which constitutes a fundamental feature of the dual oracle: namely, the ability to estimate cheaply quantities of the form $\mathbb{E}_{i\sim D}\left[\Phi(i, D(i))\right]$ for any *bounded* function Φ.

Theorem 4. *In the dual access model, there exists a tolerant tester for uniformity with query complexity $O\big(\frac{1}{(\varepsilon_2-\varepsilon_1)^2}\big)$.*

This hinges on observing that the distance to uniformity can be rewritten as

$$\mathrm{d_{TV}}(D,\mathcal{U}) = \frac{1}{2}\sum_{i\in[n]}\left|D(i)-\frac{1}{n}\right| = \sum_{i\colon\, D(i)>\frac{1}{n}}\Big(1-\frac{1}{nD(i)}\Big)\cdot D(i) = \mathbb{E}_{i\sim D}\left[\Phi(i, D(i))\right]$$

[8] In the sense that the evaluation oracle, being simulated via another type of oracle, is not only noisy but also allowed to err on a small set of points.

for $\Phi(i, D(i)) \stackrel{\text{def}}{=} (1 - \frac{1}{nD(i)})\mathbb{1}_{\{D(i)>\frac{1}{n}\}}$. Moreover, by massaging further this equality, the result above can be easily extended to other distributions than uniform, and even to the case of two unknown distributions:

Corollary 1. *In the dual access model, there exists a tolerant tester for identity to a known distribution with query complexity* $O\big(\frac{1}{(\varepsilon_2-\varepsilon_1)^2}\big)$.

Corollary 2. *In the dual access model, there exists a tolerant tester for closeness between two unknown distributions with query complexity* $O\big(\frac{1}{(\varepsilon_2-\varepsilon_1)^2}\big)$. *As noted in the next subsection, this is optimal (up to constant factors).*

Interestingly, this tester can be made robust to multiplicative noise, i.e. can be shown to work even when the answers to the EVAL queries are only accurate up to a factor $(1+\gamma)$ for $\gamma > 0$.

Lower Bound. In this subsection, we show that the upper bounds of Lemma 4, Corollaries 1 and 2 are tight.

Theorem 5. *In the dual access model, performing* $(\varepsilon_1, \varepsilon_2)$*-testing for uniformity requires sample complexity* $\Omega\big(\frac{1}{(\varepsilon_2-\varepsilon_1)^2}\big)$ *(the bound holds even when only asking* ε_1 *to be* $\Omega(1)$*).*

4 Entropy and Support Size

4.1 Additive and Multiplicative Estimations of Entropy

In this section, we describe simple algorithms to perform both additive and multiplicative estimation (which in turns directly implies tolerant testing) of the *entropy* $H(D)$ of the unknown distribution D, defined as

$$H(D) \stackrel{\text{def}}{=} -\sum_{i\in[n]} D(i)\log D(i) \in [0, \log n]$$

We remark that Batu et al. ([6], Theorem 14) gives a similar algorithm, based on essentially the same approach but relying on a Chebyshev bound, yielding a $(1+\gamma)$-multiplicative approximation algorithm for entropy with sample complexity $O\big((1+\gamma)^2 \log^2 n/\gamma^2 h^2\big)$, given a lower bound $h > 0$ on $H(D)$.

Guha et al. ([19], Theorem 5.2) then refined their result, using as above a threshold for the estimation along with a multiplicative Chernoff bound to get the sample complexity down to $O\big(\log n/\gamma^2 h\big)$ – thus matching the $\Omega(\log n/\gamma(2+\gamma)h)$ lower bound of [6] (Theorem 18); we recall their results for multiplicative estimation of the entropy below[9].

[9] In particular, note that translating their lower bound for additive estimation implies that the dependence on n of our algorithm is tight.

Theorem 6 (Upper bound [[19], Theorem 5.2]). *Fix $\gamma > 0$. In the dual access model, there exists an algorithm that, given a parameter $h > 0$ and the promise that $H(D) \geq h$, estimates the entropy within a multiplicative $(1+\gamma)$ factor, with sample complexity $\Theta\left(\frac{\log n}{\gamma^2 h}\right)$.*

Theorem 7 (Lower bound [[6], Theorem 18]). *Fix $\gamma > 0$. In the dual access model, any algorithm that, given a parameter $h > 0$ and the promise that $H(D) = \Omega(h)$, estimates the entropy within a multiplicative $(1+\gamma)$ factor must have sample complexity $\Omega\left(\frac{\log n}{\gamma(2+\gamma)h}\right)$.*

Observe that the additive bound we give (based on a different cutoff threshold), however, still performs better in many cases, e.g. $\Delta = \gamma h > 1$ and $h > 1$; and does not require any *a priori* knowledge on a lower bound $h > 0$. Moreover, we believe that this constitutes a good illustration of the more general technique used, and a good example of what the dual model allows: approximation of quantities of the form $\mathbb{E}_{i\sim D}\left[\Phi(i, D(i))\right]$, where Φ is any *bounded* function of both an element of the domain and its probability mass under the distribution D.

Additive Estimate. The key idea is to observe that for a distribution D, the entropy $H(D)$ can be rewritten as

$$H(D) = \sum_{x\in[n]} D(x)\log\frac{1}{D(x)} = \mathbb{E}_{x\sim D}\left[\log\frac{1}{D(x)}\right] \tag{1}$$

Carefully estimating this expectation (in particular, making sure to handle appropriately the elements x for which $D(x)$ is very small, as for them the random variable $\log\frac{1}{D(x)}$ cannot be bounded) leads to the following theorem:

Theorem 8. *In the dual access model, there exists an algorithm estimating the entropy up to an additive Δ, with sample complexity $\Theta\left(\frac{\log^2\frac{n}{\Delta}}{\Delta^2}\right)$.*

or, in terms of tolerant testing (i.e., distinguishing distributions with entropy at most Δ_1 from those with entropy at least Δ_2):

Corollary 3. *In the dual access model, there exists an (Δ_1, Δ_2)-tolerant tester for entropy with sample complexity $\tilde{\Theta}\left(\frac{\log^2 n}{(\Delta_1-\Delta_2)^2}\right)$.*

4.2 Additive Estimation of Entropy for Monotone Distributions

In the previous section, we saw how to obtain an additive estimate of the entropy of the unknown distribution, using essentially $O\left(\log^2 n\right)$ sampling and evaluation queries; moreover, this dependence on n is optimal. However, one may wonder if, by taking advantage of *cumulative* queries, it becomes possible to obtain a better query complexity. We partially answer this question, focusing on a particular class of distributions for which the cumulative dual query access seems particularly well-suited: namely the class of *monotone* distributions[10].

Before describing how this assumption can be leveraged to obtain an exponential improvement in the sample complexity for cumulative dual query algorithms, we first show that given only *dual* access to a distribution promised to be $o(1)$-close to monotone, no such speedup can hold. By establishing (see Remark 1) that the savings obtained for (close to) monotone distributions are only possible with cumulative dual access, this will yield a separation between the two oracles, proving the latter is strictly more powerful.

Theorem 9. *In the dual access model, any algorithm that estimates the entropy of distributions $O(1/\log n)$-close to monotone to an additive constant must make $\Omega(\log n)$ queries to the oracle.*

Upper Bound: Exponential Speedup for Cumulative Dual Oracles. We now establish the positive result in the case of algorithms given cumulative dual query access. Note that Batu et al. [6] already consider the problem of getting a (multiplicative) estimate of the entropy of D, under the assumption that the distribution is monotone; and describe (both in the evaluation-only and sample-only models) polylog(n)-query algorithms for this task, which work by recursively splitting the domain in a suitable fashion to get a partition into near uniform and negligible intervals.

The main insight here (in addition to the mere fact that we allow ourself a stronger type of access to D) is to use, instead of an *ad hoc* partition of the domain, a specific one tailored for monotone distributions, introduced by Birgé [7] – and which crucially *does not depend on the distribution itself.* By using this oblivious decomposition of the domain into $\ell = o(n)$ intervals, we can set out to approximate the entropy of the induced *flat* distribution (that we can efficiently simulate from the cumulative dual oracles). This roughly reduces the complexity parameter from n to ℓ; it only remains to use our previous approach, slightly adapted, on this flat distribution. Of course, we have to be careful not to incur too much a loss at each step, when first approximating $H(D)$ by $H(\bar{D})$, and then specifying our cutoff threshold to only keep significant contributions to $H(\bar{D})$.

Theorem 10. *In the cumulative dual access model, there exists an algorithm for monotone distributions estimating the entropy up to an additive Δ, with sample complexity $\tilde{O}\left(\log^2 \frac{\log n}{\Delta} / \Delta^2\right)$.*

Remark 1. We remark that the above result still applies if D is only guaranteed to be $O(1/\log n)$-*close* to monotone; indeed, as the "Birgé decomposition" is (crucially) robust, $\bar{D}$ will still be $O(\varepsilon)$-close to D.

[10] Recall that a distribution D over a totally ordered domain is said to be monotone if $x \preceq y$ implies $D(x) \geq D(y)$.

References

1. Bar-Yossef, Z.: The Complexity of Massive Data Set Computations. Ph.D. thesis, UC Berkeley (2002), Christos Papadimitriou
2. Batu, T., Fischer, E., Fortnow, L., Kumar, R., Rubinfeld, R., White, P.: Testing random variables for independence and identity. In: Proceedings of FOCS, pp. 442–451 (2001)
3. Batu, T., Fortnow, L., Rubinfeld, R., Smith, W.D., White, P.: Testing that distributions are close. In: Proceedings of FOCS, pp. 189–197 (2000)
4. Batu, T., Fortnow, L., Rubinfeld, R., Smith, W.D., White, P.: Testing closeness of discrete distributions. Tech. Rep. abs/1009.5397, ArXiv (2010), this is a long version of [3]
5. Batu, T., Kumar, R., Rubinfeld, R.: Sublinear algorithms for testing monotone and unimodal distributions. In: Proceedings of STOC, pp. 381–390 (2004)
6. Batu, T., Dasgupta, S., Kumar, R., Rubinfeld, R.: The complexity of approximating the entropy. SIAM Journal on Computing 35(1), 132–150 (2005)
7. Birgé, L.: On the risk of histograms for estimating decreasing densities. The Annals of Statistics 15(3), 1013–1022 (1987)
8. Blum, A., Ligett, K., Roth, A.: A learning theory approach to noninteractive database privacy. J. ACM 60(2), 12 (2013)
9. Canonne, C., Ron, D., Servedio, R.A.: Testing probability distributions using conditional samples. Tech. Rep. abs/1211.2664, ArXiV (November 2012)
10. Canonne, C., Ron, D., Servedio, R.A.: Testing equivalence between distributions using conditional samples. In: Proceedings of SODA (2014)
11. Canonne, C.L., Rubinfeld, R.: Testing probability distributions underlying aggregated data. Electronic Colloquium on Computational Complexity (ECCC) 21, 21 (2014)
12. Chakraborty, S., Fischer, E., Goldhirsh, Y., Matsliah, A.: On the power of conditional samples in distribution testing. In: Proceedings of the 4th Conference on Innovations in Theoretical Computer Science, ITCS 2013, pp. 561–580. ACM, New York (2013)
13. Chan, S.O., Diakonikolas, I., Valiant, G., Valiant, P.: Optimal Algorithms for Testing Closeness of Discrete Distributions. In: Proceedings of SODA (2014)
14. Daskalakis, C., Diakonikolas, I., Servedio, R., Valiant, G., Valiant, P.: Testing *k*-modal distributions: Optimal algorithms via reductions. In: Proceedings of SODA (2013)
15. Dinur, I., Nissim, K.: Revealing information while preserving privacy. In: Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 202–210. ACM (2003)
16. Dwork, C.: Differential privacy: A survey of results. In: Agrawal, M., Du, D.-Z., Duan, Z., Li, A. (eds.) TAMC 2008. LNCS, vol. 4978, pp. 1–19. Springer, Heidelberg (2008)
17. Dwork, C., Nissim, K.: Privacy-preserving datamining on vertically partitioned databases. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 528–544. Springer, Heidelberg (2004)
18. Goldreich, O., Ron, D.: On testing expansion in bounded-degree graphs. Tech. Rep. TR00-020, ECCC (2000)
19. Guha, S., McGregor, A., Venkatasubramanian, S.: Streaming and sublinear approximation of entropy and information distances. CoRR abs/cs/0508122 (2005)

20. Indyk, P., Levi, R., Rubinfeld, R.: Approximating and Testing k-Histogram Distributions in Sub-linear Time. In: Proceedings of PODS, pp. 15–22 (2012)
21. Ma, S.K.: Calculation of entropy from data of motion. Journal of Statistical Physics 26(2), 221–240 (1981), `http://dx.doi.org/10.1007/BF01013169`
22. Paninski, L.: A coincidence-based test for uniformity given very sparsely sampled discrete data. IEEE-IT 54(10), 4750–4755 (2008)
23. Parnas, M., Ron, D., Rubinfeld, R.: Tolerant property testing and distance approximation. J. Comput. Syst. Sci. 72(6), 1012–1042 (2006)
24. Rubinfeld, R., Servedio, R.A.: Testing monotone high-dimensional distributions. RSA 34(1), 24–44 (2009)
25. Valiant, G., Valiant, P.: A CLT and tight lower bounds for estimating entropy. Tech. Rep. TR10-179, ECCC (2010)
26. Valiant, G., Valiant, P.: Estimating the unseen: A sublinear-sample canonical estimator of distributions. Tech. Rep. TR10-180, ECCC (2010)
27. Valiant, G., Valiant, P.: Estimating the unseen: an $n/\log(n)$-sample estimator for entropy and support size, shown optimal via new CLTs. In: Proceedings of STOC, pp. 685–694 (2011); see also [25] and [26]
28. Valiant, P.: Testing symmetric properties of distributions. SICOMP 40(6), 1927–1968 (2011)

Parallel Repetition of Entangled Games with Exponential Decay via the Superposed Information Cost

André Chailloux[1] and Giannicola Scarpa[2]

[1] SECRET Project Team, INRIA Paris-Rocquencourt
[2] CWI, Amsterdam

Abstract. In a two-player game, two cooperating but non communicating players, Alice and Bob, receive inputs taken from a probability distribution. Each of them produces an output and they win the game if they satisfy some predicate on their inputs/outputs. The entangled value $\omega^*(G)$ of a game G is the maximum probability that Alice and Bob can win the game if they are allowed to share an entangled state prior to receiving their inputs.

The n-fold parallel repetition G^n of G consists of n instances of G where the players receive all the inputs at the same time and produce all the outputs at the same time. They win G^n if they win each instance of G.

In this paper we show that for any game G such that $\omega^*(G) = 1 - \varepsilon < 1$, $\omega^*(G^n)$ decreases exponentially in n. First, for any game G on the uniform distribution, we show that $\omega^*(G^n) = (1-\varepsilon^2)^{\Omega\left(\frac{n}{\log(|I||O|)} - |\log(\varepsilon)|\right)}$, where $|I|$ and $|O|$ are the sizes of the input and output sets. From this result, we show that for any entangled game G, $\omega^*(G^n) = (1-\varepsilon^2)^{\Omega(\frac{n}{Q^4\log(Q\cdot|O|)} - |\log(\varepsilon/Q)|)}$ where p is the input distribution of G and $Q = \max(\lceil\frac{1}{\min_{xy:p_{xy}\neq 0}\{\sqrt{p_{xy}}\}}\rceil, |I|)$.

To prove this parallel repetition, we introduce the concept of *Superposed Information Cost* for entangled games which is inspired from the information cost used in communication complexity.

1 Introduction

A *two-player (nonlocal) game* is played between two cooperating parties Alice and Bob which are not allowed to communicate. This game G is characterized by an input set I, an output set O, a probability distribution p on I^2 and a result function $V : O^2 \times I^2 \to \{0,1\}$. The game proceeds as follows: Alice receives $x \in I$, Bob receives $y \in I$ where (x, y) is taken according to p. Alice outputs $a \in O$ and Bob outputs $b \in O$. They win the game if $V(a, b|x, y) = 1$. The value of the game $\omega(G)$ is the maximum probability, over all strategies, with which Alice and Bob can win the game.

The n-fold parallel repetition G^n of G consists of the following. Alice and Bob get inputs $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$, respectively. Each (x_i, y_i) is taken independently according to p. They output $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$, respectively.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 296–307, 2014.

They win the game if and only if $\forall i,\ V(a_i, b_i|x_i, y_i) = 1$. In order to win the n-fold repetition, Alice and Bob can just take the best strategy for G and use it n times. If they do so, they will win G^n with probability $(\omega(G))^n$ which shows that $\omega(G^n) \geq (\omega(G))^n$.

Parallel repetition of games studies how the quantity $\omega(G^n)$ behaves. For example, if $\omega(G^n) = (\omega(G))^n$ for each n then we say that G admits perfect parallel repetition. However, there are some games for which this does not hold, for example the CHSH game [8] repeated two times. It was a long-standing open question to determine whether the value of $\omega(G^n)$ decreases exponentially in n. This was first shown by Raz [20]. Afterwards, a series of works showed improved results for specific types of games [13,19,1]. Parallel repetition for games has many applications, from direct product theorems in communication complexity [18] to hardness of approximation results [3,11,12].

In the quantum setting, it is natural to consider games where Alice and Bob are allowed to share some entangled state at the beginning of the game. In this case we talk about entangled strategies. The maximum probability that Alice and Bob can win a game G, over all the entangled strategies, is the entangled value $\omega^*(G)$. Some entangled games are witnesses for the phenomenon of quantum nonlocality, as they are special cases of the so-called Bell inequality violations. (We have a Bell inequality violation whenever $\omega^*(G) > \omega(G)$.) The study of entangled games is also greatly related to our understanding of quantum entanglement.

Perfect parallel repetition has been shown for entangled XOR games [9]. It was also shown that entangled unique games [15] admit parallel repetition with exponential decay. Finally, it was shown that any entangled game admits (a variant of) parallel repetition [16]. However, this last parallel repetition only shows a polynomial decay of $\omega^*(G^n)$. It was unknown for a large class of games whether this decay is exponential or not. Very recently two more works have been presented: a parallel repetition result with exponential decay for entangled projection games [10] and an independent work [14] similar to this one.

1.1 Contribution

The main contribution of this paper is the following theorem.

Theorem 1. *For any game G on the uniform distribution with $\omega^*(G) \leq 1 - \varepsilon$, we have:*

$$\omega^*(G^n) = (1-\varepsilon^2)^{\Omega\left(\frac{n}{\log(|I||O|)} - |\log(\varepsilon)|\right)}.$$

where $|I|$ and $|O|$ are respectively the size of the input and the output sets.

The class of entangled games with a uniform distribution is a large class of entangled games for which such parallel repetition was unknown. We can extend this result to any entangled game.

Corollary 1. *For any game G such that $\omega^*(G) \leq 1 - \varepsilon$, we have that*

$$\omega^*(G^n) = (1-\varepsilon^2)^{\Omega\left(\frac{n}{Q^4 \log(Q\cdot|O|)} - |\log(\varepsilon/Q)|\right)},$$

where $|O|$ is the size of the output set of G and $Q = \max(\lceil \frac{1}{\min_{xy:p_{xy}\neq 0}\{\sqrt{p_{xy}}\}} \rceil, |I|)$.

This corollary can be obtained directly from the previous theorem. The above corollary is the first general parallel repetition theorem for any entangled game with exponential decay. It is not as strong as usual parallel repetition theorems with exponential decay because of this dependency on Q. Notice however that Q depends only on the game G and not on n.

In order to prove this theorem, we introduce the concept of *Superposed Information Cost* of a game, an insightful concept and the cornerstone of our proof.

1.2 Superposed Information Cost

This concept is derived from the notion of information cost widely used in communication complexity [7,2,4,17]. In the setting of communication complexity, we consider a function $f(x, y)$ and suppose that Alice has some input x and Bob some input y. They want to determine the outcome of $f(x, y)$ for a certain function f with the minimal amount of communication. The interactive information cost IC of f describes the least amount of information that Alice and Bob need to have about each other's inputs in order to compute $f(x, y)$.

We want to follow a similar approach for entangled games. In entangled games, the quantum state Alice and Bob share is independent of the inputs x, y. We now give extra resources to Alice and Bob: advice states. Alice and Bob are given an advice state $|\phi_{xy}\rangle$ that can depend on their inputs. This can greatly increase their winning probability. For example, Alice could have perfect knowledge of Bob's input y, and vice-versa.

We define (informally) the information cost of a game as follows:

Information Cost for entangled games
Alice and Bob are given advice states $|\phi_{xy}\rangle$ to share that can depend on their inputs. What is the minimal amount of information that these states have to give Alice and Bob about each other's input, in order to allow them to win the game with probability 1?

This is a natural extension of the information cost to entangled games. However, it is a limited notion since we cannot relate it to the entangled value of the game. (A simple counterexample can be obtained from the CHSH game.) Therefore, we extended this notion to the case where we allow the players to be *in a superposition of their inputs.*

Superposed Information Cost (SIC) for entangled games
We extend the notion of information cost by allowing the players to have a superposition of their inputs. We then consider the amount of information that advice states have to give Alice and Bob about each other's input, in order to allow them to win with probability 1.

These notions are defined precisely in Section 3.1.

Lower Bounding the Value of Entangled Games Using the Superposed Information Cost. The reason we introduce the superposed information cost for entangled games is that we want to have an information theoretic characterization of the value of entangled games. The next theorem states that the value of any entangled game on the uniform distribution can be lower bounded by the superposed information cost (this does not hold for the non-superposed one).

Theorem 2. *For any game G with a uniform input distribution, we have $SIC(G) \geq \frac{1-\omega^*(G)}{32\ln(2)}$ or equivalently $\omega^*(G) \geq 1 - 32\ln(2) \cdot SIC(G)$.*

The Superposed information cost is additive under parallel repetition:

Proposition 1. *$SIC(G^n) = nSIC(G)$.*

Putting these two results together, we have $SIC(G^n) \geq \frac{n(1-\omega^*(G))}{32\ln(2)}$. This result shows that $SIC(G^n)$ is large when n increases and can be seen as evidence that the game G^n is hard to win and that $\omega^*(G^n)$ decreases fast.

Using SIC to Show our Parallel Repetition Theorem. We fix a game G with $\omega^*(G) = 1 - \varepsilon$ and $\omega^*(G^n) = 2^{-t}$ for some t. In order to prove our theorem, we consider a quantity S which is strongly related to $SIC(G^n)$. We show that

$$\Omega(n\varepsilon) \leq S \leq O\left(\frac{t\log(|I||O|)}{\varepsilon}\right). \tag{1}$$

The lower bound is a natural extension of the above argument about the additivity of SIC. The ingredient we need to show the upper bound is the following *communication task*:

- The players use an optimal strategy for G^n and win with probability $\omega^*(G^n) = 2^{-t}$.
- Alice sends $m = O(\frac{t\log(|I||O|)}{\varepsilon})$ bits to Bob.
- Using this message, Bob's goal is to determine with high probability whether they won most of the games or not.

Switching to a communication task and to a related quantity S seems much weaker than showing directly an upper bound on $SIC(G^n)$, but it will be enough for us. Combining these two results, we conclude that $t = \Omega(\frac{n\varepsilon^2}{\log(|I||O|)})$ or equivalently, for ε close to 0, $\omega^*(G^n) = (1-\varepsilon^2)^{\Omega(\frac{n}{\log(|I||O|)})}$.

1.3 Organization of the Paper

In this extended abstract we give the main arguments and proof sketches. We refer to the full paper [6] for the complete proofs. Section 2 contains preliminaries about entangled games. In Section 3, we define the key concept of the superposed information cost for a game and show that this quantity is additive when repeating games in parallel. In Section 4, we provide a brief organization of the main proof. In Section 5, we show Theorem 2 and some generalizations. In Section 6 we derive the upper bound of (1) (the lower bound is proven in the main paper). Finally, in Section 7 we prove our main theorem.

2 Entangled Games

The Value of an Entangled Game

Definition 1. *An entangled game $G = (I, O, V, p)$ is defined by finite input and output sets I and O as well as an accepting function $V : O^2 \times I^2 \rightarrow \{0, 1\}$ and a probability distribution $p : I^2 \rightarrow [0, 1]$.*

A strategy for the game proceeds as follows. Alice and Bob can share any quantum state. Then, Alice receives an input $x \in I$ and Bob receives an input $y \in I$ where these inputs are sampled according to p. They can perform any quantum operation but are not allowed to communicate. Alice outputs $a \in O$ and Bob outputs $b \in O$. They win the game if $V(a, b|x, y) = 1$.

The *entangled value* of a game G is the maximal probability with which Alice and Bob can win the game. From standard purification techniques, we have that w.l.o.g., Alice and Bob share a pure state $|\phi\rangle$ and their optimal strategy consists of projective measurements $A^x = \{A^x_a\}_{a\in O}$ and $B^y = \{B^y_b\}_{b\in O}$ on $|\phi\rangle$. This means that after receiving their inputs, they share a state of the form $\rho = \sum_{x,y\in I} p_{xy}|x\rangle\langle x| \otimes |\phi\rangle\langle\phi| \otimes |y\rangle\langle y|$, for some state $|\phi\rangle$.

Definition 2. *The entangled value of a game G is* $\omega^*(G) = \sup_{|\phi\rangle,A^x,B^y} \sum_{x,y,a,b} p_{xy}V(a, b|x, y)\langle\phi|A^x_a \otimes B^y_b|\phi\rangle$.

Definition 3. *A game $G = (I, O, V, p)$ is on the uniform distribution if $I = [k]$ for some k and $\forall x, y \in [k]$, $p_{xy} = \frac{1}{k^2}$. We write $p =$ Unif. when this is the case.*

Value of a Game with Advice States. Consider a game $G = (I, O, V, p)$. We are interested in the value of the game when the two players share an advice state $|\phi_{xy}\rangle$ additionally to their inputs x, y. This means that Alice and Bob share a state of the form $\rho = \sum_{x,y,a,b} p_{xy}|x\rangle\langle x| \otimes |\phi_{xy}\rangle\langle\phi_{xy}| \otimes |y\rangle\langle y|$.

Definition 4. *The entangled value of G, given that Alice and Bob share the above state ρ is $\omega^*(G|\rho) = \max_{A^x,B^y} \sum_{x,y} p_{xy}V(a, b|x, y)\langle\phi_{xy}|A^x_a \otimes B^y_b|\phi_{xy}\rangle$.*

Repetition of Entangled Games. In the n-fold parallel repetition of a game G, each player gets n inputs from I and must produce n outputs from O. Each instance of the game will be evaluated as usual by the function V. The players win the parallel repetition game if they win *all* the instances. More formally, for a game $G = (I, O, V, p)$ we define $G^n = (I', O', V', q)$, where $I' = I^{\times n}, O' = O^{\times n}, q_{xy} = \Pi_{i\in[n]} p_{x_i,y_i}$ and $V'(a, b|x, y) = \Pi_{i\in[n]} V(a_i, b_i|x_i, y_i)$. While playing G^n, we say that Alice and Bob win game i if $V(a_i, b_i|x_i, y_i) = 1$.

Majority Game. For a game $G = (I, O, V, p)$ and a real number $\alpha \in [0, 1]$ we define $G^n_\alpha = (I', O', V', p')$ as follows: $I' = I^{\times n}$, $O' = O^{\times n}$, $p'_{xy} = \Pi_{i \in [n]} p_{x_i,y_i}$ as in G^n. We define $V'(a, b|x, y) = 1 \Leftrightarrow \#\{i : V(a_i, b_i|x_i, y_i) = 1\} \geq \alpha n$.

3 Advice States, Superposed Players and Information Cost

The notion of information cost has been very useful for communication complexity. Here we derive a similar notion for entangled games.

Consider a game with advice state as defined in Section 2. The advice state can potentially greatly help the players. For example, Alice could know y and Bob could know x. We ask ourselves the following question:
For a game $G = (I = [k], O, V, p)$ such that $\omega^*(G) = 1 - \varepsilon < 1$ and a state $\rho = \sum_{x,y \in [k]} p_{xy} |x\rangle\langle x|_{\mathcal{X}} \otimes |\phi_{xy}\rangle\langle\phi_{xy}|_{\mathcal{AB}} \otimes |y\rangle\langle y|_{\mathcal{Y}}$, what is the minimum dependency that the states $\{|\phi_{xy}\rangle\}_{xy}$ must have on x, y to have $\omega^*(G|\rho) = 1$?

There are different ways of characterizing this dependency on x, y. A first possibility would be to consider the information that Alice has about y and Bob has about x while sharing ρ. However, there are cases where Alice and Bob can win a game with probability 1 using an advice state while still not learning anything about each other's input. For example, take the CHSH game [8] and consider the states $|\phi_{00}\rangle = |\phi_{01}\rangle = |\phi_{10}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)_{\mathcal{AB}}$ and $|\phi_{11}\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$. If the two players share the state $\rho = \sum_{x,y \in \{0,1\}} 1/4 |x\rangle\langle x|_{\mathcal{X}} \otimes |\phi_{xy}\rangle\langle\phi_{xy}|_{\mathcal{AB}} \otimes |y\rangle\langle y|_{\mathcal{Y}}$, Alice has no information about y and Bob has no information about x. On the other hand, if both players measure their registers $\mathcal{A}$ and $\mathcal{B}$ in the computational basis and output the results, they will win the CHSH game with probability 1 hence $\omega^*(CHSH|\rho) = 1$ while $\omega^*(CHSH) = \cos^2(\pi/8)$.

We must consider a slightly different scenario so that Alice or Bob can learn something about the other player's input. When considering the amount of information that Alice has about Bob's input y, we allow Alice to have a coherent superposition of her inputs. Similarly, we will be interested in the amount of information Bob has about x when he has a coherent superposition of his inputs.

This scenario is motivated as follows: if Alice and Bob have a common procedure to create $|\phi_{xy}\rangle$ from their respective inputs x and y, Alice can create a superposition of her inputs and they can perform the same procedure. This scenario has for example been in order to show optimal bounds for quantum bit commitment [5].

This approach leads to the definition of the superposed information cost of a game. In the next section, we give formal definitions of this notion.

3.1 The Superposed Information Cost

Consider a family of states $\{|\phi_{xy}\rangle\}_{xy}$ and a probability distribution $\{p_{xy}\}_{xy}$. Let $p_{x\cdot} = \sum_y p_{xy}$ and $p_{\cdot y} = \sum_x p_{xy}$.

Let $|L^B_x\rangle = \frac{1}{\sqrt{p_{x\cdot}}}\sum_y \sqrt{p_{xy}}|\phi_{xy}\rangle|y\rangle$ and $|L^A_y\rangle = \frac{1}{\sqrt{p_{\cdot y}}}\sum_x \sqrt{p_{xy}}|x\rangle|\phi_{xy}\rangle$. Consider the two superposed states:

$$\sigma^A = \sum_{y\in[k]} p_{\cdot y}|L^A_y\rangle\langle L^A_y|_{\mathcal{XAB}} \otimes |y\rangle\langle y|_{\mathcal{Y}}$$
$$\sigma^B = \sum_{x\in[k]} p_{x\cdot}|x\rangle\langle x|_{\mathcal{X}} \otimes |L^B_x\rangle\langle L^B_x|_{\mathcal{ABY}}.$$

Here σ^A (resp. σ^B) corresponds to ρ where Alice's input (resp. Bob's input) is put in a coherent superposition. We first define the superposed information cost of a family of states with a probability distribution.

Definition 5. *The superposed information cost* $SIC(\{|\phi_{xy}\rangle, p_{xy}\}_{xy})$ *is defined as* $SIC(\{|\phi_{xy}\rangle, p_{xy}\}_{xy}) = I(Y:XA)_{\sigma^A} + I(X:BY)_{\sigma^B}$.

Remark: This definition has good properties when the input distribution is a product distribution or close to a product distribution. One may want to consider a more general definition when considering any distribution.

We also define the superposed information cost of a shared state ρ of the form $\rho = \sum_{xy\in[k]} p_{xy}|x\rangle\langle x| \otimes |\phi_{xy}\rangle\langle\phi_{xy}| \otimes |y\rangle\langle y|$.

Definition 6. $SIC(\rho) = \inf\{SIC(\{|\phi_{xy}\rangle, p_{xy}\}_{xy})\}$ *where the infimum is taken over all families* $\{|\phi_{xy}\rangle, p_{xy}\}_{xy}$ *s.t.* $\rho = \sum_{xy\in[k]} p_{xy}|x\rangle\langle x| \otimes |\phi_{xy}\rangle\langle\phi_{xy}| \otimes |y\rangle\langle y|$.

Remark: notice that a state ρ doesn't uniquely define states $\{|\phi_{xy}\rangle, p_{xy}\}_{xy}$ because it doesn't capture the phases in the states $|\phi_{xy}\rangle$. We now define the superposed information cost of an entangled game.

Definition 7. *For any entangled game* $G = (I, O, V, p)$*, we define* $SIC(G) = \inf\{SIC(\{|\phi_{xy}\rangle\}_{xy}, \{p_{xy}\}_{xy})\}$ *where the infimum is taken over all* $(\{|\phi_{xy}\rangle\}_{xy}, \{p_{xy}\}_{xy})$ *such that the associated state* $\rho = \sum_{xy} p_{xy}|x\rangle\langle x| \otimes |\phi_{xy}\rangle\langle\phi_{xy}| \otimes |y\rangle\langle y|$ *satisfies* $\omega^*(G|\rho) = 1$.

The superposed information cost behaves nicely under parallel repetition. In the full paper, we show

Proposition 2. *For any game* G*, we have* $SIC(G^n) = n \cdot SIC(G)$.

4 Organisation of the Proof of Theorem 1

In Section 5, we show how to use the *Superposed Information Cost* of a game G to bound its entangled value $\omega^*(G)$. We first show:

Theorem 2. *For any game* G *on the uniform distribution,* $SIC(G) \geq \frac{1-\omega^*(G)}{32\ln(2)}$.

We also extend this theorem as follows:

Theorem 3. *There exists a small constant* c_0 *such that for any game* $G = (I = [k], O, V, Unif.)$ *satisfying* $\omega^*(G) = 1 - \varepsilon$*, for any game* $G' = (I = [k], O, V, p)$ *satisfying* $\frac{1}{2}\sum_{x,y}|p_{xy} - \frac{1}{k^2}| \leq c_0\varepsilon$ *and any state* $\rho = \sum_{xy} p_{xy}|x\rangle\langle x| \otimes |\phi_{xy}\rangle\langle\phi_{xy}| \otimes |y\rangle\langle y|$ *such that* $\omega^*(G'|\rho) \geq 1 - \frac{\varepsilon}{4}$*, we have that* $SIC(\rho) = \Omega(\varepsilon)$.

If $\omega^*(G) = 1-\varepsilon$, Theorem 2 claims that $SIC(G) \geq \frac{\varepsilon}{32\ln(2)}$ which gives by additivity of the superposed information cost that $SIC(G^n) \geq \frac{n\varepsilon}{32\ln(2)}$. Ideally, we would like to upper bound $SIC(G^n)$ with a function of $\omega^*(G^n)$. Unfortunately, we are not able to do this directly. In Section 6, we show the following weaker statement:

Theorem 4. *Consider a game $G = (I, O, V, Unif.)$ such that $\omega^*(G) = 1-\varepsilon$ and $\omega^*(G^n) = 2^{-t}$. Let $G^n_{1-\varepsilon/32} = (I^n, O^n, V', Unif.)$ as defined in Section 2. There exists a game $G' = (I^n, O^n, V', p)$ and a state $\xi = \sum_{xy} p_{xy}|x\rangle\langle x| \otimes |\phi_{xy}\rangle\langle\phi_{xy}| \otimes |y\rangle\langle y|$ satisfying the following properties:*

1. $H(XY)_\xi \geq 2n\log(k) - t - 1$
2. $\omega^*(G'|\xi) \geq 1 - \varepsilon/32$
3. $SIC(\xi) \leq \frac{32\log(|I||O|)}{\varepsilon}((t+1) + |\log(\varepsilon)| + 5) + 2t + 2.$

The first condition states that p is in some sense close to the uniform distribution hence G' is close to $G^n_{1-\varepsilon/32}$. This theorem is weaker than an upper bound on $SIC(G')$ which itself is weaker than an upper bound on $SIC(G^n)$, but this kind of upper bound will be enough.

In the full paper, we prove a matching lower bound.

Theorem 5. *Consider a game $G = (I = [k], O, V, Unif.)$ such that $\omega^*(G) = 1-\varepsilon$ and $\omega^*(G^n) = 2^{-t}$ with $t = o(n\varepsilon)$. Let also $G^n_{1-\varepsilon/32} = (I^n = [k^n], O^n, V', Unif.)$ as defined in Section 2. For any game $G' = (I' = [k^n], O', V', p)$ and any state $\rho = \sum_{x,y\in[k^n]} p_{xy}|x\rangle\langle x|_{\mathcal{X}} \otimes |\phi_{xy}\rangle\langle\phi_{xy}|_{\mathcal{AB}} \otimes |y\rangle\langle y|_{\mathcal{Y}}$, satisfying*

1. $H(XY)_\rho \geq 2n\log(k) - t - 1$
2. $\omega^*(G'|\rho) \geq 1 - \varepsilon/32$

we have $SIC(\rho) \geq \Omega(n\varepsilon)$.

In Section 7, we show how to use the two above theorems to conclude:

Theorem 1. *For any game $G = (I, O, V, Unif.)$ with $\omega^*(G) \leq 1-\varepsilon$, we have $\omega^*(G^n) = (1-\varepsilon^2)^{\Omega\left(\frac{n}{\log(|I||O|)} - |\log(\varepsilon)|\right)}$.*

5 Overview of Theorem 2

Theorem 2. *For any game G on the uniform distribution, $SIC(G) \geq \frac{1-\omega^*(G)}{32\ln(2)}$.*

We sketch the proof as follows. We fix a game $G = (I = [k], O, V, \text{Unif.})$ and a state $\rho = \sum_{x,y} \frac{1}{k^2}|x\rangle\langle x|_{\mathcal{X}} \otimes |\phi_{xy}\rangle\langle\phi_{xy}|_{\mathcal{AB}} \otimes |y\rangle\langle y|_{\mathcal{Y}}$ such that $\omega^*(G|\rho) = 1$. As in Section 3.1, we define $|L^A_y\rangle, |L^B_x\rangle, \sigma^A, \sigma^B$. Let $\rho^A_y = \mathrm{Tr}_{\mathcal{B}}|L^A_y\rangle\langle L^A_y|$ and $\rho^B_x = \mathrm{Tr}_{\mathcal{A}}|L^B_x\rangle\langle L^B_x|$. Intuitively, ρ^A_y (resp. ρ^B_x) corresponds to the input-superposed state that Alice (resp. Bob) has, conditioned on Bob getting y (resp. Alice getting x). Let F denote the fidelity of quantum states. We prove the following three inequalities.

1. First we show that $SIC(\rho) \geq \frac{1}{4\ln(2)}(1 - \frac{1}{k^2}\sum_{y,y'} F^2(\rho^A_y, \rho^A_{y'}) + 1 - \frac{1}{k^2}\sum_{x,x'} F^2(\rho^B_x, \rho^B_{x'}))$
2. Then we show that
$$1 - \frac{1}{k^2}\sum_{y,y'} F^2(\rho^A_y, \rho^A_{y'}) + 1 - \frac{1}{k^2}\sum_{x,x'} F^2(\rho^B_x, \rho^B_{x'}) \geq \frac{1}{8}(1 - \max_{|\Omega\rangle} \sum_{x,y\in[k]} \frac{1}{k^2}|\langle\Omega|U_x \otimes V_y|\phi_{xy}\rangle|^2)$$
for some (sets of) unitaries $\{U_x\}_x, \{V_y\}_y$.
3. Finally, we show that $(1-\max_{|\Omega\rangle}\sum_{x,y\in[k]} \frac{1}{k^2}|\langle\Omega|U_x \otimes V_y|\phi_{xy}\rangle|^2) \geq 1-\omega^*(G)$.

Putting the three inequalities together, we get

$$\begin{aligned} SIC(\rho) &\geq \frac{1}{4\ln(2)}(1 - \frac{1}{k^2}\sum_{y,y} F^2(\rho^A_y, \rho^A_{y'}) + 1 - \frac{1}{k^2}\sum_{x,x'} F^2(\rho^B_x, \rho^B_{x'})) \\ &\geq \frac{1}{32\ln(2)}(1 - \max_{|\Omega\rangle} \sum_{x,y\in[k]} \frac{1}{k^2}|\langle\Omega|U_x \otimes V_y|\phi_{xy}\rangle|^2) \quad \text{for some } \{U_x\}_x\{V_y\}_y \\ &\geq \frac{1-\omega^*(G)}{32\ln(2)}. \end{aligned}$$

Since this holds for any ρ satisfying $\omega^*(G|\rho) = 1$, we have $SIC(G) \geq \frac{1-\omega^*(G)}{32\ln(2)}$.

6 Overview of Theorem 4

In this section we sketch the proof of Theorem 4. The construction of the state ξ will directly be inspired by a communication task that we now present.

The Communication Task. Fix a game $G = (I, O, V, \text{Unif.})$ satisfying $\omega^*(G) = 1 - \varepsilon$. Let $G^n = (I^n, O^n, V_n, \text{Unif.})$ such that $\omega^*(G^n) = 2^{-t}$ for some t. We now consider the following task $H(p, m)$.

Task H(p, m)

- Alice and Bob are allowed to share any quantum state $|\phi\rangle$.
- Alice and Bob get inputs $x = x_1, \ldots, x_n$ and $y = y_1, \ldots, y_n$, with $x, y \in I^n$, following the uniform distribution.
- Alice is allowed to send m bits to Bob
- Then Alice outputs some value $a \in O^n$ and Bob outputs some value $b \in O^n$ or 'Abort'.

For each index i, we say that Alice and Bob win game i if Bob does not abort and $V(a_i, b_i|x_i, y_i) = 1$. We require the following

1. Pr[Bob does not abort] $\geq p$
2. Pr[Alice and Bob win $\geq (1 - \varepsilon/32)n$ games | Bob does not abort] $\geq (1 - \varepsilon/32)$.

Showing how to perform this task with a small amount of communication is a first step towards the construction of ξ. We consider the following protocol P that efficiently performs this task.

Protocol P for the task H(p,m)

1. Let $v \leq n$ be an integer, to be determined at the end of this section. Alice and Bob have shared randomness that correspond to v random (not necessarily different) indices $i_1, \ldots, i_v \in [n]$ as well as a state $|\phi\rangle$ that allows them to win G^n with probability at least $\frac{\omega^*(G^n)}{2} = 2^{-(t+1)}$.
2. Alice and Bob receive uniform inputs x, y. They perform a strategy that wins all n games with probability $2^{-(t+1)}$ and have some outputs $a = a_1, \ldots, a_n$ and $b = b_1, \ldots, b_n$.
3. For each index $i \in \{i_1, \ldots, i_v\}$, Alice sends x_i and a_i to Bob.
4. For each of these indices i, Bob looks at x_i, y_i, a_i, b_i and checks whether they win on all of these v games, *i.e.* , he checks that for all these indices, $V(a_i, b_i|x_i, y_i) = 1$.
5. If they do win on all of these games, Bob outputs b. Otherwise, Bob outputs 'Abort'.

Proposition 3. *The above protocol performs the task $H(p, m)$ with $p \geq 2^{-(t+1)}$ and $m = \frac{32\log(|I||O|)}{\varepsilon}((t+1) + |\log(\varepsilon)| + 5)$.*

The proof is in the full paper.

Using the Communication Task to Prove Theorem 4

The idea is the following: Alice and Bob perform protocol P for the task $H(p, m)$ performing everything in superposition, including the messages and their shared randomness. The advice state we consider is the state ρ_{NA} Alice and Bob share conditionned on Bob not aborting. This state ρ_{NA} can be written as

$$\rho_{NA} = \sum_{xy} q_{xy} |x\rangle\langle x|_{\mathcal{X}} \otimes |\phi_{xy}\rangle\langle\phi_{xy}| \otimes |y\rangle\langle y|_{\mathcal{Y}}$$

To prove the theorem, we must show the following properties for ρ_{NA}.

1. $H(XY)_{\rho_{NA}} \geq 2n\log(k) - t - 1$.
2. $\omega^*(G'|\rho_{NA}) \geq 1 - \varepsilon/32$ where $G^n_{1-\varepsilon/32} = (I', O', V', \text{Unif.})$ and $G' = (I', O', V', q)$.
3. $SIC(\rho_{NA}) \leq \frac{32\log(|I||O|)}{\varepsilon}((t+1) + |\log(\varepsilon)| + 5) + 2t + 2$.

The ideas behind the proofs of these three properties are as follows:

1. In task H(p,m), $Pr[\text{Bob does not abort}] \geq p = 2^{-t}$, when conditionning on Bob winning, we remove at most t bits of entropy from the (uniform) inputs in X, Y , the 1 in the inequality is there for technical reasons.
2. In the task H(p,m), $Pr[\text{Alice and Bob win } \geq (1-\varepsilon/32)n \text{ games} \mid \text{Bob does not abort}] \geq (1-\varepsilon/32)$. This directly implies the second property
3. In protocol P, before Alice sends her message, Bob has no information about x. Alice sends a message of size m, which gives m bits of information about Alice's input. Conditionning on Bob winning gives him an extra $2t$ bits of information. Since $m = \frac{32\log(|I||O|)}{\varepsilon}((t+1) + |\log(\varepsilon)| + 5)$ from the previous Proposition, we can conclude.

7 Final Theorem

Theorem 1. *For any game $G = (I, O, V, Unif.)$ with $\omega^*(G) \leq 1-\varepsilon$, we have:*

$$\omega^*(G^n) = (1-\varepsilon^2)^{\Omega\left(\frac{n}{\log(|I||O|)} - |\log(\varepsilon)|\right)}.$$

Proof. Let $G^n_{1-\varepsilon/32} = (I^n = [k^n], O^n, V_n, \text{Unif.})$ as defined in Section 2. Using Theorem 4, we know there exists a state $\xi = \sum_{xy} p_{xy} |x\rangle\langle x| \otimes |\phi_{xy}\rangle\langle\phi_{xy}| \otimes |y\rangle\langle y|$ and a game $G' = (I^n, O^n, V_n, p)$ satisfying

1. $H(XY)_\xi \geq 2n\log(k) - t - 1$
2. $\omega^*(G'|\xi) \geq 1 - \varepsilon/32$
3. $SIC(\xi) \leq \frac{32\log(|I||O|)}{\varepsilon}((t+1) + |\log(\varepsilon)| + 5)$,

where $2^{-t} = \omega^*(G^n)$. We now distinguish two cases

- If $t = \Omega(\varepsilon n)$ then $\omega^*(G^n) = (1-\varepsilon)^{\Omega(n)}$ and the theorem holds directly.
- If $t = o(\varepsilon n)$, we need the following argument. The state ξ satisfies all the properties of Theorem 5 which implies that $SIC(\xi) = \Omega(n\varepsilon)$. We combine the two inequalities and obtain

$$\Omega(n\varepsilon) \leq SIC(\xi) \leq \frac{32\log(|I||O|)}{\varepsilon}((t+1) + |\log(\varepsilon)| + 5).$$

It follows that $t = \Omega\left(\frac{n\varepsilon^2}{\log(|I||O|)} - |\log(\varepsilon)|\right)$, which allows us to conclude

$$\omega^*(G^n) = 2^{-t} \leq (1-\varepsilon^2)^{O\left(\frac{n}{\log(|I||O|)} - |\log(\varepsilon)|\right)}.$$

Finally, in the full paper we extend this result to general games.

Corollary 1. *For any game $G = (I, O, V, p)$ such that $\omega^*(G) \leq 1-\varepsilon$, we have*

$$\omega^*(G^n) = (1-\varepsilon^2)^{\Omega\left(\frac{n}{Q^4\log(Q\cdot|O|)} - |\log(\varepsilon/Q)|\right)},$$

where $|O|$ is the dimension of the output space of G and $Q = \max(\lceil\frac{1}{\min_{xy:p_{xy}\neq 0}(\sqrt{p_{xy}})}\rceil, |I|)$.

Acknowledgments. The authors thank Ronald de Wolf and Anthony Leverrier for helpful suggestions. Part of the work was done while A.C. was at CWI, Amsterdam. A.C. was partially supported by the European Commission under the project QCS (Grant No. 255961). G.S. was supported by de Wolf's Vidi grant 639.072.803 from the Netherlands Organization for Scientific Research (NWO).

References

1. Arora, S., Khot, S.A., Kolla, A., Steurer, D., Tulsiani, M., Vishnoi, N.K.: Unique games on expanding constraint graphs are easy: extended abstract. In: Proceedings STOC 2008, pp. 21–28 (May 2008)
2. Bar-Yossef, Z., Jayram, T.S., Kumar, R., Sivakumar, D.: An information statistics approach to data stream and communication complexity. J. Comput. Syst. Sci. 68(4), 702–732 (2004)
3. Bellare, M., Goldreich, O., Sudan, M.: Free bits, pcps, and nonapproximability—towards tight results. SIAM J. Comput. 27(3), 804–915 (1998)
4. Braverman, M.: Interactive information complexity. In: Proceedings of STOC 2012, pp. 505–524 (May 2012)
5. Chailloux, A., Kerenidis, I.: Optimal bounds for quantum bit commitment. In: Proceedings of FOCS 2011, pp. 354–362 (October 2011)
6. Chailloux, A., Scarpa, G.: Parallel Repetition of Entangled Games with Exponential Decay via the Superposed Information Cost. arXiv:1310.7787 (October 2013)
7. Chakrabarti, A., Shi, Y., Wirth, A., Yao, A.: Informational complexity and the direct sum problem for simultaneous message complexity. In: Proceedings of the FOCS 2001, pp. 270–288 (October 2001)
8. Clauser, J.F., Horne, M.A., Shimony, A., Holt, R.A.: Proposed experiment to test local hidden-variable theories. Phys. Rev. Lett. 23, 880–884 (1969)
9. Cleve, R., Slofstra, W., Unger, F., Upadhyay, S.: Perfect parallel repetition theorem for quantum xor proof systems. Comput. Complex. 17(2), 282–299 (2008)
10. Dinur, I., Steurer, D., Vidick, T.: A parallel repetition theorem for entangled projection games, arXiv:1310.4113 (October 2013)
11. Feige, U.: A threshold of ln n for approximating set cover. J. ACM 45(4), 634–652 (1998)
12. Håstad, J.: Some optimal inapproximability results. J. ACM 48(4), 798–859 (2001)
13. Holenstein, T.: Parallel repetition: simplifications and the no-signaling case. In: Proceedings of STOC 2007, pp. 411–419 (May 2007)
14. Jain, R., Pereszlyi, A., Yao, P.: A parallel repetition theorem for entangled two-player one-round games under product distributions. arXiv:1311.6309 (November 2013)
15. Kempe, J., Regev, O., Toner, B.: Unique games with entangled provers are easy. In: Proceedings of the FOCS 2008, pp. 457–466 (October 2008)
16. Kempe, J., Vidick, T.: Parallel repetition of entangled games. In: Proceedings of STOC 2011, pp. 353–362 (May 2011)
17. Kerenidis, I., Laplante, S., Lerays, V., Roland, J., Xiao, D.: Lower bounds on information complexity via zero-communication protocols and applications. In: Proceedings of FOCS 2012, pp. 500–509 (October 2012)
18. Parnafes, I., Raz, R., Wigderson, A.: Direct product results and the GCD problem, in old and new communication models. In: Proceedings of STOC 1997, pp. 363–372 (May 1997)
19. Rao, A.: Parallel repetition in projection games and a concentration bound. In: Proceedings STOC 2008, pp. 1–10 (May 2008)
20. Raz, R.: A parallel repetition theorem. SIAM J. Comput. 27(3), 763–803 (1998)

The Bose-Hubbard Model is QMA-complete*

Andrew M. Childs[1,2], David Gosset[1,2], and Zak Webb[2,3]

[1] Department of Combinatorics & Optimization, University of Waterloo
[2] Institute for Quantum Computing, University of Waterloo
[3] Department of Physics & Astronomy, University of Waterloo

Abstract. The Bose-Hubbard model is a system of interacting bosons that live on the vertices of a graph. The particles can move between adjacent vertices and experience a repulsive on-site interaction. The Hamiltonian is determined by a choice of graph that specifies the geometry in which the particles move and interact. We prove that approximating the ground energy of the Bose-Hubbard model on a graph at fixed particle number is QMA-complete. In our QMA-hardness proof, we encode the history of an n-qubit computation in the subspace with at most one particle per site (i.e., hard-core bosons). This feature, along with the well-known mapping between hard-core bosons and spin systems, lets us prove a related result for a class of 2-local Hamiltonians defined by graphs that generalizes the XY model. By avoiding the use of perturbation theory in our analysis, we circumvent the need to multiply terms in the Hamiltonian by large coefficients.

1 Introduction

The problem of approximating the ground energy of a given Hamiltonian is a natural quantum analog of classical constraint satisfaction. Many authors have considered the computational complexity of such quantum ground state problems. For a variety of classes of Hamiltonians and a suitable notion of approximation, this task is complete for the complexity class QMA, the quantum version of NP with two-sided error (see reference [2] for a recent review). These results provide evidence that approximating the ground energy of such quantum systems is intractable.

The first such example is the Local Hamiltonian problem introduced by Kitaev [3]. A k-local Hamiltonian acts on a system of n qubits and can be written as a sum of terms, each acting nontrivially on k qubits. The k-Local Hamiltonian problem is a promise problem related to the task of approximating the ground energy of a k-local Hamiltonian. Given such a Hamiltonian and two thresholds a and b, one is asked to determine if the ground energy is below a or above b (promised that one of these conditions holds). Kitaev's original work showed that 5-local Hamiltonian is QMA-complete [3]; subsequent works proved QMA-completeness of the 3-local Hamiltonian problem [4], the 2-local Hamiltonian

* Reference [1] is a detailed technical version of this extended abstract.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 308–319, 2014.

problem [5], and the 2-local Hamiltonian problem with interactions between qubits restricted to a two-dimensional lattice [6].

The complexity of similar computational problems related to other classes of Hamiltonians has also been considered. These include Hamiltonians in one dimension [7, 8], frustration-free Hamiltonians [9, 10], and stoquastic Hamiltonians (Hamiltonians with no "sign problem") [11, 12], among others.

The QMA-hardness of ground energy problems for local Hamiltonians acting on qubits has implications for Hamiltonians acting on indistinguishable particles (bosons or fermions) due to formal mappings between these systems. By applying such mappings to the Local Hamiltonian problem, one can show that certain bosonic [13] and fermionic [14] Hamiltonian problems are QMA-hard. A more restrictive class of QMA-complete fermionic Hamiltonians was considered by Schuch and Verstraete, who showed that the Hubbard model with a site-dependent magnetic field is QMA-complete [15]. This is a specific model of interacting electrons (i.e., spin-$\frac{1}{2}$ fermions) on a two-dimensional lattice, with a magnetic field that may take different values and point in different directions (in three dimensions) at distinct sites of the lattice.

Many of the QMA-complete problems considered previously have the property that the form of the terms in the Hamiltonian is part of the specification of the instance. For example, a 2-local Hamiltonian is specified by a graph, indicating pairs of qubits where terms in the Hamiltonian act, along with a 2-local Hermitian operator for each edge. In the Hubbard model considered in reference [15], there is a similar freedom in the choice of magnetic field at each site. A recent classification of local Hamiltonian problems [16] likewise applies only to models with adjustable coefficients. In fact, these results typically require coefficients that grow with the problem size.

In contrast, here we consider a system of interacting bosons with fixed movement and interaction terms. Specifically, we consider the Bose-Hubbard model, which has one of the simplest interactions between particles that conserves total particle number. Although the Bose-Hubbard model is traditionally defined on a lattice and with negative hopping strength [17], here we consider its extension to a general graph, with positive hopping strength.

We consider undirected graphs without multiple edges and with at most one self loop per vertex. Any such graph G (with vertex set V) can be specified by its adjacency matrix, a symmetric 0-1 matrix denoted $A(G)$. The Bose-Hubbard model on G with hopping strength t_{hop} and interaction strength J_{int} has the Hamiltonian

$$H_G = t_{\text{hop}} \sum_{i \in V} \sum_{j \in V} A(G)_{ij}\, a_i^\dagger a_j + J_{\text{int}} \sum_{k \in V} n_k(n_k - 1) \tag{1.1}$$

where $a_i^\dagger$ creates a boson at vertex i and $n_i = a_i^\dagger a_i$ counts the number of bosons at vertex i. Our results apply to the Bose-Hubbard model for any fixed positive hopping strength $t_{\text{hop}} > 0$ and any fixed positive (i.e., repulsive) interaction strength $J_{\text{int}} > 0$. Unlike other QMA-hardness results, in our work the coefficients $t_{\text{hop}}, J_{\text{int}}$ are not inputs to the problem; rather, each fixed choice defines a computational problem and we prove QMA-completeness for each of them.

Observe that the Bose-Hubbard Hamiltonian (1.1) conserves the total number of particles $N = \sum_{k\in V} n_k$. We focus on the space of N-particle states, which can be identified with the symmetric subspace of $(\mathbb{C}^{|V|})^{\otimes N}$ (as we discuss in more detail in Section 3). The first term in (1.1) allows particles to move between vertices; the second term is an interaction between particles that assigns an energy penalty for each vertex that is occupied by more than one particle. The Bose-Hubbard model is an example of a multi-particle quantum walk, a generalization of quantum walk to systems with more than one walker.

Recently we showed that the Bose-Hubbard model on a graph can perform efficient universal quantum computation [18]. Sometimes universality goes hand-in-hand with QMA-completeness, e.g., for local Hamiltonians, whose dynamics are BQP-complete [19] and whose ground energy problem is QMA-complete [3]. However, not all classes of Hamiltonians with universal dynamics have QMA-complete ground energy problems. For example, the dynamics of stoquastic local Hamiltonians are BQP-complete (as follows from [20] and time reversal), whereas the corresponding ground energy problem is in AM [11] and hence unlikely to be QMA-hard. Similarly, the ground energy problem for a Bose-Hubbard model with $t_{\text{hop}} < 0$ is also in AM [11], whereas the dynamics of such Hamiltonians are universal [18]. The ferromagnetic Heisenberg model on a graph provides an even starker contrast: its dynamics are BQP-complete (as can be inferred from [18] using a correspondence between spins and hard-core bosons) but its ground energy problem is trivial since the ground space is the symmetric subspace.

2 Overview of Results and Techniques

In this paper we define the Bose-Hubbard Hamiltonian problem and characterize its complexity. In this problem one is given a graph G and a number of particles N and asked to approximate the ground energy of the Bose-Hubbard Hamiltonian (1.1) in the N-particle sector (in a precise sense described in Section 3). We prove that this problem is QMA-complete.

To prove QMA-hardness of the Bose-Hubbard Hamiltonian problem, we show that in fact a notable special case of this problem, called Frustration-Free Bose Hubbard Hamiltonian, is QMA-hard. In this problem one is asked (roughly) to determine if the ground energy of the Bose-Hubbard Hamiltonian (1.1) in the N-particle sector is close to N times its single-particle ground energy (i.e., N times the smallest eigenvalue of the adjacency matrix $A(G)$). This is always a lower bound on the N-particle energy, and when it is achieved we say the N-particle ground states are frustration free. A frustration-free state has the special property that it has minimal energy for both terms in (1.1), and in particular it is annihilated by the interaction term. Frustration-free states therefore live in the subspace of *hard-core* bosons, with at most one boson per vertex.

Furthermore, we prove a reduction from Frustration-Free Bose-Hubbard Hamiltonian to an eigenvalue problem for a class of 2-local Hamiltonians defined by graphs. The two problems are related by a well-known mapping between hard-core bosons and spin systems. Specifically, given a graph G (with vertex set V) we consider the Hamiltonian

$$O_G = \sum_{\substack{A(G)_{ij}=1\\ i\neq j}} \frac{\sigma_x^i\sigma_x^j + \sigma_y^i\sigma_y^j}{2} + \sum_{A(G)_{ii}=1} \frac{1-\sigma_z^i}{2} \tag{2.1}$$

where $\sigma_x, \sigma_y, \sigma_z$ are the Pauli matrices. Note that this Hamiltonian commutes with the magnetization operator $M_z = \sum_{i=1}^{|V|} \frac{1-\sigma_z^i}{2}$ and has a sector for each of its eigenvalues $M_z \in \{0, 1, \ldots, |V|\}$. We reduce Frustration-Free Bose-Hubbard Hamiltonian (with N particles on a graph G) to the problem of approximating the smallest eigenvalue of (2.1) within the sector with magnetization $M_z = N$. We call this the XY Hamiltonian problem because of its connection to the XY model from condensed matter physics. Since this problem is contained in QMA, our reduction shows it to be QMA-complete.

We also obtain another result that may be of independent interest. In Appendix A of [1] we give a self-contained proof that computing the smallest eigenvalue of a sparse, efficiently row-computable [21] symmetric 0-1 matrix (the adjacency matrix of a graph) is QMA-complete. This can alternatively be viewed as a result about the QMA-completeness of a single-particle quantum walk on a graph with at most one self-loop per vertex. To prove this, we use a mapping from circuits to graphs that is also used in our main result. Note that Janzing and Wocjan used a similar construction to design a BQP-complete problem [20].

Proof Techniques

We prove our main result by direct reduction from quantum circuit satisfiability. We introduce several new techniques in order to do this using the Bose-Hubbard model on an unweighted graph.

Kitaev's original proof of QMA-hardness of the Local Hamiltonian problem encodes a QMA verification circuit using ideas from a computationally universal Hamiltonian proposed by Feynman [19]. This Hamiltonian uses a "clock register" to record the progress of the computation; in an appropriate basis, the Hamiltonian is a quantum walk on a path whose vertices represent the steps of the computation. Other proofs of QMA-hardness have used other encodings of the temporal structure of a verification circuit into a quantum state. In our construction, we encode the history of an n-qubit verification circuit in the state of n interacting particles on a graph, where each particle encodes a single qubit.

Our construction uses a class of graphs we define called *gate graphs*. Gate graphs are built from a basic subgraph whose single-particle ground states encode the history of a simple single-qubit computation. By combining copies of this basic unit, we define gadgets with other functionality. (Note that these gadgets realize some desired behavior exactly; they are not "perturbative gadgets" in the sense of [5, 22].) In particular, we design gadgets for two-qubit gates such that each ground state of the two-particle Bose-Hubbard model encodes a two-qubit

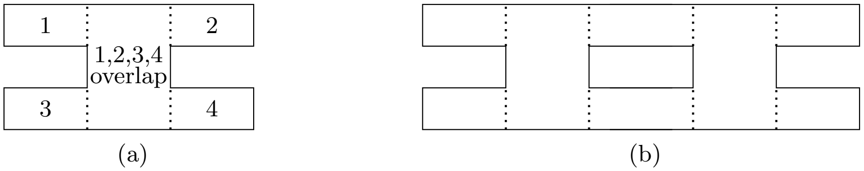


Fig. 2.1. We design graphs for two-qubit gates with overlapping regions as in (A). Regions 1 and 2 are associated with the first encoded qubit and regions 3 and 4 with the second encoded qubit. One could imagine designing a graph for a circuit with two-qubit gates U_1 followed by U_2 by connecting the corresponding gadgets as in (B). In the text we describe a challenge with this approach.

computation. We now give a high-level description of how these gadgets work and how we use them to construct a graph for a QMA verification circuit.

For each two-qubit gate U from a fixed universal set, we design a graph G_U that can be divided into four overlapping regions as shown schematically in Figure 2.1(a). (The specific graphs we use for two-qubit gates each have 4096 vertices and are described using the gate graph formalism.) The two-particle Bose-Hubbard model on this graph has ground states that encode the two-qubit computation. To describe them it is helpful to first consider the single-particle ground states, i.e., the ground states of the adjacency matrix $A(G_U)$. This matrix has 16 orthonormal single-particle ground states $|\rho^{i,U}_{z,a}\rangle$. Each index $i \in \{1,2,3,4\}$ is associated with the corresponding region in the graph, as $|\rho^{i,U}_{z,a}\rangle$ is supported entirely within region i. The index $z \in \{0,1\}$ corresponds to the computational basis states of a single encoded qubit. Note that, since $A(G_U)$ is a real matrix, the complex conjugate of any eigenstate is also an eigenstate with the same eigenvalue. The index $a \in \{0,1\}$ is associated with this freedom, i.e., $|\rho^{i,U}_{z,1}\rangle = |\rho^{i,U}_{z,0}\rangle^*$. The ground space of the two-particle Bose-Hubbard model on G_U is spanned by 16 states, indexed by two choices $z_1, z_2 \in \{0,1\}$ of computational basis states for the encoded qubits and two bits $a_1, a_2 \in \{0,1\}$ associated with complex conjugation. These states can be represented as symmetric states in the Hilbert space $\mathbb{C}^{4096} \otimes \mathbb{C}^{4096}$; they are

$$\begin{aligned}&\frac{1}{2}(|\rho^{1,U}_{z_1,a_1}\rangle|\rho^{3,U}_{z_2,a_2}\rangle + |\rho^{3,U}_{z_2,a_2}\rangle|\rho^{1,U}_{z_1,a_1}\rangle)\\ &+\frac{1}{2}\sum_{x_1,x_2\in 0,1} U(a_1)_{x_1,x_2,z_1,z_2}(|\rho^{1,U}_{x_1,a_1}\rangle|\rho^{3,U}_{x_2,a_2}\rangle + |\rho^{3,U}_{x_2,a_2}\rangle|\rho^{1,U}_{x_1,a_1}\rangle)\end{aligned}$$

where $U(0) = U$ is the two-qubit gate of interest and $U(1) = U^*$ is its elementwise complex conjugate. Observe that each of these states is a superposition of a term where both particles are on the left-hand side of the graph, encoding a two-qubit input state $|z_1\rangle|z_2\rangle$, and a term where both particles are on the right-hand side of the graph, encoding the two-qubit output state $U(a_1)|z_1\rangle|z_2\rangle$ where either U or its complex conjugate has been applied. While we might prefer the ground states to only encode the computation corresponding to U, we must include the possibility of U^* because the Hamiltonian is real. The same issue arises for n-qubit verification circuits. Fortunately, the complex conjugate of a circuit is equally useful for QMA verification.

It is natural to attempt to construct a graph for an n-qubit verification circuit by combining gadgets for each of the two-qubit gates. However, there is an obstacle to this approach, as illustrated by the example of a two-qubit circuit consisting of only two gates U_1 and U_2. One could construct a graph for such a circuit as shown schematically in Figure 2.1(b), where the two-qubit gadgets for U_1 and U_2 are connected in some unspecified way in the middle. However, not every ground state of the two-particle Bose-Hubbard model on such a graph encodes a computation. For example, there could be a ground state where one of the particles is in the single-particle state $|\rho_{z,a}^{1,U_1}\rangle$ localized on the left side of the graph and the other particle is in the state $|\rho_{z,a}^{2,U_2}\rangle$ with support on a disjoint region of the graph on the right-hand side. To eliminate such spurious ground states, we develop a method to enforce *occupancy constraints* on the locations of particles in gate graphs using the Bose-Hubbard interaction. Although this interaction only directly penalizes simultaneous occupation of the same vertex, we show how to simulate terms that penalize simultaneous occupation of different regions of the graph. We formalize this method by proving an "Occupancy Constraints Lemma" for gate graphs.

In summary, our construction of the graph for an n-qubit verification circuit proceeds in two steps. We first construct a graph G by connecting two-qubit gadgets for each of the gates in the circuit. As discussed above, the ground space of the n-particle Bose-Hubbard model on G includes a subspace of states that encode computations and a subspace of states that do not. We construct a set of occupancy constraints that are only satisfied by states in the former subspace. We then apply the Occupancy Constraints Lemma to obtain another gate graph where each N-particle ground state encodes a computation.

Unlike many previous works, we do not use perturbation theory in our analysis. Instead, we use a "Nullspace Projection Lemma" (used implicitly in [23]) that characterizes the smallest nonzero eigenvalue of a sum of two positive semidefinite matrices $H_A + H_B$ in terms of the smallest nonzero eigenvalue of H_A and the smallest nonzero eigenvalue of H_B restricted to the nullspace of H_A. This Lemma allows us to establish an eigenvalue promise gap (i.e., to bound the ground energies of yes instances away from those of no instances) without having to multiply terms in the Hamiltonian by large coefficients, something that is not allowed in the setting of the Bose-Hubbard model on a graph. Whereas QMA-hardness proofs such as those of [4, 5, 6, 15, 16] require multiplying terms in the Hamiltonian by unphysical, problem-size dependent coefficients, our approach avoids this. To the best of our knowledge, our proof would not be much simpler if we only demanded constant-size coefficients; the further restriction that the model is defined entirely by a graph is an extra benefit with little additional cost.

3 Definitions and Results

In this Section we introduce the Bose-Hubbard model and a related spin model, and formally state our results.

3.1 The Bose-Hubbard Model on a Graph

We consider the Bose-Hubbard model on a graph G, where the Hamiltonian is given by (1.1). While our complexity-theoretic results apply to the Bose-Hubbard model for any strictly positive hopping and interaction strengths, we set $t_{\text{hop}} = J_{\text{int}} = 1$ for convenience.

In the second-quantized formulation of the Bose-Hubbard model used in (1.1), the Hamiltonian H_G acts on the Fock space with orthonormal basis vectors specified by the number of bosons at each vertex. For our purposes, it will be more convenient to work in an equivalent (first-quantized) basis.

Consider the Hilbert space $(\mathbb{C}^{|V|})^{\otimes N}$ where each basis state $|i_1\rangle \ldots |i_N\rangle$ corresponds to an N-tuple of vertices $(i_1, ..., i_N) \in V^N$. Define the linear operator Sym that symmetrizes over all $N!$ permutations of the N particles:

$$\text{Sym}(|i_1\rangle \ldots |i_N\rangle) = \frac{1}{\sqrt{N!}} \sum_{\pi \in S_N} |i_{\pi(1)}\rangle \ldots |i_{\pi(N)}\rangle.$$

Every state in the Fock space can be uniquely paired with a state in

$$\mathcal{Z}_N(G) = \text{span}\{\text{Sym}(|i_1, \ldots, i_N\rangle) \colon i_1, \ldots, i_N \in V\}$$

since the two spaces have the same dimension. A natural bijection sends a basis state $\text{Sym}(|i_1\rangle \ldots |i_N\rangle)$ to the Fock state with $|\{j \colon i_j = v\}|$ bosons at each vertex v.

If we restrict our attention to the N-particle sector, then the Bose-Hubbard Hamiltonian (with $t_{\text{hop}} = J_{\text{int}} = 1$) acts as the operator

$$H_G^N = \sum_{w=1}^{N} A(G)^{(w)} + \sum_{k \in V} \hat{n}_k \left(\hat{n}_k - 1\right) \tag{3.1}$$

on the space $\mathcal{Z}_N(G)$, where the number operator is $\hat{n}_i = \sum_{w=1}^{N} |i\rangle\langle i|^{(w)}$ (see for example [24, §64]). Here a superscript (w) indicates that an operator acts nontrivially on subsystem w.

While H_G^N is defined as a $|V|^N \times |V|^N$ matrix in the space $(\mathbb{C}^{|V|})^{\otimes N}$, we consider its restriction

$$\bar{H}_G^N = H_G^N \Big|_{\mathcal{Z}_N(G)}$$

to the bosonic N-particle subspace $\mathcal{Z}_N(G)$. It is convenient to add a term proportional to the identity to obtain a positive semidefinite operator. Letting $\mu(G)$ denote the smallest eigenvalue of the adjacency matrix $A(G)$, we consider

$$H(G, N) = \bar{H}_G^N - N\mu(G)$$

and we write $\lambda_N^1(G)$ for the smallest eigenvalue of $H(G,N)$. Clearly $\lambda_N^1(G) \geq 0$ since the interaction term is positive semidefinite. Also note that, given the graph G, the smallest eigenvalue $\mu(G)$ of its adjacency matrix can be efficiently approximated using a classical polynomial-time algorithm, so the complexity of approximating $\lambda_N^1(G)$ is equivalent to the complexity of approximating the ground energy of $\bar{H}_G^N$. (Note that here the graph is specified explicitly by its adjacency matrix. In other contexts one might consider a graph specified compactly, e.g., by a circuit that computes rows of its adjacency matrix. Then the situation is more complex since the input size can be much smaller than the number of vertices in the graph. Indeed, we prove in Appendix A of [1] that approximating the smallest eigenvalue of such a graph is QMA-complete.)

When $\lambda_N^1(G) = 0$, the ground energy of the N-particle Bose-Hubbard model $\bar{H}_G^N$ is equal to N times the one-particle energy $\mu(G)$. Then we say that the N-particle Bose-Hubbard model is *frustration free.*

3.2 Complexity of the Bose-Hubbard Model

Given a K-vertex graph G and a number of particles N, how hard is it to approximate the ground energy of the N-particle Bose-Hubbard model $\bar{H}_G^N$ on G? We consider the following decision version of this computational problem.

*Problem 1 (**Bose-Hubbard Hamiltonian**).* We are given a K-vertex graph G, a number of particles N, a real number c, and a precision parameter $\epsilon = \frac{1}{T}$. The positive integers N and T are provided in unary; the graph is specified by its adjacency matrix, which can be any $K \times K$ symmetric 0-1 matrix. We are promised that either the smallest eigenvalue of $\bar{H}_G^N$ is at most c (yes instance) or is at least $c + \epsilon$ (no instance) and we are asked to decide which is the case.

In this problem c is provided in a straightforward manner, with enough precision to resolve ϵ, i.e., using $\mathcal{O}(\log|c| + \log T)$ bits. The input size is therefore $\Theta(K^2 + T + N + \log|c|)$ bits. We prove that this problem is QMA-complete, providing evidence that approximating the ground energy of the N-particle Bose-Hubbard model on a graph G is intractable.

Theorem 1. *Bose-Hubbard Hamiltonian is QMA-complete.*

The proof of this Theorem has two parts.

The easy part is to show that Bose-Hubbard Hamiltonian is contained in QMA. The basic strategy of Arthur's verification protocol is to measure the energy of the Bose-Hubbard Hamiltonian in the state given to him by Merlin, using phase estimation and Hamiltonian simulation. Arthur accepts if the energy is small enough and rejects otherwise. We give a more detailed description of the verification procedure in Section 3 of [1].

The more involved part is to show that Bose-Hubbard Hamiltonian is QMA-hard. For this we show that any instance of a QMA problem can be converted (in deterministic polynomial time on a classical computer) into an equivalent instance of Bose-Hubbard Hamiltonian. In fact, our reduction proves a slightly stronger result, namely that a notable extremal case of Bose-Hubbard Hamiltonian is already QMA-hard. We now discuss this special case.

Recall from the previous section that the ground energy of the N-particle Bose-Hubbard model is at least N times the single-particle ground energy $\mu(G)$, i.e., $\lambda_N^1(G) \geq 0$. We can ask if this inequality is close to equality, i.e., is the N-particle Bose-Hubbard model close to being frustration free?

*Problem 2 (**Frustration-Free Bose-Hubbard Hamiltonian**).* We are given a K-vertex graph G, a number of particles $N \leq K$, and a precision parameter $\epsilon = \frac{1}{T}$. The integer $T \geq 4K$ is provided in unary; the graph is specified by its adjacency matrix, which can be any $K \times K$ symmetric 0-1 matrix. We are promised that either $\lambda_N^1(G) \leq \epsilon^3$ (yes instance) or $\lambda_N^1(G) \geq \epsilon + \epsilon^3$ (no instance) and we are asked to decide which is the case.

For concreteness, we have made some specific choices in defining this problem. Our proof that it is QMA-hard also applies, for example, to variants of the problem where ϵ^3 is replaced (in both places it appears) by ϵ^α for any constant $\alpha \in \{1, 2, 3, \ldots\}$. We use the version with $\alpha = 3$ as stated above to facilitate a reduction to the XY Hamiltonian problem.

The requirement $T \geq 4K$ ensures that ϵ is small so that, for a yes instance, the system is very close to being frustration free. We choose the specific threshold $4K$ for concreteness.

The restriction $N \leq K$ is without loss of generality since the problem is trivial otherwise. To see this, note that any state with more than K particles is orthogonal to the nullspace of the interaction term since there are always two or more particles located at one vertex; hence $\lambda_N^1(G) \geq 2$ whenever $N \geq K + 1$.

Frustration-Free Bose-Hubbard Hamiltonian is a special case of Bose-Hubbard Hamiltonian with $c = N\mu(G) + \epsilon^3$. To prove that Bose-Hubbard Hamiltonian is QMA-hard, it therefore suffices to prove that Frustration-Free Bose-Hubbard Hamiltonian is QMA-hard. The bulk of our technical work [1] is concerned with the proof of this fact, following the strategy outlined in Section 2.

3.3 Complexity of the XY Hamiltonian Problem

We reduce Frustration-Free Bose-Hubbard Hamiltonian to an eigenvalue problem for a class of 2-local Hamiltonians defined by graphs. The reduction is based on a well-known mapping between hard-core bosons and spin systems.

We define the subspace $\mathcal{W}_N(G) \subset \mathcal{Z}_N(G)$ of N hard-core bosons on a graph G to consist of the states where each vertex of G is occupied by either 0 or 1 particle, i.e.,

$$\mathcal{W}_N(G) = \text{span}\{\text{Sym}(|i_1, \ldots, i_N\rangle) \colon i_j \in V,\ i_j \neq i_k \text{ for distinct } j, k \in [N]\}.$$

A basis for $\mathcal{W}_N(G)$ is the subset of Fock states with at most one particle per vertex, which can be labeled by bit strings with Hamming weight N. The space

$\mathcal{W}_N(G)$ can thus be identified with the weight-N subspace

$$\mathrm{Wt}_N(G) = \mathrm{span}\{|z\rangle : z \in \{0,1\}^{|V|}, \textstyle\sum_i z_i = N\}$$

of a $|V|$-qubit Hilbert space. We consider the restriction of H_G^N to the space $\mathcal{W}_N(G)$, which can equivalently be written as the $|V|$-qubit Hamiltonian O_G from equation (2.1) restricted to the space $\mathrm{Wt}_N(G)$.

Note that the Hamiltonian O_G conserves the total magnetization (Hamming weight) $M_z = \sum_{i=1}^{|V|} \frac{1-\sigma_z^i}{2}$ along the z axis. We define $\theta_N(G)$ to be the ground energy of O_G in the sector with magnetization N. We show that approximating this quantity is QMA-complete.

*Problem 3 (**XY Hamiltonian**).* We are given a K-vertex graph G, an integer $N \leq K$, a real number c, and a precision parameter $\epsilon = \frac{1}{T}$. The positive integer T is provided in unary; the graph is specified by its adjacency matrix, which can be any $K \times K$ symmetric 0-1 matrix. We are promised that either $\theta_N(G) \leq c$ (yes instance) or else $\theta_N(G) \geq c + \epsilon$ (no instance) and we are asked to decide which is the case.

Theorem 2. *XY Hamiltonian is QMA-complete.*

We prove QMA-hardness of XY Hamiltonian by reduction from Frustration-Free Bose-Hubbard Hamiltonian. The proof of Theorem 2 appears in Appendix B of [1].

4 Extensions and Open Questions

Our result shows that approximating the ground energy of the Bose-Hubbard model on a graph at fixed particle number is likely intractable. In showing this, we introduce techniques that we expect will be useful in other contexts. Here we briefly discuss some related questions for future work.

One might consider the complexity of variants of the Bose-Hubbard Hamiltonian problem. For example, one could consider the problem with negative hopping (i.e., $t_{\text{hop}} < 0$), with attractive interactions (i.e., $J_{\text{int}} < 0$), or both. For negative hopping, the results of [11] show that the problem is in AM; we do not know if it is AM-hard. For attractive interactions, the problem is clearly in QMA (the verification procedure described in Section 3 of [1] applies independent of the signs of $t_{\text{hop}}, J_{\text{int}}$), but again we do not know the true complexity.

One can define other variants of the Bose-Hubbard Hamiltonian problem by lifting the restriction to fixed particle number.

One could also consider other classes of graphs. The graphs we consider in this paper are described by symmetric 0-1 matrices and have at most one self-loop per vertex. We do not know if the model remains QMA-hard on simple graphs, i.e., without any self-loops.

There are many open questions concerning the complexity of the ground energy problem for other quantum systems defined by graphs. For example, one could consider fermions or bosons on a graph with nearest-neighbor interactions. One could also consider quantum spin models defined on graphs such as the XY model or the antiferromagnetic Heisenberg model. Both of these examples correspond to Hamiltonians that conserve magnetization, so one could consider the ground energy problem with or without a restriction to a fixed-magnetization sector. This would complement existing results about the complexity of computing the lowest-energy configuration of classical spin models defined by graphs (for example, the antiferromagnetic Ising model on a graph is NP-complete, as it is equivalent to max cut).

As emphasized previously, the Hamiltonians we consider are determined entirely by a choice of graph, with the same type of movement and interaction terms applied throughout the graph. It might be interesting to find other QMA-complete problems with similar features, such as a version of Local Hamiltonian with only one type of local term. Analogous classical constraint satisfaction problems with a fixed type of constraint are well known (e.g., Exact Cover and Not-All-Equal SAT) and have been widely studied. Along similar lines, it might be interesting to understand when local Hamiltonian problems remain QMA-hard with constant-size coefficients. Nagaj and Mozes have shown that the 3-local Hamiltonian problem has this property [25], but whether the same holds for the 2-local Hamiltonian problem remains open.

Acknowledgments. This work was supported in part by NSERC; the Ontario Ministry of Research and Innovation; the Ontario Ministry of Training, Colleges, and Universities; and the US ARO.

References

[1] Childs, A.M., Gosset, D., Webb, Z.: The Bose-Hubbard model is QMA-complete. arXiv:1311.3297

[2] Bookatz, A.D.: QMA-complete problems. arXiv:1212.6312

[3] Kitaev, A.Y., Shen, A.H., Vyalyi, M.N.: Classical and Quantum Computation. American Mathematical Society (2002)

[4] Kempe, J., Regev, O.: 3-Local Hamiltonian is QMA-complete. Quantum Inf. Comput. 3(3), 258–264 (2003)

[5] Kempe, J., Kitaev, A., Regev, O.: The complexity of the local Hamiltonian problem. SIAM J. Comput. 35(5), 1070–1097 (2006)

[6] Oliveira, R., Terhal, B.M.: The complexity of quantum spin systems on a two-dimensional square lattice. Quantum Inf. Comput. 8(10), 900–924 (2008)

[7] Aharonov, D., Gottesman, D., Irani, S., Kempe, J.: The power of quantum systems on a line. Commun. Math. Phys. 287, 41–65 (2009)

[8] Gottesman, D., Irani, S.: The quantum and classical complexity of translationally invariant tiling and Hamiltonian problems. Theory Comput. 9(2), 31–116 (2013)

[9] Bravyi, S.: Efficient algorithm for a quantum analogue of 2-SAT. Contemp. Math. 536, 33–48 (2011)

[10] Gosset, D., Nagaj, D.: Quantum 3-SAT is QMA$_1$-complete. In: 54th FOCS, pp. 756–765 (2013)

[11] Bravyi, S., DiVincenzo, D.P., Oliveira, R.I., Terhal, B.M.: The complexity of stoquastic local Hamiltonian problems. Quantum Inf. Comput. 8(5), 361–385 (2008)

[12] Bravyi, S., Terhal, B.: Complexity of stoquastic frustration-free Hamiltonians. SIAM J. Comput. 39(4), 1462–1485 (2009)

[13] Wei, T.C., Mosca, M., Nayak, A.: Interacting boson problems can be QMA hard. Phys. Rev. Lett. 104(4), 040501 (2010)

[14] Liu, Y.K., Christandl, M., Verstraete, F.: Quantum computational complexity of the N-representability problem: QMA complete. Phys. Rev. Lett. 98(11), 110503 (2007)

[15] Schuch, N., Verstraete, F.: Computational complexity of interacting electrons and fundamental limitations of density functional theory. Nature Phys. 5(10), 732–735 (2009)

[16] Cubitt, T., Montanaro, A.: Complexity classification of local Hamiltonian problems. arXiv:1311.3161

[17] Fisher, M.P.A., Weichman, P.B., Grinstein, G., Fisher, D.S.: Boson localization and the superfluid-insulator transition. Phys. Rev. B 40, 546–570 (1989)

[18] Childs, A.M., Gosset, D., Webb, Z.: Universal computation by multiparticle quantum walk. Science 339(6121), 791–794 (2013)

[19] Feynman, R.P.: Quantum mechanical computers. Optics News 11, 11–20 (1985)

[20] Janzing, D., Wocjan, P.: BQP-complete problems concerning mixing properties of classical random walks on sparse graphs. arXiv:quant-ph/0610235

[21] Aharonov, D., Ta-Shma, A.: Adiabatic quantum state generation and statistical zero knowledge. In: 35th STOC, pp. 20–29 (2003)

[22] Jordan, S.P., Farhi, E.: Perturbative gadgets at arbitrary orders. Phys. Rev. A 77(6), 062329 (2008)

[23] Mizel, A., Lidar, D.A., Mitchell, M.: Simple proof of equivalence between adiabatic quantum computation and the circuit model. Phys. Rev. Lett. 99(7), 070502 (2007)

[24] Landau, L.D., Lifshitz, E.M.: Quantum mechanics: non-relativistic theory. Pergamon Press (1994)

[25] Nagaj, D., Mozes, S.: New construction for a QMA complete three-local Hamiltonian. J. Math. Phys. 48(7), 072104 (2007)

Characterization of Binary Constraint System Games [*]

Richard Cleve[1] and Rajat Mittal[2]

[1] Institute for Quantum Computing and School of Computer Science,
University of Waterloo

[2] Institute for Quantum Computing and Department of Combinatorics and Optimization, University of Waterloo

Abstract. We investigate a simple class of multi-prover interactive proof systems (classical non-local games), called *binary constraint system (BCS) games*, and characterize those that admit a perfect entangled strategy (i.e., a strategy with value 1 when the provers can use shared entanglement). Our characterization is in terms of a system of matrix equations. One application of this characterization is that, combined with a recent result of Arkhipov, it leads to a simple algorithm for determining whether certain *restricted* BCS games have a perfect entangled strategy, and, for the instances that do not, for bounding their value strictly below 1. An open question is whether, for the case of *general* BCS games, making this determination is computationally decidable. Our characterization might play a useful role in the resolution of this question.

Keywords: Quantum information, entanglement, binary constraint systems.

1 Introduction

Constraint systems and various two-player non-local games associated with them have played an important role in computational complexity theory (probabilistic interactive proof systems [7,5,12,4] and the hardness of approximation [12]) as well as quantum information (pertaining to the power of entanglement [6,8,18,9]).

We investigate the computational complexity of determining the value of a game given its description. Quantumly (when the players are allowed to possess any entangled state at the beginning), it is not even currently known that the problem is computable. This is the current state of affairs even for gapped versions of the problem, where $\varepsilon > 0$ and the goal is to distinguish between these cases: (a) the existence of a perfect strategy (i.e., with value 1); and (b) all strategies have value $\leq 1 - \varepsilon$. We refer to [9] for a detailed introduction to quantum non-local games and quantum strategies.

For a very special class of non-local games, called *XOR games*, a characterization in terms of semidefinite programs exists that makes the problem of

[*] Full version is available at `http://arxiv.org/abs/1209.2729`

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 320–331, 2014.

approximating their value tractable (see [9] and references therein). Also, an XOR game has a perfect quantum strategy if and only if it has a perfect classical strategy—which can be characterized by a linear system of equations. Thus, it is easy to determine whether or not an XOR game has a perfect entangled strategy.

We consider a generalization of XOR games known as *binary constraint system (BCS)* games. For such games, even determining the existence of a perfect strategy is not currently known to be computable. We characterize perfect strategies for BCS games in terms of solutions to certain systems of equations in which the variables are binary observables (involutory matrices). The known entangled strategies for BCS games have been based on such binary observables, and our main result is to shows that *any* perfect strategy for *any* BCS game must be based on such binary observables.

A parity BCS game is a BCS game where the constraints can be expressed as parities of variables. Recently, Arkhipov [3] gave an elegant algorithm for determining if a certain restricted type of parity BCS game (where every variable appears in at most two constraints) has a perfect entangled strategy. Arkhipov's methodology uses our characterization in that it assumes that any perfect entangled strategy is based on binary observables. The methodology in [3] apparently does not generalize to unrestricted parity BCS games (without the above restriction). The problem determining whether a parity BCS game has a perfect entangled strategy is not currently known to be computable.

We also give a method that upper bounds the value of BCS games strictly below 1 in certain cases of interest (but we do not know how to do this in general).

1.1 Binary Constraint System Games

A *binary constraint system (BCS)* consists of n binary variables, $v_1, v_2, \ldots, v_n$, and m constraints, $c_1, c_2, \ldots, c_m$, where each c_j is a binary-valued function of a subset of the variables. For convenience, we may write the constraints as equations. An example of a BCS (with $n = 9$ and $m = 6$) is

$$\begin{array}{lcl} v_1 \oplus v_2 \oplus v_3 = 0 & \qquad & v_1 \oplus v_4 \oplus v_7 \;=\; 0 \\ v_4 \oplus v_5 \oplus v_6 = 0 & & v_2 \oplus v_5 \oplus v_8 \;=\; 0 \\ v_7 \oplus v_8 \oplus v_9 = 0 & & v_3 \oplus v_6 \oplus v_9 \;=\; 1 \end{array} \tag{1}$$

(this BCS is related to the version of Bell's theorem introduced by Mermin [14], that is discussed further in the next section). If, as in this example, all the constraints are functions of the parity of a subset of variables we call the system a *parity BCS*. A BCS is *satisfiable* if there exists a truth assignment to the variables that satisfies every constraint. The above example is easily seen to be unsatisfiable (since summing all the equations modulo 2 yields $0 = 1$).

We can associate a two-player non-local game with each BCS that proceeds as follows. There are two cooperating players, Alice and Bob, who cannot communicate with each other once the protocol starts, and a verifier. The verifier randomly (uniformly) selects one constraint c_s and one variable x_t from c_s. The verifier sends s to Alice and t to Bob. Alice returns a truth assignment to all variables in c_s and Bob returns a truth assignment to variable x_t. The verifier accepts the answer if and only if:

1. Alice's truth assignment *satisfies* the constraint c_s;
2. Bob's truth assignment for x_t is *consistent* with Alice's.

Strategies where Alice and Bob employ no entanglement are called *classical.* Strategies where they employ entanglement are called *quantum* (or *entangled*). A strategy is *perfect* if it always succeeds.

It is not too hard to see that there exists a perfect *classical* strategy for a BCS game if and only if the underlying BCS is satisfiable. It is interesting that there exist perfect entangled strategies for BCS games for some unsatisfiable BCSs.

1.2 Mermin's Quantum Strategies

Mermin [14,15] made a remarkable discovery about sets of observables with certain properties that has consequences for quantum strategies for BCS games[1] that are unsatisfiable—in particular the following two games. The left side of Fig. 1 summarizes the BCS specified by the aforementioned system of equations (1). We refer to this BCS as the *magic square.* Similarly, the right side of Fig. 1 summarizes another BCS consisting of ten variables and five constraints, where each constraint is related to the parity of four variables. We refer to this BCS as the *magic pentagram.*

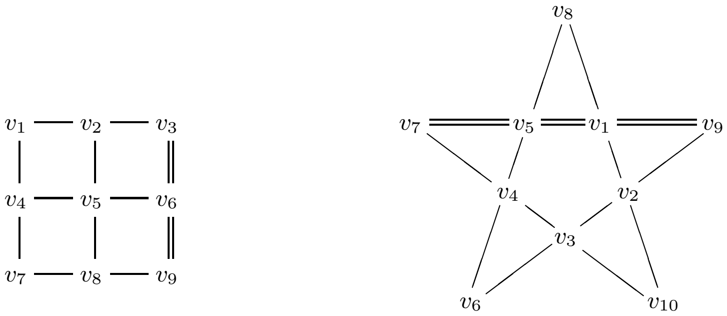


Fig. 1. Structure of two BCSs: (a) magic square (left) and (b) magic pentagram (right). Each straight line indicates a parity constraint on its variables of 0 for single lines, and 1 for double lines.

[1] Mermin's original paper was written in the language of no-hidden-variables theorems, along the lines of the Kochen Specker Theorem; however, it discusses implications regarding Bell inequality violations, and these can be interpreted as non-local games where quantum strategies exist that outperform classical strategies. The connection is made more explicit by Aravind [1,2].

To understand Mermin's strategies, we first define a *quantum satisfying assignment* of a BCS as a relaxation of a classical satisfying assignment, in the following manner. First translate each $\{0,1\}$-variable v_j into a $\{+1,-1\}$-variable $V_j = (-1)^{v_j}$. Then the parity of any sequence of variables is their product—and, in fact, every boolean function can be uniquely represented as a multilinear polynomial over $\mathbb{R}$ (e.g., for the binary OR-function (in $\{+1,-1\}$ domain), the polynomial is $(V_1V_2 + V_1 + V_2 - 1)/2$). Now we can define a *quantum satisfying assignment* as an assignment of finite-dimensional Hermitian operators $A_1, A_2, \ldots, A_n$ to the variables $V_1, V_2, \ldots, V_n$ (respectively) such that:

(a) Each A_j is a binary observable in that its eigenvalues are in $\{+1,-1\}$ (i.e., $A_j^2 = I$).
(b) All pairs of observables, A_i, A_j, that appear within the same constraint are commuting (i.e., they satisfy $A_iA_j = A_jA_i$).
(c) The observables *satisfy* each constraint $c_s : \{+1,-1\}^k \to \{+1,-1\}$ that acts on variables $V_{i_1}, \ldots, V_{i_k}$, in the sense that the multilinear polynomial equation $c_s(A_{i_1}, \ldots, A_{i_k}) = -I$ is satisfied (since c_s is arbitrary, we can assume right hand side of the polynomial to be -1).

This is a relaxation of the standard "classical" notion of a satisfying assignment (which corresponds to the case of one-dimensional observables). Quantum satisfying assignments for the two BCSs in Figure 1 are shown in Figure 2.

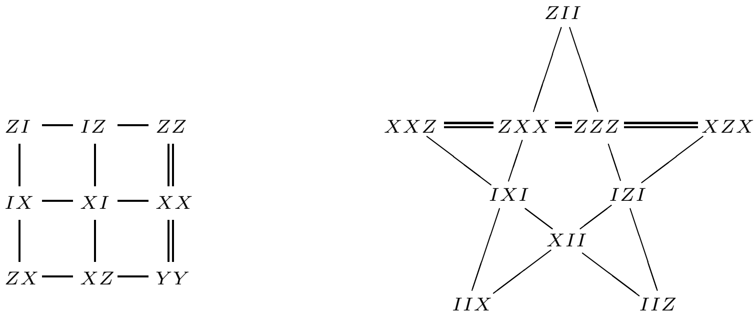


Fig. 2. Quantum satisfying assignments for: (a) magic square (left) and (b) magic pentagram (right). (X, Y, and Z are the usual 2×2 Pauli matrices, and juxtaposition means tensor product.)

There is a construction (implicit in [14] and explicit in [2] for the magic square) that converts these quantum satisfying assignments into perfect strategies—and this is easily extendable to any quantum satisfying assignment of a BCS. For completeness, we summarize the known construction. The entanglement is of the form $|\psi\rangle = \frac{1}{\sqrt{d}} \sum_{j=1}^{d} |j\rangle|j\rangle$, where d is the dimension of the observables. Alice associates observables $A_1, A_2, \ldots, A_n$ with the variables and Bob associates their transposes $A_1^T, A_2^T, \ldots, A_n^T$ (with respect to the computational basis) with the variables. On input s, Alice measures her observables that correspond to the variables in constraint c_s. At this point, it should be noted that this is a well-defined measurement since condition (b) implies that these observables are mutually commuting. Also, on input t, Bob measures his observable A_t^T. Condition (c) implies that Alice's output satisfies the constraint. Finally, Alice and Bob give consistent values for variable v_t because $\langle\psi|A_t \otimes A_t^T|\psi\rangle = \langle\psi|A_t \cdot A_t \otimes I|\psi\rangle = \langle\psi|\psi\rangle = 1$. The first equality follows from the fact that for the maximally entangled $|\psi\rangle$, $B \otimes A^T|\psi\rangle = B \cdot A \otimes I|\psi\rangle$.

1.3 General BCS Games

A natural computational problem is: given a description of a BCS as input, determine whether or not it has a perfect entangled strategy. A more general problem is to compute the maximum (or supremum) value of all entangled strategies.

For *classical* strategies, the problem of determining whether or not a perfect strategy exists is the same as finding out whether the underlying constraint system is feasible or not. It is NP-hard for general BCS games and in polynomial time for parity BCS games (where the problem reduces to solving a system of linear equations in modulo 2 arithmetic). For *quantum* strategies, we are currently not aware of *any* algorithm that determines whether or not an arbitrary parity BCS game has a perfect strategy (i.e., presently we do not even know that the problem is *decidable*).

In Section 2, we prove a converse to the construction of entangled strategies from quantum satisfying assignments in Section 1.2. Namely, we show that any perfect quantum strategy that uses countable-dimensional entanglement implies the existence of a quantum satisfying assignment.

It can be easily seen that not all BCS games have perfect quantum strategies, by this example

$$v_1 \oplus v_2 = 0 \qquad\qquad v_1 \oplus v_2 = 1. \tag{2}$$

First note that no generality is lost if we assume that Alice returns only a value for v_1 (since the value of v_2 is then uniquely determined by the constraint). The only case when they need to output different bits is when Alice is asked the second constraint and Bob is asked the second variable. Labelling the constraints as $\{0, 1\}$ for Alice and variables as $\{0, 1\}$ for Bob, it is not hard to see that such a game is equivalent to the so-called CHSH game [8], which is known to admit no perfect quantum strategy [18] (even though the quantum success probability is higher than the classical success probability [8]). In Section 3, we show how to derive upper bounds strictly below 1 on the entangled value of many parity BCSs.

2 Characterization of Perfect Strategies by Observables

Theorem 1. *For any binary constraint system, if there exists a perfect quantum strategy for the corresponding BCS game that uses finite or countably-infinite dimensional entanglement, then it has a quantum satisfying assignment.*

Proof. We start with an arbitrary binary constraint system that has variables $v_1, v_2, \ldots, v_n$ and constraints $c_1, c_2, \ldots, c_m$. Assume that there is a perfect entangled protocol for this system that uses entanglement

$$|\psi\rangle = \sum_{i=1}^{l} \alpha_i |\phi_i\rangle |\psi_i\rangle, \tag{3}$$

where $\{|\phi_1\rangle, \ldots, |\phi_l\rangle\}$ and $\{|\psi_1\rangle, \ldots, |\psi_l\rangle\}$ are orthonormal sets, $\alpha_1, \ldots, \alpha_l > 0$, and $\sum_{i=1}^{l} |\alpha_i|^2 = 1$. Here l is the Schmidt rank of the shared state—which can be set to ∞ to indicate a countably infinite set.

We consider two separate cases for Alice's strategy. In the first case, she applies an arbitrary *projective* measurement to the first register of $|\psi\rangle$. In the second case, Alice can apply an arbitrary POVM measurement to the first register of $|\psi\rangle$. For the definition and differences between these two measurements, we refer the reader to [16].

We will prove that quantum satisfying assignment exists in the first case. Then we will show that the second case can be reduced to first one, hence proving the theorem.

Case 1: Projective Measurements for Alice. For each $s \in \{1, 2, \ldots, m\}$, let c_s be a constraint consisting of r_s variables. Therefore, the set of outcomes for Alice is $\{0,1\}^{r_s}$. These can be associated with orthogonal projectors Π_a^s ($a \in \{0,1\}^{r_s}$). From these projectors, we can define the r_s individual bits of the outcome as the binary observables

$$A_s^{(j)} = \sum_{a \in \{0,1\}^{r_s}} (-1)^{a_j} \Pi_a^s, \tag{4}$$

for $j \in \{1, \ldots, r_s\}$ (Here we adopt the notation that observable $A_s^{(j)}$ corresponds to the variable in position j of constraint s). It is easy to check that $\{A_s^{(j)} : j \in \{1, \ldots, r_s\}\}$ is a set of commuting binary observables. We have defined a binary observable for Alice for each variable in the context of each constraint that includes it. For example, in the case of the magic square (Eqns. (1)), there is a binary observable $A_3^{(1)}$ for v_7 in the context of the third constraint and a binary observable $A_4^{(3)}$ for v_7 in the context of the fourth constraint. We have not yet shown that $A_3^{(1)} = A_4^{(3)}$ (constraint independent).

The measurements for Bob are (without loss of generality) binary observables B_t for each variable v_t ($t \in \{1, 2, \ldots, n\}$).

We need to show that the observables for Alice are the same, regardless of the constraint that they arise from (for example, for the magic square game, $A_3^{(1)} = A_4^{(3)}$). We shall use the following lemma.

Lemma 1. *Let $-I \preceq C_1, C_2, B \preceq I$ be Hermitian matrices on some Hilbert space $\mathcal{H}$. Let $|\psi\rangle \in \mathcal{H} \otimes \mathcal{H}$ be of the form*

$$|\psi\rangle = \sum_{i=1}^{l} \alpha_i |\phi_i\rangle |\psi_i\rangle, \tag{5}$$

where $\{|\phi_1\rangle, |\phi_2\rangle, \ldots, |\phi_l\rangle\}$ *and* $\{|\psi_1\rangle, |\psi_2\rangle, \ldots, |\psi_l\rangle\}$ *are orthonormal bases for* $\mathcal{H}$*,* $\alpha_1, \alpha_2, \ldots, \alpha_l > 0$*, and* $\sum_{i=1}^{l} |\alpha_i|^2 = 1$*. Then, for the Hermitian matrices* $\{B, C_1, C_2\}$*, if* $\langle\psi|B \otimes C_1|\psi\rangle = \langle\psi|B \otimes C_2|\psi\rangle = 1$ *then* $C_1 = C_2$*.*

Proof (Lemma 1). Consider the vectors $w = B \otimes I|\psi\rangle$, $u_1 = I \otimes C_1|\psi\rangle$, and $u_2 = I \otimes C_2|\psi\rangle$. These are vectors with length at most 1 and we have $w \cdot u_1 = w \cdot u_2 = 1$, which implies that $u_1 = w = u_2$. Therefore,

$$\begin{aligned} 0 &= I \otimes C_1|\psi\rangle - I \otimes C_2|\psi\rangle && (6)\\ &= (I \otimes (C_1 - C_2)) \left(\sum_{i=1}^{l} \alpha_i |\phi_i\rangle|\psi_i\rangle \right) && (7)\\ &= \sum_{i=1}^{l} \alpha_i |\phi_i\rangle (C_1 - C_2)|\psi_i\rangle, && (8) \end{aligned}$$

which implies that $(C_1 - C_2)|\phi_i\rangle = 0$, for all $i \in \{1, 2, \ldots\}$. This implies that $C_1 = C_2$, which completes the proof of the lemma. □

Returning to the proof of Theorem 1, let $t \in \{1, 2, \ldots, n\}$ and $A_s^{(j)}$ and $A_{s'}^{(j')}$ be any two observables of Alice corresponding to the same variable v_t. Since Alice's binary observables associated with constraint c_s are commuting, we can assume that Alice begins her measurement process by measuring $A_s^{(j)}$, while Bob measures B_t. Since these two measurements must yield the same outcome, we have $\langle\psi|A_s^{(j)} \otimes B_t|\psi\rangle = 1$. Similarly, $\langle\psi|A_{s'}^{(j')} \otimes B_t|\psi\rangle = 1$. Therefore, applying Lemma 1, we have $A_s^{(j)} = A_{s'}^{(j')}$, which establishes that Alice's observables are constraint independent.

In addition to consistency between Alice and Bob, Alice's output bits must satisfy the constraint c_s (recall that c_s can be expressed as a multilinear polynomial over $\mathbb{R}$). That is,

$$\langle\psi|c_s(A_s^{(1)}, \ldots, A_s^{(r_s)}) \otimes I|\psi\rangle = -1. \tag{9}$$

By invoking Lemma 1 again, with $C_1 = -c_s(A_s^{(1)}, \ldots, A_s^{(r_s)})$, $C_2 = I$, $B = I$, we can deduce that $c_s(A_s^{(1)}, \ldots, A_s^{(r_s)}) = -I$.

At this point, it is convenient to rename Alice's observables to A_t, for each $t \in \{1, 2, \ldots, n\}$ (which we can do because we proved they are constraint independent). The observables associated with each constraint commute and their product has the required parity.

We will finally prove that a finite-dimensional set of observables must exist. If l is finite then there is nothing to prove, so assume it is countably infinite. Since, for all $t \in \{1, 2, \ldots, n\}$, $\langle\psi|A_t \otimes B_t|\psi\rangle = \langle\psi\,(A_t \otimes I)\,|\,(I \otimes B_t)\,\psi\rangle = 1$, we have $A_t \otimes I|\psi\rangle = I \otimes B_t|\psi\rangle$, so

$$\sum_{i=1}^{\infty} \alpha_i \left(A_t|\phi_i\rangle\right)|\psi_i\rangle = \sum_{i=1}^{\infty} \alpha_i |\phi_i\rangle \left(B_t|\psi_i\rangle\right). \tag{10}$$

Both sides of Eq. (10) are Schmidt decompositions of the same quantum state. Now we can use the fact that the Schmidt decomposition is unique up to a change of basis for the subspace associated with each distinct Schmidt coefficient. Consider any Schmidt coefficient with multiplicity d (each Schmidt coefficient appears with finite multiplicity because $\sum_{i=1}^{\infty}|\alpha_i|^2 = 1$). Suppose, without loss of generality, that $\alpha_1 = \alpha_2 = \cdots = \alpha_d = \alpha$. Then the span of $\{A_t|\phi_i\rangle : i \in \{1, 2, \ldots, d\}\}$ equals the span of $\{|\phi_i\rangle : i \in \{1, 2, \ldots, d\}\}$. In other words, A_t leaves the subspace spanned by $\{|\phi_i\rangle : i \in \{1, 2, \ldots, d\}\}$ fixed. By similar reasoning, B_t leaves the subspace spanned by $\{|\psi_i\rangle : i \in \{1, 2, \ldots, d\}\}$ fixed. Therefore, there exist bases in which A_t and B_t have block decompositions of the form

$$A_t = \begin{pmatrix} A'_t & 0 & 0 & \ldots \\ 0 & A''_t & 0 & \ldots \\ 0 & 0 & A'''_t & \ldots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \qquad B_t = \begin{pmatrix} B'_t & 0 & 0 & \ldots \\ 0 & B''_t & 0 & \ldots \\ 0 & 0 & B'''_t & \ldots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{11}$$

with one block for the subspace of each Schmidt coefficient. We can take, say, the d-dimensional observables from the first block $\{A'_t : t \in \{1, 2, \ldots, n\}\}$ as a quantum satisfying assignment (which changes the effective entanglement to a d-dimensional maximally entangled state).

Case 2: POVM Measurements for Alice. A POVM measurement can be expressed as a projective measurement in a larger Hilbert space that includes ancillary qubits, as shown in Figure 3. Again we can define binary observables for j^{th} variable in a constraint s as in Case 1.

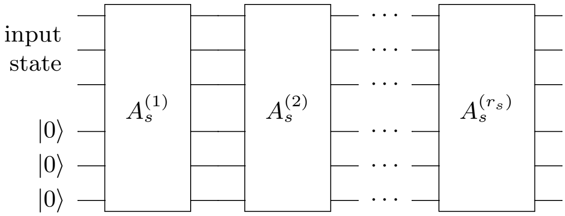


Fig. 3. Alice's POVM measurement on receiving input s expressed in Stinespring form (Case 2)

$$A_s^{(j)} = \sum_{a \in \{0,1\}^{r_s}} (-1)^{a_j} \Pi_a, \tag{12}$$

these observables act on the larger Hilbert space $\mathcal{H}_s \otimes \mathcal{H}_p$. Here $\mathcal{H}_s$ ($\mathcal{H}_p$) represents the Hilbert space for the entangled (private) qubits. Like before, the

$\{A_s^{(j)} : j \in \{1, \ldots, r_s\}\}$ is a set of commuting binary observables. Since these observables commute, without loss of generality, any of the corresponding variables can be measured first by Alice.

We will focus on the first measurement done by Alice given some constraint. Let us suppress the superscript and subscript for brevity of notation. Say, Alice uses observable A for the first measurement corresponding to variable t. This defines a projective measurement $(\Pi_0 = \frac{A+I}{2}, \Pi_1 = \frac{I-A}{2})$ on $\mathcal{H}_s \otimes \mathcal{H}_p$.

Suppose that the reduced entangled state on Alice's side is ρ. Then Alice's strategy is to apply the channel which adds the ancilla qubits to ρ and then applies the measurement (Π_0, Π_1). Using the Kraus operators of this channel, we can come up with *equivalent* POVM elements E_0, E_1 acting on the Hilbert space $\mathcal{H}_s$. Here equivalent means, for all $i \in \{0, 1\}$ and $|\phi\rangle \in \mathcal{H}_s$,

$$\langle \phi, 00\ldots0|\Pi_i|\phi, 00\ldots0\rangle = \langle \phi|E_i|\phi\rangle. \tag{13}$$

Similarly, Bob has POVM elements (F_0, F_1) to measure variable t. Since their strategy is perfect, they always answer with same bit when asked for the variable t, which implies

$$\langle \psi|E_0 \otimes F_0|\psi\rangle + \langle \psi|E_1 \otimes F_1|\psi\rangle = 1. \tag{14}$$

This can be simplified to

$$\langle \psi|(E_0 - E_1) \otimes (F_0 - F_1)|\psi\rangle = 1. \tag{15}$$

Now we use the following lemma to prove that (E_0, E_1) is actually a projective measurement (similarly (F_0, F_1) is projective).

Lemma 2. *Let $|\psi\rangle \in \mathcal{H}_A \otimes \mathcal{H}_B$ be such that $|\psi\rangle = \sum_{i=1}^n \alpha_i |\phi_i\rangle|\psi_i\rangle$, where $\alpha_1, \alpha_2, \ldots, \alpha_n > 0$. If we have two POVM measurements, (E_0, E_1) on $\mathcal{H}_A$ and (F_0, F_1) on $\mathcal{H}_B$, such that*

$$\langle \psi|(E_0 - E_1) \otimes (F_0 - F_1)|\psi\rangle = 1 \tag{16}$$

then (E_0, E_1) and (F_0, F_1) are projective measurements.

Proof (Lemma 2). We will prove that (E_0, E_1) is a projective measurement. The proof for (F_0, F_1) is the same.

Notice that E_0 and E_1 are simultaneously diagonalizable (they are both Hermitian and $E_0 + E_1 = I$). The dimension of the system is n which can be set to ∞ to indicate that it is countably infinite. In the basis which diagonalizes them,

$$E_0 = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix} \quad \text{and} \quad E_1 = \begin{pmatrix} 1-\lambda_1 & & & \\ & 1-\lambda_2 & & \\ & & \ddots & \\ & & & 1-\lambda_n \end{pmatrix}.$$

This implies that E_0 and E_1 can be thought of as a probability distribution on 2^n projective measurements in the following way. For each $S \subseteq [n]$, define the projectors $\Pi_0^S = \sum_{i\in S} |i\rangle\langle i|$ and $\Pi_1^S = I - \Pi_0^S$, and $p_S = \prod\limits_{i\in S} \lambda_i \prod\limits_{i\notin S}(1-\lambda_i)$. Note that $\sum\limits_{S\subseteq[n]} p_S = 1$. It is straightforward to verify that

$$E_0 = \sum_{S\subseteq[n]} p_S \Pi_0^S \quad \text{and} \quad E_1 = \sum_{S\subseteq[n]} p_S \Pi_1^S. \tag{17}$$

By Eqns. (16), (17), and linearity,

$$\sum_{S\subseteq[n]} p_S \, \langle\psi|(\Pi_0^S - \Pi_1^S)\otimes(F_0 - F_1)|\psi\rangle = 1. \tag{18}$$

In the above equation, p_S's sum up to 1, and the term multiplied to them is at most 1. By an averaging argument, for all S with $p_S > 0$,

$$\langle\psi|(\Pi_0^S - \Pi_1^S)\otimes(F_0 - F_1)|\psi\rangle = 1. \tag{19}$$

Using Lemma 1, the $(\Pi_0^S - \Pi_1^S)$ have to be same for all S with $p_S > 0$. Hence, there can be at most one p_S with non-zero probability. Hence (E_0, E_1) is a projective measurement.

□

Now we know that (E_0, E_1) is a projective measurement. Also, using Eq. (13), any eigenvector $|\phi\rangle$ of E_i can be converted into an eigenvector $|\phi, 00\cdots 0\rangle$ for Π_i with same eigenvalue. Then, in the basis where eigenvectors of the form $|\phi, 00\cdots 0\rangle$ are listed first,

$$\Pi_0 = \left(\begin{array}{c|c} E_0 & 0 \ \cdots \ 0 \\ \hline 0 & \\ \vdots & M_0 \\ 0 & \end{array}\right) \quad \text{and} \quad \Pi_1 = \left(\begin{array}{c|c} E_1 & 0 \ \cdots \ 0 \\ \hline 0 & \\ \vdots & M_1 \\ 0 & \end{array}\right). \tag{20}$$

It is given that the observables $\Pi_0 - \Pi_1$ corresponding to different variables in the same context commute. It follows that the observables $E_0 - E_1$ corresponding to different variables in the same context also commute. Hence the proof for Case 2 follows from Case 1.

□

From the argument at the end of the first case, it follows that if we have a perfect strategy using countably infinite entanglement then it can be converted into a strategy having finite entanglement. The generic conversion (Sec. 1.2) of quantum satisfying assignments to a quantum strategy uses maximally entangled state. Hence Theorem 1 shows that if there is a perfect strategy for a BCS game then there exist a perfect strategy which uses maximally entangled state.

3 Proving Gaps on the Maximum Success Probability

Due to space constraints, the content of this section is omitted; however, it is available in the full version of this paper [10], which can be accessed at `http://arxiv.org/abs/1209.2729`.

The main result in this section is an upper bound below 1 on the entangled value of some BCS games of interest, under the assumption that the entanglement is a maximally mixed state (of arbitrarily high dimension).

4 Related Work

After the results of this article were made public, Arkhipov [3] studied the restricted case of parity BCS games where every variable appears in at most two constraints. He showed that these games have a perfect entangled strategy if and only if a related *dual* graph of the game is non-planar. The result combines elegant techniques with Kuratowski's theorem and our characterization of perfect strategies (in the sense that [3] makes use of our characterization).

More recently, Ji [13] showed that interesting examples like quantum chromatic number and Kochen-Specker sets can be described in the BCS game framework. He used special gadgets, called *commutativity* gadgets, to show reductions between various BCS's which preserve satisfiability using quantum assignments. Also, he showed that, for all k, there exists a parity BCS game which requires at least k entangled qubits to play perfectly.

5 Open Questions

There are many questions left open by this work. We have a characterization of perfect strategies for BCS games. It shows that there always exists a perfect strategy using maximal entanglement if a perfect entangled strategy exist. Still, given a game, deciding whether it has a perfect strategy is open.

There are questions pertaining to the *optimal* values of BCS games (the maximum success probability achievable), such as problem of computing these values, or approximations of them. Another question is whether there always exists an optimal strategy for a BCS game which uses maximally entangled states.

All of the above questions can be asked for general non-local games too. For the case of XOR games, the optimal value is given by a semidefinite program [9,18]. This shows how to compute the optimal value of the game and that there always exist an optimal strategy which uses maximally entangled states [9]. It is also known for graph coloring games (like BCS games) that there always exists a perfect strategy using maximal entanglement (if a perfect entangled strategy exist) [11]. But whether this is true for general games that have perfect strategies remains open.

Acknowledgments. We are grateful for discussions about this project with many people, including Alex Arkhipov, Harry Buhrman, Sevag Gharibian, Tsuyoshi Ito, Kazuo Iwama, Zhengfeng Ji, Hirotada Kobayashi, François Le Gall, Laura Mancinska (for pointing out an error in a previous version of this manuscript pertaining to the analysis of the case of POVM measurements), Oded Regev, Florian Speelman, Sarvagya Upadhyay, John Watrous, and Ronald de Wolf. Some of this work took place while the first author was visiting Amsterdam's CWI in 2011. This work is partially supported by Canada's NSERC and CIFAR.

References

1. Aravind, P.K.: Bell's Theorem without inequalities and only two distant observers. Found. Phys. Lett. 15(4), 397–405 (2002)
2. Aravind, P.K.: Quantum mysteries revisited again. Am. J. Phys. 72, 1303–1307 (2004)
3. Arkhipov, A.: Extending and characterizing quantum magic games. arXiv:1209.3819 [quant-ph] (2012)
4. Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and the hardness of approximation problems. J. ACM 45(3), 501–555 (1998)
5. Babai, L., Fortnow, L., Lund, C.: Non-deterministic exponential time has two-prover interactive protocols. Computational Complexity 1, 3–40 (1991)
6. Bell, J.S.: On the Einstein-Podolsky-Rosen paradox. Physics 1, 195–200 (1964)
7. Ben-Or, M., Goldwasser, S., Kilian, J., Wigderson, A.: Multi-prover interactive proofs: How to remove intractability assumptions. In: Proc. of the 20th ACM Symp. on Theory of Computing (STOC 1988), pp. 113–131 (1988)
8. Clauser, J.F., Horne, M.A., Shimony, A., Holt, R.A.: Proposed experiment to test local hidden-variable theories. Phys. Rev. Lett. 23(15), 880–884 (1969)
9. Cleve, R., Høyer, P., Toner, B., Watrous, J.: Consequences and limits of nonlocal strategies. In: Proc. of the 19th IEEE Conf. on Computational Complexity (CCC 2004), pp. 236–249 (2004)
10. Cleve, R., Mittal, R.: Characterization of binary constraint system games (2013), http://arxiv.org/abs/1209.2729
11. Cameron, P.J., Montanaro, A., Newman, M.W., Severini, S., Winter, A.: On the quantum chromatic number of a graph. The Electronic Journal of Combinatorics [Electronic Only] 14(1):Research Paper R81, 15 (2007)
12. Feige, U., Goldwasser, S., Lovász, L., Safra, S., Szegedy, M.: Interactive proofs and the hardness of approximating cliques. J. ACM 43(2), 268–292 (1996)
13. Ji, Z.: Binary constraint system games and locally commutative reductions. arXiv:1310.3794 [quant-ph] (2013)
14. Mermin, N.D.: Simple unified form for the major no-hidden-variables theorems. Phys. Rev. Lett. 65(27), 3373–3376 (1990)
15. Mermin, N.D.: Hidden variables and the two theorems of John Bell. Rev. Mod. Phys. 65(3), 803–815 (1993)
16. Nielsen, M.A., Chuang, I.L.: Quantum computation and quantum information. Cambridge University Press, Cambridge (2000)
17. Speelman, F.: Personal communication (2011)
18. Cirel'son, B.S.: Quantum generalizations of Bell's inequality. Lett. in Math. Phys. 4(2), 93–100 (1980)

Fast Algorithms for Constructing Maximum Entropy Summary Trees[⋆]

Richard Cole[1,⋆⋆] and Howard Karloff[2]

[1] Computer Science Department, Courant Institute, NYU
[2] Yahoo! Research

Abstract. Karloff and Shirley recently proposed "summary trees" as a new way to visualize large rooted trees (Eurovis 2013) and gave algorithms for generating a maximum-entropy k-node summary tree of an input n-node rooted tree. However, the algorithm generating optimal summary trees was only pseudo-polynomial (and worked only for integral weights); the authors left open existence of a polynomial-time algorithm. In addition, the authors provided an additive approximation algorithm and a greedy heuristic, both working on real weights.

This paper shows how to construct maximum entropy k-node summary trees in time $O(k^2 n + n \log n)$ for *real* weights (indeed, as small as the time bound for the greedy heuristic given previously); how to speed up the approximation algorithm so that it runs in time $O(n + (k^4/\epsilon) \log(k/\epsilon))$, and how to speed up the greedy algorithm so as to run in time $O(kn + n \log n)$. Altogether, these results make summary trees a much more practical tool than before.

1 Introduction

How should one draw a large n-node rooted tree on a small sheet of paper or computer screen? Recently, in Eurovis 2013, Karloff and Shirley [4] proposed a new way to visualize large trees. While the best introduction to summary trees appears in [4], here we give a necessarily short description. A user has an n-node node-weighted tree T and wants to draw a k-node summary S of T on a small screen or sheet of paper, k being user-specified. We begin with an informal, bottom-up, operational description. Two type of contraction are performed: subtrees are contracted to single nodes that represent the corresponding subtrees; similarly multiple sibling subtrees (subtrees whose roots are siblings) are contracted to single nodes representing them. The node resulting from the latter contraction is called a group node. The one constraint is that each node in the summary tree have at most one child that is a group node. An example is shown in Figure 1 (based on a figure in [4]).

Next, we give a more formal description. Let T_v denote the subtree of T rooted at v. We name each node of S by the set of nodes of T that it represents.

[⋆] A full version of the paper is available at `http://arxiv.org/abs/1404.5660`

[⋆⋆] Richard Cole's work was supported in part by NSF grant CCF-1217989.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 332–343, 2014.

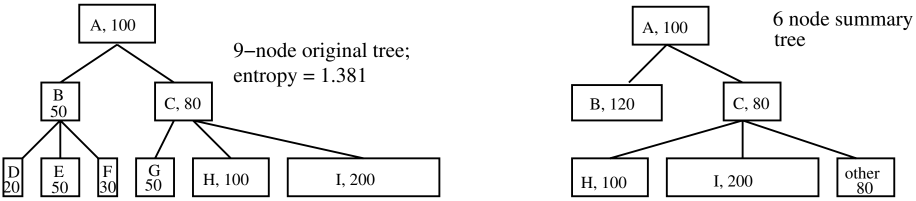


Fig. 1. On the left, a 9-node tree with node weights, and to the right, a 6-node summary tree of the original 9-node tree

The following comprise the possible *summary trees for* T_v: If T_v has just one node, the only summary tree is the one node $\{v\}$. Otherwise, a summary tree for T_v is one of:

1. a one-node tree $V(T_v)$ (the set of nodes in T_v); or
2. a singleton node $\{v\}$ and summary trees for the subtrees rooted at the children of v (and edges from $\{v\}$ to the roots of these summary trees); or
3. a singleton node $\{v\}$, a node $other_v$ representing a non-empty subset U_v of v's children and all the descendants of the nodes $x \in U_v$, and for each of v's children $x \notin U_v$ a summary tree for T_x (and edges from $\{v\}$ to $other_v$ and to the roots of the summary trees for each T_x).[1] Sometimes we will overload the term $other_v$ by using it to denote the subset U_v.

We allow arbitrary nonnegative real weights w_v on the nodes v of the input tree T. The weight of a node in a summary tree is defined to be the sum of the weights of the corresponding nodes in T. Paper [4] defined the entropy of a k-node summary tree with nodes of weights $W_1, W_2, ..., W_k$ to be $-\sum_{i=1}^{k} p_i \lg p_i$, where $p_i = W_i/W$ and W is the sum of all node weights, the usual information-theoretic entropy. Paper [4] then proposed that the most informative summary trees are those of maximum entropy. As noted in [4], this is a natural way to think about the information contained in a node-weighted tree. For given a bound on the number of nodes available in a summary tree, it seems plausible that a best summary tree is one of maximum entropy, because it is theoretically the most informative. This provided a principled way to identify the best k-node summary tree, in contrast to more heuristic and operational rules in prior work.

The fact that $other_v$ is an *arbitrary* non-empty subset of v's potentially large set of children is what makes finding maximum entropy summary trees difficult. Indeed, [4] resorted to using a dynamic program over the node weights (which worked provided that the weights were integral) and which led to a final running time of $O(K^2nW)$, where W is the sum of the node weights and K is the maximum k for which one is interested in finding a k-node summary tree. Given K, the dynamic program finds maximum entropy k-node summary trees for $k = 1, 2, \ldots, K$; from now on we assume that the user specifies K and k-node

[1] $other_v$ sets of size 1 are covered by Cases 2 and 3, but this redundancy is convenient for the algorithm description.

Table 1. Running times of the algorithms; $W_0 = O((K/\epsilon)\log(K/\epsilon))$

	Optimal Entropy	Greedy	ϵ-Approximate
Known results	$O(K^2 nW)$ [4]	$O(K^2 n + n\log n)$ [4]	$O(K^2 nW_0)$ [4]
New results	$O(K^2 n + n\log n)$	$O(Kn + n\log n)$	$O(n + K^3 W_0 + W_0 \log W_0)$

summary trees are found for all $k \leq K$. The algorithm worked well when W was small, but failed to terminate on two of the five data sets used in [4].

The key to obtaining a running time independent of W is to develop a fuller understanding of the structure of maximum entropy summary trees. Our new understanding readily yields a truly polynomial-time algorithm. The main remaining challenge is to create and analyze an effective implementation. We give an algorithm running in time $O(K^2 n + n\log n)$ [2]; it generates maximum entropy summary trees even for real weights, assuming, of course, a real-arithmetic model of computation, which is necessary (even for integral weights) because of the computation of logarithms. This result is based on a structural theorem which shows that the *other* sets, while allowed to be arbitrary, can be assumed, without loss of generality, to have a simple structure.

To deal with the case of real weights or exceedingly large integral weights, [4] gave an algorithm based on scaling, rounding, and algorithmic discrepancy theory which builds a summary tree whose entropy is within ϵ *additively* of the maximum, in time $O(K^2 nW_0)$, where W_0 is $O((K/\epsilon)\log(K/\epsilon))$. Keep in mind here that K is meant to be small, e.g., 100 or 500, while n is meant to go to infinity, and also that W_0 is a function only of K and ϵ (and neither of n nor W). The key here was to show that scaling the real input weights to have sum W_0, rounding them using algorithmic discrepancy theory, and then running the exact dynamic program previously mentioned on the rounded weights caused a loss of only ϵ in the final entropy.

This paper shows that the same algorithm can be implemented in time $O(n + K^3 W_0 + W_0 \log W_0)$; this is linear time if n is larger than the other terms. The key here is to notice that if the sum of integral weights is W_0, which is small, and $n \gg W_0$, then most nodes have rounded weight 0. Surely one shouldn't have to devote a lot of time to nodes of weight 0, and our algorithm, by effectively replacing n by $O(W_0)$, exploits this intuition.

Last, [4] proposed a fast greedy algorithm to generate summary trees. Running in time $O(K^2 n + n\log n)$ (though [4] overlooked the $n\log n$ time needed for sorting), the algorithm never took longer than six seconds to run on the data sets of [4]. This paper shows that a simple modification to the greedy code, neither suggested in [4] nor implemented in the associated C code, specifically, not computing a k-node summary tree of a tree rooted at a node having fewer than k descendants, decreases the running time bound of the greedy algorithm

[2] Actually, this can be reduced to $O(K^2 n)$ time by using a combination of fast selection and sorting instead of sorting alone in various places.

from $O(K^2 n + n \log n)$ to $O(Kn + n \log n)$. While the modification is trivial, its analysis is not.

Taken together, these new results show that maximum entropy summary trees are a much more practical tool than was previously known.

A number of the proofs are omitted for lack of space. They can be found in the full version of the paper (see http://arxiv.org/abs/1404.5660).

Previous Work. Traditionally tree visualization involved either visualizing the entire tree or allowing the user to interactively specify tree parts of interest. Approaches taken include "Degree-of-interest trees" [2,3], "hyperbolic browsers" [5], and the "accordion drawing technique" [1,7]. "Space-filling" layouts, e.g., treemaps [9], are another popular method. Paper [6] is a recent survey on techniques for drawing large graphs. Also see [4] for other relevant previous work.

2 Structural Theorem

This section proves a structural theorem which implies that maximum entropy summary trees can be computed in polynomial time, in a real-arithmetic model of computation. We begin by relating our approach to the greedy algorithm from [4]. Let v be a node of an input tree and suppose that $\{v\}$ appears in the summary tree. Recall that $other_v$ denotes the group child of v, if any.

Definition 1. *1. The* size s_v *of a node* v *in* T *is the sum of the weights of its descendants.*

2. n_v *denotes the number of descendants of* v *(including* v*).*

3. d_v *denotes the degree of* v*, the number of children it has.*

4. $\langle v_1, v_2, ..., v_{d_v} \rangle$ *denotes the children of* v *when sorted into nondecreasing order by size. (Fix one sorted order for each* v*, breaking ties arbitrarily.)*

5. The prefixes *of* $\langle v_1, v_2, ..., v_{d_v} \rangle$ *are the sequences* $\langle v_1, v_2, ..., v_i \rangle$ *and sets* $\{v_1, v_2, ..., v_i\}$ *for* $i \geq 0$.

The greedy algorithm in [4] sorted and then processed the children of each node in nondecreasing order by size; more about this later. It finds a maximum entropy summary tree among those in which for each v, either $other_v$ does not exist or is a nonempty prefix of $\langle v_1, v_2, ..., v_{d_v} \rangle$, but this need not be the optimal summary tree. In fact, [4] gives a 7-node tree T for which the uniquely optimal 4-node summary tree has an $other_v$ node which is not a prefix of v's children. In their example, the greedy algorithm achieves approximately 1 bit of entropy, but the optimal summary tree achieves approximately 1.5 bits. This example proves that restricting $other_v$ to be a prefix of the list of v's children can lead to summary trees of suboptimal entropy. Consequently, [4] resorted to a pseudo-polynomial-time dynamic program in order to find the optimal *other* sets.

The definition of summary trees allows $other_v$ to represent an *arbitrary* nonempty subset of v's children (and all their descendants). However, in this paper we prove the surprising fact that, without loss of generality, in every summary tree *of maximum entropy*, $other_v$ can be assumed to have a special form, a simple extension of the "prefix" form used in the greedy algorithm from [4].

Definition 2. *The* near-prefixes *of $\langle v_1, v_2, \ldots, v_{d_v}\rangle$ are the sequences $\langle v_1, v_2, \ldots, v_i; v_j\rangle$ and the sets $\{v_1, v_2, \ldots, v_i; v_j\}$ where $i \geq 0$, $j \geq i+2$, and $j \leq d_v$. v_j is called the non-prefix element. This terminology is also applied to the sequence $\langle T_{v_1}, T_{v_2}, \ldots, T_{v_{d_v}}\rangle$ of trees rooted at $v_1, v_2, \ldots, v_{d_v}$, respectively.*

We prove the following structural theorem:

Theorem 1. *For each k, $1 \leq k \leq n$, there is a maximum entropy k-node summary tree S in which, for every node v, $other_v$, when present, is either a prefix or a near-prefix of $\langle T_{v_1}, T_{v_2}, \ldots, T_{v_{d_v}}\rangle$.*

Proof. For any summary tree R of an n-node tree T, let $M = 2n+1$ and define $\Phi(R) = \sum_{v: other_v \text{ exists}} M^{n-d_R(v)} \sum_{j: v_j \in other_v} j$, where $d_R(v)$ denotes the depth in R of the node $other_v$. Among all maximum entropy summary trees for T, let S be one for which $\Phi(S)$ is minimum. (The role of Φ will be to enable tie-breaking among equal-weight summary trees.)

Lemma 1. *Let v be a node of T such that $other_v$ exists in S. If $v_i \notin other_v$ and $v_j \in other_v$, where $i < j$, then T_{v_i} is represented by two or more nodes in S.*

Proof. Suppose, for a contradiction, that T_{v_i} is represented by a single node. Consider the following alternate summary tree S': S' is obtained from S by replacing v_j in $other_v$ by v_i, and by representing T_{v_j} by a single node. The number of nodes in the summary tree remains k.

Let s_0 denote the sum of the sizes of all the children of v in $other_v - \{v_j\}$. (Here "$other_v$" refers to $other_v$ before the change.) Then W times the increase in entropy in going from S to S' is given by

$$I = (s_0 + s_{v_i}) \lg \frac{W}{s_0 + s_{v_i}} + s_{v_j} \lg \frac{W}{s_{v_j}} - (s_0 + s_{v_j}) \lg \frac{W}{s_0 + s_{v_j}} - s_{v_i} \lg \frac{W}{s_{v_i}}.$$

The derivative of this term with respect to s_{v_i} is $\lg \frac{s_{v_i}}{s_0 + s_{v_i}} \leq 0$. As $i < j$, $s_{v_i} \leq s_{v_j}$, and thus I is necessarily nonnegative (for it declines to 0 at $s_{v_i} = s_{v_j}$); consequently, there is a nonnegative increase in entropy, and hence S' is also a maximum entropy summary tree. Furthermore, if d is the depth of $other_v$ in S, then $\Phi(S') - \Phi(S) \leq -(j-i)M^{n-d} < 0$, which contradicts the assumption that S is a maximum entropy summary tree of minimum $\Phi(S)$. ■

Lemma 2. *Let v be a node in T such that $other_v$ exists in S. If $v_i \notin other_v$ and $v_{i+1} \in other_v$, then $v_j \notin other_v$ for all $j > i+1$.*

Proof. Suppose, for a contradiction, that $v_j \in other_v$, for some $j > i+1$.

By Lemma 1, T_{v_i} is represented by two or more nodes in S. Hence $\{v_i\}$ appears as a node in the summary tree, and $\{v_i\}$ has one or more children in S. In S, let x be a descendant of $\{v_i\}$ of maximum depth in S. Node x is a proper descendant of $\{v_i\}$.

We will show now that combining node x with another node in a specified way yields a summary tree of T_{v_i} with one fewer node and having entropy at most s_{v_i} smaller. Node x is not $\{v_i\}$. Let y be x's parent in S. Node $y = \{u\}$ for some node u in T (since every nonleaf in a summary tree represents a single node of T). There are four cases to analyze, but before turning to them, we state the following simple lemma which we will need; it can be proven by calculus.

Lemma 3. *If $a, b \geq 0$, $-a \lg a - b \lg b + (a+b)\lg(a+b) \leq a+b$.*

Let s_x, for a node x in summary tree S, denote the sum of the weights of all the nodes of T represented by x. (For a node of the form *other*$_v$, we mean the sum of the *sizes* of all the children of v in *other*$_v$, or equivalently, the sum of the weights of all their descendants.)

Now we begin the case analysis. Let d be the depth in S of node $\{v_i\}$.

1. y's only child in S is x.
   We combine nodes x and $y = \{u\}$ into a node z representing T_u. Recall that w_u denotes u's weight. Then W times the entropy decrease equals

$$\begin{aligned} s_x \lg(W/s_x) + w_u \lg(W/w_u) - (s_x + w_u)\lg(W/(s_x+w_u)) \\ = \quad -s_x \lg s_x - w_u \lg w_u + (s_x + w_u)\lg(s_x + w_u) \\ \leq \quad s_x + w_u \quad \text{(by Lemma 3)} \quad = \; s_z \; \leq \; s_{v_i}. \end{aligned}$$

   This change leaves Φ unchanged.
2. x has a sibling in S and *other*$_u$ does not exist.
   Hence x is either $\{\alpha\}$ or T_α for some node $\alpha \in T$.
   We create a new *other*$_u$ node by combining x with an arbitrary sibling x' of x. Because x is of maximum depth in S, x' is either of the form $\{\beta\}$ (node β in T has no children) or T_β, for some β in T. The resulting entropy decrease equals

$$\begin{aligned} s_x \lg(W/s_x) + s_{x'} \lg(W/s_{x'}) - (s_x + s_{x'})\lg(W/(s_x+s_{x'})) \\ = \quad -s_x \lg s_x - s_{x'} \lg s_{x'} + (s_x + s_{x'})\lg(s_x + s_{x'}) \\ \leq \quad s_x + s_{x'} \quad \text{(by Lemma 3)} \quad \leq \; s_{v_i}. \end{aligned}$$

   This change can increase Φ by at most $2n \cdot M^{n-(d+1)}$, because the depth of the new *other*$_u$ node is at least $d+1$.
3. x has a sibling in S and $\{x\} =$ *other*$_u$.
   We choose an arbitrary sibling x' of x and add it to *other*$_u$. The entropy calculation is the same as for Case 2. This change can increase Φ by at most $n \cdot M^{n-(d+1)}$, where d is the depth of $\{v_i\}$ in S.
4. x has a sibling in S, *other*$_u$ exists, and and $\{x\} \neq$ *other*$_u$.
   We add x to *other*$_u$. Let x' be the node *other*$_u$. The calculations are exactly the same as in Case 3.

In all four cases, the decrease in entropy is at most s_{v_i} and the increase in Φ is at most $2nM^{n-d-1}$.

Now we show how to generate a new maximum entropy summary tree S'. To get S', combine x as above with either its parent or a sibling, thereby decreasing the number of summary tree nodes by one, and then split off v_{i+1} from $other_v$ and create a node to represent $T_{v_{i+1}}$, thereby increasing the number of summary tree nodes back to k. Now, let s_0 denote the sum of the sizes of all the children of v in $other_v - \{v_{i+1}, v_j\}$. W times the increase in entropy from this two-part change to S is at least

$$\left[(s_0 + s_{v_j}) \lg \frac{1}{s_0 + s_{v_j}} + s_{v_{i+1}} \lg \frac{1}{s_{v_{i+1}}} - (s_0 + s_{v_{i+1}} + s_{v_j}) \lg \frac{1}{s_0 + s_{v_{i+1}} + s_{v_j}}\right] - s_{v_i}$$

$$= (s_0 + s_{v_j}) \lg \frac{s_0 + s_{v_{i+1}} + s_{v_j}}{s_0 + s_{v_j}} + s_{v_{i+1}} \lg \frac{s_0 + s_{v_{i+1}} + s_{v_j}}{s_{v_{i+1}}} - s_{v_i} \geq s_{v_{i+1}} - s_{v_i} \geq 0.$$

(The first inequality follows because $s_{v_j} \geq s_{v_{i+1}}$, which implies that $(s_0 + s_{v_{i+1}} + s_{v_j})/s_{v_{i+1}} \geq 2$.) But this is a nonnegative increase in entropy, proving that S' is a maximum entropy summary tree.

Splitting off v_{i+1} from $other_v$ decreases Φ by at least M^{n-d}, because the depth of the $other_v$ node equals the depth of node v_i, which is d. Hence the total $\Delta\Phi$ is at most $-M^{n-d} + 2n \cdot M^{n-d-1} = -M^{n-d}(1 - 2n/M) < 0$, a contradiction to the fact that S is a maximum entropy summary tree of minimum Φ. ■

This completes the proof of Theorem 1. ■

Theorem 2. *For all v, if $other_v$ exists, then $|other_v| \geq d_v - K + 2$.*

Proof. Each child of v not in $other_v$ contributes at least one node to the final summary tree, which has order $k \leq K$, and hence the number of children not in $other_v$ cannot exceed $K - 2$ (for one node is needed to represent $\{v\}$). ■

3 The Exact Algorithm

Relabel the nodes as $1, 2, ..., n$, with the root being node 1, the nodes at depth d getting consecutive labels, and the children of a node being labeled with increasing consecutive labels in nondecreasing size order. (This can be done by processing the nodes in nondecreasing order by depth, with all the children of node v processed consecutively in nondecreasing order by size.) This relabeling costs $O(n \log n)$ time,[3] because $\sum_v (d_v \log d_v) \leq \sum_v (d_v \log n) \leq n \log n$.

The description and the implementation of the algorithm are simplified if we compute what we call the"pseudo-entropy," of summary trees for T_v rather than their entropy. The *pseudo-entropy p-ent*(S_v) of a tree S_v with nodes of weights $W_1, W_2, \ldots, W_k$ is simply $-\sum p_i \log p_i$, where $p_i = W_i/W$ and W is the weight of T (and *not* of T_v). Clearly, if S_v is part of a summary tree S for T, then S_v

[3] In fact, the relative order, at node v, of its $d_v - K + 1$ smallest-sized children does not matter since they must all be included in $other_v$. This allows us to perform just a partial sort at each node, in which the $d_v - K + 1$ smallest-size children are identified by selection and then the remaining at most $K - 1$ children are sorted. This improves the $O(n \log n)$ term to $O(n \log K)$ which is dominated by $O(nK)$.

contributes $-\sum p_i \log p_i$ to the entropy of S. Let $\text{ent}(S_v)$ denote the entropy of tree S_v. Then

$$\begin{aligned}\text{ent}(S_v) &= -\sum_i \frac{W_i}{W_v}\log\frac{W_i}{W_v} \;=\; -\left[\frac{W}{W_v}\sum_i \frac{W_i}{W}\log\frac{W_i}{W} + \sum_i W_i W\log\frac{W}{W_v}\right]\\ &= -\frac{W}{W_v}\text{p-ent}(S_v) - \log\frac{W}{W_v}.\end{aligned}$$

Thus the same tree optimizes the entropy and the pseudo-entropy.

We will be using a dynamic programming algorithm. To simplify the presentation we will only describe how to compute the maximum pseudo-entropy for a k-node summary tree for T_v, for each node v and for all k, $1 \le k \le \min\{K, n_v\}$.

The algorithm will first seek to find the value of the pseudo-entropy for optimal k-node summary trees when $other_v$ is restricted to being a prefix set, and then when $other_v$ is restricted to being a near-prefix set containing v_j as its non-prefix element, for each possible v_j in turn, i.e., for $\max\{3, d_v - K + 3\} \le j \le d_v$. Thus the algorithm will consider $\min\{d_v - 1, K - 1\}\min\{d_v, K - 1\}$ classes of candidate $other_v$ sets.

To describe the algorithm it will be helpful to introduce the notion of a summary forest. A k-node *summary forest for* T_v is a $(k+1)$-node summary tree for T_v from which v has been excised (leaving a forest). We will also call this a summary forest for $T_{v_1}, T_{v_2}, \ldots, T_{v_{d_v}}$. A summary forest for $T_{v_1}, T_{v_2}, \ldots, T_{v_l}$ is defined analogously, for $1 \le l \le d_v$.

To find the pseudo-entropy-optimal k-node summary trees for T_v, for $1 \le k \le K$, we first find the pseudo-entropy of optimal k-node summary forests for $T_{v_1}, T_{v_2}, \ldots, T_{v_l}$, for $\max\{1, d_v - K + 2\} \le l \le d_v$. The optimal k-node summary trees for T_v are then obtained by attaching $\{v\}$ as a root node to the trees in the optimal $(k-1)$-node summary forests for $T_{v_1}, T_{v_2}, \ldots, T_{v_{d_v}}$.

Now we explain how to find these optimal summary forests. In turn, we consider each of the up-to-$\max\{1, K - 1\}$ possible classes of $other_v$ nodes: the prefix $other_v$ nodes, and for each j with $\max\{3, d_v - K + 3\} \le j \le d_v$, the class of near-prefix $other_v$ nodes including v_j as the non-prefix element.

First, we describe the handling of the candidate prefix $other_v$ nodes. We start with optimal k-node summary trees for T_{v_1}, for $1 \le k \le K - 1$. Inductively, suppose that we have computed (the entropy of) optimal k-node summary forests for $T_{v_1}, \ldots, T_{v_l}$. We find optimal k-node summary forests for $T_{v_1}, \ldots, T_{v_l}, T_{v_{l+1}}$ as follows. For $k = 1$, the forest comprises a single $other_v$ node. For each $k > 1$, we choose the highest entropy among the following options: an optimal h-node summary forest for $T_{v_1}, \ldots, T_{v_l}$ plus an optimal $(k - h)$-node summary tree for $T_{v_{l+1}}$, for $1 \le h < k$.

The correctness of this procedure is immediate: for $k = 1$ clearly the only summary forest is a one-node forest. For $k > 1$, $T_{v_{l+1}}$ cannot be represented by the $other_v$ node (since we are discussing the handling of the *prefix other$_v$* nodes) and so it must be represented by one tree in the summary forest; this implies that $T_{v_1}, T_{v_2}, \ldots, T_{v_l}$ must also be represented by one or more trees in the summary forest. Of course, the representation of each of the parts must be optimal.

Our algorithm considers all possible ways of partitioning the nodes in the summary forest among these two parts; consequently it finds an optimal forest.

The process when v_j is the non-prefix node in $other_v$ is essentially identical. There are two changes: (i) $other_v$ is initialized to contain T_{v_j} (rather than .eing the empty set) and (ii) the incremental sweep skips tree T_{v_j}. The correctness argument is as in the previous paragraph.

Finally, to obtain optimal k-node summary forests for $T_{v_1}, T_{v_2}, \ldots, T_{v_{d_v}}$ one simply takes the best among the k-node forests computed for the different classes of candidate $other_v$ nodes. Again, correctness is immediate.

Theorem 3. *The running time of the algorithm is $O(K^2 n + n \log n)$.*

NOTE. Our time bound is $O(K^2 n + n \log n)$ to build K maximum-entropy summary trees, or $O(Kn + (n \log n)/K)$ amortized time for each. There is an obvious lower bound of $\Omega(n + K^2)$ to build all K trees, since one has to read an n-node tree and produce trees having $1, 2, 3, \ldots, K$ nodes. Hence there cannot be a $O(n)$-time algorithm that generates all K trees, since it would violate the lower bound when K is $\omega(\sqrt{n})$. Of course, conceivably there is a linear-time algorithm to build a maximum-entropy k-node summary tree for a single value of k.

Proof. The running time is the sum of three terms:
(1) $O(n \log n)$, for sorting the children of all nodes by size.
(2) $O(Kn)$ for initializations. In fact, the initializations for node v take time $O(K \cdot \min\{d_v, K-1\})$, which is $O(Kn)$ time in total.
(3) For each node v, the cost of processing node v_l when processing each of the classes of candidate $other_v$ nodes. Let $\langle v_a, v_{a+1}, \ldots, v_{v_d} \rangle$ be the sequence of nodes processed when considering the candidate prefix $other_v$ sets (nodes $v_1, \ldots, v_{a-1}$ are the nodes guaranteed to be in $other_v$). When processing the near-prefix candidate $other_v$ sets with non-prefix element v_j, the same sequence will be processed except that v_j will be omitted. For the class of prefix candidate sets, the cost for processing v_{l+1}, for $a \le l < v_d$, is $\min\{K-1, n_{v_a} + n_{v_{a+1}} + \cdots + n_{v_l}\} \cdot \min\{K-1, n_{v_{l+1}}\} \le \min\{K-1, n_{v_1} + n_{v_2} + \cdots + n_{v_l}\} \cdot \min\{K-1, n_{v_{l+1}}\}$, for we are seeking k-node summary forests for $1 \le k \le K-1$, and the number of nodes in a summary tree cannot be more than the number of nodes available in the relevant subtrees of T. The same bound applies for each of the remaining classes of candidate $other_v$ sets and there are at most $K-1$ of these classes. Since the number of child nodes being processed when computing at node v is $d_v - a + 1 \le d_v$, the obvious upper bound here is $O(K^3 \cdot d_v)$. Summed over all v, this totals $O(K^3 \cdot n)$. However, Corollary 1 below shows that $\sum_{\text{non-leaf } v} \sum_l \min\{n_{v_1} + n_{v_2} + \cdots + n_{v_l}, K\} \cdot \min\{n_{v_{l+1}}, K\} \le 2Kn$, giving an overall time of $O(n \log n + K^2 n)$. ■

4 A Lemma for Running Time Analysis

In this section we state a lemma underlying the running time analysis of both the greedy algorithm and the exact algorithm. Let n be a positive integer and let T be a rooted, n-node tree, and *for this section only*, let $v_1, v_2, ..., v_{d_v}$ be v's children in *any* order.

Definition 3. *Relative to T, let $cost(v)$ be defined for all $v \in T$ as follows. If v is a leaf, $cost(v) = 0$. If v is not a leaf, $cost(v) = [\sum_{i=1}^{d_v} cost(v_i)] + [\sum_{i=1}^{d_v-1} \min\{n_{v_1} + n_{v_2} + \cdots + n_{v_i}, K\} \cdot \min\{n_{v_{i+1}}, K\}]$.*

Lemma 4. [Proof omitted.] *For all v, $cost(v) \le n_v^2$ if $n_v \le K$, and $cost(v) \le 2Kn_v - K^2$, if $n_v > K$.*

Corollary 1. [Proof omitted.] *For $K \ge 1$, $\sum_{non\text{-}leaf\ v}[\sum_{i=1}^{d_v-1} \min\{n_{v_1} + n_{v_2} + \cdots + n_{v_i}, K\} \cdot \min\{n_{v_{i+1}}, K\}] \le 2Kn$.*

5 Greedy Algorithm

The greedy algorithm proposed in [4] is precisely the algorithm proposed herein for the exact solution but with the *other* sets restricted to being prefix sets. In [4] Greedy was shown to run in time $O(K^2n + n \log n)$. Here, we shave off a factor of K from the first term.

Corollary 2. *(of Lemma 4).* [Proof omitted.] *The time needed by the greedy algorithm to generate summary trees of orders $k = 1, 2, \ldots, K$ is $O(Kn+n \log n)$.*

6 Improved Approximation Algorithm

In this section we describe an algorithm that computes an approximately entropy-optimal k-node summary tree. Our algorithm relies on the following outline from [4]:

1. One can rescale the weights in a tree to make them sum up to any positive integral value W_0, while leaving the entropy of any summary tree unchanged. (This is obvious.)
2. One can use algorithmic discrepancy theory to round each resulting real node weight w_v to value w'_v equal to either $\lfloor w_v \rfloor$ or $1 + \lfloor w_v \rfloor$ such that for each node $v \in T$, $|\sum_{u \in T_v} w'_u - \sum_{u \in T_v} w_u| \le 1$ *for all v simultaneously*, without changing the overall sum.
3. Using Naudts's theorem [8] that almost identical probability distributions have almost identical entropy, one can prove, for some integer W_0 which is $O((K/\epsilon) \log(K/\epsilon))$, that if one finds a maximum entropy summary tree T^* for the modified weights (w'_v), then T^* has entropy (measured according to the *original* weights w_v) at most ϵ less than that of the truly maximum entropy summary tree.

Suppose that the weights on T are integral and sum to W_0. Clearly the number of nodes of positive weight cannot exceed W_0; however, the 0-weight nodes could far outnumber the positive-weight nodes. Indeed, that is exactly what happens if $n \gg W_0$.

Our algorithm exploits the fact that little processing is needed for most of the 0-weight nodes. In fact, we will need to compute summary trees for only the non-zero weight nodes and for at most $2(W_0 - 1)$ 0-weight nodes.

The algorithm works with a tree T', a reduced version of T in which some 0-weight nodes have been removed. The following notation will be helpful. $F_T(v,k)$ denotes the maximum pseudo-entropy of a k-node summary tree of T_v, where T_v is a subtree of tree T; similarly, $F_{T'}(v,k)$ denotes the maximum pseudo-entropy of a k-node summary tree of T'_v, where T'_v is a subtree of tree T'.

T' is obtained from T as follows: for each positively-sized node v in T, if v has one or more size-0 children, remove them and their descendants and replace them all by a single 0-weight child. Clearly optimal summary trees in T' form optimal summary trees in T (for the only difference in summarizing T is that we could add 0-weight nodes no longer present in T', and these would contribute 0 to the entropy). Note that if v is a 0-weight non-leaf node in T' then it must have non-zero size (assuming T has at least one positive-weight node). The following result is immediate.

Lemma 5. *Let T have n nodes and T' have n' nodes. Let v be a node in T' with $n(v)$ descendants in T and $n'(v)$ descendants in T'. Then $F_T(v,k) = F_{T'}(v,k)$ for $1 \le k \le n'(v)$. For $n'(v)+1 \le k \le n$, $F_T(v,k) = F_{T'}(v,n'(v))$.*

Note that $F_{T'}(v,n'(v))$ is attained by a partition of the set of v's children in T' into singletons.

(Now of course we *have* changed the problem, since T' might have fewer than K nodes. However, if this happens, then optimal summary trees of T having more than $|T'|$ nodes have no more entropy than optimal summary trees of T having exactly $|T'|$ nodes.)

Even after the reduction it may be the case that $|T'| \gg W_0$, for T' might still contain long paths of 0-weight nodes in which each node has only one positively-sized child. However, the following lemmas show that they add little to the cost of computing optimal summary trees.

Lemma 6. *Let v be a 0-weight node in T' with a single child u. Then for $2 \le k \le |T'_v|$, $F_{T'}(v,k+1) = F_{T'}(u,k)$; also $F_{T'}(v,1) = F_{T'}(u,1)$.*

Proof. For $k \ge 2$, the $(k+1)$-node summary tree for T'_v adds a zero-weight node $\{v\}$ to the k-node summary tree for T'_u. For $k=1$ both trees have a single node of weight w_u.

Lemma 7. *Let v be a 0-weight node in T' with exactly two children, a 0-weight leaf v_1 and a child u of positive size. Then for $3 \le k \le |T'_v|$, $F_{T'}(v,k+2) = F_{T'}(u,k)$; also $F_{T'}(v,2) = F_{T'}(v,1) = F_{T'}(u,1)$.*

The proof of this lemma is essentially the same as that of Lemma 6. The following corollary is immediate.

Corollary 3. *Let $v_1, v_2, \ldots, v_l$, for $l > 1$, be a descending path of 0-weight nodes in T' such that each v_i, $1 \le i \le l$ either has one child, or has exactly two children* *one of which is a 0-weight leaf. Further suppose that l' of these nodes are in the second category. Node v_l must have a child of positive size (as otherwise $v_1 \neq v_l$ would be a size-0 non-leaf). Let u be the child of v_l of positive size. Then for $1 \leq k \leq |T'_{v_1}| - (l + l')$, $F_{T'}(v_1, k + l + l') = F_{T'}(u, k)$; and for $j \leq l + l'$, $F_{T'}(v_1, j) = F_{T'}(u, 1)$.*

This corollary implies that given the entropies of optimal entropy summary trees at a node u at the bottom of a maximal path of 0-weight nodes one can obtain the entropies of the optimal entropy summary trees at node v_1 at the top of the path in time $O(K)$.

At the remaining nodes in T' we perform the same computation as in the exact algorithm. As we can show, there are $O(W_0)$ such nodes, which leads to the following running time bound.

Theorem 4. [Proof omitted.] *The approximation algorithm to obtain a summary tree that has entropy within an additive ϵ of the optimal summary runs in time $O(n + W_0 \cdot K^3)$, where $W_0 = O((K/\epsilon) \log(K/\epsilon))$.*

Acknowledgments. We thank the referees for their helpful suggestions.

References

1. Beermann, D., Munzner, T., Humphreys, G.: Scalable, Robust Visualization of Very Large Trees. In: Proc. EuroVis, pp. 37–44 (2005)
2. Card, S.K., Nation, D.: Degree-of-Interest Trees: A Component of an Attention-Reactive User Interface. In: Proceedings of the Working Conference on Advanced Visual Interfaces, pp. 231–245 (2002)
3. Heer, J., Card, S.K.: DOI Trees Revisited: Scalable, Space-Constrained Visualization of Hierarchical Data. In: Advanced Visual Interfaces, pp. 421–424 (2004), http://vis.stanford.edu/papers/doitrees-revisited
4. Karloff, H., Shirley, K.: Maximum Entropy Summary Trees. In: Eurovis 2013 (2013), www2.research.att.com/~kshirley/KarloffShirleyWebsite.pdf
5. Lamping, J., Rao, R., Pirolli, P.: A Focus+Context Technique Based on Hyperbolic Geometry For Visualizing Large Hierarchies. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 401–408 (1995), http://dx.doi.org/10.1145/223904.223956
6. von Landesberger, T., Kuijper, A., Schreck, T., Kohlhammer, J., van Wijk, J.J., Fekete, J.-D., Fellner, D.W.: Visual Analysis of Large Graphs: State-of-the-Art and Future Research Challenges. Computer Graphics Forum 30(6), 1719–1749 (2011)
7. Munzner, T., Guimbretiere, R., Tasiran, S., Zhang, L., Zhou, Y.: TreeJuxtaposer: Scalable tree comparison using Focus+Context with guaranteed visibility. ACM Transactions on Graphics 22(3), 453–462 (2003)
8. Naudts, J.: Continuity of a Class of Entropies and Relative Entropies. Reviews in Mathematical Physics 16(6), 809–822 (2004)
9. Shneiderman, B.: The Eyes Have It: A Task by Data Type Taxonomy for Information Visualization. In: Proceedings of the IEEE Symposium on Visual Languages, pp. 336–343 (1996)

Thorp Shuffling, Butterflies, and Non-Markovian Couplings*

Artur Czumaj[1] and Berthold Vöcking[2]

[1] Department of Computer Science and Centre for Discrete Mathematics and its Applications,
University of Warwick
A.Czumaj@warwick.ac.uk
[2] Department of Computer Science, RWTH Aachen University
voecking@cs.rwth-aachen.de

Abstract. *Thorp shuffle* is a simple model for a random riffle shuffle that for many years has eluded good analysis. In Thorp shuffle, one first cuts a deck of cards in half, and then starts dropping the cards from the left or right hand as with an ordinary shuffle, so that at each time, one chooses the left or right card with probability $\frac{1}{2}$ and drops it, and then drops the card from the opposite hand. Then one continues this inductively until all cards have been dropped. The question is how many times one has to repeat this process to randomly shuffle a deck of n cards. Despite its rather simple description and wide interest in understanding its behavior, Thorp shuffle has been very difficult to analyze and only very recently, Morris showed that Thorp shuffle mixes in a polylogarithmic number of rounds.

In our main result, we show that if Thorp shuffle mixes sequences consisting of $n-k$ distinct elements together with k identical elements (so-called *k-partial n-permutations*) with $k = \Theta(n)$, then $\mathcal{O}(\log^2 n)$ rounds are sufficient to randomly mix the input elements. In other words, $\mathcal{O}(\log^2 n)$ Thorp shuffles with n input elements randomly permutes any set of cn elements with any $c < 1$, or, equivalently, is almost cn-wise independent. The key technical part of our proof is a novel analysis of the shuffling process that uses *non-Markovian coupling*. While non-Markovian coupling is known to be more powerful than the Markovian coupling, our treatment is one of only a few examples where strictly non-Markovian coupling can be used to formally prove a strong mixing time. Our non-Markovian coupling is used to reduce the problem to the analysis of some random process in networks (in particular, when n is a power of two then this is in a butterfly network), which we solve using combinatorial and probabilistic arguments.

Our result can be used to randomly permute any number of elements using Thorp shuffle: If the input deck has N cards, then add another set of $0.01N$ "empty" cards and run $\mathcal{O}(\log^2 N)$ Thorp shuffles. Then, if we remove the empty cards, the obtained deck will have the original N cards randomly permuted.

We also analyze a related shuffling process that we call *Perfect shuffle*. We cut a deck of n cards into two halves, randomly permute each half, and then perform one step of Thorp shuffle. Apart from being interesting on its own, our motivation to study this process is that a single Perfect shuffle is very similar to $\mathcal{O}(\log n)$ Thorp shuffles, and thus understanding of Perfect shuffle can shed some light on the performance of Thorp shuffle. We apply coupling to show that Perfect shuffle mixes in $\mathcal{O}(\log \log n)$ steps, which we conjecture to be asymptotically tight.

* Research supported in part by the Centre for Discrete Mathematics and its Applications (DIMAP), EPSRC award EP/D063191/1.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 344–355, 2014.

1 Introduction

Let n be an even number. Thorp [21] proposed the following card shuffling process:

Thorp shuffle:

- Cut the deck of n card into exactly two halves.
- Repeat until both halves are empty:
  - ◇ drop the top card from the left half or the right half, using a fair random coin flip;
  - ◇ then drop the card from the other half.

In 1973, Thorp [21] asked about the number of times Thorp shuffle has to be repeated to (almost) randomly shuffle (mix) a deck of n cards. (The original motivation of Thorp to study Thorp shuffle was to understand the ways of taking advantage of poor shuffling in casino games.) Despite its rather simple description and also interest in understanding its behavior, Thorp shuffle has proven to be very difficult to analyze. It has been conjectured (see, e.g., [2]) that after $\Theta(\log^2 n)$ shuffles the obtained permutation of cards will be almost random[1], but despite many attempts, for a long time no proof of $o(n)$ mixing time bounds has been found. Only recently Morris [15,16], using the method of evolving sets proved that the mixing time of Thorp shuffle on $n = 2^d$ cards is polylogarithmic in n, i.e., it is $\log^{\mathcal{O}(1)} n$. The precise bound for the mixing time proved by Morris [16] was $\mathcal{O}(\log^{44} n)$ (though the preliminary version of [16], which appeared in [15], claimed a stronger bound which has been later retracted). This has been later improved to $\mathcal{O}(\log^{29} n)$ by Montenegro and Tetali [14, Theorem 6.15], who used spectral profile techniques as well as the method of evolving sets. Very recently, Morris improved the analysis and showed the mixing time of $\mathcal{O}(\log^4 n)$ [17], without the assumption that n is a power of two, and then again, Morris [18] improved that bound to $\mathcal{O}(\log^3 n)$ *for n being a power of* 2. While this is a polylogarithmic mixing time, it is still off from the conjectured mixing time of $\mathcal{O}(\log^2 n)$ [2, Example 12].

Thorp Shuffle and Butterfly Networks. When n is a power of 2, then the Thorp shuffle (more precisely, its $\log_2 n$ repetitions) can be modeled by a random process of passing the cards through a *butterfly network*, where at each stage of the butterfly, the neighboring cards are interchanged with probability $\frac{1}{2}$. Butterfly networks have been used in computer science in various setting, and here our use has flavor similar to that used for sorting (comparator) networks [13].

A *butterfly network* with $n = 2^d$ inputs is a layered network with $d{+}1$ levels $0, \dots, d$ consisting of nodes and *switches*. There are n nodes at each level, each node corresponding to a distinct binary number of d bits, which is called the *label* of that node. The nodes are connected by *switches*, with $\frac{n}{2}$ switches connecting the nodes between any two consecutive levels. A switch between levels ℓ and $\ell + 1$ connects two nodes

[1] Even though we would like to generate a uniformly random permutation, this is impossible for the processes considered in this paper and we will settle with *almost random permutations*. By that we mean that the total variation distance (the L_1-distance) between the distribution of the obtained permutation and the uniformly random permutation is $o(1)$. In view of that, we will abuse the notation and refer to such *almost random permutations* as random permutations.

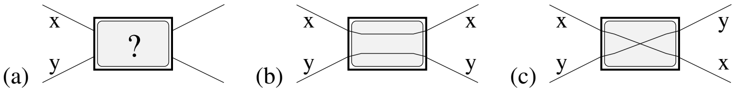


Fig. 1. A *switch* and its two possible outcomes (depending on the outcome of coin toss)

from level ℓ to two nodes with identical labels from level $\ell + 1$. For each node x with label $\langle x_{d-1}, x_{d-2}, \ldots, x_0 \rangle$ that is at level ℓ, $0 \le \ell \le d-1$, x is connected to a switch between levels ℓ and $\ell + 1$ together with a node $\bar{x}$ that has label $\langle x'_{d-1}, x'_{d-2}, \ldots, x'_0 \rangle$, where $x'_i = 1 - x_i$ if $i = \ell$, and $x'_i = x_i$ otherwise.

While in sorting networks (cf. [13]) the role of a switch (called there a comparator) is to sort the numbers from the two incoming inputs, in our case, the switch performs a uniformly random choice of which incoming element from level ℓ will go to which outgoing node at level $\ell + 1$. In view of this, in this paper a *switch* (Figure 1) will be a gate with two inputs and two outputs that takes two inputs and return them as the outputs in a random order. We define a *butterfly shuffle* (Figure 2) to be the process that takes as the input n elements and run them through the butterfly network with switches making random selections of the outputs. With this, the following lemma is known.

Lemma 1. *Let n be a power of two. Then, applying* $\log_2 n$ *Thorp shuffles is equivalent to a butterfly shuffle.* □

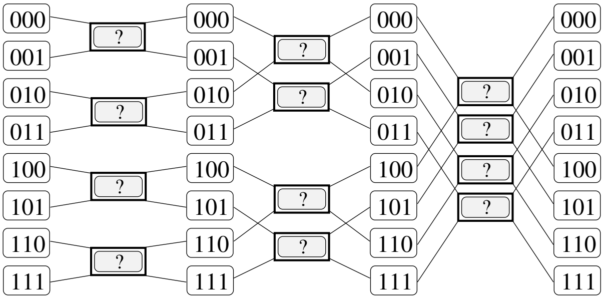


Fig. 2. A butterfly shuffle for $n = 8$

Because of Lemma 1, our goal is to analyze the *random butterfly process* which is the process of repeatedly applying butterfly shuffle to the input permutation. (See Theorem 4 for a claim extending this analysis to arbitrary values of n.)

Random Switching Networks. A construction similar to that given above can be generalized to arbitrary switching networks (see, e.g., [5,20]). A *switching network* $\mathcal{N}$ with n inputs is a layered network with n nodes in each layer. The connection between the nodes is only between the nodes in two consecutive layers. Every node at layer ℓ is either (directly) connected to a unique node at layer $\ell + 1$, or is connected as the input to a switch with two inputs and two outputs. Every node at layer $\ell + 1$ has either a connection from a unique node at layer ℓ or is connected as the output of a switch with two inputs and two outputs.

A *random switching network* takes as its input an arbitrary set of elements and move them along network $\mathcal{N}$ from the first layer to the last layer. If an element λ is located in a node x at layer ℓ in $\mathcal{N}$ that is connected directly to a node y at layer $\ell + 1$, then λ moves from x to y. If x is connected to a switch between layers ℓ and $\ell + 1$, then a coin is tossed at random for that switch, and depending on the outcome, λ moves to one of the outputs of the switch (and the element from the other input moves to the other output).

1.1 New Contributions

A *k-partial n-permutation* is any sequence $\langle x_0, \dots, x_{n-1} \rangle$ consisting of k 0s and $n - k$ distinct elements from $\{1, \dots, n-k\}$. The set of all k-partial n-permutations is denoted by $\mathbb{S}_{n,k}$; $\mathbb{S}_{n,k} = \{(x_0, \dots, x_{n-1}) : |\{j \in \{0, 1, \dots, n-1\} : x_j = 0\}| = k$ and $\forall_{r \in \{1,\dots,n-k\}} \exists_{j \in \{0,1,\dots,n-1\}}\ x_j = r\}$. Observe that $|\mathbb{S}_{n,k}| = \frac{n!}{k!}$.

We consider the problem of generating a uniformly random element from $\mathbb{S}_{n,k}$: a random k-partial n-permutation. Our main technical result, Theorem 4, is a formal proof that $\mathcal{O}(\log^2 n)$ Thorp shuffles will suffice to obtain an almost random k-partial n-permutation, assuming $k = \Omega(n)$.[2] Observe that this implies that $\mathcal{O}(\log^2 n)$ Thorp shuffles will generate an $(n-k)$-wise (almost) independent permutation. While this does not settle the conjecture of Aldous, it shows that $\mathcal{O}(\log^2 n)$ Thorp shuffles randomly permute almost all (but a constant fraction of) input elements. Furthermore, this can be easily used to randomly permute any number of elements using Thorp shuffle: if the input deck has N cards, then add another set of $0.01N$ "empty" cards and run $\mathcal{O}(\log^2 N)$ Thorp shuffles. Then, the obtained deck will have the original N cards randomly permuted, with the empty cards at some arbitrary positions. Therefore, if we remove the empty cards, then we obtain a random permutation of the original N elements.

A direct implication of this result is that $\mathcal{O}(\log^2 n)$ Thorp shuffles will randomly permute any sequence of 0s and 1s (Theorem 3); see [16] for earlier, weaker results.

Our analysis of the Thorp shuffle uses a novel method of analyzing convergence times of Markov chains, which we believe is the most important contribution of this paper. The proof of our main lemma uses coupling. Coupling has been used in the analysis of convergence rates of Markov chains frequently before, but in this paper we apply coupling in an interesting, non-Markovian way. While non-Markovian coupling is known to be more powerful than the Markovian coupling (cf. [9]), our treatment is one of only a few examples where strictly non-Markovian coupling can be used to formally prove a strong mixing time (see [3,6,11] for several other examples). Our non-Markovian coupling is used to reduce the problem to the analysis of some random process in networks, which we solve using combinatorial and probabilistic arguments.

[2] A similar result has been *announced* in [15], however in the journal version [16] of that paper, only the bound of $\mathcal{O}(\log^6 n)$ (which is $\mathcal{O}(d^5)$ in the notation from [16]) is proven (see the discussion in [16, Section 3] and after Theorem 1 in [16]).

It is easy to extend our result about Thorp shuffle to construct a random switching network of depth $\mathcal{O}(\log^3 n)$ that randomly permutes any set of n elements. We first apply $\mathcal{O}(\log^2 n)$ Thorp shuffles to partition the n element into two sets of size $\lceil \frac{n}{2} \rceil$ and $\lfloor \frac{n}{2} \rfloor$, respectively, at random (almost random), and then we recursively apply Thorp shuffles to each of the two sets. To ensure that the error is small in all recursive calls, for any set of elements we will perform $\mathcal{O}(\log^2 n)$ Thorp shuffles to make the partition (and so, the number of Thorp shuffles is independent of the actual size of the instance of the recursive call). This leads to a simple construction of a random switching network of depth $\mathcal{O}(\log^3 n)$ randomly permuting n elements.

Theorem 1. *There is an explicitly given random switching network of depth $\mathcal{O}(\log^3 n)$ that randomly permute any set of n elements.* □

We also consider a related random process (which we call a *Perfect shuffle*) which can be seen as a process "perfectly" simulating $\mathcal{O}(\log n)$ steps of Thorp shuffle: first randomly permute the first $n/2$ elements, then randomly permute the other $n/2$ elements, and then randomly swap elements with indices i and $\frac{n}{2} + i$ for all i, $1 \le i \le \frac{n}{2}$. Our motivation to study this process is that a single Perfect shuffle mimics $\mathcal{O}(\log n)$ Thorp shuffles, and thus understanding of Perfect shuffle can shed some light on the performance of Thorp shuffle. We apply coupling to show in Theorem 5 that Perfect shuffle mixes in $\mathcal{O}(\log \log n)$ steps, which we conjecture to be asymptotically tight.

We also notice that Thorp shuffle has been used in the past in some cryptographical applications (see [5,19] for a selection). And so, for example, recently Morris et al [19] used their results about randomly permuting partial permutations in Thorp shuffle to analyze and obtain new results for provably secure blockcipher-based schemes. The key technical result of Morris et al [19, Theorem 1] is that if $n - k \ll r(4 \log_2 n(n - k)/n)^{-r}$, then $2r \log_2 n$ Thorp shuffles will randomly permute any k-partial n-permutation; observe that this result is interesting only when $n - k \ll n/ \log_2 n$, and hence it is weaker than the result shown in this paper.

Note. This extended abstract presents only the results and main ideas behind the analysis. Detailed proofs are available in the full version of the paper.

2 Preliminaries

We consider the problem of generating a uniformly random element from $\mathbb{S}_{n,k}$, or equivalently, a random k-partial n-permutation. Our goal is to show that for n being a power of 2, $\mathcal{O}(\log^2 n)$ Thorp shuffles will suffice to obtain a random k-partial n-permutation, assuming $k = \Omega(n)$. Theorem 4 will state the result for arbitrary n.

2.1 Markov Chains, Coupling, and Non-markovian Coupling

To analyze Thorp shuffle for partial permutations, we model the process by a Markov chain over the state $\mathbb{S}_{n,k}$. If the starting state of the shuffle is $\pi_0 \in \mathbb{S}_{n,k}$, then the Markov chain $(\pi_t)_{t \in \mathbb{N}}$ is defined such that $\pi_{t+1} \in \mathbb{S}_{n,k}$ is obtained by applying Thorp shuffle to π_t. It is known that the stationary distribution of such Markov chain is uniform and our goal is to estimate the convergence rate of this Markov chain to the uniform distribution. To analyze the convergence rate we use the *coupling* approach (for a survey on the coupling approach we refer the reader to [12]; cf. also [1,4,8,14]). While typically Markovian couplings are used to analyze mixing times of Markov chains, in our analysis we heavily use *non-Markovian* features of coupling.

Let $\mathfrak{M} = (\mathcal{M}_t)_{t \in \mathbb{N}}$ be a discrete-time Markov chain with a finite state space Ω and a unique stationary distribution $\mu_{\mathfrak{M}}$. For any random variable X, let $\mathcal{L}(X)$ denote the probability distribution of X, and let $\mathcal{L}(\mathcal{M}_t \mid \mathcal{M}_0 = \omega)$ denote the probability distribution of $\mathcal{M}_t$ given that $\mathcal{M}_0 = \omega$. We are interested in studying Markov chains for which the statistical distance between $\mathcal{L}(\mathcal{M}_t \mid \mathcal{M}_0 = \omega)$ and $\mu_{\mathfrak{M}}$ tends quickly to zero, independently of $\omega \in \Omega$. To quantify this, we will use the standard measure of the distance between two distributions: the *total variation distance* between two probability distributions $\mathcal{X}$ and $\mathcal{Y}$ over the same finite domain Ω is defined as $d_{TV}(\mathcal{X}, \mathcal{Y}) = \max_{S \subseteq \Omega} |\mathbf{Pr}_{\mathcal{X}}[S] - \mathbf{Pr}_{\mathcal{Y}}[S]|$. To study behavior of a Markov chain $\mathfrak{M}$ with stationary distribution $\mu_{\mathfrak{M}}$, we define the total variation distance after t steps of $\mathfrak{M}$ with respect to the initial state $\omega \in \Omega$ as $\Delta_{\omega}^{\mathfrak{M}}(t) = d_{TV}(\mathcal{L}(\mathcal{M}_t \mid \mathcal{M}_0 = \omega), \mu_{\mathfrak{M}})$. Then, the standard measure of the convergence of a Markov chain $\mathfrak{M}$ to its stationary distribution $\mu_{\mathfrak{M}}$ is the *mixing time*, denoted by $\tau_{\mathfrak{M}}(\varepsilon)$, which is defined as $\tau_{\mathfrak{M}}(\varepsilon) = \min\{T \in \mathbb{N} : \Delta_{\omega}^{\mathfrak{M}}(t) \leq \varepsilon \text{ for each } \omega \in \Omega \text{ and each } t \geq T\}$. In this paper our main focus is on the case $\varepsilon = o(1)$, and in the case of studying (partial) permutations of n numbers, we aim at having $\varepsilon = n^{-c}$ for some constant $c \geq 1$.

A *coupling* (see, e.g., [1,4,7,8,12]) for a Markov chain $\mathfrak{M} = (\mathcal{M}_t)_{t \in \mathbb{N}}$ on state space Ω is a stochastic process $(\mathcal{X}_t, \mathcal{Y}_t)_{t \in \mathbb{N}}$ on $\Omega \times \Omega$ such that each of $(\mathcal{X}_t)_{t \in \mathbb{N}}$, $(\mathcal{Y}_t)_{t \in \mathbb{N}}$, considered independently, is a faithful copy of $\mathfrak{M}$ (i.e., $\mathcal{L}_{\omega}(\mathcal{M}_t) = \mathcal{L}_{\omega}(\mathcal{X}_t) = \mathcal{L}_{\omega}(\mathcal{Y}_t)$ for each $\omega \in \Omega$). The key result on coupling, the *Coupling Inequality* (see, e.g., [1, Lemma 3.6]), states that the total variation distance between $\mathcal{L}(\mathcal{M}_t \mid \mathcal{M}_0 = \omega)$ and its stationary distribution $\mu_{\mathfrak{M}}$ is bounded above by the probability that $\mathcal{X}_t \neq \mathcal{Y}_t$ for the worst choice of initial states $\mathcal{X}_0$ and $\mathcal{Y}_0$. One difficulty in applying coupling lies in the requirement to analyze process $(\mathcal{X}_t, \mathcal{Y}_t)_{t \in \mathbb{N}}$ on the whole space $\Omega \times \Omega$. The *path coupling* method of Bubley and Dyer [4] allows to consider a coupling only for a subset of $\Omega \times \Omega$. Further refinement is coming from an extension of path coupling method called *delayed path coupling* [3,5,6]. Comparing to standard coupling, delayed path coupling considers coupling $(\mathcal{X}_t, \mathcal{Y}_t)_{t \in \mathbb{N}}$ with $\mathcal{X}_0$ and $\mathcal{Y}_0$ being similar (like in path coupling [4]), and the goal is to design the coupling by observing the Markov chain in several steps to ensure that for some small t the value of $\mathbf{Pr}[\mathcal{X}_t \neq \mathcal{Y}_t]$ is very small (traditionally path coupling considers only $\mathbf{Pr}[\mathcal{X}_t \neq \mathcal{Y}_t]$ conditioned on $\mathcal{X}_{t-1}$ and $\mathcal{Y}_{t-1}$, whereas delayed path coupling considers $\mathbf{Pr}[\mathcal{X}_t \neq \mathcal{Y}_t]$ conditioned on $\mathcal{X}_0$ and $\mathcal{Y}_0$ only, and thus considers the coupling for multiple steps). We will analyze the convergence of Markov chains using the following lemma.

Lemma 2 (Delayed Path Coupling Lemma [3,5,6]). *Let $\mathfrak{M} = (\mathcal{X}_t)_{t \in \mathbb{N}}$ be a discrete-time Markov chain with a finite state space Ω. Let Γ be any subset of $\Omega \times \Omega$. Suppose that there is an integer D such that for every $(\mathcal{X}, \mathcal{Y}) \in \Omega \times \Omega$ there exists a sequence $\mathcal{X} = \Lambda_0, \Lambda_1, \ldots, \Lambda_r = \mathcal{Y}$, where $(\Lambda_i, \Lambda_{i+1}) \in \Gamma$ for $0 \leq i < r$, and $r \leq D$. If there exists a coupling $(\mathcal{X}_t, \mathcal{Y}_t)_{t \in \mathbb{N}}$ for $\mathfrak{M}$ such that for some $T \in \mathbb{N}$, for all $(\mathcal{X}, \mathcal{Y}) \in \Gamma$, it holds that $\mathbf{Pr}[\mathcal{X}_T \neq \mathcal{Y}_T \mid (\mathcal{X}_0, \mathcal{Y}_0) = (\mathcal{X}, \mathcal{Y})] \leq \frac{\varepsilon}{D}$, then $\tau_{\mathfrak{M}}(\varepsilon) \leq T$.*

Let us briefly discuss how the Delayed Path Coupling Lemma can be used. Our Markov chain will have the state space $\mathbb{S}_{n,k}$ of all k-partial n-permutations. We define Γ to be the set of all pairs of k-partial n-permutations $\pi_1, \pi_2 \in \mathbb{S}_{n,k}$ that differ on exactly two elements: for some ℓ and r, $\pi_1(i) = \pi_2(\ell)$ if $i = r$, $\pi_1(i) = \pi_2(r)$ if $i = \ell$, and $\pi_1(i) = \pi_2(i)$ otherwise. Then, for any two k-partial n-permutations $\pi^*, \pi^{**} \in \mathbb{S}_{n,k}$, we can easily find a sequence $\pi^* = \pi_0, \pi_1, \ldots, \pi_r = \pi^{**}$ with $r \le n$, such that each pair π_i, π_{i+1} differs on exactly two elements (that is, $(\pi_i, \pi_{i+1}) \in \Gamma$).

With this, for any π_1 and π_2 that differ on exactly two elements, we define a coupling $(\mathcal{X}_t, \mathcal{Y}_t)_{t \in \mathbb{N}}$ with $\mathcal{X}_0 = \pi_1$ and $\mathcal{Y}_0 = \pi_2$, such that each $\mathcal{X}_{t+1}$ and $\mathcal{Y}_{t+1}$ is obtained from $\mathcal{X}_t$ and $\mathcal{Y}_t$, respectively, by applying a single Thorp shuffle or a butterfly shuffle. Our goal is to ensure that the designed coupling for the random butterfly process will have $\mathbf{Pr}[\mathcal{X}_t \ne \mathcal{Y}_t] \le n^{-2}$ for some $t = \mathcal{O}(\log n)$ ($t = \mathcal{O}(\log^2 n)$ for Thorp shuffle). By Lemma 2, this will ensure that $\tau_{\mathfrak{M}}(1/n) = \mathcal{O}(\log n)$ for the random butterfly process.

3 Analyzing Partial Permutations

In this section we will analyze the *random butterfly process* for generating random k-partial n-permutations. (Let us recall that t steps of the random butterfly process correspond to $t \log_2 n$ Thorp shuffles assuming n is a power of 2. Our analysis can be extended to arbitrary n, see Theorem 4.) Our analysis uses Delayed Path Coupling.

Let $k = \Omega(n)$. Let π_1 and π_2 be two arbitrary k-partial n-permutations from $\mathbb{S}_{n,k}$ which differ on exactly two elements. Let α and β be the two elements which have distinct positions in π_1 and π_2. Let $\varpi = \lfloor \log_2(n/k) \rfloor$ and hence $\frac{n}{2^\varpi} \le k < \frac{n}{2^{\varpi-1}}$. Let $\mathbb{Z}$ (like *zeroes*) be the set of the 0 elements except possibly for the elements α and β (if either α or β is a 0). Observe that $|\mathbb{Z}| \ge \frac{n}{2^\varpi} - 1$. Let $\mathbb{Z}^* = \mathbb{Z} \cup \{\alpha, \beta\}$.

Our goal is to define a coupling $(\mathcal{X}_t, \mathcal{Y}_t)_{t \in \mathbb{N}}$ that satisfies the following conditions:

Initial state: $(\mathcal{X}_0, \mathcal{Y}_0) = (\pi_1, \pi_2)$;
Coupling: each sequence $(\mathcal{X}_t)_{t \in \mathbb{N}}$ and $(\mathcal{Y}_t)_{t \in \mathbb{N}}$ in isolation is a faithful copy of the random butterfly process;
Convergence: for certain $T = \mathcal{O}(\log n)$, with high probability: $\mathcal{X}_t = \mathcal{Y}_t$ for all $t \ge T$.

By Lemma 2, these three conditions would imply that the mixing time of the random butterfly process for generating random k-partial n-permutations is $\mathcal{O}(\log n)$.

We will define the coupling by first allowing the process $(\mathcal{X}_t)_{t \in \mathbb{N}}$ to be run arbitrarily and then we will set the sequence $(\mathcal{Y}_t)_{t \in \mathbb{N}}$ in a non-Markovian way to ensure the second and the third properties above. By non-Markovian we mean that the sequence $(\mathcal{Y}_t)_{t \in \mathbb{N}}$ will be defined only once the entire sequence $(\mathcal{X}_t)_{t \le T}$ is known, and each $\mathcal{Y}_t$ with $t \le T$ depends on the entire $(\mathcal{X}_t)_{t \le T}$. (Markovian coupling, which has been more commonly used in the past, would mean that $\mathcal{Y}_{t+1}$ depends only on $\mathcal{Y}_t$, $\mathcal{X}_t$, and $\mathcal{X}_{t+1}$.)

Our analysis splits the process into two phases, each phase corresponding to $\mathcal{O}(\log n)$ runs of the butterfly shuffles, and the final coupling will be done after seeing all random choices for $(\mathcal{X}_t)_{t \in \mathbb{N}}$ in these two phases.

Notation. When we consider the process as one of moving the input elements from left to right, that is, by applying one butterfly shuffle another another, then we use term *element* to denote the current status of a given input element, and *position* to denote its current position in the switching network. If we run the random butterfly process, then each time we have two elements that are connected by a switch (to be swapped at random), then we say that these two elements are a *match* at a given moment. There are exactly $\frac{1}{2} n \log_2 n$ matches in a single butterfly shuffle, $\frac{n}{2}$ matches in each of the $\log_2 n$ layers of the butterfly.

3.1 Overview of the Analysis for k-Partial n-Permutations

We consider the random butterfly process starting at the state $\mathcal{X}_0 = \pi_1$. We will look at the sequence $(\mathcal{X}_t)_{t \in \mathbb{N}}$ and we will analyze its properties to define the sequence $(\mathcal{Y}_t)_{t \in \mathbb{N}}$. We will follow the elements from $\mathbb{Z}^*$ and those from outside $\mathbb{Z}^*$, and every time we have a match not involving two elements from $\mathbb{Z}^*$ in $(\mathcal{X}_t)_{t \in \mathbb{N}}$, we will make at once the same choices (outcomes of the matches) for both $(\mathcal{X}_t)_{t \in \mathbb{N}}$ and $(\mathcal{Y}_t)_{t \in \mathbb{N}}$. Furthermore, for a large part of the outcomes of the matches between two elements from $\mathbb{Z}^*$ in $(\mathcal{X}_t)_{t \in \mathbb{N}}$ we will also set it identically in both copies, in $(\mathcal{X}_t)_{t \in \mathbb{N}}$ and $(\mathcal{Y}_t)_{t \in \mathbb{N}}$. However, our main focus is on a small number of appropriately selected matches between pairs of elements from $\mathbb{Z}^*$ in $(\mathcal{X}_t)_{t \in \mathbb{N}}$. Our analysis is in two stages.

First Stage. In the first stage, we will construct two disjoint full binary trees, that we call *fundamental trees*, that are obtained using the following branching process:

Initially create two trees whose roots are the two elements α and β (the elements where $\mathcal{X}_0$ and $\mathcal{Y}_0$ differ).

Suppose that we have built two trees for $\mathcal{X}_0, \mathcal{X}_1, \dots, \mathcal{X}_t$. Let $\varrho = \frac{n}{2^{\varpi+2}}$. Let v be any element corresponding to a leaf $\hat{v}$ in one of the trees at depth strictly smaller than $\log_2 \varrho$. If v is to be matched *after* one step of the random butterfly process to an element u from $\mathbb{Z}^*$ and u was not used in the construction of any of the trees so far, then we *branch at* $\hat{v}$. In that case, we branch at $\hat{v}$ and $\hat{v}$ has two new children: one corresponding to the element v and one corresponding to the element u. We perform this operation for all such leaves $\hat{v}$ at the same time, to build two trees for $\mathcal{X}_0, \mathcal{X}_1, \dots, \mathcal{X}_{t+1}$.

We continue this process for increasing t to build two trees until all leaves of both trees are at the same level and each tree has exactly $\varrho = \frac{n}{2^{\varpi+2}}$ leaves (which is why we branched only leaves at depth smaller than $\log_2 \varrho$); such trees will be called the *fundamental trees*. We can prove the following key lemma.

Lemma 3. *There is a constant c such that if we run the random butterfly process for $c \log_2 n$ steps with all switches set at random then the probability that two fundamental trees will be built is at least $1 - n^{-3}$.*

Let us discuss the intuitions behind this phase and state central properties of our construction. We observe that each fundamental tree has only one element from outside $\mathbb{Z}$: either α or β. Therefore, since all but one elements from each fundamental tree are in $\mathbb{Z}$, the process of setting the outcomes of the matches in each of the fundamental trees will correspond to the random selection of the position for α (or β) in the tree; since all elements in $\mathbb{Z}$ are identical (are zeros), we can arbitrary permute them without affecting the outcome of the process. Next, we observe that the trees do not dependent on the random outcomes of the matches during the branching. In other words, if we have two instances of the random butterfly process such that the second instance differs from the first one only in the (outcome of the) switches defining the branching for the fundamental trees in the first instance, then the second instance have the same fundamental trees (we may only have permuted elements in each of the fundamental trees). Finally, (and this is the central property of our construction) conditioned on the final positions of the leaves in the fundamental trees, the choice of the final positions of α and β is uniformly random. That is, if the tree containing α has the leaves at positions $p_1, \dots, p_\varrho$, $p_i < p_{i+1}$, then if we randomly decide the outcomes of the matches during the branching, then for every i, the probability that α will end up at position p_i is $\frac{1}{\varrho}$. The same property holds for β.

Second Stage. In the second stage, we fix the leaves of the two fundamental trees built in the first stage. Our goal is to show that if we run $\mathcal{O}(\log n)$ steps of the random butterfly process defining $(\mathcal{X}_t)_{t\in\mathbb{N}}$ then we can find a set $\mathbb{M}$ of ϱ matches which forms a perfect matching between the leaves of the two fundamental trees. That is, $\mathbb{M}$ is a set of ϱ matches in the random butterfly process, such that for every match $(v, u) \in \mathbb{M}$, v is a leaf from the first tree, u is a leaf from the second tree, and there is no other pair in $\mathbb{M}$ which contains either v or u. Once we have such a matching, then the *lexicographically first* perfect matching between the leaves of the two fundamental trees will be called the *fundamental matching* $\mathbb{M}^*$. (That is, if we number the shuffles in the way how they are generated by the Thorp shuffle, then for every other matching $\mathbb{M}$ between the leaves of the two fundamental trees there is an index ℓ, such that after having the same matches in $\mathbb{M}^*$ and $\mathbb{M}$ in the first $\ell - 1$ shuffles, the ℓ^{th} shuffle has a match in $\mathbb{M}^*$ and no match in $\mathbb{M}$.) We can prove the following key lemma about fundamental matchings.

Lemma 4. *Let us fix any two sets of ϱ disjoint positions for the leaves of the fundamental trees. There is a constant c such that if we run the random butterfly process for $c\log_2 n$ steps with all switches set at random then the probability that there is a fundamental matching is at least* $1 - n^{-3}$.

Coupling. Now, we are ready to define the coupling. We run the first stage with $\mathcal{O}(\log n)$ random butterflies and construct two fundamental trees T_1 and T_2 with ϱ leafs each. Then, we run the second stage with $\mathcal{O}(\log n)$ random butterflies and construct the fundamental matching $\mathbb{M}$ between the leaves of T_1 and T_2.[3]

Now, for both $(\mathcal{X}_t)_{t\in\mathbb{N}}$ and $(\mathcal{Y}_t)_{t\in\mathbb{N}}$, we will use *identical* outcomes for all the random choices in the switches that are not involved in the (branching) matches inside the trees T_1 and T_2, and are not among the matches in $\mathbb{M}$. Let P_1 be the set of positions that the leaves of T_1 reach at the end of the first stage and P_2 be the set of positions that the leaves of T_2 reach at the end of the first stage. For simplicity of notation, we assume that every match in $\mathbb{M}$ is of the form (p, q), where p is an element that reaches a position in P_1 at the end of the first stage and q is an element that reaches a position in P_2 at the end of the first stage; in this case, we use the notation $\mathbb{M}(p) = q$ and $\mathbb{M}(q) = p$.

[3] This process is conditioned on the fact that the two fundamental trees T_1 and T_2 and that the fundamental matching $\mathbb{M}$ have been found. If we failed to construct T_1, T_2, and $\mathbb{M}$, which by Lemmas 3 and 4 is very unlikely, then the coupling for $(\mathcal{X}_t)_{t\in\mathbb{N}}$ and $(\mathcal{Y}_t)_{t\in\mathbb{N}}$ will be the identity coupling, i.e., all switches will be set in the same way for both $(\mathcal{X}_t)_{t\in\mathbb{N}}$ and $(\mathcal{Y}_t)_{t\in\mathbb{N}}$.

We observe that since T_1 and T_2 are complete binary trees, if all the choices inside the trees T_1 and T_2 have been done at random, then α reaches the position at the end of the first stage that is uniform in P_1, β reaches the position at the end of the first stage that is uniform in P_2, and these positions are independent. With this, we define the coupling for $(\mathcal{X}_t)_{t\in\mathbb{N}}$ and $(\mathcal{Y}_t)_{t\in\mathbb{N}}$ in the first stage such that if α reaches position p at the end of the first stage and β reaches position q at the end of the first stage for the sequence $(\mathcal{X}_t)_{t\in\mathbb{N}}$, then we set the random outcomes for the sequence $(\mathcal{Y}_t)_{t\in\mathbb{N}}$ in the first stage such that α (which in $(\mathcal{Y}_t)_{t\in\mathbb{N}}$ is traversing through the tree T_2) reaches position $\mathbb{M}(p)$ at the end of the first stage, and β (which in $(\mathcal{Y}_t)_{t\in\mathbb{N}}$ is traversing through T_1) reaches position $\mathbb{M}(q)$ at the end of the first stage for the sequence $(\mathcal{Y}_t)_{t\in\mathbb{N}}$.

Next we define the coupling for the second stage. We first perform random choices for the outcomes of the matches in $\mathbb{M}$ for $(\mathcal{X}_t)_{t\in\mathbb{N}}$, and then *reverse* choices for the matches in $\mathbb{M}$ for $(\mathcal{Y}_t)_{t\in\mathbb{N}}$.

Lemma 5. *The process defining $(\mathcal{X}_t)_{t\in\mathbb{N}}$ and $(\mathcal{Y}_t)_{t\in\mathbb{N}}$ is a proper coupling.*

Next, we want to consider when $\mathcal{X}_t \neq \mathcal{Y}_t$. We first observe that without revealing the outcome of the matches in T_1, T_2, and $\mathbb{M}$, the final positions of the elements outside P_1 and P_2 are fixed. Furthermore, since in P_1 and P_2, all elements other than α and β are from $\mathbb{Z}$ (and hence all are 0s and thus indistinguishable), we only have to consider the final positions of α and β. Consider first the chain $(\mathcal{X}_t)_{t\in\mathbb{N}}$ and suppose that α finished the first stage at position p and β finished the first stage at position q. In $(\mathcal{Y}_t)_{t\in\mathbb{N}}$, the coupling ensures that in the first stage α finishes at position q and β at position p. Then, the only change in the performance of $(\mathcal{X}_t)_{t\in\mathbb{N}}$ and $(\mathcal{Y}_t)_{t\in\mathbb{N}}$ is at the two matches $(p, \mathbb{M}(p))$ and $(q, \mathbb{M}(q))$. The key property of our coupling is that since the outcomes of the matches in $\mathbb{M}$ for $(\mathcal{X}_t)_{t\in\mathbb{N}}$ are reverse with respect to the choices for the matches in $\mathbb{M}$ for $(\mathcal{Y}_t)_{t\in\mathbb{N}}$, we will have that $\mathcal{X}_T = \mathcal{Y}_T$ at the end of the second stage.

We can now summarize properties of our coupling for the random butterfly process:

- If T_1, T_2, and $\mathbb{M}$ have been successfully constructed then our coupling $(\mathcal{X}_t, \mathcal{Y}_t)_{t\in\mathbb{N}}$ ensures that $\mathcal{X}_T = \mathcal{Y}_T$ for all $T \geq T_0$, where $T_0 = \mathcal{O}(\log n)$, and
- T_1, T_2, and $\mathbb{M}$ have been successfully constructed with probability at least $1-2n^{-3}$.

We can combine our analysis with the Delayed Path Coupling Lemma to conclude:

Theorem 2. *Let n be a power of two and let $k \geq cn$ for some positive constant c. The mixing time of the Thorp shuffle for k-partial n-permutations is $\mathcal{O}(\log^2 n)$.*

It is not difficult to see that this result yields also the result for permuting 0s and 1s.

Theorem 3. *Mixing time of the Thorp shuffle for any sequence of n 0s and 1s is $\mathcal{O}(\log^2 n)$.*

While our analysis focused solely on the case when n is a power of two, our proof of Theorem 2 can be easily extended (with essentially no change in the analysis) to hold for all values n (including odd n).

Theorem 4. *Let n be any positive number and let $k \geq cn$ for some positive constant c. The mixing time of the Thorp shuffle for k-partial n-permutations is $\mathcal{O}(\log^2 n)$.*

4 Perfect Shuffle: A "better" Thorp Shuffle

In our pursuit to understand the behavior of Thorp shuffle, we also consider a related shuffling process that captures some features of Thorp shuffle. We assume n is even.

Perfect shuffle:

- Cut the deck of n card into two halves.
- Permute each half independently and uniformly at random.
- Drop the first card from the left half or the right half according to the outcome of a fair coin flip; then drop from the other half. Continue this way until both halves are empty.

Observe that Perfect shuffle mimics the behavior of Thorp shuffle with one addition feature: before the halves are shuffled, they are first randomly permuted. Therefore one can see Perfect shuffle as a process that *simulates* $\log n$ steps of Thorp shuffle, or perhaps rather *improves* $\log n$ steps of Thorp shuffle. Indeed, if n is a power of 2 then Perfect shuffle can be seen as the process of replacing the first $\log n - 1$ Thorp shuffles by a single step of randomly permuting two halves of the elements, and then applying one Thorp shuffle. Intuitively, this change should only help and speed up the shuffling process. Therefore, again, intuitively, a lower bound for the number of steps of Perfect shuffle needed to randomly shuffle a deck of $n = 2^d$ cards should lead to a lower bound for the number of steps of Thorp shuffle needed to randomly shuffle a deck of n cards, where the latter bound should be $\log n$ times the number of steps needed for Perfect shuffle. Using the coupling approach we can prove the following theorem:

Theorem 5. *The mixing time of Perfect shuffle is* $\mathcal{O}(\log \log n)$.

We conjecture that this bound is asymptotically optimal.

The Perfect shuffle process is similar to the square lattice shuffle studied by Håstad [10], who considers the operations of *randomly permuting columns and rows of a square matrix*. Håstad [10] proved that it is sufficient to repeat the square lattice shuffle a constant number of times to generate a random permutation. Our process can be seen as the process of randomly permuting columns and rows of a $2 \times N$ matrix.

5 Final Comments

Thorp Shuffling for Arbitrary Permutations. It is tempting to argue that Theorem 4 implies that the mixing time for Thorp shuffle for all permutations is $\mathcal{O}(\log^2 n)$. Unfortunately, we do not know of any straightforward reduction from randomly permuting k-partial n-permutations for $k \geq 0.01n$ to randomly permuting permutations. The following simple example shows that this task is not straightforward. Suppose that one has a process P that takes any permutation $\pi \in \mathbb{S}_n$ and outputs a randomly chosen permutation $\pi^* \in \mathbb{S}_n$ among all permutations in $\mathbb{S}_n$ that has the same parity as π. (The parity of permutation $\pi \in \mathbb{S}_n$ is either even or odd, depending on whether the minimum number of adjacent transpositions needed to obtain the identity permutation from π is even or odd, respectively.) It is easy to see that P randomly permutes 2-partial n-permutations, and hence, also all k-partial n-permutations for $k \geq 0.01n$. However, even if we repeat P an arbitrary number of times, this process will never generate a random permutation, because this process is invariant to the parity of the original permutation π.

Acknowledgements. We thank Martin Dyer, Uri Feige, and Peter Winkler for helpful discussions and comments.

References

1. Aldous, D.: Random walks of finite groups and rapidly mixing Markov chains. In: Séminaire de Probabilités XVII, 1981/82, pp. 243–297. Springer, Berlin (1983)
2. Aldous, D., Diaconis, P.: Strong uniform times and finite random walks. Advances in Applied Mathematics 8, 69–97 (1987)
3. Berenbrink, P., Czumaj, A., Steger, A., Vöcking, B.: Balanced allocations: The heavily loaded case. SIAM Journal on Computing 35(6), 1350–1385 (2006)
4. Bubley, R., Dyer, M.: Path coupling: A technique for proving rapid mixing in Markov chains. In: 38th IEEE FOCS, pp. 223–231 (1997)
5. Czumaj, A., Kanarek, P., Kutyłowski, M., Loryś, K.: Delayed path coupling and generating random permutations via distributed stochastic processes. In: SODA, pp. 271–280 (1999)
6. Czumaj, A., Kutyłowski, M.: Delayed path coupling and generating random permutations. Random Structures and Algorithms 17(3-4), 238–259 (2000)
7. Diaconis, P.: Group Representations in Probability and Statistics. Lecture Notes - Monograph Series, vol. 11. Institute of Mathematical Statistics, Hayward (1988)
8. Dyer, M., Greenhill, C.: Random walks on combinatorial objects. In: Surveys in Combinatorics, pp. 101–136. Cambridge University Press, UK (1999)
9. Griffeath, D.: A maximal coupling for Markov chains. Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete 31, 95–106 (1975)
10. Håstad, J.: The square lattice shuffle. Random Struct. and Algorithms 29, 466–474 (2006)
11. Hayes, T.P., Vigoda, E.: A non-Markovian coupling for randomly sampling colorings. In: 44th IEEE FOCS, pp. 618–627 (2003)
12. Jerrum, M.: Mathematical foundations of the Markov chain Monte Carlo method. In: Probabilistic Methods for Algorithmic Discrete Mathematics, pp. 116–165. Springer (1998)
13. Knuth, D.E.: The Art of Computer Programming, Volume 3: Sorting and Searching. Section 5.3.4: Networks for Sorting, 3rd edn., pp. 219–247. Addison-Wesley (1997)
14. Montenegro, R., Tetali, P.: Mathematical aspects of mixing times in Markov chains. Foundations and Trends in Theoretical Computer Science 1(3), 237–354 (2006)
15. Morris, B.: The mixing time of the Thorp shuffle. In: 37th ACM STOC, pp. 403–412 (2005)
16. Morris, B.: The mixing time of the Thorp shuffle. SIAM Journal on Computing 38(2), 484–504 (2008); This is the full, corrected, and updated version of [15]
17. Morris, B.: Improved mixing time bounds for the Thorp shuffle and L-reversal chain. Annals of Probabability 37(2), 453–477 (2009)
18. Morris, B.: Improved mixing time bounds for the Thorp shuffle. arXiv 0912.2759 (2009)
19. Morris, B., Rogaway, P., Stegers, T.: How to Encipher Messages on a Small Domain. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 286–302. Springer, Heidelberg (2009)
20. Rackoff, C., Simon, D.R.: Cryptographic defense against traffic analysis. In: 25th ACM STOC, pp. 672–681 (1993)
21. Thorp, E.O.: Nonrandom shuffling with applications to the game of Faro. Journal of the American Statistical Association 68, 842–847 (1973)

Dynamic Complexity of Directed Reachability and Other Problems

Samir Datta[1], William Hesse[2], and Raghav Kulkarni[3]

[1] Chennai Mathematical Institute, India
sdatta@cmi.ac.in
[2] University of Massachusetts & Google Inc
whessedk@gmail.com
[3] Center for Quantum Technologies, Singapore
kulraghav@gmail.com

Abstract. We report progress on dynamic complexity of well-known graph problems such as reachability and matching. In this model, edges are dynamically added and deleted and we measure the complexity of each update/query. We propose polynomial-size data structures for such updates for several related problems. The updates are in very low level complexity classes such as quantifier-free first order formulas, $\mathsf{AC}^0[2]$, TC^0. In particular, we show the following problems are in the indicated classes:
(a) maximum matching in non-uniform DynTC^0;
(b) digraph reachability in non-uniform $\mathsf{DynAC}^0[2]$;
(c) embedded planar digraph reachability in DynFO(= uniform DynAC^0).

Notably, the part (c) of our results yields the first non-trivial class of graphs where reachability can be maintained by first-order updates; it is a long-standing open question whether the same holds for general graphs. For (a) we show that the technique in [7] can in fact be generalized using [9] and [8] to maintain the determinant of a matrix in DynTC^0. For (b) we extend this technique with the help of two more ingredients namely isolation [1,13] and truncated approximation using rational polynomials. In fact, our proof yields $\mathsf{DynAC}^0[p]$ bound for any prime $p > 1$. For (c) we exploit the duality between cuts and cycles in planar graphs to maintain the number of crossings between carefully chosen primal and dual paths, using several new structural lemmas.

1 Introduction

Conventional complexity assumes that the entire input is available initially and it does not change with time. An alternative to this static view is the dynamic model, where the input evolves with time, such as where edges are added or deleted to a graph. We can measure the efficiency of an algorithm in such a model in terms of the (static) complexity of its updates. The particular complexity classes we focus upon in this paper are some of the simplest ones possible: classes of Boolean functions computable by bounded depth circuits. Yet polynomial-size

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 356–367, 2014.

data structures with updates in these classes are powerful enough to solve fairly complex problems, as we see in this work. We start with a brief description of this model.

1.1 The Model of Dynamic Complexity

In the *dynamic* (graph) model we start with an empty graph on a fixed-size set of vertices. The graph evolves by the addition/deletion of a single edge in every time step and an algorithm maintains the graph and polynomially many bits of auxiliary data, which allow a property such as reachablity to be queried efficiently. The dynamic complexity of the algorithm is the static complexity of each update step. If a polynomial-size data structure can be updated in a static class $\mathcal{C}$, the dynamic problem is said to belong to $\mathsf{Dyn}\mathcal{C}$. In this paper, $\mathcal{C}$ is often a complexity class defined in terms of bounded depth circuits[1] such as AC^0, $\mathsf{AC}^0[2]$, TC^0, where $\mathsf{AC}^0$is the class of polynomial size constant depth circuits with AND and OR gates of unbounded fan-in; $\mathsf{AC}^0[2]$circuits may additionally have PARITY (sum modulo 2) gates; TC^0circuits may additionally have MAJORITY gates. We encourage the reader to refer to any textbook (e.g. Vollmer [17]) for precise definitions of the standard circuit complexity classes. The archetypal example of a dynamic problem is maintaining reachability ("is there a directed path from s to t"), in a digraph. This problem is known to be maintainable in the class DynTC^0- a class where edge insertions and deletions can be maintained using TC^0circuits.

1.2 Historical Background

Dynamic complexity was introduced by Immerman-Patnaik [14] who defined the class DynFO(where the static update class is $\mathsf{Dlogtime}$-uniform AC^0) and proved that problems such as undirected connectivity and minimum spanning tree are in DynFO. DAG reachability was known to be in this class even before it was formally defined [4]. Hesse proved that directed reachability is in the slightly larger (uniform) class DynTC^0. Recently it has been shown that distance computation in undirected graphs can be maintained in DynFO[5] (see also [10] where 3-connected planar isomorphism is also shown to be maintainable in DynFO^+, a closely related class). The big open question in the area is whether it is possible to maintain directed reachability in DynFO. For surveys on dynamic complexity please see [6,16].

1.3 Our Results and Techniques

We summarize our results below:

Theorem 1. *The determinant of a matrix can be maintained in* DynTC^0.

[1] We will have occasion to refer to both the non-uniform and ($\mathsf{dlogtime}$-)uniform versions of these circuit classes and we adopt the convention that, whenever unspecified, we mean the uniform version.

Applying the reduction from rank to determinant by Mulmuley [12] and further applying the Isolation Lemma [13,1], we obtain:

Corollary 1. *(a) The rank of a matrix can be maintained in* DynTC^0*.*
(b) Maximum Matching in a graph is in non-uniform DynTC^0*.*

The complete proof will be presented in the full version of the paper. Next we show the following on reachability.

Theorem 2. *Reachability in directed graphs is in non-uniform* $\mathsf{DynAC}^0[2]$*.*

A generalization to the weighted case with possibly negative weights yields:

Corollary 2. *(a) Maintaining walks of weight at most W is in* $\mathsf{DynAC}^0[2]$*; (b) Shortest Path in a graph without negative cycle or detecting if the negative cycle exists can be maintained in* $\mathsf{DynAC}^0[2]$*; (c) Perfect Matching in grid graphs is in uniform* $\mathsf{DynAC}^0[2]$*.*

Again, we postpone the proof to the full version of the paper.

Theorem 3. *Reachability in directed embedded planar graphs is in* DynFO*.*

The main building block for Theorem 1 and Theorem 2 is a technique introduced by Hesse [7] for maintaining the reachability in directed graphs. This technique relies on a result of Hesse, Allender, and Barrington [8] showing that multiplying univariate polynomials is in TC^0. We show that this reahcability technique can in fact be generalized using the result of Mahajan and Vinay [9] on clow-sequences to maintain the determinant of a matrix in DynTC^0. This suffices for Theorem 1. For Theorem 2 we extend this technique further with the help of two more ingredients namely the Isolation Lemma (more specifically its usage in Allender, Reinhardt and Zhou [1]) and a simple but subtle use of rational polynomials in truncated form to achieve $\mathsf{DynAC}^0[2]$bound. In fact, our proof yields $\mathsf{DynAC}^0[p]$bound for the same for any prime $p > 1$, putting it in the intersection of these $\mathsf{AC}^0[p]$classes. We note that no total function outside $\mathsf{AC}^0$is known to lie in the intersection of these classes. Thus our result gets tantalizingly close to the non-uniform AC^0bound. The uniform AC^0bound would resolve the long-standing conjecture that directed reachability is in DynFO.

For Theorem 3 we exploit the duality between cuts and cycles in planar graphs to maintain information about all oriented cycles in the dual graph, giving cut sets in the original graph. There are interesting technical complexities to overcome, requiring some extra bookkeeping in our data structures. There seem to be no fundamental problems with extending the algorithm from embedded planar graphs to planar graphs. This generates hope for placing general reachability in DynFO.

1.4 Organization

The organization of this paper is as follows: Section 2 contains the preliminaries. Section 3 contains the proof of Theorem 1. Section 4 contains the proof of Theorem 2. Section 5 contains the proof of Theorem 3.

2 Preliminaries

2.1 Non-uniform Version of Isolation

Mulmuley, Vazirani, and Vazirani [13] introduced this simple but powerful lemma:

Lemma 1 (Isolation Lemma). *Given a non-empty $\mathcal{F} \subseteq 2^{[m]}$, if one assigns for each $i \in [m]$, $w_i \in [2m]$ uniformly at random then with probability at least half, the minimum weight subset of $\mathcal{F}$ is unique; where the weight of a subset S is $\sum_{i \in S} w_i$.*

A surprising part of the above lemma is that no structure is assumed on $\mathcal{F}$. Allender, Reinhardt, and Zhou observe that the above lemma can be derandomized with the help of non-uniformity.

Lemma 2 (Non-uniform Version of Isolation Lemma). *There exist m^2 weight assignments $W^{(i)} = (w_1^{(i)}, \ldots, w_m^{(i)})$ (for $i \in [m^2]$) such that: for any non-empty $\mathcal{F} \subseteq 2^{[m]}$ there exists an $i \in [m^2]$ such that the minimum weight subset of $\mathcal{F}$ with respect to $W^{(i)}$ is unique.*

2.2 Clow-sequences and Determinant

Mahajan-Vinay [9] introduced the concept of a *clow-sequence* for an adjacency matrix which extends the notion of a cycle-cover.

Definition 1. *A* clow *is a closed walk starting and ending at the same vertex. The smallest vertex in a clow is called in its* head. *A* clow sequence, *$\mathcal{W}$, is a sequence of clows with some constraints: (a) the heads of the clows in the sequence are in strictly increasing order, and, (b) the number of edges (with multiplicity) in a clow-sequence is exactly n, the size of the matrix. Define $sgn(\mathcal{W})$ to be $(-1)^{n\ +\ \#clows\ in\ \mathcal{W}}$ and weight $w(\mathcal{W})$ to be the product of its edge weights.*

Armed with the above definition we approach the main theorem of [9]:

Theorem 4 (Mahajan-Vinay[9]).

$$det(M) = \sum_{\mathcal{W}:\ clow\ sequence\ in\ M} sgn(\mathcal{W}) w(\mathcal{W})$$

This nice result is based on the observation that all clow-sequences which are not cycle-covers cancel out in pairs in the signed sum above, leaving behind the signed sum of cycle covers which is the determinant.

2.3 Polynomial Arithmetic and Hesse's Approach

Hesse [7] showed that:

Theorem 5 (Hesse[7]). *Reachability in directed graphs can be maintained in* DynTC^0.

The main idea is to maintain the counts of all s, t-walks parameterized on their lengths for each pair of vertices s, t. These are maintained as integer polynomials and [7] showed that updating them only requires addition and multiplication of polynomials, and finding powers of polynomials. For this the following theorem from Hesse, Allender and Barrington [8] is crucial:

Theorem 6 (Hesse-Allender-Barrington[8]). *Iterated product of integer polynomials is in uniform* TC^0.

We also note that the above theorem easily extends to polynomials in two (more generally any constant number of) variables. Arithmetic on polynomials over the finite field $\mathbb{F}_p$ can be done with even simpler circuits using modulo p gates:

Fact 1 (Folklore). *Iterated polynomial addition and multiplication of two polynomials over* $\mathbb{F}_p$ *is in* $\mathsf{AC}^0[p]$ *for prime* $p > 1$.

2.4 Planarity, Uniform Isolation, Flows and Cycles

The non-uniform edge weights of Lemma 2 can be replaced by a uniformly computed set of weights on certain families of planar graphs. In particular Bourke, Tiwari, and Vinodchandran show that for subgraphs of a planar grid, one can find weights so that minimum weight paths and perfect matchings are unique. This can be extended to perfect matchings in bipartite grid graphs [3].

Lemma 3 (Deterministic Isolation in Grid Graphs, [2,3]). *One can assign* $O(\log n)$*-bit long weights in uniform* AC^0 *to the edges of* $n \times n$ *grid such that in any subgraph of the grid (a) the minimum weight* s-t *path is unique; (b) minimum weight perfect matching is unique.*

We use this fact to obtain uniform circuits for dynamic updates in grid graphs.

Further we use planarity in the context of flows. Miller and Naor [11] relate the existence of flow in a planar graph with multiple sources and multiple sinks to the non-existence of negative weight cycle in the dual graph with appropriate edge weights. If the dual graph does not have a negative cycle then the distance between two nodes of the dual graph is well defined. Using this distance computation as a subroutine, Miller and Naor give an algorithm to compute the flow in the primal graph. As a special case of the flow problem they solve the perfect matching question in bipartite planar graphs. In this paper we use their reduction in the dynamic setting.

2.5 Maintaining the Determinant

In this paper, the “determinant-maintenance-problem” corresponds to queries for the value of the determinant of a dynamic $n \times n$ matrix. Any one entry of the matrix can be changed from zero to a poly(n)-bit integer, or vice-versa, in one update.

3 Proof of Theorem 1: Determinant in DynTC^0

We combine ideas from the dynamic reachability algorithm of Hesse [7], parallel computation of the determinant (Mahajan-Vinay [9]) and the TC^0 algorithm for computing the iterated product of polynomials by Hesse-Allender-Barrington [8], to yield the claimed result.

Reinterpreting the Mahajan-Vinay [9] result in terms of generating functions we observe that we just need to compute a particular coefficient of the product of a bunch of "signed" and weighted Hesse polynomials to compute the determinant. Use of [8] enables us to perform this computation in TC^0 every time a value of the determinant is queried for.

Let $g_u(x)$ be the generating function of the clows with u as a head. The polynomial g_u is similar to the the Hesse polynnomial counting the number of u, u (closed) walks. To make this correspondence precise we notice that the basic difference between the two polynomials is that unlike in an arbitrary closed walk:

1. the vertex u occurs exactly once in a clow, and,
2. no vertex smaller than u is contained in the clow with head u.

Let G_u be the subgraph of G from which all vertices smaller than u, including u iself, have been deleted. Consider the sum of the Hesse polynomials $h_{G_u}(v, w)$ for each pair of vertices v, w such that $(w, u), (u, v)$ are edges in the graph (along with a 1 for the 0-length walk). This is because along with the path w, u, v a v, w walk forms a u, u closed walk satisfying the two properties above and vice-versa.

In other words we have shown that:

$$g_u(x) = \sum_{w,v:(w,u),(u,v)\in E(G)} x^2 h_{G_u}(v, w)$$

Proposition 1. *The generating function for clow sequences is:*

$$\prod_{u\in V} (g_u(x) - 1)$$

The -1 term arises because of the sign $(-1)^{n+k} = (-1)^{n-k}$ of a clow sequence.

Notice that maintaining G_u is easy - we just update an edge in G_u iff both its endpoints are larger than u. Now we can maintain the $h(x)$ for the G_u s, which follow the following update rule (on update of edge (i, j)):

$$h_{s,t}(x) := h_{s,t}(x) + b \cdot x \cdot h_{s,i}(x) \sum_{k=0}^{\infty} (b \cdot x \cdot h_{j,i}(x))^k h_{j,t}(x), \tag{1}$$

where b is 1 for addition update and -1 for deletion update. Moreover it suffices to maintain the summation up to only first polynomially many terms. Further, Hesse-Allender-Barrington [8] show in Corollary 6.5 that iterated sum and multiplication of polynomials is in $\mathsf{DLogtime}$-uniform TC^0. Thus we do not explicitly need to maintain the product of the terms but only compute the product when the determinant is queried. This completes the proof of the 0-1 case. In case of arbitrary weights the same proof goes through with a modified update rule replacing b by $b \cdot w(i, j)$. This completes the proof of Theorem 1. □

4 Proof of Theorem 2: Reachability in $\mathsf{DynAC}^0[2]$

The basic idea is to maintain the number of s, t walks modulo 2 (as usual, parameterised on the length) instead of the exact number. Thus if the graph can be made min-unique by appropriate weighting with polynomially bounded weights i.e. if there is a unique minimum weight path between every pair of connected vertices, then it suffices to check the number of paths modulo 2 for a given pair s, t for every weight from up to n because we are guaranteed that the coefficient of the min weight path will be 1. But Reinhardt-Allender [15] provides such a weighting scheme although a non-uniform one.

We have to be a bit careful in maintaining the Hesse polynomials $f_{s,t}$ modulo 2. In Hesse's paper he reduces the problem of maintaining the polynomials to iterated integer product and integer division which are in TC^0 by [8]. However, this reduction does not work for small rings like $\mathbb{Z}_2$.

We will of course maintain $h_{s,t}(x)$ mod 2 therefore in (1) the update will be identical for addition and deletion (since $b = 1 \equiv -1$ mod 2). We will also need to deal with $g_{i,j}(x) = \sum_{k=0}^{\infty} (h_{j,i}(x)x)^k$, where and henceforth in the section, arithmetic is modulo 2 unless explicitly mentioned otherwise. We cannot compute this on the fly because we no longer have the power of TC^0at our disposal. We will side-step the issue of computing g versus maintaining it by just considering an implicit representation of it as described below.

Observe that $g_{i,j}(x) = (1 - xh_{j,i}(x))^{-1}$ by just summing up the geometric series representing it. Thus $g_{i,j}(x) \in \mathbb{Z}_2(x)$ We will maintain each $h_{s,t}(x)$ as a rational polynomial i.e. the ratio of two polynomials rather than expand it as an infinite series/truncation of infinite series. If we do it naïvely the degrees of the numerator and denominator will start to grow as more and more updates occur. A simple observation takes care of this:

Observation 7. *Let $\alpha(x) = \frac{\beta(x)}{\gamma(x)} \in \mathbb{Z}_2(x)$ be a rational function such that the constant term $\gamma(0)$ of $\gamma(x)$ is 1. Further, let $\tilde{\beta}(x), \tilde{\gamma}(x)$ be the truncations of $\beta(x), \gamma(x)$ after the degree d terms. Let $\tilde{\alpha}(x) = \frac{\tilde{\beta}(x)}{\tilde{\gamma}(x)}$. Then $\tilde{\alpha}(x)$ is well defined and in the power series expansion of $\tilde{\alpha}(x)$ all the terms of degree at most d have the same coefficients as the corresponding terms in the expansion of $\alpha(x)$.*

Thus we maintain polynomials $\tilde{\beta}_{s,t}(x), \tilde{\gamma}_{s,t}(x)$ with the intention that $\tilde{f}_{s,t}(x)$ will be the ratio of these two polynomials. Thus the update rule Equation (1) is converted to the following equations:

$$\beta'_{s,t}(x) = \tilde{\beta}_{s,t}(x) \left(\tilde{\gamma}_{j,i}(x) - x\tilde{\beta}_{j,i}(x)\right) \tilde{\gamma}_{s,i}(x)\tilde{\gamma}_{j,t}(x) + \tilde{\gamma}_{s,t}(x)\tilde{\beta}_{s,i}(x)x\tilde{\gamma}_{j,i}(x)\tilde{\beta}_{j,t}(x) \quad (2)$$

$$\gamma'_{s,t}(x) = \tilde{\gamma}_{s,t}(x)\tilde{\gamma}_{s,i}(x) \left(\tilde{\gamma}_{j,i}(x) - x\tilde{\beta}_{j,i}(x)\right) \tilde{\gamma}_{j,t}(x) \quad (3)$$

We can do this in $\mathsf{AC}^0[2]$ because of Fact 1.

Next we truncate the two polynomials above at degree d (where d is the sum of all weights in the graph $= O(n^4)$), to obtain the new values $\tilde{\beta'}_{s,t}(x), \tilde{\gamma'}_{s,t}(x)$. This is also doable in $\mathsf{AC}^0[2]$.

To answer a query about whether there exists an s,t-path in the graph, we need to check the polynomials $\tilde{\beta}_{s,t}(x)$ for the various weighting functions promised by [15] and if any of them is non-zero we know that indeed there exists such a path. Conversely, because we have shown that in a weighting under which there exists a unique min-weight path (of some weight, say $w \leq d = O(n^4)$) between s,t, it must be the case that the coefficient of x^w in the power series expansion of $\frac{\tilde{\beta}_{s,t}(x)}{\tilde{\gamma}_{s,t}(x)}$ is non-zero hence $\tilde{\beta}_{s,t}(x)$ must also be non-zero. This completes the proof of Theorem 2. □

5 Proof of Theorem 3: Planar Reachability in DynFO

In this section we show that reachability in an embedded directed planar graph can be maintained in DynFO when adding or removing edges. Maintaining the transitive closure of a directed graph when adding an edge is easy, and we handle the deletion of edges by maintaining a type of dual to the directed graph, so that removing an edge from the original graph corresponds to adding an edge to the dual, and vice versa. To complete the proof, we show that an extended reachability relation on the dual graph is first-order definable in terms of the relation on the original graph, and vice versa.

We present the proof first in the context of a changing directed graph G which is a directed subgraph of a fixed, embedded planar graph H. Maintaining our data structures when planar edges are added to or removed from H presents no serious difficulties, and the details of this are left to the full version of the paper.

Let H be an embedded planar multigraph, and H' be its planar dual, with self-loops allowed in both graphs. Observe that a cycle in H gives a cut set in H': If all edges in H' dual to (crossing) the edges of a cycle in H are removed, then the remaining graph is not connected. This will be the basic idea of our proof, but our data structures become more complex to handle directed edges and to keep track of what is inside of a cycle and what is outside of a cycle.

The directed graph G is a directed subgraph of H, with up to two oppositely directed edges of G corresponding to each edge of H. We will define a directed dual complement graph G^{ddc} that is similarly a directed subgraph of H', and with the property that an edge is in G^{ddc} iff its corresponding dual edge is missing from G, where dual edge is as defined below.

To determine whether directed paths in G and G^{ddc} cross each other, we associate a crossing number with each pair of directed paths, (p,q) where p is a path in G, and q a path in G^{ddc}. This is the net number of times p crosses q, where a crossing is positive if q crosses p in a right to left direction, when oriented in the direction that p is going. If q crosses in the opposite direction, the crossing is a negative crossing. If the pair of paths is a pair of edges (e,e'), the crossing number is 0 if they do not cross, and is 1 or -1 if the two edges cross. The crossing number of paths is just the bilinear extension of this map to sets of edges, denoted $p \otimes q$.

We can now define the dual directed complement of G, G^{ddc}, as the directed subgraph of H' that contains only the edges whose crossing number with every edge of G is 0 or -1. In other words, for each edge e of G, we remove from the graph G^{ddc} the unique edge whose crossing number with e is $+1$. By construction, then, all paths in G have a negative or zero net crossing number with all paths in G^{ddc}.

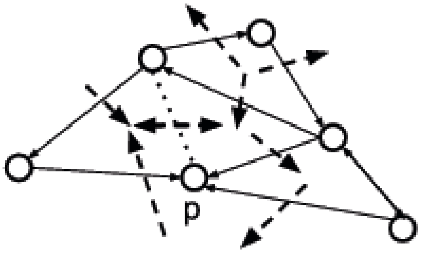


Fig. 1. Directed Dual Complement

In Figure 1, the graph G is shown with circles and solid arrows, and the graph G^{ddc} is shown with dashed arrows for directed edges, and without its nodes drawn. A cycle in G^{ddc} separates two points in G. For example, in Figure X, there is a clockwise cycle of dashed edges in G^{ddc} surrounding point p, and there is therefore no path in G connecting p to a point outside this cycle. If there was a path like that, it would have a crossing number of 1 with the cycle in G^{ddc}, which is impossible, because we excluded all edges with positive crossing numbers from G^{ddc}. It remains, now, to construct a data structure that will tell us about all cycles in G and G^{ddc}, and which points are inside and outside them.

5.1 Canonical Paths and Possible Crossing Numbers

If s and t are both inside or both outside a simple cycle in G^{ddc}, then all paths from s to t will have a net crossing number of 0 with the cycle. If the cycle separates s and t, the net crossing number will be 1 or -1. So we can pick any path from s to t to compute the net crossing number with the cycle. Therefore, we will maintain a spanning tree T of H, and a spanning tree T' of H', and will only compute the net crossing numbers of paths in G^{ddc} with paths in the spanning tree T, and vice versa.

Given a fixed path from s to t, such as the canonical spanning tree path, then if there are two paths from u to v in G^{ddc} with net crossing numbers i and j with that path from s to t, then for any k between i and j ,there is a path from u to v with crossing number k. The proof is left to the full version of the paper. This fact allows us to store only two numbers in our data structure for every such quadruple (s, t, u, v): the lowest number and highest number in the set of possible crossing numbers, if these bounds exist.

5.2 The Paths in G are First-Order Definable from the Paths in G^{ddc}

It should be clear that the bilinearity of net crossing number makes it easy to compute the new sets of possible crossing numbers when an edge is added to G^{ddc}. Let the set of possible net crossing numbers of a path from u to v in G^{ddc} with the canonical spanning tree path from s to t in H be denoted $Crossings((u,v),(s,t))$. If we denote the dual concept, of possible crossing numbers of paths from s to t in G with canonical paths in H' by $Crossings'$, then we can easily update $Crossings$ when deleting an edge from G, because this is just adding an edge to G^{ddc}. We can also update $Crossings'$ when adding an edge to G. If we can compute $Crossings'$ easily from $Crossings$, then we can maintain both of them when adding or deleting edges.

The first part of the formulas relating the two is relatively easy.

Theorem 8. *If the net crossing number of the canonical paths from s to t and u to v is k_0, then if $i \in Crossings((u,v),(s,t))$, and $j > k_0 - i$, then $j \notin Crossings'((s,t),(u,v))$*

Proof. This follows nicely from the bilinearity of $\otimes$ and the fact that the net crossing number of two cycles is zero (on a genus 0 surface, like the plane or a sphere). Let c be the canonical path from s to t in H and c' be the canonical path from u to v. Then if p is the path with crossing number i from u to v in G^{ddc} and q is any path from s to t in G, then $q - c$ is a cycle and $p - c'$ is a cycle. We will show that the crossing number of q and c' must be less than or equal to $k_0 - i$.

$$\begin{aligned} 0 &= (p - c') \otimes (q - c) \\ &= p \otimes q - c' \otimes q - p \otimes c + c' \otimes c \\ &= p \otimes q - c' \otimes q - i + k_0 . \end{aligned}$$

But since p is a path in G and q is a path in G^{ddc}, we know that $p \otimes q \leq 0$. So

$$\begin{aligned} 0 &\leq -c' \otimes q - i + k_0 \\ c' \otimes q &\leq k_0 - i . \end{aligned}$$

The opposite direction, that if $j \notin Crossings'((s,t),(u,v))$, then certain paths must exist, is more complex. The path that must exist is not necessarily a single path from u to v in G^{ddc}, but a path from a cycle in G^{ddc} containing u to a cycle containing v. The configuration of two cycles connected by a path looks like a pair of eyeglasses, so we call this the ”Eyeglass Lemma”, and prove it in the full version of the paper. The proof constructs these paths from the boundaries between regions reachable from s with crossing number i, and those not reachable from s with that crossing number. Any edge in this boundary must be in G^{ddc} or in the canonical spanning path from s to t, since otherwise the edge crossing the boundary would be in G and the path could be extended by this edge without changing the crossing number. Querying whether such an eyeglass configuration exists is a first-order query, since we can check all possible pairs of endpoints of the path between the two cycles, for the existence of the cycles at those points and the path between them.

The final lemmas, whose proofs are left to the full version of the paper, are that these quantities can be updated when the planar graph H is changed by adding or removing an edge, and that all of these quantities can be updated by first order formulas, putting the data structure in the dynamic complexity class DynFO.

6 Open Ends

A promising direction for future work on planar directed reachability is extending our proof from fixed embedded planar graphs to planar graphs whose embedding may change dramatically when adding and removing edges. It appears that a data structure maintaining the decomposition of the planar graph into 3-vertex-connected components (the SPQR-decomposition) would be necessary to keep track of this. Further directions include reducing the complexity of the update computations of this paper's algorithm from AC^0 circuits to quantifier-free first-order formulas, which use function terms but no quantifiers. These correspond to constant-depth circuits with selection gates (multiplexers), but only Boolean gates with constant fan-in. The technique of cut-cycle duality does not seem to extend much beyond the family of planar graphs. However, the techniques imported in this paper such as the Isolation Lemma and its de-randomization, might be useful to further reduce the dynamic complexity of reachability in general directed graphs. As a step towards the long-standing conjecture that reachability in arbitrary directed graphs is in DynFO, one may first want to improve our $\cap_p \mathsf{AC}^0[p]$ bound for updates, to AC^0. We note again that with the current knowledge $\cap_p \mathsf{AC}^0[p]$ may in fact be equal to AC^0.

Acknowledgements. The authors would like to thank Thomas Schwentick for bringing them together for collaboration. The first author would like to acknowledge Jenish C. Mehta's contribution in the initial part of this project without which it would not have been possible. The second author's research was supported by NSF grant CCR-987707.

References

1. Allender, E., Reinhardt, K., Zhou, S.: Isolation, matching, and counting uniform and nonuniform upper bounds. J. Comput. Syst. Sci. 59(2), 164–181 (1999)
2. Bourke, C., Tewari, R., Vinodchandran, N.V.: Directed planar reachability is in unambiguous log-space. TOCT 1(1) (2009)
3. Datta, S., Kulkarni, R., Roy, S.: Deterministically isolating a perfect matching in bipartite planar graphs. Theory Comput. Syst. 47(3), 737–757 (2010)
4. Dong, G., Su, J.: Incremental and decremental evaluation of transitive closure by first-order queries. Information and Computation 120(1), 101–106 (1995)

5. Grädel, E., Siebertz, S.: Dynamic definability. In: ICDT, pp. 236–248 (2012)
6. Hesse, W.: Dynamic computational complexity. Ph.D. thesis, U. Mass. (2003)
7. Hesse, W.: The dynamic complexity of transitive closure is in DynTC^0. Theor. Comput. Sci. 296(3), 473–485 (2003)
8. Hesse, W., Allender, E., Barrington, D.A.M.: Uniform constant-depth threshold circuits for division and iterated multiplication. J. Comput. Syst. Sci. 65(4), 695–716 (2002)
9. Mahajan, M., Vinay, V.: Determinant: Combinatorics, algorithms, and complexity. Chicago J. Theor. Comput. Sci. 1997 (1997)
10. Mehta, J.C.: Dynamic Complexity of Planar 3-connected Graph Isomorphism. In: CSR (accepted, 2014)
11. Miller, G.L., Naor, J.: Flow in planar graphs with multiple sources and sinks. SIAM J. Comput. 24(5), 1002–1017 (1995)
12. Mulmuley, K.: A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. Combinatorica 7(1), 101–104 (1987)
13. Mulmuley, K., Vazirani, U.V., Vazirani, V.V.: Matching is as easy as matrix inversion. Combinatorica 7(1), 105–113 (1987)
14. Patnaik, S., Immerman, N.: Dyn-FO: A parallel, dynamic complexity class. Journal of Computer and System Sciences 55(2), 199–209 (1997)
15. Reinhardt, K., Allender, E.: Making nondeterminism unambiguous. SIAM J. Comput. 29(4), 1118–1131 (2000)
16. Schwentick, T.: Perspectives of Dynamic Complexity. In: Libkin, L., Kohlenbach, U., de Queiroz, R. (eds.) WoLLIC 2013. LNCS, vol. 8071, pp. 33–33. Springer, Heidelberg (2013), `http://dx.doi.org/10.1007/978-3-642-39992-3-6`
17. Vollmer, H.: Introduction to circuit complexity - a uniform approach. Texts in theoretical computer science. Springer (1999)

One Tile to Rule Them All: Simulating Any Tile Assembly System with a Single Universal Tile[⋆],[⋆⋆]

Erik D. Demaine[1], Martin L. Demaine[1], Sándor P. Fekete[2],
Matthew J. Patitz[3], Robert T. Schweller[4],
Andrew Winslow[5], and Damien Woods[6]

[1] Massachussetts Institute of Technology, Cambridge, MA 02139, USA
{edemaine,mdemaine}@mit.edu
[2] TU Braunschweig, Germany
s.fekete@tu-bs.de
[3] University of Arkansas, Fayetteville, AR 72701, USA
patitz@uark.edu
[4] University of Texas–Pan American, Edinburg, TX 78539, USA
rtschweller@utpa.edu
[5] Tufts University, Medford, MA 02155, USA
awinslow@cs.tufts.edu
[6] California Institute of Technology, Pasadena, CA 91125, USA
woods@caltech.edu

Abstract. In the classical model of tile self-assembly, unit square *tiles* translate in the plane and attach edgewise to form large crystalline structures. This model of self-assembly has been shown to be capable of asymptotically optimal assembly of arbitrary shapes and, via information-theoretic arguments, increasingly complex shapes necessarily require increasing numbers of distinct types of tiles.

We explore the possibility of complex and efficient assembly using systems consisting of a single tile. Our main result shows that any system of square tiles can be simulated using a system with a single tile that is permitted to flip and rotate. We also show that systems of single tiles restricted to translation only can simulate cellular automata for a limited number of steps given an appropriate seed assembly, and that any longer-running simulation must induce infinite assembly.

Keywords: DNA computing, algorithmic self-assembly, hexagonal tiles.

[⋆] A full version of this paper can be found at http://arxiv.org/abs/1212.4756

[⋆⋆] Research of Matthew J. Patitz supported in part by NSF grant CCF-1117672. Research of Robert Schweller supported in part by NSF grant CCF-1117672. Research of Andrew Winslow supported in part by NSF grant CDI-0941538. Research of Damien Woods Supported by NSF grants 0832824 & 1317694 (the Molecular Programming Project), CCF-1219274, and CCF-1162589.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 368–379, 2014.

1 Introduction

This paper shows that many copies of a single rotatable polygonal tile type suffices to simulate any square tile assembly system.

Winfree [15] introduced the *abstract Tile Assembly Model* (aTAM) as a clean theoretical model for nanoscale self-assembling systems. In several experiments of increasing complexity and reliability [16,2,5,13,3], this model has been shown to be physically practical, with tiles composed of DNA strands. As a result, the aTAM has become the standard in theoretical work on self-assembly, with previous work exploring its abilities and limitations in terms of its ability to use computation to assemble shapes and patterns [12,1,14,6], as well comparing its computational power and simulation abilities [7,10,18].

In the aTAM, we start with a single specific tile, or a connected assembly of tiles, (called the *seed*) and repeatedly add any tile to the assembly that has enough matching glues (colored edges) to "stick" to the rest of the assembly. Each glue type (color class) has a natural number *strength*, which represents the affinity for matching glues of that type, and a global *temperature* (typically 2) of the system specifies the total required strength for a tile to attach to the assembly. Unlike Wang tiling, in the aTAM we can never throw away partially formed assemblies; in fact, the aTAM can be seen as a special kind of asynchronous, and nondeterministic, cellular automaton. See Section 2 for more details.

1.1 Our Results I: Universal Self-Assembly with One Tile

We prove in Section 5 that any aTAM system can be simulated by just a single tile, in a generalization of the aTAM model called the polygonal free-body Tile Assembly Model (pfbTAM). More precisely, we show that any temperature-τ aTAM system can be converted into a temperature-τ pfbTAM system with a single tile type t_U such that the two systems have exactly the same producible assemblies, modulo isometry. This construction is *self-seeding* in the sense that it starts from a single copy of the very same tile t; it is even a challenge to get the next copy of t to attach without uncontrolled infinite growth.

Another contribution of this paper is in our proof technique: we use a chain of four simulations. Such long chains of simulations, or reductions, are commonplace throughout the theory of computation, but not seen (so far) in self-assembly. Our single tile simulates an arbitrary square tile assembly system $\mathcal{T}$ as follows. We begin with the fact that the aTAM is intrinsically universal [7], which means that there is a single set of square tiles U that can simulate any square tile assembly system $\mathcal{T}$. Via the construction in [7], the tile assembly system $\mathcal{T}$ that we wish to simulate is first encoded as a seed assembly using tiles from the intrinsically universal tile set U to give a tile assembly system $\mathcal{U}_{\mathcal{T}}$. Next, the system $\mathcal{U}_{\mathcal{T}}$ is simulated by a "low-strength" hexagonal tile assembly system, as described below. The main result of this paper is that the resulting hexagonal assembly system can be simulated by a pfbTAM system consisting of only a single tile t_U. And, of course, our single tile t_U works for any such system $\mathcal{T}$ we wish to simulate. In particular, both the geometry and the dynamics of the simulated system are, modulo rescaling, precisely simulated: for every sequence of tile placement in the simulated system there is a sequence of blocks of tile placements of t_U in the simulator system, and vice-versa. Hence, when appropriately seeded, t_U assembles scaled-up versions of what is assembled by $\mathcal{T}$, and *in the same way* that $\mathcal{T}$ does it.

It is worth noting that the notion of intrinsic universality, with its strict notion of simulation, gives a framework to compare the power of tile assembly models that at first sight seem very different and perhaps difficult to compare. For example, in this paper we study rotatable and fliapable polygons, and translatable polyominos, yet simulation gives a way to directly compare the power of this model with the well-known square (aTAM) model. Intrinsic universality is giving rise to a kind of complexity theory for self-assembly systems allowing us to tease apart the power of different models [18].

Our universal tile t_U is a kind of geometric analog to a universal Turing machine, simultaneously simulating the shape construction and computational ability of an arbitrary tile assembly system (although our definition of simulation is in fact stronger—we care about dynamics, not merely input to output mappings). The existence of a system with just a single tile demonstrates that geometry alone (as opposed to, say, a large, although constant, number of square tile types [7]) suffices to bootstrap a system of self-assembly in even the most restrictive case where the system may only utilize copies of a single shape.

The pfbTAM model differs from aTAM in two ways: tiles consist of general simple polygons rather than squares and tiles are (possibly) permitted to rotate. Both are physically realistic aspects of self-assembling systems. For example, DNA origami [11] is a rapidly evolving technology that has been used to successfully build numerous complex shapes using strands of DNA. The technology has evolved to the point where free software automatically designs DNA to fold into essentially arbitrary desired shapes. Polyomino generalizations of square aTAM tiles have already been developed in practice [17] and studied in theory [8]. Rotation is clearly a natural attribute of all physical systems – prior work in the aTAM used a simple trick to eliminate rotation. Although our single polygonal tile t_U has such a large number of sides that its fabrication would be extremely challenging, our work here demonstrates that rotation can be used as an encoding mechanism to design systems that reuse a single tile at various rotations to achieve universal tile assembly systems. It would not be inconceivable to build a single tile with a more modest number of sides that simulates a simple square tile system.

The full version of this paper also shows the existence of a single tile that simulates any square tile system in the Wang model of plane tiling.

1.2 Our Results II: Hexagonal Tile Assembly Systems

As noted above, part of our proof involves the use hexagonal tiles: Section 4 describes aTAM systems with unit-sized hexagonal tiles on a hexagonal grid. The only previous paper considering this model [9] simply showed differences between squares and hexagons with respect to infinite constructions. Here we show that any temperature-2 square aTAM system can be simulated by a temperature-2 hexagonal aTAM system in which all glues have strength at most 1. The construction works at a scale factor of only 3: each square tile is simulated by a 3×3 block of hexagonal tiles. The main reason we use hexagons is that no such system is possible for square systems: any temperature-2 square aTAM system in which all glues have strength at most 1 cannot grow outside its bounding box (and so it cannot simulate arbitrary square-tile systems).

This result is a key step to proving our main positive result (aTAM simulation allowing translation and rotation). Specifically, we show in Section 5 how to simulate any temperature-2 hexagonal aTAM system that uses exclusively strength-1 glues with a rotatable polygon t_U that encodes different tile types by attaching at different rotation angles. Independent of their use in our constructions, hexagonal systems without strength-2 glues could be significantly easier to implement in the laboratory than square systems using both strength-1 and strength-2 glues in arbitrary arrangements on the tiles.

1.3 Our Results III: Linear-Time Computation with a Single Translation-Only Tile

In Section 6, we prove both a positive and a negative result on single-tile systems where the tile is forbidden from rotating. On the positive side, we prove that single-tile translation-only systems have non-trivial power: they are capable of time-bounded simulation of computationally universal 1D cellular automaton. Any 1D cellular automata that runs for n steps can be simulated starting with a seed assembly of $O(n)$ tiles. This is proved by first showing that single-tile translation-only systems can simulate a restricted class of multi-tile systems, which have previously been shown to simulate computationally universal 1D cellular automata.

On the negative side, we prove that translation-only single-tile systems need a seed assembly consisting of at least four tiles to carry out any non-trivial assembly. More formally, we prove that any single-tile translation-only system with a seed tile consisting of one, two, or three tiles either produces an infinite assembly, or only the seed assembly. More generally, we conjecture that no finite seed suffices for unbounded computational power with single-tile, translation-only systems, in stark contrast to our result that general single-tile pfbTAM systems have this power starting with a single-tile seed.

2 Model Definitions

The *polygonal free-body Tile Assembly Model* or *pfbTAM* generalizes self-assembly models such as the aTAM by using tiles with arbitrary polygonal shapes that may be translated and rotated. In our positive results, we only utilize tiles whose shapes are convex regular n-gons with small surface geometries. Our negative results are valid for arbitrary polygons, as discussed in Section 6.

A pfbTAM system Γ is defined as $\Gamma = (T, \Sigma, \tau, \sigma)$, where T is a *tile set* of polygonal *tiles*, Σ is a collection of *glue types*, $\tau \in \mathbb{N}$ is the *temperature* of the system, and σ is a *seed assembly* consisting of an arrangement of tiles from T and their locations. Each tile in T has a shape defined by a simple polygon (a polygon without holes), and each side of the tile has is assigned a *glue* from the collection of glue types Σ of Γ. Each glue type $g \in \Sigma$ is assigned a positive integer value called a *strength* with the exception of special *null glue* whose strength is 0.

If a pair of tiles of Γ are arranged in the plane such that their interiors do not overlap, and a pair of their edges are coincident, then these edges are said to form a *bond.* The *strength* of this bond is determined by the glues of both sides, with the strength equal to the strength of the glue if both sides have the same glue type, and zero otherwise.

A collection of tiles arranged in the plane whose interiors are pairwise disjoint is an *assembly.* The *bond graph* of an assembly is the (planar) multigraph consisting of labeled nodes for each tile, and an edge between a pair of nodes for each positive-strength bond the tiles share. An assembly is *τ-stable* if any edge-cut of the bond graph of the assembly has cut edges whose total corresponding bond strength meets or exceeds τ, the temperature of the system.

The seed assembly σ of Γ is a τ-stable assembly consisting of the tiles in T. The assembly process consists of attaching single tiles of T to a growing τ-stable assembly, beginning with σ, the seed assembly of the system. Because each single-tile attachment must yield a τ-stable assembly, a tile can attach to the growing assembly if and only if it is able to form bonds with assembly whose total strength is at least τ. Any assembly A that can be formed by this process is a *producible assembly of Γ* and is said to be *produced by Γ*. If no tile can attach to A, then A is also a *terminal assembly of Γ*. In some cases we consider pfbTAM systems in which tiles are not permitted to rotate, but instead merely translate. We call these system *translation-only pfbTAM systems.*

The well-studied aTAM and hTAM models (reviewed in Section 1) are both special cases of translation-only pfbTAM systems. An *abstract Tile Assembly Model (aTAM) system* is a translation-only pfbTAM system $\Gamma = (T, \Sigma, \tau, \sigma)$ where the tiles in T are unit squares, while a *hexagonal Tile Assembly Model (hTAM) system* is a system where the tiles in T are unit hexagons.

In Section 6 we also consider a restricted class of aTAM systems (rotated by 45°) called *pyramid aTAM systems*, proving that single-tile, translation-only pfbTAM systems are capable of simulating them. An aTAM system $\Gamma = (T, \Sigma, 2, \sigma)$ is said to be a *pyramid aTAM system* if three conditions hold. First, σ contains n tiles configured in the format described in Figure 1, with the property that all coincident tile edges have matching glues. Second, all glues in T have strength 1. Third, the tile set T and seed σ are such that no tiles can attach to the southern face of the seed. In addition, a pyramid aTAM system is said to be *double-checkerboarded* if every tile attaching to the seed assembly has distinct tile types at the locations to the southwest, south, and southeast of the tile's attachment location. An example of a coloring scheme that denotes which tiles must be of differing type is shown in Figure 1.

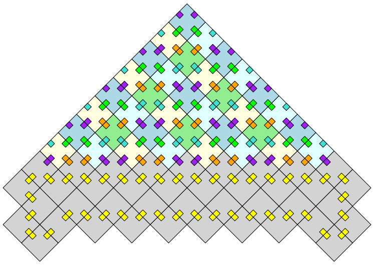


Fig. 1. Pyramid aTAM systems start with a seed assembly (gray) and grow upwards with cooperative temperature-2 bonding, yielding an assembly that is maximally pyramid-shaped. They can be used to simulate the light cone of a cellular automaton.

3 Simulation Definitions

En route to proving that pfbTAM systems with a single tile type are powerful, we prove that pfbTAM and hTAM systems can capture the behavior of, or *simulate*, aTAM systems. Below we define what it means for a pfbTAM system to simulate hTAM and aTAM systems, and for an hTAM system to simulate an aTAM system.

Loosely defined, a system simulates another if there is a mapping between the producible assemblies of both systems, such that a producible assembly A yields another producible assembly A' via a single-tile addition in one system if and only if, in the other system, the analogous assembly to A yields an analogous assembly to A' via a single-tile addition. In Sections 3.2 and 3.3 the simulating systems use a *block* of tiles to represent a single tile in the simulated system, and each single-tile addition in the simulated system is equivalent to a short sequence of single-tile additions in the simulated system, where the final addition completes the simulation of the single-tile addition in the simulated system.

3.1 Simulating hTAM Systems with pfbTAM Systems

A pfbTAM system $\Gamma_p = (T_p, \Sigma_p, \tau_p, \sigma_p)$ *simulates* an hTAM system $\Gamma_h = (T_h, \Sigma_h, \tau_h, \sigma_h)$ if there exists a mapping $\phi : T_p \times [0, 2\pi) \to T_h$ of orientations (specified by an angle in $[0, 2\pi)$) of tile in T_p to tiles in T_h such that there exists a bond graph G_{A_p} generated by a producible assembly A_p of Γ_p if and only if mapping the label of each node v of G_{A_p} to $\phi(p)$ yields a bond graph G_{A_h} of a producible assembly A_h of Γ_h.

Also, for each producible assembly A'_p produced by Γ_p via a single-tile addition to assembly A_p, an assembly A'_h in Γ_h equivalent to A'_p via ϕ can be produced by Γ_h via a single-tile addition to assembly A_h equivalent to A_f via ϕ and vice versa. In other words, equivalent assemblies are producible in both systems by equivalent sequences of tile additions.

3.2 Simulating aTAM Systems with hTAM Systems

Our definition of pfbTAM systems simulating hTAM systems uses a strict one-to-one-correspondence between tiles in the simulated and simulating systems. Here and in Section 3.3, our definition of hTAM systems simulating aTAM systems has a slightly weaker correspondence called a *c-block representation* where each tile in the simulated aTAM system corresponds to a $c \times c$ grid of tiles in the simulating hTAM system.

Let A_h be an assembly of an hTAM system $\Gamma_h = (T_h, \Sigma_h, \tau_h, \sigma_h)$ and A_a be an assembly of an aTAM system $\Gamma_a = (T_a, \Sigma_a, \tau_a, \sigma_a)$. Then A_h is a *valid c-block representation of* A_a for an odd, positive integer c and partial function $\phi : T_h \to T_a$ if two conditions hold. First, that A_h is evenly divisible in $c \times c$ blocks of tiles, as shown in Figure 2. Second, that x is in the domain of ϕ if and only if x is at the center of a $c \times c$ block.

Given a valid c-block representation A_h, define the *c-bond graph of* A_h to be a graph with a labeled node for each center tile x of a $c \times c$ block with label $\phi(x)$. The c-bond graph of A_h has an edge between two nodes corresponding to tiles x and x' if the bond graph of A_h has a length-c path between x and x' consisting of edges between tiles exclusively at angles 90° and −90°, or 120° and −30°.

Now we are ready to define simulation. We say that Γ_h *simulates* Γ_a *at scale* c, if there exists a partial function $\phi : T_h \to T_a$ such that three conditions hold. First, every tile in any producible assembly of Γ_h of size greater than $c^2 - 1$ is within distance at most c from a tile x for which $\phi(x)$ is defined. Second, there exists a producible assembly A_h of Γ_h that is a valid c-block representation for function $\phi(x)$, if and only if mapping the label of each node v in the c-block bond graph of A_h yields a bond graph of a producible assembly of Γ_a. Third, for each producible assembly A'_a of Γ_a produced by Γ_a via a single-tile addition to assembly A_a, there are equivalent c-block representation assemblies A'_h and A_h of Γ_h, such that A'_h is producible from A_h via a sequence of tile additions, for which each producible assembly created during this sequence of tile additions, the thing in Γ_h is A_h.

3.3 Simulating aTAM Systems with pfbTAM Systems

We define a c-scaled simulation of an aTAM system by a pfbTAM system by mapping $c \times c$ blocks within pfbTAM assemblies to aTAM tiles, where this mapping reads rotations of pfbTAM tiles in the blocks. A pfbTAM system $\Gamma_p = (T_p, \Sigma_p, \tau_p, \sigma_p)$ *simulates* an aTAM system $\Gamma_a = (T_a, \Sigma_a, \tau_a, \sigma_a)$ at scale $c \in \mathbb{N}$ if the following conditions hold, based on the more formal definition of [7].

First, there exists a mapping $\phi : ((T_p \cup \{\varnothing\}) \times [0, 2\pi))^{c^2} \to T_a \cup \{\varnothing\}$ of $c \times c$ blocks of tiles from T_p (with the output of ϕ depends on the orientations of those tiles, specified by a rotation angle in $[0, 2\pi)$) and empty locations (denoted $\varnothing$) to tiles in T_a and empty locations such that for every producible assembly A_p of Γ_p there is a producible assembly A_a in Γ_a, where $A_a = \phi^*(A_p)$ and for every producible assembly A_a of Γ_a there exists a producible assembly A_p in Γ_p, where $A_p = \phi^*(A_a)$ (here ϕ^* denotes the function ϕ applied to an entire assembly, in the most obvious block-wise way). We also require that A_p maps *cleanly* to A_a under ϕ^*, that is, for all non-empty $c \times c$ blocks b in A_p it is the case that at least one neighbor of $\phi(b)$ in $\phi^*(A_p)$ is non-empty, or else A_p has at most one non-empty $c \times c$ block. In other words, π may have tiles in $c \times c$ blocks representing empty space in α, but only if that position is adjacent to a tile in α.

Second, there exist producible assemblies A_a and A'_a of Γ_a such that $A_a \rightarrow_1 A'_a$ (growth by single tile addition), then for every producible assembly A_p of Γ_f, where $A_a = \phi^*(A_p)$ it is the case that there exists A'_p such that $A_p \rightarrow_* A'_p$ (growth by one or more tile additions) in Γ_f, where $A'_a = \phi^*(A'_a)$. Furthermore, for every pair of producible assemblies A_p, A'_p of Γ_f, if $A_p \rightarrow_* A'_p$, and $A_a = \phi^*(A_p)$ and $A'_p = \phi^*(A'_p)$, then $A_a \rightarrow_* A'_a$ for assemblies A_a, A'_a of Γ_a.

3.4 Simulating Pyramid aTAM Systems with Single-Tile Translation-Only pfbTAM Systems

In Section 6 we simulate *pyramid aTAM systems* (defined in Section 2), a special class of aTAM systems that have significant computational power but limited enough to permit simulation by translation-only, single-tile pfbTAM systems. Recall that a pyramid aTAM system is a restricted aTAM system in which each tile must attach to the growing assembly using exactly the southwest and southeast sides. The key idea of the simulation is to place an imaginary grid of boxes over a given assembly in the simulating translation-only system to define the position each tile (and specifically the tile's north/south position) and thus the tile of the aTAM system this tile is simulating; see Figure 3 for the idea.

We now define the mapping of an assembly in a single-tile, translation-only pfbTAM system $\Gamma_p = (T_p, \Sigma_p, 3, \sigma_p)$ to an assembly in the simulated pyramid aTAM system $\Gamma_a = (T_a, \Sigma_a, 2, \sigma_a)$. Consider a 2-stable assembly A consisting of translations of tiles of type p. Now consider the westmost, southmost tile in A. Assume this tile sits at coordinate position $(0, -x_1)$. Define a partial mapping $f : Z \times Z \rightarrow Z \times Z \times T$ that maps tile coordinate locations within an assembly to both a 2D coordinate position and a tile type in T.

Given the partial mapping f, for an assembly A produced by Γ_p, we say A simulates the assembly A' in Γ_a if A' is the assembly obtained by including each tile of type t at position (w, y), such that $f(x, y) = (w, u, t)$ for some tile in A at position (x, y). If any tile in A is at a position at which f is not defined, then A does not have a defined mapping to a square aTAM tile assembly over T.

We say that Γ_p *terminally simulates*, or simply *simulates*, Γ_a if the set of terminal assemblies of Γ_p maps exactly to the set of terminal assemblies of Γ_a. [1]

4 Low-Strength hTAM Systems Simulate aTAM Systems

In this section we prove that any temperature-τ aTAM system can be simulated by a temperature-τ hTAM system that uses only glues of strength *less than*

[1] This is a weaker definition of simulation than what is defined for the other pairs of models. While our construction actually satisfies an equivalent stronger definition, we omit the more involved simulation definition for simplicity.

τ (called *low-strength glues*). We call these hTAM systems *low-strength hTAM systems*. This is used later in Section 5 to simulate temperature-τ aTAM systems with single-tile pfbTAM systems (Lemma 2). See Figure 2 for an example.

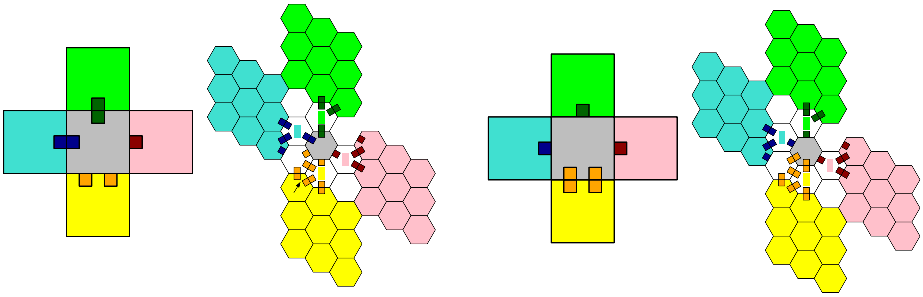


Fig. 2. Simulating a non-deterministic attachment in the aTAM via a low-strength hTAM system. The center location in the 3×3 block is a location of contention; multiple blocks compete and/or cooperate to claim it.

Lemma 1. *For any aTAM system $\Gamma_a = (T_a, \Sigma_a, \tau, \sigma_a)$ with $|\sigma_a| = 1$ and $\tau \geq 2$, there exists an hTAM system $\Gamma_h = (T_h, \Sigma_h, \tau, \sigma_h)$ that simulates Γ_a at scale 3 and has the property that all glues in Σ_h are of strength less than τ. Also, $|T_h| = O(|T_a|^2)$ and $|\sigma_h| = 3$.*

Furthermore, it is straightforward to see from the above construction that an aTAM system with seed σ_a and $|\sigma_a| \geq 1$, i.e. a seed assembly consisting of multiple tiles, can be simulated by an hTAM system, where the $9|\sigma_a|$ hexagonal tiles simulating the aTAM seed assembly are appropriately placed to represent that seed assembly. This gives the following corollary:

Corollary 1. *For any aTAM system $\Gamma_a = (T_a, \Sigma_a, \tau, \sigma_a)$ with $|\sigma_a| \geq 1$ and $\tau \geq 2$, there exists a low-strength hTAM system $\Gamma_h = (T_h, \Sigma_h, \tau, \sigma_h)$ that simulates Γ_a at scale 3. Also, $|T_h| = O(|T_a|^2)$ and $|\sigma_h| = 9|\sigma_a|$.*

5 Single-Tile pfbTAM Systems Simulate Low-Strength hTAM Systems

In this section we show that pfbTAM systems with a single tile type can simulate low-strength hTAM systems. Combining this result with Lemma 1 proves that single-tile pfbTAM systems can simulate all aTAM systems. Two-step simulation enables independent resolution of two main difficulties: using rotation as an encoding mechanism and eliminating uncontrolled growth of a single tile type with strength-τ glues.

Lemma 2. *For any low-strength hTAM system $\Gamma_h = (T_h, \Sigma_h, \tau, \sigma_h)$ with $|\sigma_h| = 3$, there is a pfbTAM system $\Gamma_p = (T_p, \Sigma_p, \tau, \sigma_p)$ with $|T_p| = 1$ and $|\sigma_p| = 3$ that simulates Γ_h at scale 1.*

Theorem 1. *(Universal Single-Tile Simulation) There exists a polygonal tile t_U such that for any aTAM system $\Gamma_a = (T_a, \Sigma_a, \tau_a, \sigma_a)$ with $|\sigma_a| = 1$ and $\tau_a \geq 2$, there exists a pfbTAM system $\Gamma_p = (\{t_U\}, \Sigma_p, 2, \sigma_p)$ simulating Γ_a.*

Here U denotes the intrinsically universal tile set [7]. As U is a fixed tile set, t_U is a tile with a constant number of sides. By adapting ideas from [4], we can eliminate the need for a multi-tile seed assembly, making the system *self-seeding.*

Theorem 2. *(Self-Seeding Single-Tile Simulation) For any aTAM system $\Gamma_a = (T_a, \Sigma_a, \tau, \sigma_a)$ with $|\sigma_a| = 1$ and $\tau \geq 2$, there exists a pfbTAM system $\Gamma_p = (T_p, \Sigma_p, \tau, \sigma_p)$ with $|T_p| = 1$ and $|\sigma_p| = 1$ that simulates Γ_a.*

6 Single-Tile Translation-Only pfbTAM Systems Simulate Cellular Automata

First we prove that single-tile pfbTAM systems where rotation is forbidden, called *translation-only* systems, are capable of arbitrary computation given an appropriately large seed. See Figure 3 for an example.

Theorem 3. *For any double-checkerboarded pyramid aTAM system $\Gamma_a = (T_a, \Sigma_a, 2, \sigma_a)$, there exists a translation-only pfbTAM $\Gamma_p = (\{t_p\}, \Sigma_p, 3, \sigma_2)$ that simulates Γ_a. Furthermore, t_p has $O(|T_a|^5)$ sides.*

Next, we prove that any single-tile, translation-only pfbTAM system with a seed assembly consisting of fewer than four tiles either produces only the seed assembly, or produces an infinite assembly.

Lemma 3. *Let S be a two-dimensional, bounded, connected, regular closed set S and $\boldsymbol{v}$ be a two-dimensional vector. Define $S + \boldsymbol{v} = \{p + \boldsymbol{v} : p \in S\}$. If $S + \boldsymbol{v} \cap S = \emptyset$, then $S + c \cdot \boldsymbol{v} \cap S = \emptyset$ for any non-zero integer c.*

Theorem 4. *For any translation-only pfbTAM system $\Gamma = (T, \Sigma, \tau, \sigma)$ with $|T| = 1$ and $|\sigma| = 1$, the set of producible assemblies of Γ is either $\{\sigma\}$ or contains assemblies of unbounded size.*

Corollary 2. *There are aTAM systems that cannot be simulated by any single-tile, translation-only pfbTAM system.*

Theorem 4 utilizes Lemma 3 and a simple observation about self-seeding systems: to form a two-tile assembly requires a strength-τ attachment between two individual tiles.

Theorem 5. *For any translation-only pfbTAM system $\Gamma = (T, \Sigma, \tau, \sigma)$ with $|T| = 1$ and $|\sigma| = 3$, the set of producible assemblies of Γ is either $\{\sigma\}$ or contains assemblies of unbounded size.*

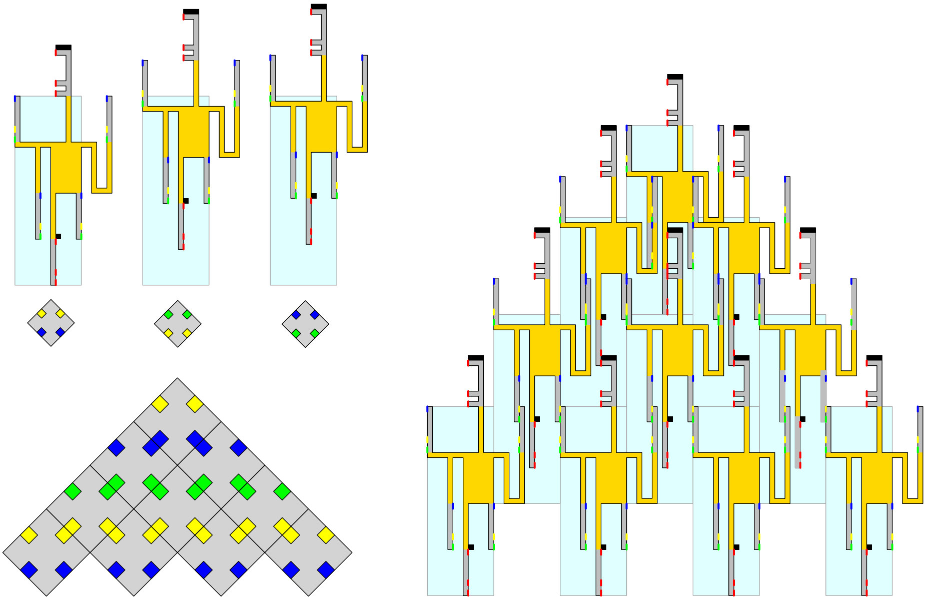


Fig. 3. An assembly of sliders (right) is mapped to a corresponding square tile assembly (left) by placing an imaginary grid (light blue)

A detailed proof is in the full paper. Matters get more involved with arbitrary seeds, and we conjecture polynomially large seeds are needed in general.

Conjecture 1. Let $\Gamma = (T, \Sigma, \tau, \sigma)$ be a translation-only pfbTAM system with $|T| = 1$ and $|\sigma| = n$. If $|\sigma| = n$, then the set of producible assemblies of Γ either contains exclusively assemblies of size $O(n^2)$ or contains assemblies of unbounded size.

Acknowledgements. This work was initiated at the 27th Bellairs Winter Workshop on Computational Geometry held on February 11-17, 2012 in Holetown, Barbados. We thank the other participants of that workshop for a fruitful and collaborative environment. In particular, we thank Brad Ballinger and Anna Lubiw for important discussions regarding Lemma 3. In addition, we thank Jarkko Kari for interesting and fruitful discussions on aperiodic tilings of the plane with few tile types.

References

1. Adleman, L., Cheng, Q., Goel, A., Huang, M.-D.: Running time and program size for self-assembled squares. In: Proceedings of 33rd Annual Symposium on Theory of Computing, pp. 740–748 (2001)

2. Barish, R.D., Rothemund, P.W., Winfree, E.: Two computational primitives for algorithmic self-assembly: Copying and counting. Nano Letters 5(12), 2586–2592 (2005)
3. Barish, R.D., Schulman, R., Rothemund, P.W., Winfree, E.: An information-bearing seed for nucleating algorithmic self-assembly. Proceedings of the National Academy of Sciences 106(15), 6054–6059 (2009)
4. Cannon, S., Demaine, E.D., Demaine, M.L., Eisenstat, S., Patitz, M.J., Schweller, R.T., Summers, S.M., Winslow, A.: Two hands are better than one (up to constant factors): Self-assembly in the 2HAM vs. aTAM. In: STACS 2013. LIPIcs, vol. 20, pp. 172–184. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (2013)
5. Chen, H.-L., Schulman, R., Goel, A., Winfree, E.: Reducing facet nucleation during algorithmic self-assembly. Nano Letters 7(9), 2913–2919 (2007)
6. Cook, M., Fu, Y., Schweller, R.: Temperature 1 self-assembly: Deterministic assembly in 3D and probabilistic assembly in 2D. In: Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 570–589 (2011)
7. Doty, D., Lutz, J.H., Patitz, M.J., Schweller, R.T., Summers, S.M., Woods, D.: The tile assembly model is intrinsically universal. In: Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 302–310 (2012)
8. Fu, B., Patitz, M.J., Schweller, R., Sheline, R.: Self-assembly with geometric tiles. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012, Part I. LNCS, vol. 7391, pp. 714–725. Springer, Heidelberg (2012)
9. Kari, L., Seki, S., Xu, Z.: Triangular and hexagonal tile self-assembly systems. In: Dinneen, M.J., Khoussainov, B., Nies, A. (eds.) Computation, Physics and Beyond. LNCS, vol. 7160, pp. 357–375. Springer, Heidelberg (2012)
10. Meunier, P.-E., Patitz, M.J., Summers, S.M., Theyssier, G., Winslow, A., Woods, D.: Intrinsic universality in tile self-assembly requires cooperation. In: SODA 2014: Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms, Portland, Oregon, pp. 752–771. SIAM (2014)
11. Rothemund, P.W.K.: Design of DNA origami. In: ICCAD 2005: Proceedings of the 2005 IEEE/ACM International Conference on Computer-Aided Design, pp. 471–478. IEEE Computer Society, Washington, DC (2005)
12. Rothemund, P.W.K., Winfree, E.: The program-size complexity of self-assembled squares (extended abstract). In: Proceedings of ACM Symposium on Theory of Computing (STOC), pp. 459–468 (2000)
13. Schulman, R., Winfree, E.: Synthesis of crystals with a programmable kinetic barrier to nucleation. Proceedings of the National Academy of Sciences 104(39), 15236–15241 (2007)
14. Soloveichik, D., Winfree, E.: Complexity of self-assembled shapes. SIAM Journal on Computing 36(6), 1544–1569 (2007)
15. Winfree, E.: Algorithmic Self-Assembly of DNA. PhD thesis, California Institute of Technology (June 1998)
16. Winfree, E., Liu, F., Wenzler, L.A., Seeman, N.C.: Design and self-assembly of two-dimensional DNA crystals. Nature 394(6693), 539–544 (1998)
17. Woo, S., Rothemund, P.W.: Stacking bonds: Programming molecular recognition based on the geometry of DNA nanostructures. Nature Chemistry 3, 620–627 (2011)
18. Woods, D.: Intrinsic universality and the computational power of self-assembly. In: MCU: Proceedings of Machines, Computations and Universality. Electronic Proceedings in Theoretical Computer Science, vol. 128, pp. 16–22 (2013)

Canadians Should Travel Randomly*

Erik D. Demaine[1], Yamming Huang[2], Chung-Shou Liao[2],
and Kunihiko Sadakane[3]

[1] Computer Science and Artificial Intelligence Laboratory,
Massachusetts Institute of Technology, Cambridge, MA 02139, USA
edemaine@mit.edu

[2] Department of Industrial Engineering and Engineering Management,
National Tsing Hua University, Hsinchu 30013, Taiwan
s100034528@m100.nthu.edu.tw, csliao@ie.nthu.edu.tw

[3] National Institute of Informatics, Tokyo, Japan
sada@nii.ac.jp

Abstract. We study online algorithms for the Canadian Traveller Problem (CTP) introduced by Papadimitriou and Yannakakis in 1991. In this problem, a traveller knows the entire road network in advance, and wishes to travel as quickly as possible from a source vertex s to a destination vertex t, but discovers online that some roads are blocked (e.g., by snow) once reaching them. It is PSPACE-complete to achieve a bounded competitive ratio for this problem. Furthermore, if at most k roads can be blocked, then the optimal competitive ratio for a deterministic online algorithm is $2k+1$, while the only randomized result known is a lower bound of $k+1$.

In this paper, we show for the first time that a polynomial time randomized algorithm can beat the best deterministic algorithms, surpassing the $2k+1$ lower bound by an $o(1)$ factor. Moreover, we prove the randomized algorithm achieving a competitive ratio of $\left(1+\frac{\sqrt{2}}{2}\right)k+1$ in pseudo-polynomial time. The proposed techniques can also be applied to implicitly represent multiple near-shortest s-t paths.

1 Introduction

Imagine attempting to drive across your favorite northern country in the dead of winter. Snow is falling in unpredictable patterns, effectively blocking certain roads from passage (either from lack of plowing or accident pile-ups). You have purchased a complete road map, modeled as a weighted graph $G = (V, E)$ whose edges represent roads and whose edge weights represent the time to traverse the edge. But you have no knowledge of which roads are blocked by weather or accidents, until you reach a vertex incident to such a road, in which case you can directly observe the blockage before attempting traversal. This problem, called the Canadian Traveller Problem (CTP), was defined by Papadimitriou and Yannakakis [14] in 1991. The objective is to design an efficient route from a

* Supported by NSC Grant 102-2221-E-007-075-MY3 and KAKENHI 23240002.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 380–391, 2014.

source to a destination under this condition of uncertainty. The major difficulty in developing a good strategy based on partial information is to make decisions without being able to predict blockages.

Prior Work on CTP. Papadimitriou and Yannakakis [14] proved that it is PSPACE-complete to devise a CTP strategy that guarantees a bounded competitive ratio. They also proved that the stochastic model of this problem, in which a probability that each edge is blocked, independent of all others, is given in advance, is #P-hard when minimizing the expected competitive ratio to the offline optimum [14]. Bar-Noy and Schieber [1] investigated several variations of the CTP from the worst-case perspective, where the objective is to find a static (offline) algorithm that minimizes the maximum travel cost [2]. They considered the k-CTP in which the number of blockages is bounded by k. Note that for an arbitrary k, the problem of designing a strategy that guarantees a given travel time remains PSPACE-complete, as has been shown in [1,14].

During recent decades, no significant progress has been made in the development of online approximation algorithms for solving the k-CTP. Basically, two simple deterministic strategies for solving this problem are available [16,17]. The *greedy* algorithm (GA) starts at a vertex v and finds the shortest v-t path using Dijkstra's algorithm [5] in a greedy manner based on the current blockage information. The other strategy, called the *reposition* algorithm (RA), proposed by Westphal [16], requires the traveller to begin at the source s, follow the shortest s-t path until she learns about a blockage on the path to t, and then returns to s and takes a new shortest s-t path based on the updated blockage information. Westphal [16] proved that no deterministic online algorithm within a $(2k+1)$-competitive ratio exists for the problem, and the simple reposition algorithm achieves the lower bound. Westphal also proved a lower bound of $k+1$ on the competitive ratios of all randomized online algorithms. Xu et al. [17] developed a similar deterministic adaptive *comparison* strategy that incorporates the concept of reposition; the approach achieves the tight deterministic lower bound as well. They also showed that the competitive ratio of the GA algorithm is exponential in k in the worst case. Huang and Liao [9] considered a generalization of the k-CTP, called the DOUBLE-VALUED GRAPH, in which each edge is associated with two possible distances. They proposed lower bounds and a simple algorithm that meets the deterministic lower bound. They also extended the k-CTP to the design of a tour through a set of vertices, in which the traveller visits each vertex and returns to the origin under the same uncertainty.

Our Results. We develop improved randomized strategies for solving the k-CTP. Whereas deterministic algorithms have been extensively studied, research on randomized approaches for solving the problem is lacking.[1] In this paper, we propose a polynomial time randomized algorithm that surpasses the deterministic lower bound of $2k+1$ by an $o(1)$ factor. In addition, the competitive ratio of

[1] Recently, Bender and Westphal proposed a randomized algorithm using the RA strategy for special graphs in which all s-t paths are vertex-disjoint [3].

this algorithm can be improved to $\left(1+\frac{\sqrt{2}}{2}\right)k+1$ in pseudo-polynomial running time. This result is the first demonstration that randomization strictly helps in the k-CTP for arbitrary graphs.

The rationale behind the proposed randomized algorithm is summarized as follows. Given a connected graph $G=(V,E)$ with a source s and a destination t in V and a distance function $d: E \rightarrow R^+$, the algorithm first selects a set S of near-shortest s-t paths whose distance cost does not exceed the product of that of the shortest s-t path and a threshold factor. Precisely, this set comprises all s-t paths of cost $(1+\alpha)d(s,t)$, where $d(s,t)$ denotes the shortest travel cost from s to t and α is a small constant. Let the set of all such paths be represented by an *apex tree* T, which is a tree-like graph that becomes a tree by removal of a vertex. Then, the traveller traverses T rather than the original graph G using an online randomized strategy under the same uncertainty until all possible s-t paths in S are blocked. We repeat the argument until she arrives at t.

Near-Shortest Paths. In order to conduct the randomized algorithm in polynomial time, we need to find all near-shortest s-t paths efficiently, which is of independent interest. There has been a considerable amount of research investigating the problem. Eppstein's well-known approach for finding ℓ shortest s-t paths or all s-t paths shorter than a given distance cost, in a directed graph $G=(V,A)$, spends constant time for each of the ℓ paths, after a fast preprocessing step that runs in $O(|V|\log|V|+|A|)$ time [6]. That is, the ℓ shortest paths can be obtained in $O(\ell)$ time. Note that ℓ may be exponential in $|V|$, even if all the ℓ paths have the same distance. In Eppstein's study, cycles of repeated vertices were allowed. Recently, Hershberger et al. [8] and Frieder and Roditty [7] studied finding the ℓ shortest simple (i.e., loopless) paths in directed graphs. In undirected graphs, Katoh et al. [11] investigated finding ℓ shortest simple paths in an undirected graph $G=(V,E)$, and their algorithm, which takes $O(\ell(|V|\log|V|+|E|))$ time, is the best known-to-date result.

In this paper, we propose an implicit representation, constructed in $O(\mu^2|E|^2)$ time and $O(\mu|E|)$ space, of all *strictly* jth-shortest s-t paths, $1 \leq j \leq \mu$, where μ is at most the summation of distances of all edges. That is, assume $d^1(s,t), d^2(s,t), d^3(s,t), \ldots$ denotes the strictly increasing sequence of all possible distinct s-t path distances, where $d^1(s,t)=d(s,t)$; our technique can represent all strictly jth-shortest paths of cost $d^j(s,t)$, $1 \leq j \leq \mu$. Note that a strictly jth-shortest s-t path can be obtained by finding ℓ shortest s-t paths for a sufficiently large value of ℓ; however, the worst-case number of such near-shortest s-t paths can be exponential in the order of G. The proposed implicit representation can guarantee the pseudo-polynomial running time of the randomized strategy.

2 Preliminaries

Given a connected graph $G=(V,E)$ with a source s and a destination t, let an s-t path p of length m be $p: s=v_1-v_2-\cdots-v_m-v_{m+1}=t$. We denote the subset of blockages in E identified by an online algorithm A during the trip as

$E_i^A = \{e_1, e_2, \ldots, e_i\} \subseteq E$, $1 \leq i \leq k$, where e_i is the ith blockage identified, and let $E_0 = \emptyset$ and E_k be the set of all blocked edges. Let $d_{E_i^A}(s,t)$ denote the travel cost from s to t, derived by an adaptive algorithm A that learns about blockage information E_i during the trip; and let $d_{E_k}(s,t)$ be the offline optimum from s to t under complete information E_k. For convenience, we denote by $d_{E_i}(s,t)$-*path* a route along which the traveller spends at most $d_{E_i}(s,t)$. For all instances, the following property is immediately obtained, where $E_1 \subseteq E_2 \subseteq \cdots \subseteq E_k$.

$$d(s,t) = d_{E_0}(s,t) \leq d_{E_1}(s,t) \leq \cdots \leq d_{E_k}(s,t). \qquad (1)$$

We refer to [4,15] and formally define the competitive ratio as follows. An online randomized algorithm A is c^A-*competitive* against an oblivious adversary for the k-CTP if

$$\mathrm{E}[d_{E_i^A}(s,t)] \leq c^A \cdot d_{E_k}(s,t) + \varepsilon, \ \ 1 \leq i \leq k,$$

where $\mathrm{E}[d_{E_i^A}(s,t)]$ is the expected travel cost of the randomized strategy A and c^A and ε are constants. To analyze the performance of online algorithms for the k-CTP, we make two basic assumptions [1,17]: one is that once a blocked edge is discovered by the traveller, the edge remains blocked forever. The other is that the given graph G remains connected even if all the blockages are eliminated.

3 Main Algorithm

Given a connected graph $G = (V, E)$ with a source s and a destination t, we require a set S of all $(1+\alpha)d_{E_i}(s,t)$-paths from s to t under blockage information E_i, where α is a constant, $0 < \alpha < 1$. In contrast to the previous studies, we find strictly jth-shortest paths, $1 \leq j \leq \mu$, for a sufficiently large μ, to derive the set S of the s-t paths. The technique for so doing will be presented later.

Algorithm 1. Greedy & Reposition Randomized Algorithm (*GRR*)

```
Input : A graph $G = (V, E)$ with a source $s$ and a sink $t$, and constants $k$ and $\alpha$;
Output : A random route from $s$ to $t$;
Let $i = 0$;                                              ▷ no blockage found
while the traveller does not arrive at $t$ do
    Find a set $S$ of all $(1+\alpha)d_{E_i}(s,t)$-paths from $s$ to $t$;
    Randomly select an $s$-$t$ path from $S$ to traverse with the following probabilities;
    $\frac{k-i}{k-i+1}$: she leaves for $t$ along any $d_{E_i}(s,t)$-path until a blockage is found;
    $\frac{1}{k-i+1}$: she follows Traverse-Tree on the remaining $s$-$t$ paths in $S$;
    if the traveller learns about the $j$th blockage on the way to $t$, $j < k$ then
        the traveller returns to $s$ and $i \leftarrow j$;
    else if the traveller learns about the last blockage then ▷ finding all blockages
        the traveller returns to $s$ and follows a $d_{E_k}(s,t)$-path to $t$;
    end if
end while
```

The main algorithm (Algorithm 1) is described as follows. First, we select a set S of all $(1+\alpha)d_{E_i}(s,t)$-paths from s to t under blockage information E_i, $0 \leq i \leq k$, for a small α. Then, the traveller uses the *reposition* RA strategy, and when restarting at s, she either selects an arbitrary $d_{E_i}(s,t)$-path from S and traverses the path with probability $\frac{k-i}{k-i+1}$, or she chooses the remaining s-t paths in S and traverses them using the Traverse-Tree procedure with probability $\frac{1}{k-i+1}$. The algorithm repeats until the traveller arrives at t.

To analyze the competitive ratio of the entire *GRR* algorithm, the traveller is supposed to be able to efficiently traverse the remaining s-t paths in S following the Traverse-Tree procedure, whose details will be introduced in the next section. More precisely, if the extra travel cost of the procedure is bounded within an acceptable range for every blockage discovered, then the ratio can be improved over the deterministic lower bound of $2k+1$. The correctness of the claim will be proved in the next section.

Claim 1: If the traveller uses the Traverse-Tree procedure on the $(1+\alpha)d_{E_i}(s,t)$-paths from s to t in S and arrives at t, $0 \leq i < k$, each blockage increases the total travel cost of the algorithm by at most $(1+\alpha)d_{E_k}(s,t)$ on average.

Theorem 1. *The k-Canadian Traveller Problem can be approximated within a competitive ratio $\left(1+\frac{\sqrt{2}}{2}\right)k+1$, when the number of blockages is up to a given constant k.*

Proof. In each iteration, r, of the while loop, let the cost of conducting the Traverse-Tree procedure be $c(r)$ for blockages discovered. When the traveller arrives at t using the *GRR* algorithm, consider the case whether she learns about all the blockages or not.

Case 1: The traveller does not learn about every blockage when she arrives at t; i.e. at least one $(1+\alpha)d_{E_{k-1}}(s,t)$-path from s to t is unblocked during the trip.

In the worst case analysis, assume the traveller learns about only one blockage in each iteration of the while loop. Based on Claim 1, $c(r) \leq (k-r+1)(1+\alpha)d_{E_k}(s,t)$ for every iteration r of the loop, where the number of remaining blockages undiscovered is $k-r+1$ in iteration r. Consequently, the expected travel cost of the *GRR* algorithm is formulated as follows:

$$\begin{aligned}
\mathrm{E}[d_{E_k^{GRR}}(s,t)] \leq & \left[\frac{k}{k+1}\cdot 2d(s,t)+\frac{1}{k+1}\cdot c(1)\right]+\frac{k}{k+1}\left[\frac{k-1}{k}\cdot 2d_{E_1}(s,t)+\frac{1}{k}\cdot c(2)\right] \\
& +\frac{k}{k+1}\cdot\frac{k-1}{k}\cdot\left[\frac{k-2}{k-1}\cdot 2d_{E_2}(s,t)+\frac{1}{k-1}\cdot c(3)\right]+\cdots\cdots \\
& +\left[\frac{1}{k+1}\cdot 2d_{E_{k-1}}(s,t)+\frac{1}{k+1}\cdot c(k)\right]+d_{E_k}(s,t) \\
\leq & \left(\frac{k}{k+1}+\frac{k-1}{k+1}+\cdots+\frac{1}{k+1}\right)\cdot 2d_{E_{k-1}}(s,t) \\
& +\left(\frac{k}{k+1}+\frac{k-1}{k+1}+\cdots+\frac{1}{k+1}\right)\cdot(1+\alpha)d_{E_k}(s,t)+d_{E_k}(s,t) \\
\leq & \left[k+\frac{1}{2}(1+\alpha)k+1\right]\cdot d_{E_k}(s,t).
\end{aligned}$$

Thus, in Case 1, the competitive ratio of the *GRR* algorithm is at most

$$\frac{\left[k + \frac{1}{2}(1+\alpha) \cdot k + 1\right] \cdot d_{E_k}(s,t)}{d_{E_k}(s,t)} = \frac{3+\alpha}{2} \cdot k + 1.$$

Case 2: The traveller learns about all the k blockages before she arrives at t; that is, each $(1+\alpha)d_{E_{k-1}}(s,t)$-path is blocked during the trip. Thus, she eventually restarts at s and follows a $d_{E_k}(s,t)$-path to t, as indicated in Line 8 of the *GRR* algorithm.

The condition implies that the distance of an offline optimal s-t path is $d_{E_k}(s,t) > (1+\alpha)d_{E_{k-1}}(s,t)$. Moreover, Claim 1 cannot be applied because the traveller cannot reach t while conducting the Traverse-Tree procedure. Thus, an upper bound on the travel cost $c(r)$ in iteration r is derived by exploiting the RA strategy [16]. In the worst case analysis, $c(r) \leq 2(k-r+1)(1+\alpha)d_{E_{k-1}}(s,t)$ for every iteration r, where the number of remaining blockages undiscovered is $k-r+1$ in iteration r. The expected total travel cost of the *GRR* algorithm is formulated in a similar manner:

$$\mathrm{E}[d_{E_k^{GRR}}(s,t)] \leq k(2+\alpha) \cdot d_{E_{k-1}}(s,t) + d_{E_k}(s,t).$$

Hence, in Case 2, the competitive ratio of the *GRR* algorithm is at most

$$\frac{k(2+\alpha) \cdot d_{E_{k-1}}(s,t) + d_{E_k}(s,t)}{d_{E_k}(s,t)} \leq \frac{k(2+\alpha) \cdot d_{E_{k-1}}(s,t)}{(1+\alpha) \cdot d_{E_{k-1}}(s,t)} + 1 = \frac{2+\alpha}{1+\alpha} \cdot k + 1.$$

By simple algebra, the competitive ratio of the *GRR* algorithm can be minimized in the two cases by letting the constant α be $\sqrt{2}-1$. So the expected competitive ratio is at most $\left(1+\frac{\sqrt{2}}{2}\right)k+1$. Note that for any small constant α', $0 < \alpha' \leq \alpha = \sqrt{2}-1$, the competitive ratio is still strictly smaller than the deterministic lower bound of $2k+1$. □

4 Apex Tree

In this section, we consider the Traverse-Tree procedure conducted in a tree-like graph, called an *apex tree*, which can represent all $(1+\alpha)d_{E_i}(s,t)$-paths from s to t, $0 \leq i < k$, in a given graph G, for a small constant α.

A graph $T = (V,E)$ is an *apex tree* if T comprises a source vertex s, a rooted tree that consists of a destination vertex t (as root) and all other vertices in V, and edges that connect s to each leaf and some internal vertices of the tree. That is, $T \setminus \{s\}$ is actually a tree that is rooted at t.

Subsequently, we claim that the Traverse-Tree procedure (see Algorithm 2) is an optimal randomized strategy for solving the k-CTP in an apex tree T, if the distance cost of every s-t path in T is assumed to be identical. Note that the worst-case instance that was reported in [16] to establish the lower bound of $k+1$ is in fact also an apex tree where all s-t paths have the same cost.

Algorithm 2. TRAVERSE-TREE

```
Input : An apex tree $T$ that represents all $(1+\alpha)d_{E_i}(s,t)$-paths from $s$ to $t$;
Output : A random route from $s$ to $t$;
Assign equal probabilities to every child of the root $t$, and sequentially repeat the
  process for each descendant of $t$ in the order of breadth-first search;
The traveller begins at $s$, and randomly selects an $s$-$t$ path according to the assigned
  probability and leaves for $t$ following that path;
while the traveller does not arrive at $t$ do
    Let the blocked edge discovered by the traveller be $e = (v_i, v_{i+1})$ along a path
      $p : s = v_1 - v_2 - \cdots - v_h = t$;
    The traveller returns to $s$ and eliminates the blocked $s$-$v_i$ path;
    while every $s$-$t$ path through the vertex $v_{i+1}$ is currently blocked do
        $i \leftarrow i + 1$;                         ▷ depth-first search order of the path $p$
    end while
    Reassign probabilities only to the subtree rooted at $v_{i+1}$ in a similar way;
    The traveller randomly selects an $s$-$t$ path through $v_{i+1}$ according to the
      assigned probability and leaves for $t$ following that path;
end while
```

Thus, the competitive ratio of the algorithm can achieve the lower bound and it follows that the algorithm is optimal.

The main concept of the Traverse-Tree procedure is to incorporate randomized operations into the reposition RA strategy and then to explore subtrees of an apex tree T in the order of *depth-first search.* Precisely, we initially distribute probabilities of path selection equally among every child of the root t. We sequentially distribute the probabilities equally to each descendant in the order of *breadth-first search.* When the traveller starts at the source s, she randomly selects an s-t path according to the assigned probability and follows the path to t. While finding a blockage on the way to t, the traveller uses the RA strategy and returns to s. We eliminate the blocked path, and reassign probabilities to the unblocked subtrees similarly. The traveller traverses the remaining routes in T by exploring the subtrees in the order of *depth-first search.* The argument repeats until the traveller arrives at the destination t.

For the k-CTP in an apex tree T, there is at least one s-t path without a blockage. Let this offline optimal s-t path be $p : s = v_1^* - v_2^* - \cdots - v_m^* = t$ and the number of children of a vertex v_j^* be c_j in the apex tree T, such that v_j^* has children $v_{j,1}, v_{j,2}, \ldots, v_{j,c_j}$. Assume the last child of each v_j^*, v_{j,c_j}, $2 \leq j \leq m$, lies on the path p; i.e. $v_{j,c_j} = v_{j-1}^*$. Moreover, suppose each subtree rooted at $v_{j,\ell}$, $1 \leq \ell \leq c_j - 1$, has $b_{j,\ell}$ blockages. Consider the expected total cost when the traveller traverses the paths other than the offline optimal path. Note that the malicious adversary does not block any edge $(s, v) \in E$. So $\sum_{j=3}^{m} \sum_{\ell=1}^{c_j-1} b_{j,\ell} = k$.

Lemma 1. *For the k-CTP in an apex tree T in which the distance costs of all s-t paths are equal, there is an optimal $(k+1)$-competitive randomized algorithm.*

Proof. Let $\mathrm{E}(s, v_i^*)$ be the expected total travel cost from s to v_i^*. For $t = v_m^*$, we evaluate the cost. If $c_m = 1$, we obtain $\mathrm{E}(s, v_m^*) \leq \mathrm{E}(s, v_{m-1}^*) + d_{E_k}(v_{m-1}^*, v_m^*)$.

If $c_m > 1$, with probability $\frac{1}{c_m}$, the traveller finds v^*_{m-1} as a predecessor of v^*_m in the first trial. Then the expected travel cost is at most $\mathrm{E}(s, v^*_{m-1}) + d_{E_k}(v^*_{m-1}, v^*_m)$. If the traveller cannot find v^*_{m-1} in the first trial, with probability $\left(1 - \frac{1}{c_m}\right)\frac{1}{c_m-1}$, she finds v^*_{m-1} in the second trial. Without loss of generality, suppose the traveller selects $v_{m,\ell}$ from $\ell = 1$ to $\ell = c_m - 1$, when finding $v^*_{m-1} = v_{m,c_m}$. In this case the expected travel cost is $\left\{2b_{m,1}d_{E_{b_{m,1}}}(s, v_{m,1}) + \mathrm{E}(s, v^*_{m-1}) + d_{E_k}(v^*_{m-1}, v^*_m)\right\}$ because the reposition algorithm may return to s $b_{m,1}$ times and find the way to v^*_{m-1} with $\mathrm{E}(s, v^*_{m-1})$ expected cost and finally go to v^*_m with cost $d_{E_k}(v^*_{m-1}, v^*_m)$. Therefore we obtain

$$\begin{aligned}
\mathrm{E}(s, v^*_m) \le{}& \tfrac{1}{c_m}\left\{\mathrm{E}(s, v^*_{m-1}) + d_{E_k}(v^*_{m-1}, v^*_m)\right\}\\
&+ (1 - \tfrac{1}{c_m})\tfrac{1}{c_m-1}\left\{2b_{m,1}d_{E_{b_{m,1}}}(s, v_{m,1}) + \mathrm{E}(s, v^*_{m-1}) + d_{E_k}(v^*_{m-1}, v^*_m)\right\}\\
&+ (1 - \tfrac{1}{c_m})(1 - \tfrac{1}{c_m-1})\tfrac{1}{c_m-2}\left\{2b_{m,1}d_{E_{b_{m,1}}}(s, v_{m,1}) + 2b_{m,2}d_{E_{b_{m,2}}}(s, v_{m,2})\right.\\
&\left.+ \mathrm{E}(s, v^*_{m-1}) + d_{E_k}(v^*_{m-1}, v^*_m)\right\} + \cdots\\
&\le \tfrac{1}{c_m}\left\{2(b_{m,1} + b_{m,2} + \cdots + b_{m,c_m-1})d_{E_k}(s,t)\right\} + \mathrm{E}(s, v^*_{m-1})\\
&+ d_{E_k}(v^*_{m-1}, v^*_m)\\
&\le \sum_{\ell=1}^{c_m-1} b_{m,\ell}d_{E_k}(s,t) + \mathrm{E}(s, v^*_{m-1}) + d_{E_k}(v^*_{m-1}, v^*_m)
\end{aligned}$$

Therefore either $c_m = 1$ or not, we obtain $\mathrm{E}(s, v^*_m) \le \sum_{\ell=1}^{c_m-1} b_{m,\ell}d_{E_k}(s,t) + \mathrm{E}(s, v^*_{m-1}) + d_{E_k}(v^*_{m-1}, v^*_m)$. Similarly, $\mathrm{E}(s, v^*_{m-1}) \le \sum_{\ell=1}^{c_{m-1}-1} b_{m-1,\ell}d_{E_k}(s,t) + \mathrm{E}(s, v^*_{m-2}) + d_{E_k}(v^*_{m-2}, v^*_{m-1})$. Therefore, $\mathrm{E}(s, v^*_m) \le \sum_{j=2}^{m}\sum_{\ell=1}^{c_j-1} b_{j,\ell}d_{E_k}$ $(s,t) + d_{E_k}(v^*_1, v^*_2) + \ldots + d_{E_k}(v^*_{m-1}, v^*_m) = (k+1)d_{E_k}(s,t)$, because $\sum_{\ell=1}^{c_2-1} b_{2,\ell} = 0$. The expected total cost of the algorithm, including the distance cost of the offline optimal path, is $(k+1)d_{E_k}(s,t)$. The competitive ratio achieves the lower bound for any randomized online algorithms in apex trees. □

Based on the above proof, if the distance cost of each s-t path in T is at most $(1+\alpha)d_{E_i}(s,t)$, $0 \le i < k$, then the upper bound on the expected total travel cost of the algorithm is $\sum_{j=3}^{m}\sum_{\ell=1}^{c_j-1} b_{j,\ell}(1+\alpha)d_{E_k}(s,t) = k(1+\alpha)d_{E_k}(s,t)$, excluding the cost of the offline optimal path. The next theorem follows immediately and proves the claim made in Section 3, i.e., each blockage increases the total travel cost by at most $(1+\alpha)d_{E_k}(s,t)$ on average.

Theorem 2. *For the k-CTP in an apex tree T in which each s-t path is a $(1+\alpha)d_{E_i}(s,t)$-path, $0 \le i < k$, the competitive ratio of the Traverse-Tree procedure is at most $(1+\alpha)k+1$.*

5 Implicit Representation of Near-Shortest Paths

Given a connected undirected graph $G = (V, E)$ with a source s and a destination t, we give two simple data structures for storing all shortest to strictly μth-shortest s-t paths, for a large value of μ. The first representation is for storing shortest to strictly jth-shortest simple s-t paths, provided that $d^j(s,t)$ is given, $1 \leq j \leq \mu$, and the second one is for representing possibly non-simple paths whose distances are from the cost $d(s,t)$ to $d^\mu(s,t)$.

5.1 Strictly Second-Shortest Paths

For each vertex $v \neq s$, let $S^j(v)$ be the vertex set that comprises all v's predecessors, each of which is v's preceding neighbor lying in a strictly ith-shortest s-v path, $1 \leq i \leq j$. To represent all shortest to strictly jth-shortest s-t paths, we define the jth-*shortest path digraph*, denoted by $D^j(G) = (V^j, A^j)$ of G, where an arc $\overrightarrow{(u,v)} \in A^j$ if and only if there exists a strictly ith-shortest s-t path p in G, $1 \leq i \leq j$, such that $(u,v) \in p$; all isolated vertices are eliminated in V^j. The previous studies [10,12,13] investigated the strictly second-shortest path problem (i.e., *next-to-shortest path*) based on the *shortest path digraph*, i.e., $D^1(G) = (V^1, A^1)$. Notably, $D^1(G)$ is acyclic and can be constructed in $O(|V|^2)$ time [10,12]. Moreover, for every vertex $v \neq s$, $d^1(s,v)$ and $S^1(v)$ can also be obtained. Kao et al.'s algorithm [10] can derive the cost of a strictly second-shortest s-t path, $d^2(s,t)$ in $O(|V|^2)$ time, given the shortest path digraph $D^1(G)$.

```
procedure Find-2nd-Shortest($G, s, t, D^1(G)$)    ▷ find each $d^2(s,v)$ and $S^2(v)$
    Use Kao et al.'s algorithm to compute $d^2(s,t)$ based on $D^1(G)$;
    Initialize a queue $Q = \{t\}$, $D^2(G) = (V^1, \emptyset)$, and $S^2(v) = \emptyset$, $\forall v \in V$;
    while the queue $Q \neq \emptyset$ do
        $u \leftarrow$ Dequeue($Q$);
        for each neighbor $w$ adjacent to $u$ and $\overrightarrow{(u,w)} \notin A^2$ do ▷ breath-first search
            if $d^2(s,u) - d(w,u) \geq d^1(s,w)$ then
                $A^2 \leftarrow A^2 \cup \overrightarrow{(w,u)}$ and $S^2(u) \leftarrow S^2(u) \cup \{w\}$;
                if $d^2(s,u) - d(w,u) > d^1(s,w)$ or
                  $d^2(s,u) - d(w,u) = d^1(s,w)$ and $w \in V \setminus V^1$ then
                    $d^2(s,w) \leftarrow d^2(s,u) - d(w,u)$ and $V^2 \leftarrow V^2 \cup \{w\}$;
                end if
                Enqueue($Q, w$) if $w$ is not in the queue $Q$;
            end if
        end for
    end while
end procedure
```

We present the Find-2nd-Shortest procedure to search for all strictly second-shortest simple s-t paths and to construct the representation $D^2(G) = (V^2, A^2)$. The main step of this procedure is just a breadth-first search. We start at t and traverse backward all other vertices until s, and to determine whether each vertex lies on a strictly second-shortest s-t path. The correctness of the procedure follows from the optimal substructure property; moreover, the resulting graph $D^2(G)$, which is a directed acyclic graph from s to t, can represent an apex tree.

Note that in an apex tree, there is a unique path from each vertex to t, while in $D^2(G)$ there may exist multiple paths to t. However, this is not an obstacle to use of our algorithm in $D^2(G)$ if we obtain a random path from t to s. The traveller randomly selects one of incoming edges (or outgoing edges) to a vertex, and traverses the edge. The traveller excludes already visited vertices when she randomly selects a path.

In addition, the procedure can be generalized to finding all strictly third-shortest to μth-shortest simple s-t paths, provided that the cost $d^j(s,t)$ is given, $3 \leq j \leq \mu$; note that the property holds between every strictly $(j-1)$th-shortest and jth-shortest paths. However, Kao et al.'s method cannot be straightforwardly extended to compute $d^j(s,t)$, $j \geq 3$. We leave it as an open problem to find an efficient way to derive $d^j(s,t)$, $j \geq 3$.

Clearly, the data structure $D^2(G)$ can be constructed in polynomial time and linear space. In each iteration of the loop in the *GRR* algorithm, we find all shortest to strictly second-shortest s-t paths whose cost is assumed to be at most $d^2(s,t) = (1+\alpha')d(s,t)$, for some $\alpha' > 0$. The competitive ratio would be $(\frac{2+\alpha'}{1+\alpha'})k + 1 < 2k+1$, and therefore, the *GRR* algorithm can surpass the deterministic lower bound for the k-CTP in polynomial time.

```
procedure Find-Multiple-Shortest($G, s, t, \mu$)
    Duplicate each edge in the input graph $G = (V, E)$ to make $G$ directed;
    Let $S_0 = \{t\}$, $D_V(t) = \{0\}$; $D_E(\overrightarrow{(u,v)}) = \{\infty\}$ $\overrightarrow{(u,v)} \in E$; $D_V(v) = \{\infty\}$ $v \in V$;
    for $i = 0$ to $\mu|E|$ do
        Let $S_{i+1} = \emptyset$;
        for each $v \in S_i$ do
            for each $e = \overrightarrow{(u,v)} \in E$ do
                $S_{i+1} \leftarrow S_{i+1} \cup \{u\}$;
                Let $L = \{d(u,v) + \ell \mid \ell \in D_V(v)\}$;
                Let $D_E(\overrightarrow{(u,v)})$ be the set of the smallest $\mu$ values in $D_E(\overrightarrow{(u,v)}) \cup L$;
            end for
        end for
        for each $u \in V$ do
            for each $e = \overrightarrow{(u,v)} \in E$ do
                Let $D_V(u)$ be the set of the smallest $\mu$ values in $D_V(u) \cup D_E(\overrightarrow{(u,v)})$;
            end for
        end for
    end for
end procedure
```

5.2 Strictly μth-Shortest Paths

We compute the sets of possibly non-simple strictly jth-shortest paths for $j = 1, 2, \ldots, \mu$ using the Find-Multiple-Shortest procedure. This is again a breadth-first search traversing from t backward to s. We have some notation. In the ith iteration, we keep a set S_i of vertices which are reached from t using exactly i edges, and initially, $S_0 = \{t\}$. Let a set $D_V(u)$ store shortest to strictly μth-shortest distances from u to t; that is, $D_V(s) = \{d^1(s,t), d^2(s,t), \ldots, d^\mu(s,t)\}$. Let two sets $D_E(\overrightarrow{(u,v)})$ and $D_E(\overrightarrow{(v,u)})$ for an edge $e = (u,v)$ store distances from u to t and distances from v to t, respectively, using the edge e. Precisely, $\ell \in D_E(\overrightarrow{(u,v)})$ for an edge $e = (u,v)$ if and only if there is a u-t path of distance $\ell \in D_V(u)$ through the edge e. Note that the breadth-first search traverses a vertex multiple times. It is enough to repeat $\mu|E|$ times the iteration because a strictly μth-shortest s-t path uses at most $\mu|E|$ edges.

```
procedure Implicit-Representation($G, s, t, L$)
    Let $i = 0$, $S_i = \{s\}$, $V' = \{s\}$, $E' = \emptyset$, $L(v) = \emptyset\ \forall v \in V$, and $L(s) = D_V(s) \cap L$;
    while $S_i \neq \emptyset$ do
        Let $S_{i+1} = \emptyset$;
        for each $u \in S_i$ do
            for each $e = \overrightarrow{(u,v)} \in E$ do
                Let $L = \{\ell - d(u,v) \mid \ell \in L(u)\}$;
                $L(v) \leftarrow L(v) \cup (D_V(v) \cap L)$;
                if $L(v) \neq \emptyset$ then
                    $S_{i+1} \leftarrow S_{i+1} \cup \{v\}$, $V' \leftarrow V' \cup \{v\}$, and $E' \leftarrow E' \cup \{e\}$;
                end if
            end for
        end for
        $i \leftarrow i + 1$;
    end while
end procedure
```

Based on this data structure, we can construct a graph $G' = (V', E')$ which represents all (possibly non-simple) s-t paths whose distances are in a given set $L \subseteq \{d^1(s,t), \ldots, d^\mu(s,t)\}$ using the Implicit-Representation procedure. Starting from s, we traverse an edge $e = (u,v)$ only if the set $D_E(\overrightarrow{(u,v)})$ or $D_E(\overrightarrow{(v,u)})$ of distances to t contains the given set of distances. Similarly, in the ith iteration, we keep a set S_i of vertices which are reached from s using exactly i edges, and initially, $S_0 = \{s\}$. Let $L(v)$ be the set of distances from v to t for each vertex $v \in V'$. Precisely, if $\ell \in L(v)$, then $\ell \in D_V(v)$ and there is a v-t path of cost ℓ. The number of iterations of the procedure is also at most $\mu|E|$.

The resulting graph G' is a directed graph from s to t. However because there may exist non-simple paths, the graph may have cycles. We can convert G' into a directed acyclic graph G'' as follows. For each $\ell \in L(v)$, we create a vertex v_ℓ, and for each pair of vertices v_ℓ and $w_{\ell'}$, we create an edge from v_ℓ to $w_{\ell'}$ if and only if $(v,w) \in E'$ and $\ell - d(v,w) = \ell'$. The graph G' is converted so that vertices can be duplicated to eliminate multiple edges. Each vertex is identified with the name of the vertex in the original graph and the unique distance to t. Then the new graph containing those newly inserted vertices and edges becomes a directed acyclic graph. Therefore, the graph G'' can represent an apex tree.

In summary, the simple implicit representation has at most $\mu|V|$ vertices and at most $\mu|E|$ edges, and it can be constructed in $O(\mu^2|E|^2)$ time. Hence, we can obtain a set of paths whose distances are at most $(1+\alpha)d(s,t)$ by setting μ to be the summation of distances of all edges. The number of iterations of the loop in the *GRR* algorithm is at most k and thus the whole process takes $O(k\mu^2|E|^2)$ time in the worst case; that is, the proposed $\big(1+\frac{\sqrt{2}}{2}\big)k+1$-competitive randomized algorithm runs in pseudo-polynomial time.

References

1. Bar-Noy, A., Schieber, B.: The Canadian traveller problem. In: Proc. of the 2nd ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 261–270 (1991)
2. Ben-David, S., Borodin, A.: A new measure for the study of online algorithms. Algorithmica 11(1), 73–91 (1994)
3. Bender, M., Westphal, S.: An optimal randomized online algorithm for the k-Canadian traveller problem on node-disjoint paths. Journal of Comb. Opt. (2013) (published online)
4. Borodin, A., El-Yaniv, R.: Online computation and competitive analysis. Cambridge University Press, Cambridge (1998)
5. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik 1(1), 269–271 (1959)
6. Eppstein, D.: Finding the k shortest paths. SIAM Journal on Computing 28(2), 652–673 (1998)
7. Frieder, A., Roditty, L.: An experimental study on approximating K shortest simple paths. In: Demetrescu, C., Halldórsson, M.M. (eds.) ESA 2011. LNCS, vol. 6942, pp. 433–444. Springer, Heidelberg (2011)
8. Hershberger, J., Maxel, M., Suri, S.: Finding the k shortest simple paths: A new algorithm and its implementation. ACM Trans. on Algorithms 3(4), 45 (2007)
9. Huang, Y., Liao, C.S.: The Canadian traveller problem revisited. In: Chao, K.-M., Hsu, T.-s., Lee, D.-T. (eds.) ISAAC 2012. LNCS, vol. 7676, pp. 352–361. Springer, Heidelberg (2012)
10. Kao, K.H., Chang, J.M., Wang, Y.L., Juan, J.S.T.: A quadratic algorithm for finding next-to-shortest paths in graphs. Algorithmica 61, 402–418 (2011)
11. Katoh, N., Ibaraki, T., Mine, H.: An efficient algorithm for k shortest simple paths. Networks 12(4), 411–427 (1982)
12. Krasikov, I., Noble, S.D.: Finding next-to-shortest paths in a graph. Information Processing Letters 92, 117–119 (2004)
13. Lalgudi, K.N., Papaefthymiou, M.C.: Computing strictly-second shortest paths. Information Processing Letters 63, 177–181 (1997)
14. Papadimitriou, C.H., Yannakakis, M.: Shortest paths without a map. Theoretical Computer Science 84(1), 127–150 (1991)
15. Sleator, D., Tarjan, R.E.: Amortized efficiency of list update and paging rules. Communications of the ACM 28, 202–208 (1985)
16. Westphal, S.: A note on the k-Canadian traveller problem. Information Processing Letters 106, 87–89 (2008)
17. Xu, Y.F., Hu, M.L., Su, B., Zhu, B.H., Zhu, Z.J.: The Canadian traveller problem and its competitive analysis. Journal of Comb. Opt. 18, 195–205 (2009)

Efficiency Guarantees in Auctions with Budgets

Shahar Dobzinski[1] and Renato Paes Leme[2]

[1] Weizmann Institute
dobzin@gmail.com
[2] Google Research NYC
renatoppl@google.com

Abstract. In settings where players have limited access to liquidity, represented in the form of budget constraints, efficiency maximization has proven to be a challenging goal. In particular, the social welfare cannot be approximated by a better factor than the number of players. Therefore, the literature has mainly resorted to Pareto-efficiency as a way to achieve efficiency in such settings. While successful in some important scenarios, in many settings it is known that either exactly one truthful auction that always outputs a Pareto-efficient solution, or that no truthful mechanism always outputs a Pareto-efficient outcome. Moreover, since Pareto-efficiency is a binary property (is either satisfied or not), it cannot be circumvented as usual by considering approximations. To overcome impossibilities in important setting such as multi-unit auctions with decreasing marginal values and private budgets, we propose a new notion of efficiency, which we call *liquid welfare*. This is the maximum amount of revenue an omniscient seller would be able to extract from a certain instance. For the aforementioned setting, we give a deterministic $O(\log n)$-approximation for the liquid welfare in this setting.

We also study the liquid welfare in the traditional setting of additive values and public budgets. We present two different auctions that achieve a 2-approximation to the new objective. Moreover, we show that no truthful algorithm can guarantee an approximation factor better than 4/3 with respect to the liquid welfare.

1 Introduction

Auctions started being regularly held in Europe around the middle of the 18th century - originally being used to sell antiques and artwork in English auction houses and agricultural produce such as flowers in the Netherlands [21]. In the last decades of the 20th century, however, auctions started being deployed in an incredibly larger scale: privatization auctions in Eastern Europe, sale of spectrum in the US, auctions for rights to explore natural resources, among others. The new scale brought various new challenges – among them, how to deal with the disconnect between players *willingness to pay* (value) and *ability to pay* (budget). Studying the FCC auctions, Bulow, Levin and Milgrom [7] observe the following:

> *"According to our theory, it is bidders budgets, as opposed to their license values, that determine average prices in a spectrum auction."*

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 392–404, 2014.

In fact, in any setting where the magnitude of the financial transactions is very large, budgets play a major role in the auction. One of the prime examples in internet advertisement – the choice of budget to spend is the first question asked to advertisers in the interface of Google Adwords, even before they are asked bids or keywords. Much work has been devoted to understanding the impact of budget constraints in sponsored search auctions [1,15,10]. For a more extensive discussion on the source of financial constraints, we refer to Che and Gale [9].

Despite being widely relevant in practice, it is not clear how to design *efficient* auctions in the presence of budget constraints. When efficiency means social welfare maximization, a folklore result states that, when n is the number of players, no incentive-compatible auction can do better than an n-approximation. Even weaker notions such as Pareto efficiency, are impossible to be achieved through incentive compatible mechanisms in important settings such as *private budgets* and *decreasing marginal values*. Our main goal in this paper is to search for alternative notions of efficiency that are more suitable for budgeted settings.

Due to its practical relevance, many theoretical investigations have been devoted to analyzing auctions for budget constrained agents. The impact of budgets on the revenue of standard auctions was analyzed in Che and Gale [9] and Benoit and Krishna [4], and mechanisms that optimize (exactly or approximately) revenue were designed by Laffont and Robert [17], Malakhov and Vohra [19], Pai and Vohra [23], Borgs et al [6] and Chawla et al [8].

When the objective is welfare efficiency rather then revenue, the literature has early stumbled upon impossibility results. The traditional social welfare measure, the sum of player's values for their outcomes, is known to be very poorly approximable under budget constraints, even when budgets are known to the auctioneer. This motivates the search for truthful auctions satisfying weaker notions of efficiency. Dobzinski, Lavi and Nisan [12] suggest studying Pareto-efficient auctions: the outcome of an auction is *Pareto-efficient* if there is no other outcome (allocation and payments) where no agent (bidders or auctioneer) is worse-off and at least one agent is better off. When budgets are public, they give a truthful and Pareto-efficient multi-unit auction based on Ausubel's clinching framework [2]. Furthermore, they show that this auction is the unique truthful auction that always produces Pareto-efficient solution. A sequence of follow-ups designed Pareto-efficient auctions for budget to different settings: Bhattacharya et al [5], Fiat et al [14], Colini-Baldeschi et al [10] and Goel et al [15,16].

Beyond Pareto Efficiency? Pareto efficiency has therefore emerged as the de-facto standard for measuring efficiency when bidders are budget constrained. Indeed, most of the aforementioned papers provide positive results by offering new auctions. Yet, this is far from being a complete solution from both theoretical and practical point of views. We now elaborate on this issue.

In a sense, the uniqueness result of Dobzinski et al [12] for public budgets – that shows that the clinching auction is the only Pareto efficient, truthful auction – may be viewed negatively. Rare are the cases in practice in which the designer sole goal is to obtain a Pareto efficient allocation. A more realistic view is that theory provides the designer a toolbox of complementary methods and techniques designed to obtain various different goals (efficiency, revenue maximization, fairness, computational efficiency, etc.) and balance between them. The composition of these tools as well as their adaptation to the specifics of the setting and fine tuning is the designer's task. A uniqueness result – although extremely appealing from a pure theoretical perspective – implies that the designer's toolbox contains only one tool, obviously an undesirable scenario.

Furthermore, although from a technical point of view the analysis of the existing algorithms is very challenging and the proof techniques are quite unique to each setting, the auctions themselves are all variants of the same basic clinching idea of Ausubel [2]. Again, it is obviously preferable to have more than just one bunny in the hat that will help us design auctions for these important settings.

The situation is obviously even more severe in more complicated settings, where even this lonely bunny is not available. For example, for private budgets and additive multi-unit auctions, an impossibility was given by Dobzinski et al [12], for heterogenous items and public budgets by Fiat, Leonardi, Saia and Sankowski [14] and Dütting, Henzinger and Starnberger [13] and for as multi-unit auctions with subadditive valuations and public budgets by Goel, Mirrokni and Paes Leme [15] and Lavi and May [18].

Alternatives to Pareto Efficiency. Our main goal is to research alternatives to Pareto efficiency for budget constrained agents. We start by observing that a Pareto efficiency is a binary notion: an allocation is either Pareto efficient or not, and there is no sense of one allocation being "more Pareto efficient" than the other. This is in contrast with efficiency in quasi linear environments where the traditional welfare objective induces a total order on the allocations.

The main goal of this paper is to suggest a new measure of efficiency for budgeted settings. The desiderata for this measure are: (i) it is quantifiable, i.e., attaches a value to each outcome; (ii) is achievable, i.e., can be approximated by truthful mechanisms and (iii) allows different designs that approximate welfare.

The measure we propose is called the *liquid welfare*. Before defining it, we give a revenue-motivated definition of the traditional social welfare in unbudgeted settings and show how it naturally generalizes to budgeted settings. One can view the traditional welfare of a certain outcome as the maximum revenue an omniscient seller can obtain from that outcome. If each agent i has value $v_i(x_i)$ for a certain outcome x_i, the omniscient seller can extract revenue arbitrarily close to $\sum_i v_i(x_i)$ by offering this outcome to each player i for price $v_i(x_i) - \epsilon$. This definition generalizes naturally to budgeted settings. Given an outcome, x_i, the *willingness-to-pay* of agent i is $v_i(x_i)$, which is the maximum he would give for this outcome in case he had unlimited resources. His *ability-to-pay*, however, is B_i, which is the maximum amount of money available to him. We define his *admissibility-to-pay* as the maximum value he would admit to pay for this outcome, which is the minimum between his willingness-to-pay and his ability-to-pay. The liquid welfare of a certain outcome is defined as the total admissibility-to-pay. Formally $\bar{\mathbf{W}}(x) = \sum_i \min(v_i(x_i), B_i)$.

An alternative view is as follows: efficiency should be measured only with respect to the funds available to the bidder at the time of the auction, and not the additional liquidity he might gain after receiving the goods he won. The liquid welfare objective frees the auctioneer from considering the hypothetical use the bidders will make of the items they win in the auction, and thus can focus only on the resources available to them at the time of the auction.

This objective satisfies our first requirement: it associates each outcome with an objective measure. Also, it is achievable. In fact, the clinching auction [12], which is the base for all auction achieving Pareto-efficient outcomes for budgeted settings, provide a 2-approximation for the liquid welfare objective. To show that this allows flexibility in the design, we show a different auction that also provides a 2-approximation and reveals a connection between our liquid welfare objective and the notion of market equilibrium.

It is appropriate to discuss the applicability and limitations of the liquid welfare objective. We start by illustrating a setting for which it is *not* applicable. If one were to auction hospital beds or access to doctors, it would be morally repugnant to privilege players based on their ability-to-pay. Therefore, we are not interested in claiming that the liquid welfare objective is the only alternative to Pareto efficiency, but rather argue that in *some* settings it produces reasonable results. Developing other notions of efficiency is an important future direction.

Yet, in many settings capping the welfare of the agents by their budgets makes perfect sense. Consider designing a market like internet advertising which aims at a good mix of good efficiency and revenue. In practice, players that bring more money to the market provide health to the market and improve efficiency. In real markets, there are practices to encourage wealthier players to enter the market. Therefore, privileging such players in the objective is somewhat natural.

An interesting question is whether one can have a truthful mechanism for additive valuations with public budgets that provides an approximation ratio better than 2. We show a lower bound of $\frac{4}{3}$. Closing the gap remains an open question, but we do show that for the special case of 2 players with identical public budgets there is a truthful auction that provides a matching upper bound.

We then move on to consider a setting in which truthful auction that always output Pareto-efficient solution do not exist: multi-unit auctions with decreasing marginal valuations and private budgets. For this setting we borrow ideas from Bartal, Gonen and Nisan [3] and provide a deterministic $O(\log n)$ approximation to the liquid welfare. This can be adapted to the case of subadditive valuations with an approximation of $O(\log^2 n)$ and to indivisible goods with $O(\log m)$-approximation where m is the number of goods.

Related Work. We have already surveyed results designing mechanisms for budget-constrained agents. We now focus on surveying results directly related to our efficiency measure and to our philosophical approach to efficiency maximization. As far as we know, the liquid welfare was first appeared in Chawla et al. [8] as an implicit upper bound on the revenue that a mechanism can extract.

Independently and simultaneously, two other approaches were proposed to provide quantitative guarantees for budgeted settings. Devanur, Ha and Hartline [11] show that the welfare of the clinching auction is a 2-approximation to the welfare of the best envy-free equilibrium. Their approach, however, is restricted to settings with *common budgets*, i.e., all agents have the same budget.

Syrgkanis and Tardos [24] leave the realm of truthful mechanisms and study the set of Nash and Bayes-Nash equilibria of simple mechanisms. For a wide class of mechanisms they show that the *traditional* welfare in equilibrium of such mechanism is a constant fraction of the optimal liquid welfare objective (which they call *effective welfare*). Their approach differs from ours in two ways: first they study auctions in equilibrium while we focus on incentive compatible auctions. Second, the guarantee in their mechanism is that the welfare of the allocation obtained is always greater than some fraction of the liquid welfare. The guarantee of our mechanisms is stronger: we construct mechanisms in which the *liquid welfare* is always greater than some fraction of the liquid welfare, which implies in particular that the welfare is greater than some fraction of the liquid welfare (since the welfare of an allocation is at least its liquid welfare).

Summary of Our Results. In this paper we have proposed to study the liquid welfare. We provided two truthful algorithms that guarante a 2 approximation to this objective for the setting of multi-unit auctions with public budgets. For the harder setting of multi-unit auctions with subadditive valuations and private budgets we provided a truthful $O(\log^2 n)$-approximation algorithm. For submodular bidders, the same mechanism provides an $O(\log n)$ approximation.

The main problem that we leave open is to determine whether there is a constant-approximation mechanism for multi unit auctions with private budgets. This is even open if all valuations are additive. More generally, are there truthful algorithms that provides a good approximation for *combinatorial* auctions? On top of that, notice that computational issues might come into play: while all of the constructions that we present in this paper happen to be computationally efficient, there might be a gap between the power of truthful algorithms in general and the power of computationally efficient truthful algorithms.

2 Preliminaries

2.1 Environments of Interest and Auction Basics

We consider n players and a set X of outcomes (also called environment). For each player, let $v_i : X \to \mathbb{R}_+$ be the valuation function for player i. We consider that agents are budgeted quasi-linear, i.e., each agent i has a budget B_i and for an outcome x and for payments $\pi_1, \ldots, \pi_n$, the utility of agent i is: $u_i = v_i(x_i) - \pi_i$ if $\pi_i \leq B_i$ and $-\infty$ o.w. Below, we list a set of environments we are interested:

1. *Divisible-multi-unit auctions and additive bidders:* $X = \{(x_1, \ldots, x_n); \sum_i x_i = s\}$ for some constant s and $v_i(x_i) = v_i \cdot x_i$, so we can represent the valuation function of each agent by a single real number $v_i \geq 0$.

2. *Divisible-multi-unit auctions with decreasing marginal bidders:* $X = \{(x_1, \ldots, x_n); \sum_i x_i = s\}$ for some constant s and $v_i : \mathbb{R}_+ \rightarrow \mathbb{R}_+$ is a monotone non-decreasing concave function. A generalization of decreasing marginal valuations is subadditive valuations, i.e., $v_i(x_1 + x_2) \leq v_i(x_1) + v_i(x_2)$ for every x_1, x_2.
3. 0/1 *environments:* $X \subseteq \{0,1\}^n$ and $v_i(x_i) = v_i$ if $x_i = 1$ and $v_i(x_i) = 0$ otherwise. Again the valuation is represented by a single $v_i \geq 0$.

An auction for a particular setting elicits the valuations of the players and budgets $B_1, \ldots, B_n$ and outputs an outcome $x \in X$ and payments $\pi_1, \ldots, \pi_n$ for each agents respecting budgets, i.e., such that $\pi_i \leq B_i$ for each agent. We will distinguish between *public budgets* and *private budgets* mechanisms. In the former, the auctioneer has access to the true budget of each agent[1]. In the later case, agents need to be incentivized to report their true budget. In either case, the valuations of each agent are private. We will focus on designing mechanisms that are incentive compatible (a.k.a. truthful), i.e., are such that agents utilities are maximized once they report their true value in the public budget case and their true value and budget in the private budget case. We will also require mechanisms to be individually rational, i.e., agents always derive non-negative utility upon bidding their true value.

In the case of divisible multi-unit auctions and additive bidders, the valuations can be represented by real numbers, So we can see the auctions as a pair of functions $x : \mathbb{R}^n_+ \times \mathbb{R}^n_+ \rightarrow \mathbb{R}^n_+$ and $\pi : \mathbb{R}^n_+ \times \mathbb{R}^n_+ \rightarrow \mathbb{R}^n_+$ that map (v, B) to a vector of allocations $x(v, B) \in \mathbb{R}^n_+$ and a vector of payments $\pi(v, B) \in \mathbb{R}^n_+$. The set of functions that induce incentive compatible and individually rational auctions are characterized by Myerson's Lemma:

Lemma 1 (Myerson [22]). *A pair of functions (x, π) define an incentive-compatible and individually rational auction iff (i) for each v_{-i}, $x_i(v_i, v_{-i})$ is monotone non-decreasing in v_i and (ii) the payments are such that: $\pi_i(v_i, v_{-i}) = v_i \cdot x_i(v_i, v_{-i}) - \int_0^{v_i} x_i(u, v_{-i}) du$.*

2.2 Efficiency Measures

The traditional efficiency measure in mechanism design is the *social welfare* which associates for each outcome x, the objective: $\mathbf{W}(x) = \sum_i v_i(x)$. It is known that one cannot even approximate the optimal welfare in budgeted settings in an incentive-compatible way, even if the budgets are known and equal. The result is folklore. We sketch the proof in the full version for completeness.

Lemma 2 (Folklore). *Consider the divisible-multi-unit auctions and additive bidders. There is no α-approximate, incentive compatible and individually rational mechanism $x(v), \pi(v)$ with $\alpha < n$. For $\alpha = n$ there is the mechanism that allocates the item at random to one player and charges nothing.*

[1] Most of the literature on auctions for budgeted settings [12,14,15,10,16] falls in this category, including classical references (Laffont and Robert [17] and Maskin [20]).

Due to impossibility results of this flavor, efficiency was mainly achieved in the literature through *Pareto efficiency*. We say that an outcome (x, π) with $x \in X$ and $\pi_i \leq B_i$ is Pareto-efficient if there is no alternative outcome where the utility of all the agents involved (including the auctioneer, being his utility the revenue $\sum_i \pi_i$) does not decrease and at least one agent improves. Formally, (x, π) is Pareto optimal iff there is no (x', π'), $x' \in X$, $\pi'_i \leq B_i$ such that:

$$u'_i = v_i \cdot x'_i - \pi'_i \geq u_i = v_i \cdot x_i - \pi_i, \forall i \text{ and } \textstyle\sum_i \pi'_i \geq \sum_i \pi_i \text{ and } \sum_i v_i x'_i > \sum_i v_i x_i$$

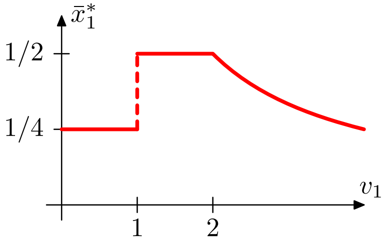
In particular, if the budgets are infinity (or simply very large), the only Pareto-optimal outcomes are those maximizing social welfare. For divisible-multi-unit auctions with additive bidders, this is achieved by the Adaptive Clinching Auction of Dobzinski, Lavi and Nisan [12]. Moreover, the authors show that this is the only incentive-compatible, individually-rational auction that achieves Pareto-optimal outcomes. The auction is further analyzed in Bhattacharya et al [5] and Goel et al [16]. In this paper we propose the *liquid welfare* objective:

Definition 1 (Liquid Welfare). *In a budgeted setting, we define the liquid welfare associated with outcome $x \in X$ by $\bar{\mathbf{W}}(x) = \sum_i \min\{v_i(x), B_i\}$.*

We will refer to the optimal liquid welfare as $\bar{\mathbf{W}}^* = \max_{x \in X} \bar{\mathbf{W}}(x)$. It is instructive yet straightforward to see that:

Lemma 3. *For divisible-multi-unit auctions and additive bidders, the optimal liquid welfare $\bar{\mathbf{W}}^*$ occurs for $\bar{x}^*_i = \min\left(\frac{B_i}{v_i}, [1 - \sum_{j<i} \bar{x}^*_j]^+\right)$ where players are sorted in non-increasing order of value, i.e., $v_1 \geq v_2 \geq \ldots \geq v_n$.*

An easy observation is that the optimal allocation for $\bar{\mathbf{W}}^*$ is not monotone in v_i, and hence cannot be implemented truthfully. For example, consider 3 agents with values $v_1 = v, v_2 = 1, v_3 = 2$ and budgets $B_1 = 1, B_2 = \frac{1}{4}, B_3 = 1$. Now, notice that $\bar{x}^*_1(v_1)$ is not monotone in v_1 as depicted in Figure 1.



$$\bar{x}^*_1(v_1, v_{-1}) = \begin{cases} 1/4, & 0 \leq v_1 \leq 1 \\ 1/2, & 1 \leq v_1 \leq 2 \\ 1/v_1, & 2 \leq v_1 \end{cases}$$

Fig. 1. Depiction of the first component of $\bar{x}^* = \text{argmax}_x \bar{\mathbf{W}}(x)$ for a 3 agent instance with $v = (v_1, 1, 2)$ and $B = (1, 1/4, 1)$. The figure highlights the non-monotonicity of the optimal solution $\bar{x}^*(v)$.

2.3 VCG and the Liquid Welfare Objective

The reader might suspect, however, that a modification of VCG might take care of optimizing the liquid welfare benchmark. This is indeed true for a couple of very simple settings. For example, for selling one indivisible item, a simple Vickrey auction on modified values: $\bar{v}_i = \min\{v_i, B_i\}$ provides a truthful mechanism that exactly optimizes the liquid welfare objective. More generally:

Theorem 2 (0/1 Environments). *Given a* $0/1$*-environment* $X \subseteq \{0,1\}^n$ *with valuations* $v_i(x) = v_i$ *if* $x_i = 1$ *and zero otherwise. Then running VCG on modified values* $\bar{v}_i = \min\{v_i, B_i\}$ *is incentive compatible and exactly optimized the liquid welfare objective* $\bar{\mathbf{W}}^*$*.*

The proof is trivial. This slightly generalizes to other simple environments of interest, for example, matching markets, where there are n agents and n indivible items and each agent i has a value v_{ij} for item j and possible outcomes are perfect matchings. Running VCG on $\bar{v}_{ij} = \min\{v_{ij}, B_i\}$ provides an incentive compatible mechanism that also exactly approximated $\bar{\mathbf{W}}^*$.

This technique, however, does not generalize past those few special cases as we show in the full version.

3 A First 2-Approximation: The Clinching Auction

In the previous section we defined our proposal for an efficiency measure in budgeted settings: the liquid welfare objective $\bar{\mathbf{W}}$. The second item in the desiderata for a new efficiency measure is that it is achievable, i.e., it could be optimized or well-approximated by an incentive compatible mechanism. In this section we show, for the setting of divisible multi-unit auctions with additive bidders, a mechanism that provides a 2-approximation for the liquid welfare, while still producing Pareto-efficient outcomes. The mechanism we use is the Adaptive Clinching Auction [12,5,16]. In the next section we provide a different truthful auction that is also a 2-approximation, and show that with respect to the liquid welfare the new auction is better on an instance-by-instance basis.

The clinching auction can be described by means of an ascending price clock procedure. The auction starts with the good unallocated and for every price p (considered in increasing order and in ϵ increments), the demand of each agent is computed, i.e., the maximum amount of the good an agent would like to be allocated at any given price. The amount an agent can *clinch* at price p is the amount leftover of the good minus the amount demanded by all other agents. At any given price, an agent is allocated his *clinched* amount at the current price.

In the full version we formally describe the clinching auction and introduce the concept of the *clinching interval* – which correspond to the price interval in which agents actively acquire goods. Finally. we prove our main result:

Theorem 3. *The clinching auction is a* 2*-approximation to the liquid welfare objective. That is, given* n *agents with values per unit* v_i *and budgets* B_i*, let* x, π *be the outcome of the clinching auction for such input. Then,* $\bar{\mathbf{W}}(x) \geq \frac{1}{2}\bar{\mathbf{W}}^*$*.*

In the full version we show that the bound proved in Theorem 3 is tight. Also, the following corollary about the revenue of the clinching auction follows from our proof:

Corollary 1 (Revenue). *If the clinching auction allocates items to more than one player, then its revenue is at least* $\frac{1}{2} \cdot \bar{\mathbf{W}}^*$.

4 A 2-Approximation via Market Equilibrium

We defined a quantifiable measure of efficiency (Section 2) and showed it can be approximated by an incentive-compatible mechanism (Section 3). The remaining item in the list of desiderata was to show that our efficiency measure allows for different designs. Here we show that we have "an extra bunny in the hat", an auction that also achieves a 2-approximation to the liquid welfare objective and is *not* based on Ausubel's clinching technique. Instead, it is based on the concept of Market Equilibrium.

Borrowing inspiration from general equilibrium theory, consider a market with n buyers each endowed with B_i dollars and willing to pay v_i per unit for a certain divisible good. This is the special case where there is only one product in the market. In this case, a price p is called a *market clearing price* if each buyer can be assigned an optimal basket of goods (in the particular of a single product, an optimal amount of the good) such that there is no surplus or deficiency of any good. Observe that there is one such price and that allocations can be computed once the price is found. Our Uniform Price Auction simply computes the market clearing price and allocates according to it. This defines the allocation. The payments are computed using the Myerson's formula for this allocation and happen to be different than the clearing price.

Definition 4 (Uniform Price Auction). *Consider n agents with values $v_1 \geq \ldots \geq v_n$ (i.e., ordered without loss of generality) and budgets B_i. Consider the auction that allocates one unit of a divisible good in the following way: let k be the maximum integer such that $\sum_{j=1}^k B_j \leq v_k$, then:*

- *Case I: if $\sum_{j=1}^k B_j > v_{k+1}$ allocate $x_i = \frac{B_i}{\sum_{j=1}^k B_j}$ for $i = 1, \ldots, k$ and nothing for the remaining players.*
- *Case II: if $\sum_{j=1}^k B_j \leq v_{k+1}$ allocate $x_i = \frac{B_i}{v_{k+1}}$ for $i = 1, \ldots, k$, $x_{k+1} = 1 - \sum_{j=1}^k x_j$ and nothing for the remaining players.*

Payments are defined through Myerson's integral (Lemma 1).

Case I corresponds to the case where the market clearing price of the Fisher Market instance is $p = \sum_{j=1}^k B_j$. Case II coresponds to the case where the Market clearing price is $p = v_{k+1}$. First we show that this auction induces an incentive-compatible auction that does not exceed the budgets of the agents – and thus is a valid auction for this setting. Then we show that it is a 2-approximation to the liquid welfare benchmark. Proofs are in the full version.

Lemma 4 (Monotonicity). *The allocation function of the Uniform Price Auction is monotone, i.e., $v_i \mapsto x_i(v_i, v_{-i})$ is non-decreasing.*

Lemma 5 (Budget Feasibility). *The payments that make this auction incentive-compatible do not exceed the budgets.*

Theorem 5. *The Uniform Price Auction is an incentive compatible 2-approximation to the liquid welfare objective.*

The same example used for showing that the analysis for the Clinching Auction was tight can be used for showing that the analysis for the Uniform Price Auction is tight. We discuss it in detail in the full version.

One of the advantages in having a quantifiable measure of efficiency is that we can compare two different outcomes and decide which one is "better". In this section we show that although the worst-case guarantees of the clinching auction and of the uniform-price auction are identical, the liquid welfare of the uniform-price auction is *always* (weakly) dominates that of the clinching auction. We refer to the full version for a proof.

Theorem 6. *Consider n players with valuations $v_1 \geq \ldots \geq v_n$ and budgets $B_1, \ldots, B_n$. Let x^{c} and x^{u} be the outcomes of the Clinching and Uniform Price Auctions respectively. Then: $\bar{\mathbf{W}}(x^{\mathrm{u}}) \geq \bar{\mathbf{W}}(x^{\mathrm{c}})$.*

5 A Lower Bound and Some Matching Upper Bounds

In the previous sections, we showed two different auctions that are incentive compatible 2-approximations to the optimal liquid welfare for the setting of multi-unit auctions with additive valuations. Inthe full version we investigate the limits of the approximability of the liquid welfare. By the observation depicted in Figure 1, it is clear that an exact incentive compatible mechanism is not possible for this setting. First, we present a $\frac{4}{3}$ lower bound and show matching upper bounds for some special cases. We refer to the appendix for the full details.

6 Subadditive Bidders with Private Budgets

Finally, we consider the setting where players have subadditive valuations and private budgets. This is a notoriously hard setting for Pareto-optimality. In fact, considering either subadditive valuations or privated budgets alone already produces an impossibility result for achieving Pareto-efficient outcomes.

We will have one divisible good and each player has a subadditive valuation $v_i : [0,1] \to \mathbb{R}_+$ and a budget B_i. This setting differs from the previously considered in the sense that budgets B_i are private information of the players.

The auction we propose is inspired in a technique by Bartal, Gonen and Nisan [3]. To describe it, we use the following notation: $\bar{v}(x_i) = \min\{v_i(x_i), B_i\}$. Now, consider the following selling procedure:

Definition 7 (Sell-Without-r). *Let r be a player. Consider the following mechanism to sell half the good, to players $i \neq r$ using the information about $\bar{v}_r(\frac{1}{2})$.*

Divide the segment $[0, \frac{1}{2}]$ into $k = 8\log(n)$ parts, each of size $\frac{1}{2k}$. Associate part $i = 1, \ldots, k$ with price per unit $p_i = \frac{2^i}{8}\bar{v}_r(\frac{1}{2})$. Order arbitrarily all players but player r. Each player different than r, in his turn, takes his most profitable (unallocated) subset of $[0, \frac{1}{2}]$ under the specified prices. Players are not allowed to pay more than their budget.

More precisely, let $p : [0, \frac{1}{2}] \to \mathbb{R}_+$ be such that for $x \in [\frac{1}{2k}(i-1), \frac{1}{2k}i]$, $p(x) = p_i = \frac{2^i}{8}\bar{v}_r(\frac{1}{2})$. Now, for $i = 1, \ldots, r-1, r+1, \ldots, n$, let x_i maximize $v_i(x_i) - \int_{z_i}^{z_i+x_i} p(t)dt$ where $z_i = \sum_{j<i} x_j$, conditioned on the payment being below the budget, i.e., $\int_{z_i}^{z_i+x_i} p(t)dt \leq B_i$. Set the payment as: $\pi_i = \int_{z_i}^{z_i+x_i} p(t)dt$.

Sell-Without-r is used in our main construction for this section:

Definition 8 (Estimate-and-Price). *Given one divisible good and n players with valuations $v_i(\cdot)$ and budgets B_i, consider the following auction: let $r_1 = \arg\max_i \bar{v}_i(\frac{1}{2})$ and $r_2 = \arg\max_{i \neq r_1} \bar{v}_i(\frac{1}{2})$. We say that r_1 is the* pivot player. *Let (x, π) be the outcome of Sell-Without-r_1 for players $[n] \setminus r_1$ and let (x', π') be the outcome of Sell-Without-r_2 for players $[n] \setminus r_2$.*

For players $i \neq r_1$, allocate x_i and charge π_i. For r_1 if $v_{r_1}(x'_{r_1}) - \pi'_{r_1} \geq v_{r_1}(\frac{1}{2}) - 2 \cdot \bar{v}_{r_2}(\frac{1}{2})$ allocate him x'_{r_1} and charge π'_{r_1} and if not, allocate $\frac{1}{2}$ and charge $2 \cdot \bar{v}_{r_2}(\frac{1}{2})$.

First, notice that the auction defined above is feasible, since r_1 is allocated at most half of the good and the players in $[n] \setminus r_1$ get allocated at most half of the good. Then we argue that this auction is incentive compatible (in the full version).

Lemma 6. *The Estimate-and-Price auction is incentive compatible for players with private budgets.*

This leads to our main result for submodular and subadditive bidders: we show that the Estimate and Price Auction is a logarithmic approximation for the liquid welfare objective. The details as well as a discussion of extensions of this result can be found in the full version.

Theorem 9. *For submodular bidders, the Estimate-and-Price auction is a truthful $O(\log n)$-approximation to the liquid welfare objective. For subadditive bidders, the same auction is an $O(\log^2 n)$-approximation to the liquid welfare.*

Acknowledgments. The first author is the incumbent of the Lilian and George Lyttle Career Development Chair. His work is supported in part by the I-CORE program of the planning and budgeting committee and the Israel Science Foundation 4/11 and by EU CIG grant 618128.

References

1. Aggarwal, G., Muthukrishnan, S., Pál, D., Pál, M.: General auction mechanism for search advertising. In: WWW, pp. 241–250 (2009)
2. Ausubel, L.M.: An efficient ascending-bid auction for multiple objects. American Economic Review 94 (1997)
3. Bartal, Y., Gonen, R., Nisan, N.: Incentive compatible multi unit combinatorial auctions. In: TARK, pp. 72–87 (2003)
4. Benoit, J.-P., Krishna, V.: Multiple-object auctions with budget constrained bidders. Review of Economic Studies 68(1), 155–179 (2001)
5. Bhattacharya, S., Conitzer, V., Munagala, K., Xia, L.: Incentive compatible budget elicitation in multi-unit auctions. In: SODA, pp. 554–572 (2010)
6. Borgs, C., Chayes, J.T., Immorlica, N., Mahdian, M., Saberi, A.: Multi-unit auctions with budget-constrained bidders. In: ACM EC (2005)
7. Bulow, J., Levin, J., Milgrom, P.: Winning play in spectrum auctions. Working Paper 14765, National Bureau of Economic Research (March 2009)
8. Chawla, S., Malec, D.L., Malekian, A.: Bayesian mechanism design for budget-constrained agents. In: ACM EC, pp. 253–262 (2011)
9. Che, Y.-K., Gale, I.: Standard auctions with financially constrained bidders. Review of Economic Studies 65(1), 1–21 (1998)
10. Colini-Baldeschi, R., Henzinger, M., Leonardi, S., Starnberger, M.: On multiple keyword sponsored search auctions with budgets. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012, Part II. LNCS, vol. 7392, pp. 1–12. Springer, Heidelberg (2012)
11. Devanur, N.R., Ha, B.Q., Hartline, J.D.: Prior-free auctions for budgeted agents. In: EC (2013)
12. Dobzinski, S., Lavi, R., Nisan, N.: Multi-unit auctions with budget limits. Games and Economic Behavior 74(2), 486–503 (2012)
13. Dütting, P., Henzinger, M., Starnberger, M.: Auctions with heterogeneous items and budget limits. In: Goldberg, P.W. (ed.) WINE 2012. LNCS, vol. 7695, pp. 44–57. Springer, Heidelberg (2012)
14. Fiat, A., Leonardi, S., Saia, J., Sankowski, P.: Single valued combinatorial auctions with budgets. In: ACM EC, pp. 223–232 (2011)
15. Goel, G., Mirrokni, V.S., Paes Leme, R.: Polyhedral clinching auctions and the adwords polytope. In: STOC, pp. 107–122 (2012)
16. Goel, G., Mirrokni, V.S., Paes Leme, R.: Clinching auctions with online supply. In: SODA (2013)
17. Laffont, J.-J., Robert, J.: Optimal auction with financially constrained buyers. Economics Letters 52(2), 181–186 (1996)
18. Lavi, R., May, M.: A note on the incompatibility of strategy-proofness and pareto-optimality in quasi-linear settings with public budgets. In: Chen, N., Elkind, E., Koutsoupias, E. (eds.) Internet and Network Economics. LNCS, vol. 7090, pp. 417–417. Springer, Heidelberg (2011)
19. Malakhov, A., Vohra, R.V.: Optimal auctions for asymmetrically budget constrained bidders. Working paper (December 2005)

20. Maskin, E.S.: Auctions, development, and privatization: Efficient auctions with liquidity-constrained buyers. European Economic Review 44(4-6), 667–681 (2000)
21. McAfee, R.P., McMillan, J.: Auctions and bidding. Journal of Economic Literature 25(2), 699–738 (1987)
22. Myerson, R.: Optimal auction design. Mathematics of Operations Research 6(1), 58–73 (1981)
23. Pai, M., Vohra, R.: Optimal auctions with financially constrained bidders. Working Paper (2008)
24. Syrgkanis, V., Tardos, É.: Composable and efficient mechanisms. In: STOC (2013)

Parameterized Complexity of Bandwidth on Trees

Markus Sortland Dregi and Daniel Lokshtanov

Department of Informatics, University of Bergen, Norway

Abstract. The bandwidth of a n-vertex graph G is the smallest integer b such that there exists a bijective function $f : V(G) \to \{1, ..., n\}$, called a layout of G, such that for every edge $uv \in E(G)$, $|f(u) - f(v)| \leq b$. In the BANDWIDTH problem we are given as input a graph G and integer b, and asked whether the bandwidth of G is at most b. We present two results concerning the parameterized complexity of the BANDWIDTH problem on trees.

First we show that an algorithm for BANDWIDTH with running time $f(b)n^{o(b)}$ would violate the Exponential Time Hypothesis, even if the input graphs are restricted to be trees of pathwidth at most two. Our lower bound shows that the classical $2^{O(b)}n^{b+1}$ time algorithm by Saxe [SIAM Journal on Algebraic and Discrete Methods, 1980] is essentially optimal.

Our second result is a polynomial time algorithm that given a tree T and integer b, either correctly concludes that the bandwidth of T is more than b or finds a layout of T of bandwidth at most $b^{O(b)}$. This is the first parameterized approximation algorithm for the bandwidth of trees.

1 Introduction

A layout for a graph G is a bijective function $\alpha : V(G) \to \{1, \ldots, |V(G)|\}$, and the bandwidth of the layout α is the maximum over all edges $uv \in E(G)$ of $|\alpha(u) - \alpha(v)| \leq b$. The bandwidth of G is the smallest integer b such that G has a layout of bandwidth b. In the BANDWIDTH problem we are given as input a graph G and an integer b and the goal is to determine whether the bandwidth of G is at most b. In the optimization variant we are given G and the task is to find a layout with smallest possible bandwidth.

The problem arises in sparse matrix computations, where given an $n \times n$ matrix A and an integer k, the goal is to decide whether there is a permutation matrix P such that PAP^T is a matrix whose all non-zero entries lie within the k diagonals on either side of the main diagonal. Standard matrix operations such as inversion and multiplication as well as Gaussian elimination can be sped up considerably if the input matrix A can be transformed into a matrix PAP^T of small bandwidth [1].

BANDWIDTH is one of the most well-studied NP-complete [2, 3] problems. The problem remains NP-complete even on very restricted subclasses of trees, such as caterpillars of hair length at most 3 [4]. Furthermore, it is NP-hard to approximate the bandwidth within any constant factor, even on trees [5]. The best approximation algorithm for BANDWIDTH on general graphs is by Dungan and Vempala [6], this algorithm has approximation ratio $(\log n)^3$. For trees Gupta [7] gave a slightly better approximation algorithm with ratio $(\log n)^{9/4}$, while for caterpillars a $O(\frac{\log n}{\log \log n})$-approximation [8] can be achieved.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 405–416, 2014.

One could argue that the BANDWIDTH problem is most interesting when the bandwidth of the graph is very small compared to the size of the graph. Indeed, when the bandwidth of G is *constant* the matrix operations discussed above can be implemented in linear time. For each $b \geq 1$ it is possible to recognize the graphs with bandwidth at most b in time $2^{O(b)}n^{b+1}$ using the classical algorithm of Saxe [9]. At this point it is very natural to ask how much Saxe's algorithm can be improved. Our first main result is that assuming the Exponential Time Hypothesis of Impagliazzo, Paturi and Zane [10], no sigificant improvement is possible, even on very restricted subclasses of trees. In particular we show the following theorem.

Theorem 1. *Assuming the Exponential Time Hypothesis there is no $f(b)n^{o(b)}$ time algorithm for* BANDWIDTH *of trees of pathwidth at most* 2.

The proof of Theorem 1 also implies that BANDWIDTH is $W[1]$-hard on trees of pathwidth at most 2 (see [11–13] for an introduction to parameterized complexity).

As a counterweight to the bad news of Theorem 1 we give the first approximation algorithm for BANDWIDTH of trees whose approximation ratio depends only on the bandwidth b, and not on the size of the graph. Specifically we give a polynomial time algorithm that given as input a tree T and integer b either correctly concludes that the bandwidth of T is greater than b or outputs a layout of width at most $b^{O(b)}$. A key subroutine of our algorithm for trees is an approximation algorithm for the bandwidth of caterpillars with ratio $O(b^3)$. Our algorithm for trees outperforms the $(\log n)^{9/4}$-approximation algorithm of Gupta [7] whenever $b = o(\frac{\log \log n}{\log \log \log n})$. Our algorithm is the first *parameterized approximation* algorithm for the BANDIWTH problem on trees, that is an algorithm with approximation ratio $g(b)$ and running time $f(b)n^{O(1)}$. A parameterized approximation algorithm for the closely related TOPOLOGICAL BANDWIDTH problem has been known for a while [14], while the existence of a parameterized approximation algorithm for BANDWIDTH, even on trees was unknown prior to this work.

An interestng aspect of our approximation algorithm is the way we lower bound the bandwidth of the input tree T. It is well known that the bandwidth of a graph G is lower bounded by its *pathwidth*, and by its *local density*[1]. One might wonder how far these lower bounds could be from the true bandwidth of G. It was conjectured that the answer to this question is "not too far", in particular that any graph with pathwidth c_1 and local density c_2 would have bandwidth at most c_3 where c_3 is a constant depending only on c_1 and c_2. Chung and Seymour [15] gave a counterexample to this conjecture by constructing a special kind of trees, called *cantor combs*, with pathwidth 2, local density at most 10, and bandwidth approximately $\frac{\log n}{\log \log n}$. Our approximation algorithm essentially shows that the only structures driving up the bandwidth of a tree are pathwidth, local density and cantor comb-like subgraphs.

Related Work. There is a vast literature on the BANDWIDTH problem. For an example the problem has been extensively studied from the perspective of approximation algoritms [5–8, 16], parameterized complexity [9, 17, 18], polynomial time algorithms on restricted classes of graphs [19–22], and graph theory [15, 23]. We focus here on the study of algorithms for BANDWIDTH for small values of b.

[1] A definition of these notions can be found in the preliminaries.

Following the $2^{O(b)}n^{b+1}$ time algorithm of Saxe [9], published in 1980, there was no progress on algorithms for the recognition of graphs of constant bandwidth. With the advent of parameterized complexity in the late 80's and early 90's [11] it became an intriguing open problem whether one could improve the algorithm of Saxe to remove the dependency on b in the exponent of n, and obtain a $f(b)n^{O(1)}$ time algorithm.

In a seminal paper from 1994, Bodlaender, Fellows, and Hallet [17] proved that a number of layout problems do not admit fixed parameter tractable algorithms unless FPT=W[t] for every $t \geq 1$, a collapse considered by many to be almost as unlikely as P=NP. In the same paper Bodlaender, Fellows, and Hallet [17] claim that their techniques can be used to show that a $f(b)n^{O(1)}$ time algorithm for BANDWIDTH would also imply FPT=W[t] for every $t \geq 1$. Downey and Fellows ([11], page 468) further claim that the techniques of [17] imply that even fixed parameter algorithm for BANDWIDTH *on trees* would yield the same collapse. Unfortunately a full version of [17] substantiating these claims is yet to appear.

Outline. In Section 2 we define notations and give the necessary background. In Section 3 we give an outline of the proof our algorithmic results and in Section 4 we give some concluding remarks and open problems. The proof of Theorem 1 is omitted due to its technical nature and space constraints. For the proof of Theorem 1 we refer to the full version of the paper [24].

2 Preliminaries

All graphs in this paper are undirected and unweighted and we will mostly use standard notation. We mention that for a graph G and a vertex v by $\deg(v)$ we refer to the degree of v and by $\deg(G)$ we refer to the maximum degree in G. Furthermore, diam(G) is the diameter of G. When removing a set of vertices X from a graph G we will use the notation $G - X$.

If a function f is defined on a set X and $Y \subseteq X$ we will use the notation $f(Y)$ for $\cup_{y \in Y} f(y)$. When it is clear from the context that we are referring to a vertex set of a graph, we will refer to just the graph. Furthermore, when a function f is defined on the vertex set of a graph, we will sometimes use the sloppy notation $f(G)$ instead of $f(V(G))$.

For intervals of natural numbers we will use the notation $[n]$ for the interval $[1, \ldots, n]$. A k-coloring of a graph G is a function from $V(G)$ to $[k]$ such that two adjacent vertices are given different values. The chromatic number of G, denoted $\chi(G)$ is the minimum k such that there is a k-coloring of G.

A tree is a connected graph without any cycles. A caterpillar is a tree T with a path B as a subgraph, such that all vertices of degree 3 or more lie on B. We then say that B is a backbone of T and every connected component of $T - B$ referred as a stray, or sometimes as a hair. We say that a caterpillar is of stray length s if there exists a backbone such that all strays are of size at most s. An interval graph is a graph such that there exists a function from $V(G)$ into intervals of $\mathbb{N}$ such that the images of two vertices have a non-empty intersection if and only if the two vertices are adjacent.

A *tree decomposition* $\mathcal{T}$ of a graph G is a pair (T, X) with $T = (I, M)$ being a tree and $X = \{X_i \mid i \in I\}$ a collection of subsets of V such that: *(I)* $\bigcup_{i \in I} X_i = V$, *(II)* for every edge uv there is a bag X_i such that both u and v are contained in X_i and *(III)* for every vertex $v \in V$ the set $\{i \in I \mid v \in X_i\}$ induces a tree in T. The *treewidth* of a tree decomposition $\mathcal{T}$, denoted $\mathrm{tw}(G, \mathcal{T}) = \max_{i \in I} |X_i| - 1$ and the treewidth of a graph G is defined as $\mathrm{tw}(G) = \min\{\mathrm{tw}(G, \mathcal{T}) \mid \mathcal{T}$ is a tree decomposition of $G\}$. A *path decomposition* $\mathcal{P}$ of a graph is a tree decomposition such that T is a path. And the *pathwidth* of a graph G, denoted $\mathrm{pw}(G)$ is the minimum width over all path decompositions.

A *linear ordering* α of a set S is a bijection between S and $[|S|]$. Given a graph $G = (V, E)$ and a linear ordering α over V, the *bandwidth* of α denoted $\mathrm{bw}(G, \alpha) = \max_{uv \in E} |\alpha(u) - \alpha(v)|$. And furthermore, the bandwidth of G denoted $\mathrm{bw}(G) = \min\{\mathrm{bw}(G, \alpha) \mid \alpha$ is a linear ordering over $V\}$. We say that α is a *k-bandwidth ordering* of some graph G if $\mathrm{bw}(G, \alpha) \leq k$. And we say that a bandwidth ordering α of G is optimal if $\mathrm{bw}(G, \alpha) = \mathrm{bw}(G)$.

Let u and v be a pair of vertices of a graph G and α an ordering of $V(G)$. We then say that u is left of v in α if $\alpha(u) < \alpha(v)$ and that u is right of v if $\alpha(v) < \alpha(u)$. A sparse ordering β of a graph G is an injective function from $V(G)$ to $\mathbb{Z}$. And the bandwidth of a sparse ordering β of G, denoted $\mathrm{bw}(G, \beta) = \max_{uv \in E} |\beta(u) - \beta(v)|$. We say that a linear ordering α of G is a compression of a sparse ordering β of G if for every pair of vertices u, v in G it holds that $\beta(u) < \beta(v)$ if and only if $\alpha(u) < \alpha(v)$.

Definition 2. *For a graph G we define the local density of G as*

$$D(G) = \max_{G' \subseteq G} \frac{|V(G')| - 1}{diam(G')}.$$

The following proposition will be used repeatedly in our arguments.

Proposition 3 (Folklore). *For every graph G,* $D(G) \leq \mathrm{bw}(G)$ *and* $\mathrm{pw}(G) \leq \mathrm{bw}(G)$.

For a graph T, an integer b and a b-bandwidth ordering α we provide the following definitions. Given a set of vertices $Y \subseteq V(T)$ we define the *inclusion interval of* Y, denoted $I(Y)$ as $[\min \alpha(Y), \max \alpha(Y)]$ and for two vertices u and v we define $I(u, v)$ as $I(\{u, v\})$ or equivalently $[\min\{\alpha(u), \alpha(v)\}, \max\{\alpha(u), \alpha(v)\}]$. Given a subgraph H of T we define $I(H)$ as $I(V(H))$. Whenever necessary, we will use subscript to avoid confusion about which ordering is considered.

We will differentiate the parametrized version of a problem (parameterized by the natural parameter) from the classical one by putting a p in front of the name, i.e. p-BANDWIDTH is the parameterized version of BANDWIDTH. We will face two other problems in this paper. The first one is CLIQUE, where given a graph G and an integer k, one is asked whether there is a clique of size k in G. The second one is EVEN CLIQUE, which is an instance of CLIQUE where you are promised that k is an even number. Both of the problems will be discussed in their parametrized form.

3 Approximation Algorithms

In this section we will provide FPT-approximation algorithms for p-BANDWIDTH on trees and caterpillars. Given a caterpillar T and a positive integer b, `CatAlg` either returns a $48b^3$-bandwidth ordering of T or correctly concludes that $bw(T) > b$. To obtain this we define an obstruction for bandwidth on caterpillars inspired by Chung & Seymour [25] and search for these objects. Based on the appearance of these objects in T we construct an interval graph such that either the interval graph has low chromatic number or the bandwidth of T is large. If the interval graph has low chromatic number we use a coloring of this graph to give a low bandwidth layout of T.

Given a tree T and positive integers b and p such that $\mathrm{pw}(T) \leq p$, `TreeAlg` either returns a $(768b^3)^p$-bandwidth ordering of T or correctly concludes that $bw(T) > b$. The high level outline of the algorithm is as follows. The algorithm first decomposes the tree into several connected components of smaller pathwidth and recurses on these. Then it builds a host graph for T that is a caterpillar, applies `CatAlg` on the host graph. Finally it combines the result of `CatAlg` with the results from the recursive calls, to give a $(768b^3)^p$-bandwidth ordering of T. Since the pathwidth of a graph is known to be bounded above by its bandwidth, it follows that `TreeAlg` is an FPT-approximation.

3.1 An FPT-Approximation for the Bandwidth of Trees

The aim of this section is to give a FPT-approximation for p-BANDWIDTH on trees, namely an $(768b^3)^b$-approximation. This algorithm crucially uses a $48b^3$-approximation of p-BANDWIDTH on caterpillars as a subroutine. We provide such an algorithm, namely the algorithm `CatAlg`, in Section 3.2. In the remainder of this section we give a $(768b^3)^b$-approximation for trees under the assumption that `CatAlg` is a $48b^3$-approximation of p-BANDWIDTH on caterpillars with running time $O(bn^3)$.

Recursive Path Decompositions and Other Simplifications. In this section we will present some decomposition results crucial for our algorithm. First we define recursive path decompositions, which will allow us to partition our graph into several components of slightly lower complexity. The recursive decomposition is used to call the algorithm recursively on easier instances, and then combine the layouts of these instances to a low bandwidth layout of the input tree.

Definition 4. *Let T be a tree and $P, T^1, \ldots, T^t$ induced subgraphs of T such that $V(T) = V(P) \cup \bigcup V(T^i)$. Then we say that $P, T^1, \ldots, T^t$ is a p-*recursive path decomposition *of T if P is a path in T and for every i it holds that T^i is a connected component of $T - P$ and* $\mathrm{pw}(T^i) < p$.

Lemma 5 (*). *Given a tree T of pathwidth at most p, a p-recursive path decomposition $P, T^1, \ldots, T^t$ of T can be found in $O(n)$ time.*

Whenever a result is marked by a star, we refer to the full version of the paper [24] for the proof.

Definition 6. *Let T be a tree and $P, T^1, \ldots, T^t$ a p-recursive path decomposition of T. We construct the* simplified instance *T_S of T with respect to $P, T^1, \ldots, T^t$ as follows. First we add P to T_S. Then, for every T^i we first add a path P^i such that $|V(P^i)| = |V(T^i)|$ and then we add an edge from one endpoint of P^i to $N(T^i)$.*

Observe that the simplified instance T_S is a caterpillar with backbone P.

Lemma 7 (*). *Let T be a tree, $P, T^1, \dots, T^T$ be a p-recursive path decomposition of T and T_S the corresponding simplified instance, then* $\mathrm{bw}(T_S) \leq 2\mathrm{bw}(T)$

Let T be a graph, v a vertex of T and α a b-bandwidth ordering of T. Let β' be a sparse ordering such that for every $u \in T$

$$\beta'(u) = \begin{cases} 2[\alpha(v) - \alpha(u)] & \text{if } \alpha(u) \leq \alpha(v) \text{ and} \\ 2[\alpha(u) - \alpha(v)] - 1 & \text{otherwise.} \end{cases}$$

and let β be the bandwidth ordering obtained by compressing β'. We then say that β is α *right folded* around v. Observe that $\mathrm{bw}(T, \beta) \leq 2\mathrm{bw}(T, \alpha)$.

Algorithm and Correctness. We are now ready to describe algorithm `TreeAlg` and prove its correctness. Pseudocode for `TreeAlg` is given in Algorithm 1.

```
Input: A tree $T$ and positive integers integers $p$ and $b$ such that $\mathrm{pw}(T) \leq p$.
Output: A $(768b^3)^p$-bandwidth ordering of $T$ or conclusion that $\mathrm{bw}(T) > b$.

if $p = 1$ then
    return CatAlg$(T, b)$
end
Find a $p$-recursive path decomposition $P, T^1, \dots, T^t$ of $T$.
Let $\alpha_1 =$ TreeAlg$(T^1, p-1, b), \dots, \alpha_t =$ TreeAlg$(T^t, p-1, b)$.
if there is an $\alpha_i = \perp$ then
    return $\perp$
end
Let $T_s$ be the simplified instance of $T$ with respect to $P, T^1, \dots, T^t$.
Let $\alpha_s =$ CatAlg$(T_s, 2b)$.
if $\alpha_s = \perp$ then
    return $\perp$
end
For every $i$, let $\beta_i$ be $\alpha_i$ right folded around $N(P) \cap T^i$.
For every $v \in P$, let $\alpha(v) = \alpha_s(v)$.
For every $P_i$ of $T_s$ and every $v \in P_i$ of distance $d$ from $P$ in $T_s$, let $\alpha(\beta_i^{-1}(d)) = \alpha_s(v)$.
return $\alpha$
```

Algorithm 1. `TreeAlg`

Lemma 8 (*). *Given a tree T and two integers p an b such that* $\mathrm{pw}(T) \leq p$*,* `TreeAlg` *terminates in $O(pbn^3)$ time.*

Lemma 9. *Given a tree T and positive integers b and p such that* $\mathrm{pw}(T) \leq p$*,* `TreeAlg` *either returns a $O((768b^3)^p)$-bandwidth ordering of T or correctly concludes that* $\mathrm{bw}(T) > b$ *in time $O(pbn^3)$.*

Proof. The running time follows directly from Lemma 8 and hence it remains to prove the correctness of the algorithm. This we will do by induction on p. For $p = 1$ the correctness follows directly from the correctness of `CatAlg` and hence it remains to prove the induction step. First we consider the case when the algorithm concluded that $\mathrm{bw}(T) > b$. Either there is an α_i such that $\alpha_i = \perp$ or $\alpha_s = \perp$. If $\alpha_i = \perp$ it follows by the induction hypothesis and the fact that bandwidth is preserved on subgraphs that the algorithm concluded correctly. Now we consider the case when $\alpha_s = \perp$. It follows from the correctness of `CatAlg` that $\mathrm{bw}(T_s) > 2b$ and hence by Lemma 7 it follows that $\mathrm{bw}(T) > b$.

It remains to consider the case when the algorithm returns a bandwidth ordering α. Then, by the induction hypothesis α_i is a $(768b^3)^{p-1}$-bandwidth ordering of T^i for every i. Furthermore, α_s is a $384b^3$-bandwidth ordering for T_s, since $48(2b)^3 = 384b^3$. Let u and v be two neighbouring vertices of T. If u and v are vertices in P it follows from $\mathrm{bw}(T_s, \alpha_s) \leq 384b^3$ that $|\alpha(u) - \alpha(v)| \leq 384b^3$. Next, we consider the case when either u or v is a vertex in P. Assume without loss of generality that $u \in P$ and let T^j be such that $v \in T^j$. By the definition of β_j it follows that $\beta_j(v) = 1$. It follows that $|\alpha(u) - \alpha(v)| = |\alpha_s(u) - \alpha_s(w)|$ where $\mathrm{dist}(u, w) = 1$, and hence u and w are neighbours in T_s and it follows directly that $|\alpha(u) - \alpha(v)| \leq 384b^3$. We will now consider the case when u and v are vertices of T^j for some j. Let u' be the vertex in P^j of distance $\beta(u)$ from P and v' the vertex in P^j of distance $\beta(v)$ from P. It follows that $|\alpha(u) - \alpha(v)| = |\alpha_s(u') - \alpha_s(v')| \leq \mathrm{dist}(u', v')384b^3 = |\beta_j(u) - \beta_j(v)|384b^3 \leq |\alpha_j(u) - \alpha_j(v)|768b^3 \leq (768b^3)^p$, which completes the proof.

Note that one in the case of $p = 1$ also could solve the instance exactly by Assmann [19]. It would decrease the approximation ratio to $(768b^3)^{p-1}$.

Theorem 10. *There exists an algorithm that given a tree T and a positive integer b either returns a $(768b^3)^b$-bandwidth ordering of T or correctly concludes that $\mathrm{bw}(T) > b$ in time $O(b^2n^3)$.*

Proof. This follows directly from $\mathrm{pw}(T) \leq \mathrm{bw}(T)$ and Lemma 9.

The proof of Theorem 10 assumed the existence of a $48b^3$-approximation algorithm for caterpillars. In the next section we give such an algorithm.

3.2 An FPT-Approximation for the Bandwidth of Caterpillars

The bandwidth of caterpillars is, somewhat surprisingly, a well-studied problem. Assmann et al. [19] proved that the bandwidth of caterpillars of stray length 1 and 2 is polynomial time computable. Monien [4] completed the story of polynomial time computability by proving that BANDWIDTH on caterpillars of stray length 3 is NP-hard. Furtermore, Haralambides [26] gave an $O(\log n)$ approximation algorithm, which later was improved to $O(\log n/ \log\log n)$ by Feige & Talwar [8]. We now give the first FPT-approximation of p-BANDWIDTH on caterpillars, namely a $48b^3$-approximation.

Skewed Cantor Combs. Chung & Seymour [25] defined *Cantor combs*. These are very special caterpillars defined in such a way that they have small local density, but high bandwidth. The definition of Cantor combs is very strict - it precisely defines the length of all the paths in the caterpillars. For our purposes we need a more general definition which captures all caterpillars that are "similar enough" to Cantor combs. We call such caterpillars *skewed Cantor combs*, and we will prove that they also have high bandwidth. Our algorithm will scan for skewed Cantor combs as an obstruction for bandwidth and if none of big enough size are found it will construct a $48b^3$-bandwidth ordering based on the appearance of smaller versions of these objects.

For positive integers $k \leq b$ we now define a *skewed b-Cantor comb* of depth k, denoted $S_{b,k}$ inductively as follows. $S_{b,1}$ is a path of length 1. For the induction step to be well-defined we mark two vertices of every skewed b-Cantor comb as end vertices. For an $S_{b,1}$ the two vertices are the end vertices. For $k > 1$ we start with two skewed b-Cantor combs of depth $k - 1$, lets call them S and S' and furthermore let x, y and x', y' be their end vertices respectively. Connect y to x' by a path P of length at least 2. Furthermore, let Q be a stray connected to an internal vertex v of P. Mark x and y' as the end vertices of the construction and let B be the path from x to y'. Let d be the maximum distance from v to any vertex in B. If Q has at least $2(b-1)d$ vertices we say that the graph described is a skewed b-Cantor comb of depth k.

Lemma 11 (*). *Let $\hat{S}_{b,k}$ be a skewed b-Cantor comb of depth k and α an optimal bandwidth ordering of $\hat{S}_{b,k}$. Furthermore, let x and y be the end vertices of $\hat{S}_{b,k}$ and B the path from x to y. Then there exists an edge uv of $\hat{S}_{b,k}$ such that $I(u,v) \cap I(B)$ is non-empty and $|\alpha(u) - \alpha(v)| = \mathrm{bw}(\hat{S}_{b,k})$.*

Lemma 12 (*). *For $b \geq k \geq 1$, the bandwidth of any $S_{b,k}$ is at least k.*

Directions. Given a caterpillar T and a backbone $B = \{b_1, \ldots, b_k\}$ we define $pos(P)$ for every stray P in T with respect to B, as the integer i such that P is attached to the vertex b_i. Furthermore, we let $|P|$ denote $|V(P)|$.

Definition 13. *Let T be a caterpillar, $B = \{b_1, \ldots, b_k\}$ a backbone of T and b a positive integer. Furthermore, let* depth *be a function from the strays of T with respect to B to $\mathbb{N}$. For every stray Q we let*

- $X_Q = \left\{P \mid pos(P) + \frac{|P|}{2b} < pos(Q) \text{ and } pos(Q) - \frac{|Q|}{2b} \leq pos(P) - \frac{|P|}{2b}\right\}$ *and*
- $Y_Q = \left\{P \mid pos(Q) < pos(P) - \frac{|P|}{2b} \text{ and } pos(P) + \frac{|P|}{2b} \leq pos(Q) + \frac{|Q|}{2b}\right\}$.

Let $x_Q = \max(\text{depth}(X_Q))$ and $y_Q = \max(\text{depth}(Y_Q))$. We say that Q is pushed east *if $x_Q > y_Q$,* pushed west *if $x_Q < y_Q$ and* lifted *if $x_Q = y_Q$.*

We say that a skewed b-Cantor comb of depth k is centered around the stray Q, where Q is as in the definition of $S_{b,k}$. For a caterpillar T we say that a *backbone B is maximized* if for every other backbone B' it holds that $|B'| \leq |B|$.

We will now describe an algorithm `FindSCC` that given a caterpillar T, a maximized backbone B of T and a positive integer b searches for skewed Cantor combs in T. Let *depth* be a function from the strays of T with respect to B into $\mathbb{N}$. As an invariant, *depth* promises there to be a skewed $(b+1)$-Cantor comb centered around Q of depth *depth*(Q). The exception is if *depth*(Q) is 0, then the stray is so short that we ignore it and we hence make no promises with respect to skewed $(b+1)$-Cantor combs. Initially, for every stray Q let *depth*(Q) be 2 if $|Q| \geq 4b$ and 0 otherwise. Observe that the invariant is true due to B being a maximized backbone.

Now we search for a stray Q that is lifted such that both x_Q and y_Q are at least *depth*(Q). It such a Q is found, increase *depth*(Q) by one. Observe that there is in fact a skewed $(b+1)$-Cantor comb centered around Q of this depth (*depth*(Q) after the incrementing). Run this procedure until such a stray Q can not be found or until *depth*(Q) reaches $b+1$ for some stray. Observe that we can for every stray evaluate x_Q and y_Q in $O(n^2)$. And since this is done at most $O(bn)$ times, the running time of `FindSCC` is bounded by $O(bn^3)$.

The reader should note that `FindSCC` does not detect all skewed b-Cantor combs. In fact, it searches only for a stricter version and might overlook the deep skewed $(b+1)$-Cantor combs in a caterpillar. But, as it turns out, these stricter versions are sufficient for our purposes. From now on, we will assume that the function applied when evaluation whether a stray is pushed west or east, is the depth function calculated by running `FindSCC`.

Definition 14. *For a caterpillar T, a maximized backbone $B = \{b_1, \ldots, b_l\}$ of T and a positive integer b we define the* directional stray graph *as the following interval graph: for every stray P add the interval*

- *$[pos(P)48b^3 - 12b^2|P|, pos(P)48b^3]$ if P is pushed west and*
- *$[pos(P)48b^3, pos(P)48b^3 + 12b^2|P|]$ otherwise.*

We say that an interval originating from a stray pushed west is west oriented *and visa versa.*

Lemma 15 (*). *Let T be a caterpillar, b a positive integer, G_I some directional stray graph of T and x and y two natural numbers such that $x < y$. Then either there are at most $2b$ intervals of length at least $y - x$ in G_I starting within $[x, y]$, or $\mathrm{bw}(T) > b$.*

Lemma 16. *Let T be a caterpillar, b a positive integer and G_I some directional stray graph of T. Then either $\chi(G_I) < 12b^2$ or $\mathrm{bw}(T) > b$.*

Proof. Assume for a contradiction that $\chi(G_I) \geq 12b^2$ and that $\mathrm{bw}(T) \leq b$. Then there is a number w such that at least $12b^2$ of the intervals of G_I contains w. This follows from the well-known result that $\chi(G_I)$ equals the size of the maximum clique of G_I, since G_I is an interval graph. Let I be the set of all east oriented intervals containing w and assume without loss of generality that I is of size at least $6b^2$. Discard the elements of I with the highest starting value and let $[x', y']$ be a discarded element. Observe that at most $2b$ elements were discarded due to the local density bound. Hence we now have at least $6b^2 - 2b$ elements left. We will start by giving a lower bound on the length of the intervals in I. Consider an element $[x, y]$ of shortest length in I. By definition $x < x' \leq y$ and by construction $x' - x \geq 48b^3$, hence $y - x \geq 48b^3$ and it follows that all elements of I are of length at least $48b^3$.

Let $[x_2, y_2]$ be a shortest interval in I and recall that the stray P^2 corresponding to the interval is attached to the backbone vertex b_{c_2} for $c_2 = x_2/48b^3$. Furthermore, $|P^2| = (y_2 - x_2)/12b^2 > 48b^3/12b^2 = 4b$. Since the backbone used when constructing

G_I is maximized it follows that the distance from b_{c_2} to any endpoint of the backbone is at least 4 and hence there is an $S_{b+1,2}$ centered around b_{c_2}.

Discard all intervals with their starting point within $[x_2 - 2(y_2 - x_2), y_2]$ in I. We know that at most $6b$ elements are discarded by Lemma 15. Now let $[x_3, y_3]$ be a shortest interval in I and recall that the stray P^3 corresponding to the interval is attached to the backbone vertex b_{c_3} for $c_3 = x_3/(12b^2)$. Observe that $|y_3 - x_3| > |x_3 - x_2|$ and that $|y_2 - x_2| < \frac{1}{2}|x_3 - x_2|$ and hence $|y_3 - x_3| > |x_3 - x_2| > \frac{1}{2}|x_3 - x_2| + |y_2 - x_2|$. If follows immediately that $\frac{|y_3 - x_3|}{2b(12b^2)} > \frac{|x_3 - x_2|}{48b^3} + \frac{|y_2 - x_2|}{2b(12b^2)}$ and by replacement $\frac{|V(P^3)|}{2b} > |c_3 - c_2| + \frac{|V(P^2)|}{2b}$.

Let S be the $S_{b+1,2}$ centered around b_{c_2} and recall that by definition the distance from b_{c_2} to any backbone vertex of S is bounded from above by $\frac{|V(P^2)|}{2b}$. It follows that the distance from b_{c_3} to any backbone vertex of S is bounded by $\frac{|V(P^3)|}{2b}$. Since $[x_3, y_3]$ is east oriented there is another $S_{b+1,i}$ centered around a stray $\bar{P}^2$ such that $\text{pos}(\bar{P}^2) + \frac{|\bar{P}^2|}{2b} < c_3$ and $\text{pos}(\bar{P}^2) - \frac{|\bar{P}^2|}{2b} \geq c_3 - \frac{|P^3|}{2b}$ for some $i \geq 2$. By definition, the $S_{b+1,i}$ contains an $S_{b+1,2}$ as a subgraph in such a way that there is an $S_{b+1,3}$ centered around c_3. Discard all intervals with starting points within $[x_3 - 2(y_3 - x_3), x_3]$ and repeat the argument to obtain a $S_{b+1,4}$. We keep repeating the argument until we obtain a $S_{b+1,b+1}$

Notice that we can do this as we are discarding at most $6b$ vertices each time, repeating the procedure $b - 1$ times and I contains at least $6b^2 - 2b > 6b(b - 1)$ intervals. This completes the proof, as we know from Lemma 12 that $\text{bw}(S_{b+1,b+1}) \geq b + 1$.

Algorithm and Correctness

Theorem 17. *There exists an algorithm that given a caterpillar T and a positive integer b either returns a $48b^3$-bandwidth ordering of T or correctly concludes that $\text{bw}(T) > b$ in time $O(bn^3)$.*

Proof. Recall that `FindSCC` runs in $O(bn^3)$ time. Furthermore, a coloring of G_I can be found in $O(n)$ time by Golumbic [27]. Observe that every other step of the algorithm trivially runs in $O(n)$ time. And hence the algorithm runs in $O(bn^3)$ time. If `CatAlg` returns $\perp$, then $\chi(G_I) \geq 12b^2$. It follows from Lemma 16 that $\text{bw}(T) > b$ and hence the conclusion is correct. We will now prove that α is a sparse ordering of $V(T)$ of bandwidth at most $48b^3$. It is clear that for any edge $uv \in E(T)$ it holds that $|\alpha(u) - \alpha(v)| \leq 48b^3$. It remains to prove that α is an injective function. Assume for a contradiction that there are two vertices u, v such that $\alpha(u) = \alpha(v)$. Observe that $\alpha(u) \equiv 0 \mod (48b^3)$ if and only if u is a backbone vertex of T. This comes from the fact that $\chi(G_I) < 12b^2$. And since it is clear from the algorithm that no two vertices of the backbone are given the same position we can assume that neither u nor v is a backbone vertex. It follows that $\alpha(u) \equiv c(P) \mod (12b^2)$ where P is the stray containing u. Observe that the algorithm gives unique positions to all vertices from the same stray and hence u and v must belong to two different strays given the same color. Let P_u be the stray containing u and P_v the strain containing v. Furthermore, let $[x_u, y_u]$ and $[x_v, y_v]$ be the corresponding intervals in G_I. Observe that $I(P_u) \subseteq [x_u, y_u]$ and $I(P_v) \subseteq [x_v, y_v]$ and hence $[x_u, y_u] \cap [x_y, y_v] \neq \emptyset$, which is a contradiction, completing the proof.

```
Input: A caterpillar $T$ and a positive integer $b$.
Output: A $48b^3$-bandwidth ordering of $T$ or conclusion that $\mathrm{bw}(T) > b$.
Let $B = \{b_1, \dots, b_k\}$ be a maximized backbone of $T$.
Construct the directional stray graph $G_I$ of $T$ with respect to $B$.
Find a minimum coloring of $G_I$.
if $\chi(G_I) \geq 12b^2$ then
    return $\perp$.
end
Let $\alpha(b_i) = 48b^3(n + i)$.
Let $\mathcal{P}$ be the collection of strays in $T$ with respect to $B$.
For every stray $P$ in $\mathcal{P}$ let $C(P)$ be the color of the interval representing the stray.
for every $P \in \mathcal{P}$ do
    Let $p_1, \dots, p_k$ be the vertices of $P$ such that $\mathrm{dist}(B, p_i) < \mathrm{dist}(B, p_{i+1})$ for every $i$.
    Let $\{u\} = N(P)$.
    if $P$ is pushed west then
        Let $\alpha(p_i) = \alpha(u) + C(P) - i12b^2$ for every $i$.
    end
    else
        Let $\alpha(p_i) = \alpha(u) + C(P) + (i - 1)12b^2$ for every $i$.
    end
end
return Compressed version of $\alpha$.
```

Algorithm 2. `CatAlg`

4 Concluding Remarks

We have shown that the classical $2^{O(b)}n^{b+1}$ time algorithm of Saxe [9] for the BANDWIDTH problem is essentially optimal, even on trees of pathwidth at most 2. On trees of pathwidth 1, namely caterpillars with hair length 1, the problem is known to be polynomial time solvable. On the positive side, we gave the first approximation algorithm for BANDWIDTH on trees with approximation ratio being a function of b and independent of n. Our approximation algorithm is based on pathwidth, local density and a new obstruction to bounded bandwidth called skewed Cantor combs. We conclude with a few open problems. *(I)* Does BANDWIDTH admit a parameterized approximation algorithm on general graphs? *(II)* Does BANDWIDTH admit an approximation algorithm on trees with approximation ratio polynomial in b? What if one allows the algorithm to have running time $f(b)n^{O(1)}$? *(III)* Does there exist a function f such that any graph G with pathwidth at most c_1, local density at most c_2, and containing no S_{c_3,c_3} as a subgraph has bandwidth at most $f(c_1, c_2, c_3)$?

References

1. George, A., Liu, J.W.: Computer Solution of Large Sparse Positive Definite. Prentice Hall Professional Technical Reference (1981)
2. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1979)

3. Papadimitriou, C.H.: The np-completeness of the bandwidth minimization problem. Computing 16(3), 263–270 (1976)
4. Monien, B.: The bandwidth minimization problem for caterpillars with hair length 3 is np-complete. SIAM Journal on Algebraic Discrete Methods 7(4), 505–512 (1986)
5. Dubey, C., Feige, U., Unger, W.: Hardness results for approximating the bandwidth. Journal of Computer and System Sciences 77(1), 62–90 (2011)
6. Dunagan, J., Vempala, S.: On euclidean embeddings and bandwidth minimization. In: Goemans, M.X., Jansen, K., Rolim, J.D.P., Trevisan, L. (eds.) RANDOM 2001 and APPROX 2001. LNCS, vol. 2129, pp. 229–240. Springer, Heidelberg (2001)
7. Gupta, A.: Improved bandwidth approximation for trees. In: SODA, pp. 788–793 (2000)
8. Feige, U., Talwar, K.: Approximating the bandwidth of caterpillars. Algorithmica 55(1), 190–204 (2009)
9. Saxe, J.B.: Dynamic-programming algorithms for recognizing small-bandwidth graphs in polynomial time. SIAM Journal on Algebraic Discrete Methods 1(4), 363–369 (1980)
10. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? J. Comput. Syst. Sci. 63(4), 512–530 (2001)
11. Downey, R.G., Fellows, M.R.: Parameterized complexity, vol. 3. Springer (1999)
12. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer-Verlag New York, Inc. (2006)
13. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford University Press (2006)
14. Marx, D.: Parameterized complexity and approximation algorithms. Comput. J. 51(1), 60–78 (2008)
15. Chung, F.R.K., Seymour, P.D.: Graphs with small bandwidth and cutwidth. Discrete Mathematics 75(1-3), 113–119 (1989)
16. Feige, U.: Approximating the bandwidth via volume respecting embeddings. J. Comput. Syst. Sci. 60(3), 510–539 (2000)
17. Bodlaender, H.L., Fellows, M.R., Hallett, M.T.: Beyond np-completeness for problems of bounded width: hardness for the w hierarchy. In: STOC, pp. 449–458 (1994)
18. Golovach, P.A., Heggernes, P., Kratsch, D., Lokshtanov, D., Meister, D., Saurabh, S.: Bandwidth on at-free graphs. Theor. Comput. Sci. 412(50), 7001–7008 (2011)
19. Assmann, S., Peck, G., Syslo, M., Zak, J.: The bandwidth of caterpillars with hairs of length 1 and 2. SIAM Journal on Algebraic Discrete Methods 2(4), 387–393 (1981)
20. Heggernes, P., Kratsch, D., Meister, D.: Bandwidth of bipartite permutation graphs in polynomial time. J. Discrete Algorithms 7(4), 533–544 (2009)
21. Kleitman, D.J., Vohra, R.V.: Computing the bandwidth of interval graphs. SIAM Journal on Discrete Mathematics 3(3), 373–375 (1990)
22. Yan, J.H.: The bandwidth problem in cographs. Tamsui Oxford Journal of Mathematical Sciences (13), 31–36 (1997)
23. Chinn, P.Z., Chvatalova, J., Dewdney, A.K., Gibbs, N.E.: The bandwidth problem for graphs and matrices-a survey. Journal of Graph Theory 6(3), 223–254 (1982)
24. Dregi, M.S., Lokshtanov, D.: Parameterized complexity of bandwidth on trees. CoRR abs/1404.7810 (2014)
25. Chung, F.R., Seymour, P.D.: Graphs with small bandwidth and cutwidth. Discrete Mathematics 75(1), 113–119 (1989)
26. Haralambides, J., Makedon, F., Monien, B.: Bandwidth minimization: an approximation algorithm for caterpillars. Mathematical Systems Theory 24(1), 169–177 (1991)
27. Golumbic, M.C.: Algorithmic graph theory and perfect graphs, vol. 57. Elsevier, Amsterdam (2004)

Testing Equivalence of Polynomials under Shifts[⋆]

Zeev Dvir[1,⋆⋆], Rafael Mendes de Oliveira[2], and Amir Shpilka[3,⋆⋆⋆]

[1] Department of Computer Science and Department of Mathematics,
Princeton University
zeev.dvir@gmail.com

[2] Department of Computer Science, Princeton University
rmo@cs.princeton.edu

[3] Department of Computer Science, Technion — Israel Institute of Technology,
Haifa, Israel
shpilka@cs.technion.ac.il

Abstract. Two polynomials $f, g \in \mathbb{F}[x_1, \ldots, x_n]$ are called *shift-equivalent* if there exists a vector $(a_1, \ldots, a_n) \in \mathbb{F}^n$ such that the polynomial identity $f(x_1 + a_1, \ldots, x_n + a_n) \equiv g(x_1, \ldots, x_n)$ holds. Our main result is a new randomized algorithm that tests whether two given polynomials are shift equivalent. Our algorithm runs in time polynomial in the circuit size of the polynomials, to which it is given black box access. This complements a previous work of Grigoriev [Gri97] who gave a deterministic algorithm running in time $n^{O(d)}$ for degree d polynomials.

Our algorithm uses randomness only to solve instances of the Polynomial Identity Testing (PIT) problem. Hence, if one could de-randomize PIT (a long-standing open problem in complexity) a de-randomization of our algorithm would follow. This establishes an equivalence between de-randomizing shift-equivalence testing and de-randomizing PIT (both in the black-box and the white-box setting). For certain restricted models, such as Read Once Branching Programs, we already obtain a deterministic algorithm using existing PIT results.

1 Introduction

In this paper we address the following problem, which we call *Shift Equivalence Testing* (SET). Given two polynomials $f, g \in \mathbb{F}[\mathbf{x}]$ (we use boldface letters to denote vectors), decide whether there exists a shift $\mathbf{a} \in \mathbb{F}^n$ such that $f(\mathbf{x}+\mathbf{a}) \equiv g(\mathbf{x})$ and output one if it exists. The symbol $\equiv$ is used to denote polynomial identity (the polynomials should have the same coefficients). We will focus mainly on the

[⋆] Full version of this paper is available at http://eccc.hpi-web.de/report/2014/003/

[⋆⋆] Research supported by NSF grants CCF-1217416 and CCF-0832797.

[⋆⋆⋆] The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 257575.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 417–428, 2014.

case where $\mathbb{F}$ is a field of characteristic zero (such as the rational numbers) or has a sufficiently large positive characteristic.

Observe that f is shift-equivalent to the zero polynomial if and only if f itself is the zero polynomial. Hence, SET is a natural generalization of the well-known Polynomial Identity Testing problem (PIT) in which we need to test whether $f(\mathbf{x}) \equiv 0$ given access to a succinct representation of f (say, as a circuit). A classical randomized algorithm by Schwartz-Zippel-DeMillo-Lipton [Sch80, Zip79, DL78] is known for PIT: evaluate f on a random input (from a large enough domain) and test if f evaluates to zero on that point. If f is non-zero, then it is not zero on a random point with very high probability. In contrast, it is not clear at all how to devise a randomized algorithm for SET. Unlike PIT, which is a 'co-NP' type problem (there is short proof that a polynomial is *not* zero), the SET problem is an '$\mathrm{RP}^{\mathrm{NP}}$' type problem (there is a short witness (the shift itself) that polynomials *are* shift equivalent, and verifying that witness is in RP).

The problem of equivalence of polynomials under shifts of the input first appeared in the works of Grigoriev, Lakshman, Saunders and Karpinski [GK93, GL95, LS94] (see also references therein), in the context of finding sparse shifts of a polynomial. That is, they were interested in finding a shift that will make a given polynomial sparse, if such a shift indeed exists. The main motivation for this question comes from considering polynomials in their sum-of-monomials representation (also called dense representation or depth-2 circuit complexity), and the goal is to find a shift that will make the representation more succinct. Later, in [Gri97], Grigoriev asked the following question: given two polynomials $f, g \in \mathbb{F}[\mathbf{x}]$, is there an efficient algorithm that can find whether there exists a shift $\mathbf{a} \in \mathbb{F}^n$ such that $f(\mathbf{x}+\mathbf{a}) \equiv g(\mathbf{x})$? In the same paper, Grigoriev gave algorithms for three versions of this problem: one deterministic for characteristic zero, one randomized for large enough characteristic $0 < p$ and one quantum for characteristic 2. The running time of Grigoriev's algorithms was polynomial in the dense representation. That is, for polynomials of degree d in n variables, the running time was $n^{O(d)}$ (which is an upper bound on the number of coefficients). In this paper, we address the same question as Grigoriev, but assume that the polynomials are given in some succinct representation (say, as arithmetic circuits). In this representation, one can hope for running time which is polynomial in the size of the given circuits (which can be exponentially small relative to the dense representation). For example the determinant polynomial has n^2 variables and degree n but can be given as a circuit of size $n^{O(1)}$ in the succinct representation.

Our main result is a new randomized (two-sided error) algorithm for SET. The algorithm runs in time polynomial in the circuit size of the given polynomials. In fact, we only require *black-box* access to the polynomials f and g and a bound on their degree and circuit size. Our algorithm is obtained as a reduction to the PIT problem. Hence, if we were able to perform deterministic PIT, we could also perform deterministic SET. For certain interesting restricted models of arithmetic computation, this already gives deterministic SET. For general circuits, our results show that it is equivalently hard to de-randomize PIT and SET, which is somewhat surprising as by the explanation above it seems as if SET is a much harder problem than PIT.

Below, we will state our results in the most general way, assuming f and g belong to some circuit classes closed under certain operations. The reason for doing this is that, in this way, one can see exactly what conditions are required to de-randomize the algorithm. That is, what kind of deterministic PIT is required to derive deterministic SET (in general we require PIT for a slightly larger class).

1.1 Formal Statement of Our Results

Our results rely on closure properties of the underlying circuit classes.

Definition 1. *Given a class of arithmetic circuits $\mathcal{M}$ we will say that $\mathcal{M}$ is closed under an operator $A : \mathbb{F}[\mathbf{x}] \mapsto \mathbb{F}[\mathbf{x}]$ if the following property holds. Let f be an n-variate polynomial of total degree d that is computed by a circuit of size s from $\mathcal{M}$. Then we require that $A(f)$ is computed by a circuit of size $\mathsf{poly}(n, d, s)$ from $\mathcal{M}$.*

For instance, one operator that is very common and under which all of the most studied circuit classes are closed is the restriction operator, namely, the operator that substitutes some of the variables of $f(\mathbf{x})$ by field elements. It is easy to see that by substituting some variables by field elements, the new polynomial will also be computed by a circuit of size less than s, and in general the new polynomial will also belong to the same class as f.

In addition, we will need to discuss closure under three different operators:

- **Directional partial derivatives:** The partial derivatives $\frac{\partial f}{\partial x_i}$ of a polynomial f are defined in the usual sense (over finite fields we use the formal definition for polynomials). We define the *first order partial derivative of f in direction* $\mathbf{a} \in \mathbb{F}^n$ to be

$$f^{(1)}(\mathbf{a}, \mathbf{x}) \triangleq \sum_{t=1}^{n} a_t \cdot \frac{\partial f}{\partial x_t}(\mathbf{x}).$$

  Apart from the class of general circuits (and formulas) that are closed under taking first order derivatives [BS83], the class of sparse polynomials (depth 2 circuits) is also closed under directional partial derivatives. Note, however, that depth-3 circuits with at most k multiplication gates, also known as $\Sigma\Pi\Sigma(k)$ circuits, are *not known* to be closed under directional partial derivatives as these might increase the top fanin.
- **Homogeneous components:** If $f \in \mathbb{F}[\mathbf{x}]$ is a polynomial of degree d, we will denote the homogeneous component of degree k of f by $H^k(f(\mathbf{x}))$. General circuits and formulas are closed under taking homogeneous components, and the same also holds for the class of sparse polynomials.
- **Shifts:** Here we require that a class will be closed under the operation $f(\mathbf{x}) \mapsto f(\mathbf{x} + \mathbf{a})$ for some $\mathbf{a} \in \mathbb{F}^n$. Again, circuits and formulas are closed to shifts, however, the class of sparse polynomials is not.

We now describe our main result that solves the SET problem given a PIT algorithm.

Theorem 1 (Main Theorem). *Let $\mathbb{F}$ be a field of characteristic zero. Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two circuit classes such that*

1. *$\mathcal{M}_1$ is closed under taking homogeneous components and closed under (first-order) directional derivatives.*
2. *$\mathcal{M}_2$ is closed under taking shifts.*
3. *We have a (white-box) black-box PIT algorithm $\mathcal{P}$ for polynomials in $\mathcal{M}_1, \mathcal{M}_2$ and for polynomials of the form $f - g$, where $f \in \mathcal{M}_1$ and $g \in \mathcal{M}_2$.*

Then, there exists an algorithm $\mathcal{S}$ that, given (white-box) black-box access to polynomials $f \in \mathcal{M}_1, g \in \mathcal{M}_2$ and a bound d on the their degree, returns $\mathbf{a} \in \mathbb{F}^n$ so that $g(\mathbf{x} + \mathbf{a}) \equiv f(\mathbf{x})$, if such a shift exists, or returns FAIL, if none exist.

Furthermore:

- *The running time of $\mathcal{S}$ is polynomial in the running time of $\mathcal{P}$ and in the other parameters (n, d).*
- *If the PIT algorithm $\mathcal{P}$ is deterministic then so is $\mathcal{S}$.*
- *All of the above holds also for the case when $\mathbb{F}$ is a finite field with characteristic greater than d.*

Combining Theorem 1 with Schwartz-Zippel-DeMillo-Lipton, we obtain a randomized SET algorithm for any pair of polynomials.

Theorem 2 (Randomized SET for Pairs of Polynomials). *Let $\mathbb{F}$ be a field of characteristic zero or of characteristic larger than d. There exists a randomized algorithm that, given black box access to $f, g \in \mathbb{F}[\mathbf{x}]$ of degree at most d, returns $\mathbf{a} \in \mathbb{F}^n$ such that $g(\mathbf{x} + \mathbf{a}) \equiv f(\mathbf{x})$, if such a shift exists, or FAIL otherwise. The algorithm runs in time $\mathsf{poly}(n, d, \log(1/\varepsilon))$, where ϵ is the probability or returning a wrong answer (i.e., FAIL if a shift exists or a shift if none exists).*

Remark 1. An interesting fact about Theorem 2 is that the algorithm we obtain has a two sided error (this can be seen from the proof). This fact is in contrast to the fact that most randomized algorithms in the algebraic setting have one-sided error.

Theorem 1 already leads to deterministic algorithms for certain restricted models. For instance, in the recent works of Forbes and Shpilka [FS12, FS13] and of Forbes, Saptharishi and Shpilka [FSS13], the authors obtain a quasi-polynomial deterministic PIT algorithm for read-once oblivious algebraic branching programs (ROABPs). Their result, together with our algorithm, imply that we can find out whether two ROABPs are shift-equivalent in deterministic quasi-polynomial time. Since this class also captures tensors,[1] an application of our result is that we can find out whether two tensors are shift-equivalent in quasi-polynomial time (we refer the reader to [FS13, FSS13] for definitions of ROABPs and tensors).

[1] We note that the work [ASS13] also gives a black-box PIT algorithm for tensors.

Corollary 1. *There is a deterministic quasi-polynomial time algorithm that given black-box access to two polynomials f and g computed by read-once oblivious algebraic branching programs, decides whether there exists $\mathbf{a} \in \mathbb{F}^n$ such that $f(\mathbf{x}+\mathbf{a}) \equiv g(\mathbf{x})$ and in case that such a shift exists, the algorithm outputs one.*

As the class of sparse polynomials is closed under taking homogeneous components and under first order directional derivatives (a directional derivative blows up the size of the circuit by at most a factor of n) we obtain the following corollary.

Corollary 2. *Let $\mathcal{M}_2$ be any circuit class so that*

1. *$\mathcal{M}_2$ is closed under shifts.*
2. *There is a deterministic PIT algorithm testing if $f - g$ is zero for sparse f and $g \in \mathcal{M}_2$.*

Then, we can test whether f and g are shift-equivalent deterministically in time $\mathsf{poly}(n, s)$

As an application of our main theorem in the white-box model, we note that Saha et al. gave a polynomial time algorithm for testing whether a given sparse polynomial equals a $\Sigma\Pi\Sigma(k)$ circuit [SSS13]. Since their algorithm works in the white-box model, we can utilize it in the variant of our main theorem in the white-box model to find whether a given sparse polynomial and a polynomial in $\Sigma\Pi\Sigma(k)$ are shift-equivalent. We also note that we can make their algorithm work in the black-box case as well. Using the reconstruction algorithms of [Shp09, KS09] we can first reconstruct the $\Sigma\Pi\Sigma(k)$ circuit in quasi-polynomial time. We can also interpolate the sparse polynomial in polynomial time (for interpolation of sparse polynomials see e.g. [KS01]) and then apply our methods together with the PIT algorithm of Saha et al. to solve the shift-equivalence problem.[2]

1.2 Overview of the Algorithm

In this section we give a short overview of our algorithm and its analysis. Assume we are given $f(\mathbf{x})$ and $g(\mathbf{x})$ and we have to find whether there exists $\mathbf{a} \in \mathbb{F}^n$ such that $f(\mathbf{x}+\mathbf{a}) \equiv g(\mathbf{x})$.

In the highest level, our algorithm will produce a candidate shift $\mathbf{a}$ such that $f(\mathbf{x}+\mathbf{a}) \equiv g(\mathbf{x})$ and then use PIT on the polynomial $f(\mathbf{x}) - g(\mathbf{x}-\mathbf{a})$, to check that the solution $\mathbf{a}$ is indeed a good shift. We need to perform the PIT on the polynomial $f(\mathbf{x}) - g(\mathbf{x}-\mathbf{a})$ because $\mathcal{M}_2$ is closed under shifts. Since we will test the validity of our candidate shift, this approach allows us to assume from the beginning on that there exists a good shift $\mathbf{a} \in \mathbb{F}^n$.

Since we assume that a shift exists, we will have that $\deg(f) = \deg(g) = d$. Let us also denote $f(\mathbf{x}) = \sum_{i=0}^{d} H^i(f(\mathbf{x}))$ where each $H^i(f)$ is homogeneous of

[2] Note that the reconstruction algorithm of [Shp09, KS09] returns so-called generalized $\Sigma\Pi\Sigma(k)$ circuits. Nevertheless, one can observe that the algorithm of Saha et al. works for such circuits as well.

degree i and similarly, $g(\mathbf{x}) = \sum_{i=0}^{d} H^i(g(\mathbf{x}))$. Since $f(\mathbf{x}+\mathbf{a}) \equiv g(\mathbf{x})$, we can also write $g(\mathbf{x})$ as

$$g(\mathbf{x}) = \sum_{i=0}^{d} H^i(f(\mathbf{x}+\mathbf{a})). \tag{1}$$

By equation (1), we have that

$$H^d(g(\mathbf{x})) \equiv H^d(f(\mathbf{x}+\mathbf{a})) \equiv H^d(f(\mathbf{x})).$$

Thus, we know that the homogeneous parts of highest degree of f and g must agree.

Next we move to degree $d-1$. A quick calculation gives

$$H^{d-1}(g(\mathbf{x})) = H^{d-1}(f(\mathbf{x}+\mathbf{a})) = H^{d-1}(f(\mathbf{x})) + \sum_{k=1}^{n} a_k \cdot \frac{\partial H^d(f(\mathbf{x}))}{\partial x_k}.$$

Therefore,

$$H^{d-1}(g(\mathbf{x})) - H^{d-1}(f(\mathbf{x})) = \sum_{k=1}^{n} a_k \cdot \frac{\partial H^d(f(\mathbf{x}))}{\partial x_k},$$

which implies that the polynomial $H^{d-1}(g(\mathbf{x})) - H^{d-1}(f(\mathbf{x}))$ is a directional derivative of $H^d(f(\mathbf{x}))$ and therefore belongs to the class $\mathcal{M}_1$. This observation is crucial, since it allows us to not require $\mathcal{M}_2$ to be closed under taking homogeneous parts.

After making this observation, notice that we want to find a vector $\mathbf{b} \in \mathbb{F}^n$ such that

$$H^{d-1}(g(\mathbf{x})) - H^{d-1}(f(\mathbf{x})) = \sum_{k=1}^{n} b_k \cdot \frac{\partial H^d(f(\mathbf{x}))}{\partial x_k}, \tag{2}$$

Observe that (2) is a polynomial equation that is linear in the entries of $\mathbf{b}$. Hence, for each $\mathbf{v} \in \mathbb{F}^n$, polynomial equation (2) becomes the following linear equation in the entries of $\mathbf{b}$:

$$H^{d-1}(g(\mathbf{v})) - H^{d-1}(f(\mathbf{v})) = \sum_{k=1}^{n} b_k \cdot \frac{\partial H^d(f(\mathbf{v}))}{\partial x_k}. \tag{3}$$

We can solve polynomial equation (2) by solving a system of linear equations on $\mathbf{b}$ given by a set of carefully chosen equations of the form (3). If we have a hitting set $\mathcal{H}$ for the class $\mathcal{M}_1$, the system of equations becomes the set of all equations of the form (3) for which $\mathbf{v} \in \mathcal{H}$. See full version of the paper for more details on how to solve this system in the white-box model. From now on, we will refer to a polynomial equation by the system of linear equations associated with it.

It turns out that if our circuit class is closed under directional derivatives, that is, if the polynomial $\sum_{k=1}^{n} a_k \cdot \frac{\partial H^d(f(\mathbf{x}))}{\partial x_k}$ (which equals the polynomial

$H^{d-1}(g(\mathbf{x})) - H^{d-1}(f(\mathbf{x}))$) belongs to the same circuit class as $f(\mathbf{x})$ (or a slightly larger class), and if we have a hitting set for the class of polynomials of the form $\sum_{k=1}^{n} a_k \cdot \frac{\partial H^d(f(\mathbf{x}))}{\partial x_k}$, for every $\mathbf{a} \in \mathbb{F}^n$, then we can solve system of equations (3) and find some solution $\mathbf{b}$ such that

$$H^{d-1}(g(\mathbf{x})) = H^{d-1}(f(\mathbf{x}+\mathbf{b})) = H^{d-1}(f(\mathbf{x})) + \sum_{k=1}^{n} b_k \cdot \frac{\partial H^d(f(\mathbf{x}))}{\partial x_k}.$$

If we allow randomness then we can also solve this system of equations without having a hitting set, since we can randomly guess a hitting set and use it to solve the system of linear equations.

Note that at this point we might have $\mathbf{b} \neq \mathbf{a}$. Hence, we have found a shift $\mathbf{b}$ that makes the homogeneous parts of degree d and $d-1$ in f and g equal. We now consider the homogeneous components of degree $d, d-1$ and $d-2$.

Here we have the system of equations (on the variables $\mathbf{c}$)

$$H^{d-1}(g(\mathbf{x})) - H^{d-1}(f(\mathbf{x})) = \sum_{k=1}^{n} c_k \cdot \frac{\partial H^d(f(\mathbf{x}))}{\partial x_k}, \tag{4}$$

and

$$\begin{aligned} H^{d-2}(g(\mathbf{x})) &= H^{d-2}(f(\mathbf{x}+\mathbf{c})) \qquad (5)\\ &= H^{d-2}(f) + \sum_{k=1}^{n} c_k \cdot \frac{\partial H^{d-1}(f(\mathbf{x}))}{\partial x_k} + \sum_{\ell,k=1}^{n} c_\ell c_k \frac{\partial^2 H^d(f(\mathbf{x}))}{\partial x_\ell \partial x_k}. \qquad (6)\end{aligned}$$

And now we seem to be in trouble as this is a system of quadratic equations in the entries of $\mathbf{c}$. Here comes another crucial observation. Recall that we have found $\mathbf{b}$ such that

$$H^{d-1}(g(\mathbf{x})) = H^{d-1}(f(\mathbf{x})) + \sum_{k=1}^{n} b_k \cdot \frac{\partial H^d(f(\mathbf{x}))}{\partial x_k}.$$

We also have that

$$H^{d-1}(g(\mathbf{x})) = H^{d-1}(f(\mathbf{x}+\mathbf{a})) = H^{d-1}(f(\mathbf{x})) + \sum_{k=1}^{n} a_k \cdot \frac{\partial H^d(f(\mathbf{x}))}{\partial x_k}.$$

Hence,

$$\sum_{k=1}^{n} (a_k - b_k) \cdot \frac{\partial H^d(f(\mathbf{x}))}{\partial x_k} = 0.$$

This means that the directional derivative of $H^d(f(\mathbf{x}))$ in direction $\mathbf{a}-\mathbf{b}$ is zero. Or, in other words, that the polynomial $H^d(f(\mathbf{x}))$ is fixed along that direction. This means that no matter how many derivatives we take along direction $\mathbf{a}-\mathbf{b}$ we always get the zero polynomial.

In particular, this gives

$$\sum_{\ell,k=1}^{n} a_\ell a_k \frac{\partial^2 H^d(f(\mathbf{x}))}{\partial x_\ell \partial x_k} = \sum_{\ell,k=1}^{n} b_\ell b_k \frac{\partial^2 H^d(f(\mathbf{x}))}{\partial x_\ell \partial x_k},$$

as both sides compute the second directional derivatives in directions **a** and **b**, respectively.

Since **c** must satisfy equation (4), we also have that

$$\sum_{k=1}^{n} (c_k - b_k) \cdot \frac{\partial H^d(f(\mathbf{x}))}{\partial x_k} = 0.$$

And therefore, by the observation above we have that

$$\sum_{\ell,k=1}^{n} a_\ell a_k \frac{\partial^2 H^d(f(\mathbf{x}))}{\partial x_\ell \partial x_k} = \sum_{\ell,k=1}^{n} c_\ell c_k \frac{\partial^2 H^d(f(\mathbf{x}))}{\partial x_\ell \partial x_k} = \sum_{\ell,k=1}^{n} b_\ell b_k \frac{\partial^2 H^d(f(\mathbf{x}))}{\partial x_\ell \partial x_k}. \quad (7)$$

Going back, and using (7), we now have that system (5) is equivalent to the system

$$H^{d-2}(g(\mathbf{x})) = H^{d-2}(f) + \sum_{k=1}^{n} c_k \cdot \frac{\partial H^{d-1}(f(\mathbf{x}))}{\partial x_k} + \sum_{\ell,k=1}^{n} b_\ell b_k \frac{\partial^2 H^d(f(\mathbf{x}))}{\partial x_\ell \partial x_k}. \quad (8)$$

By using the fact that

$$\begin{aligned} H^{d-2}(g(\mathbf{x})) &= H^{d-2}(f(\mathbf{x} + \mathbf{a})) \\ &= H^{d-2}(f(\mathbf{x})) + \sum_{k=1}^{n} a_k \cdot \frac{\partial H^{d-1}(f(\mathbf{x}))}{\partial x_k} + \sum_{\ell,k=1}^{n} a_\ell a_k \frac{\partial^2 H^d(f(\mathbf{x}))}{\partial x_\ell \partial x_k} \\ &= H^{d-2}(f(\mathbf{x})) + \sum_{k=1}^{n} a_k \cdot \frac{\partial H^{d-1}(f(\mathbf{x}))}{\partial x_k} + \sum_{\ell,k=1}^{n} b_\ell b_k \frac{\partial^2 H^d(f(\mathbf{x}))}{\partial x_\ell \partial x_k} \end{aligned}$$

we have

$$H^{d-2}(g(\mathbf{x})) - H^{d-2}(f(\mathbf{x})) - \sum_{\ell,k=1}^{n} b_\ell b_k \frac{\partial^2 H^d(f(\mathbf{x}))}{\partial x_\ell \partial x_k} = \sum_{k=1}^{n} a_k \cdot \frac{\partial H^{d-1}(f(\mathbf{x}))}{\partial x_k}.$$

Therefore, we have that polynomial $H^{d-2}(g(\mathbf{x})) - H^{d-2}(f(\mathbf{x})) - \sum_{\ell,k=1}^{n} b_\ell b_k \frac{\partial^2 H^d(f(\mathbf{x}))}{\partial x_\ell \partial x_k}$ is a directional derivative of $H^{d-1}(f(\mathbf{x}))$ and therefore is in $\mathcal{M}_1$. Notice once more that we need to make this simulation to avoid assuming that $\mathcal{M}_2$ is closed under homogeneous parts.

Since we already computed **b**, we have access to the polynomial $H^{d-2}(g(\mathbf{x})) - H^{d-2}(f(\mathbf{x})) - \sum_{\ell,k=1}^{n} b_\ell b_k \frac{\partial^2 H^d(f(\mathbf{x}))}{\partial x_\ell \partial x_k}$ and therefore we can look for a solution to both systems of equations (4) and (8) (as linear systems in the coefficients of **c**).

Once we find such a solution, say $\mathbf{c}$, we can use it to set up a new system of equations involving the homogeneous components of degree $d-3$ and so on.

Thus, our algorithm works in iterations. We start by solving a system of linear equations. We then use the solution that we found to set up another system and then we find a common solution to both systems. We use the solution that we have found to construct a third system of equations and then solve all three systems together etc. Throughout the iterations, we also make sure that we keep working with polynomials that only belong to the circuit class $\mathcal{M}_1$, so that we avoid requiring $\mathcal{M}_2$ to be closed under homogeneous parts. At the end we have a solution for all systems, and at this point it is not difficult to verify, that if such a shift $\mathbf{a}$ exists, then the solution that we found is indeed a valid shift. This can be verified by running one PIT for checking whether the shift of f that we have found and g are equivalent.

All the steps above can be completed using randomness, including solving the black-box system of equations, or using PIT for the relevant circuit classes.

1.3 Related Work

The works of Grigoriev, Lakshman, Saunders and Karpinski [GK93, GL95, LS94], try to solve the problem of finding sparse shifts of given polynomials, in order to make their representation more succinct. In [GK93], Grigoriev and Karpinski studied the problem of finding sparse affine-shifts of multivariate polynomials $f(\mathbf{x})$, that is, transformations of the form $\mathbf{x} \mapsto A\mathbf{x}+\mathbf{b}$ where A is full-rank, which make the input polynomial $f(A\mathbf{x}+\mathbf{b})$ sparse. In [LS94], the authors consider the problem of finding sparse shifts of univariate polynomials, and of determining uniqueness of a sparse shift. Given an input polynomial $f(x)$, they use a criterion based on the vanishing of the Wronskian of some carefully designed polynomials, which depend on the derivatives $f^{(i)}(x)$, in order to obtain an efficient algorithm for the univariate case.

Later, in [Gri97], Grigoriev gave three algorithms for the SET problem, which were polynomial in the size of the dense representation of the input polynomials. His algorithms were based on a structural result about the set of shifts that stabilize the polynomial, that is, the set of points $\mathbf{a} \in \mathbb{F}^n$ for which $f(\mathbf{x}+\mathbf{a}) \equiv f(\mathbf{x})$. We denote this stabilizer by $\mathcal{S}_f$. He noticed that $\mathcal{S}_f$ is a subspace of $\mathbb{F}^n$ and that the set of shifts that are solutions to the SET problem with input polynomials f, g, which we denote $\mathcal{S}_{f,g}$, is a coset of $\mathcal{S}_f$. After this observation, Grigoriev established the following recursive relations between $\mathcal{S}_{f,g}$ and $\mathcal{S}_{\frac{\partial f}{\partial x_i},\frac{\partial g}{\partial x_i}}$, for each x_i:

$$\mathcal{S}_{f,g} = \bigcap_{i=1}^{n} \mathcal{S}_{\frac{\partial f}{\partial x_i},\frac{\partial g}{\partial x_i}} \cap \{\mathbf{a} \in \mathbb{F}^n \ : \ f(\mathbf{a}) = g(0)\}.$$

From these relations, Grigoriev devised a recursive algorithm that finds $\mathcal{S}_{f,g}$ by finding the subspaces corresponding to $\mathcal{S}_{\frac{\partial f}{\partial x_i},\frac{\partial g}{\partial x_i}}$. Because this recursive procedure will find all the subspaces $\mathcal{S}_{\frac{\partial f}{\partial m},\frac{\partial g}{\partial m}}$ for every monomial m of degree less than or equal to $d = \max(d_f, d_g)$, the running time of his algorithm is bounded by

$n^{O(d)}$. Our approach is different from Grigoriev's in the sense that we avoid the recursive relations and find a shift by iteratively constructing a shift which makes f and g agree on their homogeneous parts of up to a certain degree, starting from the homogeneous parts of highest degree down to the homogeneous parts of lowest degree (i.e., the constant term).

The study of equivalences of general polynomials under affine transformations, which we refer to as affine-equivalence, was started by Kayal in [Kay12] (note that this generalizes the problem studied in [GK93]). We say that f and g are affine-equivalent if there exists a matrix A and a shift $\mathbf{b}$ such that $f(\mathbf{x}) = g(A\mathbf{x}+\mathbf{b})$. In this work, Kayal analyzes whether a given polynomial f can be obtained by an affine transformation of a given polynomial g, where g is usually taken to be a "complete" polynomial in some arithmetic circuit class, such as the Determinant or Permanent polynomials. In his paper, Kayal establishes NP-hardness of the general problem of determining affine-equivalence between two arbitrary polynomials. Moreover, he provides randomized algorithms for the affine-equivalence problem when one of the polynomials is the Permanent or the Determinant and the affine transformations $\mathbf{x} \mapsto A\mathbf{x} + \mathbf{b}$ are of a special form (in the case of Determinant and Permanent, the matrix A must be invertible). Kayal provides randomized algorithms for some other classes of homogeneous polynomials, and for more details we refer the reader to the paper [Kay12]. Our work is different from Kayal's work since in our setting we are only interested in shift-equivalences, and in this feature we are less general than Kayal's work, but we also consider larger classes of polynomials, in which case we are more general than Kayal's work.

Following the online publication of this manuscript, an anonymous reader pointed out an alternative way to solve the SET problem using a randomized algorithm. This approach uses a lemma due to Carlini [Car06] (see also [Kay12, Lemma 17]) and an argument implicit in Kayal's work [Kay12, Section 7.3]. We now discuss and compare this alternative approach to ours.

In his lemma, Carlini uses a linear transformation on the variables in order to get rid of "redundant variables," that is, variables x_i for which (after a suitable change of basis) the derivative $\frac{\partial f}{\partial x_i}$ of the polynomial is zero. The idea is to use this lemma to eliminate the "redundant variables" and work only with the "essential variables." Once we find such a linear transformation, one can solve Equation (2) (there will be at most one solution, since there are no more redundant variables). Then, we reduce the original problem to another SET problem on lower degree polynomials by subtracting the homogeneous part of largest degree. We give the details (which do not appear elsewhere in the literature) in the full version of the paper. In a sense, this approach is almost identical to ours. In the first step of our algorithm we solve Equation (2) and get an affine subspace as solution. This affine subspace can be thought of as being composed of the space of all assignments to the "non-essential" variables shifted by the unique solution. Then, in the next step, we prune this space further according to the essential variables of the degree $d-1$ part etc. The advantages of our approach come in when trying to de-randomize SET using deterministic PIT for restricted classes. When following Carlini's lemma and reducing the number of variables, one needs to solve PIT for the composition of the original circuits with a linear transformation. This is not necessary in our approach, which has weaker PIT requirements. While some circuit classes are closed under linear transformations, this is not the case in general. For example, the class of sparse polynomials is not closed under linear transformations. Thus, one will not be able to deduce polynomial time algorithms to certain instances of SET like those that follow from our approach (see Corollary 2 and the discussion following it).

Another issue with the algorithm obtained from Carlini's lemma is that in each step of the recursion we need to subtract an affine shift of the homogeneous component of maximal degree from each of the polynomials. Thus, we need PIT for classes that are closed under linear combinations of polynomials from the class. However, some restricted circuit classes do not satisfy such closure properties. For example, when executing this algorithm on depth-3 circuits with bounded top fan-in, we may get, at some step of the algorithm, a depth-3 circuit with unbounded top fan-in and so we will not be able to use current deterministic algorithms.

2 Conclusion and Open Questions

In this paper, we reduced the problem of shift-equivalence to the problem of solving PIT, and as a consequence of this reduction we obtained a polynomial-time randomized algorithm for the shift-equivalence problem, over characteristic zero or when the characteristic of the base field is larger than the degrees of the polynomials.

We gave some examples for classes of circuits where this can be performed deterministically in quasi-polynomial time. One example where we "almost" have such a result is when testing whether a given sparse polynomial is equivalent to a shift of another sparse polynomial. Note that while the class of sparse polynomials is closed under partial derivatives and homogeneous components, it is not closed under shifts and so we cannot use our approach. Nevertheless, it is quite likely that this simple case can be solved using other techniques.

Acknowledgments. The authors would like to thank an anonymous reader for the remark on the usage of Carlini's lemma and of Kayal's implicit approach to give the alternative solution to the SET problem in the randomized case.

The third author would like to thank Michael Forbes, Ankit Gupta, Elad Haramaty, Swastik Kopparty, Ramprasad Saptharishi and Shubhangi Saraf for helpful discussions on related problems.

References

[ASS13] Agrawal, M., Saha, C., Saxena, N.: Quasi-polynomial hitting-set for set-depth-d formulas. In: STOC, pp. 321–330 (2013)

[BS83] Baur, W., Strassen, V.: The complexity of partial derivatives. Theoretical Computer Science 22(3), 317–330 (1983)

[Car06] Carlini, E.: Reducing the number of variables of a polynomial. In: Algebraic geometry and geometric modeling, pp. 237–247. Springer, Heidelberg (2006)

[DL78] DeMillo, R.A., Lipton, R.J.: A probabilistic remark on algebraic program testing. Inf. Process. Lett. 7(4), 193–195 (1978)

[FS12] Forbes, M.A., Shpilka, A.: On identity testing of tensors, low-rank recovery and compressed sensing. In: Proceedings of the 44th Annual STOC, pp. 163–172 (2012)

[FS13] Forbes, M.A., Shpilka, A.: Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs. In: Proceedings of the 54th Annual FOCS (2013)

[FSS13] Forbes, M.A., Saptharishi, R., Shpilka, A.: Pseudorandomness for multilinear read-once algebraic branching programs, in any order. Electronic Colloquium on Computational Complexity (ECCC) 20, 132 (2013)

[GK93] Grigoriev, D., Karpinski, M.: A zero-test and an interpolation algorithm for the shifted sparse polynomials. In: Moreno, O., Cohen, G., Mora, T. (eds.) AAECC 1993. LNCS, vol. 673, pp. 162–169. Springer, Heidelberg (1993)

[GL95] Grigoriev, D., Lakshman, Y.N.: Algorithms for computing sparse shifts for multivariate polynomials. In: Proceedings of the 1995 International Symposium on Symbolic and Algebraic Computation, ISSAC 1995, pp. 96–103. ACM, New York (1995)

[Gri97] Grigoriev, D.: Testing shift-equivalence of polynomials by deterministic, probabilistic and quantum machines. Theoretical Computer Science 180(1-2), 217–228 (1997)

[Kay12] Kayal, N.: Affine projections of polynomials: extended abstract. In: Proceedings of the 44th symposium on Theory of Computing, STOC 2012, pp. 643–662. ACM, New York (2012)

[KS01] Klivans, A., Spielman, D.: Randomness efficient identity testing of multivariate polynomials. In: Proceedings of the 33rd Annual STOC, pp. 216–223 (2001)

[KS09] Karnin, Z.S., Shpilka, A.: Reconstruction of generalized depth-3 arithmetic circuits with bounded top fan-in. In: Proceedings of the 24th Annual CCC, pp. 274–285 (2009)

[LS94] Lakshman, Y.N., Saunders, B.D.: On computing sparse shifts for univariate polynomials. In: Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC 1994, pp. 108–113. ACM, New York (1994)

[Sch80] Schwartz, J.T.: Fast probabilistic algorithms for verification of polynomial identities. J. ACM 27(4), 701–717 (1980)

[Shp09] Shpilka, A.: Interpolation of depth-3 arithmetic circuits with two multiplication gates. SIAM J. on Computing 38(6), 2130–2161 (2009)

[SSS13] Saha, C., Saptharishi, R., Saxena, N.: A case of depth-3 identity testing, sparse factorization and duality. Computational Complexity, 1–31 (2013)

[Zip79] Zippel, R.: Probabilistic algorithms for sparse polynomials. In: Ng, K.W. (ed.) EUROSAM 1979 and ISSAC 1979. LNCS, vol. 72, pp. 216–226. Springer, Heidelberg (1979)

Optimal Analysis of Best Fit Bin Packing

György Dósa[1] and Jiří Sgall[2]

[1] Department of Mathematics, University of Pannonia, Veszprém, Hungary
dosagy@almos.vein.hu

[2] Computer Science Institute of Charles University, Praha, Czech Republic
sgall@iuuk.mff.cuni.cz

Abstract. In early seventies it was shown that the *asymptotic* approximation ratio of BestFit bin packing is equal to 1.7. We prove that also the *absolute* approximation ratio for BestFit bin packing is exactly 1.7, improving the previous bound of 1.75. This means that if the optimum needs Opt bins, BestFit always uses at most $\lfloor 1.7 \cdot \mathrm{OPT} \rfloor$ bins. Furthermore we show *matching lower bounds* for all values of Opt, i.e., we give instances on which BestFit uses exactly $\lfloor 1.7 \cdot \mathrm{OPT} \rfloor$ bins. Thus we completely settle the worst-case complexity of BestFit bin packing after more than 40 years of its study.

1 Introduction

Bin packing is a classical combinatorial optimization problem in which we are given an instance consisting of a sequence of items with rational sizes between 0 and 1, and the goal is to pack these items into the smallest possible number of bins of unit size. BestFit algorithm packs each item into the most full bin where it fits, possibly opening a new bin if the item does not fit into any currently open bin. A closely related FirstFit algorithm packs each item into the first bin where it fits, again opening a new bin only if the item does not fit into any currently open bin.

Johnson's thesis [8] on bin packing together with Graham's work on scheduling [6,7] belong to the early influential works that started and formed the whole area of approximation algorithms. The proof that the asymptotic approximation ratio of FirstFit and BestFit bin packing is 1.7 given by Ullman [14] and subsequent works by Garey et al. and Johnson et al. [5,10] were among these first results on approximation algorithms.

We prove that also the *absolute* approximation ratio for BestFit bin packing is exactly 1.7, i.e., if the optimum needs Opt bins, BestFit always uses at most $\lfloor 1.7 \cdot \mathrm{Opt} \rfloor$ bins. This builds upon and substantially generalizes our previous upper bound for FirstFit from [3]. For the comparison of the techniques, see the beginning of Section 4. Furthermore we show *matching lower bounds* for *all* values of Opt, i.e., we give instances on which BestFit and FirstFit use exactly $\lfloor 1.7 \cdot \mathrm{Opt} \rfloor$ bins. This is also the first construction of an instance that has absolute approximation ratio exactly 1.7 for an arbitrarily large Opt.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 429–441, 2014.

Note that the upper bound for BestFit is indeed a generalization of the bound for FirstFit: The items in any instance can be reordered so that they arrive in the order of bins in the FirstFit packing. This changes neither FirstFit, nor the optimal packing. Thus it is sufficient to analyze FirstFit on such instances. On the other hand, on them BestFit behaves exactly as FirstFit, as there is always a single bin where the new item fits. Thus any lower bound for FirstFit applies immediately to BestFit and any upper bound for FirstFit is equivalent to a bound for BestFit for this very restricted subset of instances. To demonstrate that the extension of the absolute bound from FirstFit to BestFit is by far not automatic, we present a class of any-fit-type algorithms for which the asymptotic bound of 1.7 holds, but the absolute bound does not.

History and Related Work. The upper bound on BestFit (and FirstFit) was first shown by Ullman in 1971 [14]; he proved that for any instance, $\mathrm{BF}, \mathrm{FF} \leq 1.7 \cdot \mathrm{Opt} + 3$, where BF, FF and Opt denote the number of bins used by BestFit, FirstFit and the optimum, respectively. Still in seventies, the additive term was improved first in [5] to 2 and then in [4] to $\mathrm{BF} \leq \lceil 1.7 \cdot \mathrm{Opt} \rceil$; due to integrality of BF and Opt this is equivalent to $\mathrm{BF} \leq 1.7 \cdot \mathrm{Opt} + 0.9$. Recently the additive term of the asymptotic bound was improved for FirstFit to $\mathrm{FF} \leq 1.7 \cdot \mathrm{Opt} + 0.7$ in [16] and to $\mathrm{FF} \leq 1.7 \cdot \mathrm{Opt}$ in [3].

The absolute approximation ratio of FirstFit and BestFit was bounded by 1.75 by Simchi-Levy [13]. Recent improvements again apply only to FirstFit: after bounds of $12/7 \approx 1.7143$ by Xia and Tan [16] and Boyar et al. [1] and $101/59 \approx 1.7119$ by Németh [11], the tight bound of $\mathrm{FF} \leq 1.7 \cdot \mathrm{Opt}$ was given in our previous work [3].

For the lower bound, the early works give examples both for the asymptotic and absolute ratios. The example for the asymptotic bound gives $\mathrm{FF} = 17k$ whenever $\mathrm{Opt} = 10k + 1$, thus it shows that the asymptotic upper bound of 1.7 is tight, see [14,5,10]. For the absolute ratio, an example is given with $\mathrm{FF} = 17$ and $\mathrm{Opt} = 10$, i.e., an instance with approximation ratio exactly 1.7 [5,10], but no such example was known for large Opt. In our previous work [3] we have given lower bound instances with $\mathrm{BF} = \mathrm{FF} = \lfloor 1.7 \cdot \mathrm{Opt} \rfloor$ whenever $\mathrm{Opt} \not\equiv 0, 3 \pmod{10}$.

We have mentioned only directly relevant previous work. Of course, there is much more work on bin packing, in particular there exist asymptotic approximation schemes for this problem, as well as many other algorithms. We refer to the survey [2] or to the recent excellent book [15].

Organization of the Paper. The crucial technique of the upper bound is a combination of amortization and weight function analysis, following the scheme of our previous work [12,3]. We present it first in Section 2 to give a simple proof of the asymptotic bound for BestFit and any-fit-type algorithms. We prove the lower bound in Section 3, as it illustrates well the issues that we need to deal with in the upper bound proof, which is then given in Section 4. Most proofs are omitted, but we try to explain the main ideas behind them. For a version with all proofs, see `http://iuuk.mff.cuni.cz/~sgall/ps/BF.pdf`.

2 Notations and the Simplified Asymptotic Bound

Let us fix an instance I with items $a_1, \ldots, a_n$ and denote the number of bins in the BestFit and optimal solutions by BF and Opt, respectively. We will often identify an item and its size. For a set of items A, let $s(A) = \sum_{a \in A} a$, i.e., the total size of items in A and also for a set of bins $\mathcal{A}$, let $s(\mathcal{A}) = \sum_{A \in \mathcal{A}} s(A)$. Furthermore, let $S = s(I)$ be the total size of all items of I. Obviously $S \leq \text{Opt}$.

We classify the items by their sizes: items $a \leq 1/6$ are **small**, items $a \in (1/6, 1/2]$ are **medium**, and items $a > 1/2$ are **huge**. A bin is called a **k-bin** or **k^+-bin**, if it contains exactly k items or at least k items, respectively, in the final packing. Furthermore, the **rank** of a bin is the number of medium and huge items in it. An item is called **k-item** if BF packs it into a k-bin.

The bins in the BF packing are ordered by the time they are opened (i.e., when the first item is packed into them). Expressions like "before", "after", "first bin", "last bin" refer to this ordering. At any time during the packing, the **level of a bin** is the total size of items currently packed in it, while by **size of a bin** we always mean its final level. A **level of an item** a denotes the level of the bin where a is packed, just before the packing of a.

The following properties of BestFit follow easily from its definition.

Lemma 2.1. *At any moment, in the* BF *packing the following holds:*

(i) *The sum of levels of any two bins is greater than 1. In particular, there is at most one bin with level at most* 1/2.

(ii) *Any item a with level at most* 1/2 *(i.e., packed into the single bin with level at most* 1/2*) does not fit into any bin open at the time of its arrival, except for the bin where the item a is packed.*

(iii) *If there are two bins B, B' with level at most* 2/3*, in this order, then either B' contains a single item or the first item in B' is huge.* □

To illustrate our technique, we now present a short proof of the asymptotic ratio 1.7 for BestFit. It uses the same weight function as the traditional analysis of BestFit. (In some variants the weight of an item is capped to be at most 1, which makes almost no difference in the analysis.) To use amortization, we split the weight of each item a into two parts, namely its bonus $\overline{w}(a)$ and its scaled size $\overline{\overline{w}}(a)$, defined as

$$\overline{w}(a) = \begin{cases} 0 & \text{if } a \leq \frac{1}{6}\,, \\ \frac{3}{5}(a - \frac{1}{6}) & \text{if } a \in \left(\frac{1}{6}, \frac{1}{3}\right)\,, \\ 0.1 & \text{if } a \in \left[\frac{1}{3}, \frac{1}{2}\right]\,, \\ 0.4 & \text{if } a > \frac{1}{2}\,. \end{cases}$$

For every item a we define $\overline{\overline{w}}(a) = \frac{6}{5}a$ and its weight is $w(a) = \overline{\overline{w}}(a) + \overline{w}(a)$. For a set of items B, $w(B) = \sum_{a \in B} w(a)$ denotes the total weight, similarly for $\overline{w}$ and $\overline{\overline{w}}$.

It is easy to observe that the weight of any bin B, i.e., of any set with $s(B) \leq 1$, is at most 1.7: The scaled size of B is at most 1.2, so we only need to check that

$\overline{w}(B) \leq 0.5$. If B contains no huge item, there are at most 5 items with non-zero $\overline{w}(a)$ and $\overline{w}(a) \leq 0.1$ for each of them. Otherwise the huge item has bonus 0.4; there are at most two other medium items with non-zero bonus and it is easy to check that their total bonus is at most 0.1. This implies that the weight of the whole instance is at most $1.7 \cdot \text{Opt}$.

The key part is to show that, on average, the weight of each BF-bin is at least 1. Lemma 2.2 together with Lemma 2.1 implies that for almost all bins with two or more items, its scaled size plus the bonus of the *following* such bin is at least 1.

Lemma 2.2. *Let B be a bin such that $s(B) \geq 2/3$ and let c, c' be two items that do not fit into B, i.e., $c, c' > 1 - s(B)$. Then $\overline{\overline{w}}(B) + \overline{w}(c) + \overline{w}(c') \geq 1$.*

Proof. If $s(B) \geq 5/6$, then $\overline{\overline{w}}(B) \geq 1$ and we are done. Otherwise let $x = 5/6 - s(B)$. We have $0 < x \leq 1/6$ and thus $c, c' > 1/6 + x$ implies $\overline{w}(c), \overline{w}(c') > \frac{3}{5}x$. We get $\overline{\overline{w}}(B) + \overline{w}(c) + \overline{w}(c') > \frac{6}{5}(\frac{5}{6} - x) + \frac{3}{5}x + \frac{3}{5}x = 1$. □

Any BF-bin D with a huge item has $\overline{w}(D) \geq 0.4$ and $\frac{6}{5}s(D) > 0.6$, thus $w(D) > 1$.

For the amortization, consider all BF-bins B with two or more items, size $s(B) \geq 2/3$, and no huge item. For any such bin except for the last one choose C as the next bin with the same properties. Since C has no huge item, its first two items c, c' have level at most 1/2 and by Lemma 2.1(ii) they do not fit into B. Lemma 2.2 implies $\overline{\overline{w}}(B) + \overline{w}(C) \geq \overline{\overline{w}}(B) + \overline{w}(c) + \overline{w}(c') \geq 1$.

Summing all these inequalities (note that each bin is used at most once as B and at most once as C) and $w(D) > 1$ for the bins with huge items we get $w(I) \geq \text{BF} - 3$. The additive constant 3 comes from the fact that we are missing an inequality for at most three BF-bins: the last one from the amortization sequence, possibly one bin B with two or more items and $s(B) < 2/3$ (cf. Lemma 2.1(iii)) and possibly one bin B with a single item and $s(B) < 1/2$ (cf. Lemma 2.1(i)). Combining this with the previous bound on the total weight, we obtain $\text{BF} - 3 \leq w(I) \leq 1.7 \cdot \text{Opt}$ and the asymptotic bound follows.

This simple proof holds for a wide class of any-fit-type algorithms: Call an algorithm a RAAF (relaxed almost any fit) algorithm, if it uses the bin with level at most 1/2 only when the item does not fit into any previous bin (Lemma 2.1(i) implies that there is always at most one such bin). Our proof of the asymptotic ratio can be tightened so that the additive constant is smaller:

Theorem 2.3. *For any RAAF algorithm A and any instance of bin packing we have $A \leq \lfloor 1.7 \cdot \text{Opt} + 0.7 \rfloor \leq \lceil 1.7 \cdot \text{Opt} \rceil$.* □

The proof is given in the full version, where we also give an example of a RAAF algorithm which does not satisfy the absolute bound of 1.7. The asymptotic bound for almost any fit (AAF) algorithms was proved in [8,9], where the original AAF condition prohibits packing an item in the smallest bin if that bin is unique and the item does fit in some previous bin (but the restriction holds also if the smallest bin is larger than 1/2). Theorem 2.3 improves the additive term and generalizes the bound from AAF to the slightly less restrictive RAAF condition (although it seems that the original proof also uses only the RAAF condition).

3 Lower Bound

The high level scheme of the lower bound for $\text{Opt} = 10k$ is this: For a tiny $\varepsilon > 0$, the instance consists of Opt items of size approximately $1/6$, followed by Opt items of size approximately $1/3$, followed by Opt items of size $1/2+\varepsilon$. The optimum packs in each bin one item from each group. BestFit packs the items of size about $1/6$ into $2k$ bins with 5 items, with the exception of the first and last of these bins that will have 6 and 4 items, respectively. The items of size about $1/3$ are packed in pairs. To guarantee this packing, the sizes of items differ from $1/6$ and $1/3$ in both directions by a small amount δ_i which is exponentially decreasing, but greater than ε for all i. This guarantees that only the item with the largest δ_i in a bin is relevant for its final size and this in turn enables us to order the items so that no additional later item fits into these bins.

Theorem 3.1. *For all values of* Opt*, there exists an instance I with* $\text{FF} = \text{BF} = \lfloor 1.7 \cdot \text{Opt} \rfloor$. □

4 Upper Bound

At the high level, we follow the weight function argument from the simple proof in Section 2. As we have seen, the BF packing in the lower bound contains three types of bins that play different roles. To obtain the tight upper bound, we analyze them separately. For two of these types we can argue easily that the weight of each bin is at least 1: First, the bins with size at least $5/6$, called big bins below; these are the initial bins in the lower bound containing the items of size approximately $1/6$. Second, the 1-bins, called dedicated bins; these are the last bins with items $1/2+\varepsilon$ in the lower bound. The remaining bins, common bins, are the middle bins of size approximately $2/3$ with items of size around $1/3$ (except for the first bin) in the lower bound. They are analyzed using the amortization lemma. This general scheme has several obstacles which we describe now, together with the intuition behind their solution.

Obstacle 1. There can be one dedicated bin with item $d_0 < 1/2$. We need to change its bonus to approximately 0.4, to guarantee a sufficient weight of this bin. This in turn possibly forces us to decrease the bonus of one huge item f_1 to 0.1, if d_0 and f_1 are in the same Opt-bin, so that the Opt-bin has weight at most 1.7.

Obstacle 2. The amortization lemma needs two items that do not fit into the previous bin. Unlike FirstFit, BestFit does not guarantee this, if the first item in a bin is huge. If this first item is not f_1, we can handle such bins, called huge-first bins, similarly as dedicated bins. If this happens for f_1, we need to argue quite carefully to find the additional bonus. This case, called the freaky case, is the most complicated part of our analysis.

Obstacle 3. Even on the instances similar to our lower bound, the amortization leaves us with the additive term of 0.1, because we cannot use the amortization on the last common bin, and its scaled weight is only about 0.8 if its size is around 2/3. Here the parity of the items of size around 1/3 comes into play: Typically they come in pairs in BF-bins, as in the lower bound, but for odd values of Opt one of them is missing or is in a FirstFit bin of 3 or more items. This allows us to remove the last 0.1 of the additive term, using the mechanism of an exceptional set, see Definition 4.9.

Obstacle 4. If the last common bin is smaller than 2/3, the problem with amortizing it is even larger. Fortunately, then the previous common bins are larger than 2/3 and have additional weight that can compensate for this, using a rather delicate argument, see Proposition 4.11.

Notations and Preliminary Lemmata

We classify the BF bins into four groups.

Any 1-bin D is a **dedicated bin**; $\mathcal{D}$ denotes the set of all dedicated bins and δ their number. If some dedicated bin has size at most 1/2, denote the item in it d_0 and let $\Delta = 1/2 - d_0$; otherwise d_0 and Δ are undefined. Lemma 2.1(i) implies that there is at most one such item; also we shall see that in the tightest case Δ is close to 0.

If d_0 is defined, there may exist a (unique) huge item in its Opt-bin. In that case, denote it f_1 for the rest of the proof and leave f_1 undefined otherwise. Furthermore, if f_1 is the first item in a BF-bin, denote that bin F for the rest of the proof; otherwise let F undefined. Note that F cannot be a 1-bin as otherwise d_0 would fit there contradicting Lemma 2.1(i). Let f_2 be the second item in F.

If the first item of a 2^+-bin H is huge and $H \neq F$, we call H a **huge-first** bin; $\mathcal{H}$ denotes the set of all huge-first bins and η their number.

If a 2^+-bin B satisfies $s(B) \geq 5/6$ and it is not in $\mathcal{H}$, we call it a **big bin**; $\mathcal{B}$ denotes the set of all big bins and β their number.

Any remaining bin (i.e., any 2^+-bin with size less than 5/6 and the first item small or medium, and also F if it is defined and not a big bin) is called a **common bin**; $\mathcal{C}$ denotes the set of all common bins, and γ their number.

An item is called an **H-item**, if it is d_o or a huge item different from f_1 (if defined). Note that each Opt-bin and each BF-bin contains at most one H-item.

The definitions imply that in every big or common bin different from F (if defined), the first item is small or medium. Then Lemma 2.1(ii) implies that the first two items of the bin do not fit into any previous bin.

Throughout the proof we distinguish two cases depending on the bin F.

If F is not defined, or it is a big bin, or f_2 does not fit into any previous common bin, then we call this the **regular case** and all the common bins, including F if it is defined and a common bin, are called **regular bins**.

If F is defined and it is a common bin, and f_2 would fit into some previous common bin at the time of its packing, fix one such bin G for the rest of the proof. We call this the **freaky case** and F the **freaky bin**. All the other common bins are called **regular bins**.

In both cases, denote the set of all regular bins by $\mathcal{R}$ and their number by ρ, furthermore number the regular bins $C_1, \ldots, C_\rho$, ordered by the time of their opening. In the freaky case, let g be the index of bin G in this ordering, i.e., let $C_g = G$. Note that $\rho = \gamma$ in the regular case and $\rho = \gamma - 1$ in the freaky case.

In the following lemma we significantly reduce the set of instances that we need to consider in our proof. Our goal is to reorder or remove the items in the sequence so that BestFit packs most items similarly as FirstFit. For these transformations, we use two important properties of BestFit that follow from its definition. First, if we remove all the items from a BF-bin from the instance, the packing of the remaining items into the remaining bins does not change; often we use this so that we move the items to a later position in the instance and then this implies that the packing of the initial segment before the new position of the moved items does not change. Second, if two instances lead to the same configuration and we extend them by the same set of items, then the resulting configurations are also the same, where the configuration is the current multiset of levels of BF-bins. (This does not hold for FirstFit, as permuting the bins can change the subsequent packing, but the configuration is the same.)

Lemma 4.1. *If there exists an instance with* BF $> 1.7 \cdot$ Opt*, then there exists such an instance I that in addition satisfies the following properties:*

(i) *All the* 1*-items form a final segment of the input instance.*

(ii) *If a* BF *bin B contains an item a such that for all other* BF *bins B' we have $a + s(B') > 1$ then B is an* 1*-bin.*

(iii) *In each* BF 2^+*-bin, the first two items are a_{j-1} and a_j for some j (i.e., they are adjacent in I). Furthermore, these two items are packed into different bins in* Opt.

(iv) *Suppose that for a* BF 3^+*-bin B, the first item in B is not huge, and no new bin is opened after opening B and before packing the third item into B. Then the first three items packed into B are a_{j-2}, a_{j-1} and a_j for some j (i.e., they are adjacent in I). Furthermore, these three items are packed into different bins in* Opt.

(v) *Suppose that a_j is the last item packed into a* BF *bin B. Then for all $j' > j$, we have $a_{j'} + s(B) > 1$ (i.e., no later items fit into B). Consequently, no later item has level $s(B)$ or larger in* BF *packing.* □

For the rest of the proof we assume that our instance I satisfies the properties from Lemma 4.1. The following lemma states the consequences for the common bins: The medium items are packed as in FirstFit and the small items are restricted to only first few common bins.

Lemma 4.2. (i) *Any item $a_j > 1/6$ packed into a regular bin C_i has the property that at the time of its packing, a_j does not fit into any previous common bin.*

(ii) *If a small item a_j is packed into a common bin, then this is a common bin with the largest level at the time of packing a_j. Except for C_1 and F, any small item in a common bin has level at least $2/3$.*

(iii) *From the moment when there are two common bins with level at least $2/3$ on, no small item arrives. In particular, no small item is packed into a common bin opened later than C_2.*

(iv) *If $a_j \in C_2$ is small, some $a_k > 1/6$, $k > j$ (i.e., a_k is after a_j), is packed into C_1.* □

In the next lemma we state some properties important for the freaky case. For the rest of the proof, let g_0 denote the item in bin G guaranteed by the next lemma. Note that the lemma implies that there are at least three items packed into G, as there are two other items in G when F opens.

Lemma 4.3. *In the freaky case, the* BF *packing satisfies the following:*

(i) *There exists an item g_0 that is packed into bin G such that g_0 arrives after f_2 and $s(F) + g_0 > 1$. Furthermore, $s(F) + s(G) > 1 + d_0$.*

(ii) *If the regular bins C_i and C_k are opened before F then $s(F) > 2/3$ and $s(C_i) + s(C_k) + s(F) > 2$.* □

Lemma 4.4. *In the* BF *packing the following holds:*

(i) *The total size of any $k \geq 2$* BF*-bins is greater than $k/2$.*

(ii) *If d_0 is defined, then $s(\mathcal{H} \cup \mathcal{D}) \geq (\delta + \eta)/2 + (\delta + \eta - 2)\varDelta$.*

(iii) *The total number of huge-first and dedicated bins is $\delta + \eta \leq$* Opt.

(iv) *Suppose that C is a regular bin of size $s(C) = 2/3 - 2x$ for some $x \geq 0$. For any bin B before C we have $s(B) > 2/3 + x$ and for any regular or big bin B after C we have $s(B) > 2/3 + 4x$.*

(v) *Suppose we have a set $\mathcal{A}$ of k common and big bins such that there are at least 3 common bins among them. Then $s(\mathcal{A}) > 2k/3$.* □

Now we assume that the instance violates the absolute ratio 1.7 and derive some easy consequences that exclude some degenerate cases. Note that the values of $1.7 \cdot \text{Opt}$ are multiples of 0.1 and BF is an integer, thus $\text{BF} > 1.7 \cdot \text{Opt}$ implies $\text{BF} \geq 1.7 \cdot \text{Opt} + 0.1$. Typically we derive a contradiction with the lower bound $S \leq \text{Opt}$ on the optimum.

Lemma 4.5. *If* $\text{BF} > 1.7 \cdot \text{Opt}$ *then the following holds:*

(i) $\text{Opt} \geq 7$.

(ii) *No common bin C has size $s(C) \leq 1/2$.*

(iii) *The total number of dedicated and huge-first bins is bounded by $\eta + \delta \geq 5$. If d_0 is not defined then there is no huge-first bin, i.e., $\eta = 0$.*

(iv) *The number of regular bins is at least $\rho \geq \text{Opt}/2 + 1 > 4$. If* $\text{BF} \geq 1.7 \cdot \text{Opt} + \tau/10$ *for some integer $\tau \geq 1$ then $\rho > (\text{Opt} + \tau)/2$.* □

The Weight Function, Amortization, Exceptional Set

Now we give the modified and final definition of the weight function. The weight is modified only for d_0 and f_1 and their modified bonus is at least 0.1. Thus Lemma 2.2 still holds, as its proof uses at most 0.1 of bonus for each item.

Definition 4.6. *The weight function w, bonus $\overline{w}$ and scaled size $\overline{\overline{w}}$ are defined as follows:*

If d_0 is defined, we define $\overline{w}(d_0) = 0.4 - \frac{3}{5}\Delta$.
If f_1 is defined, we define $\overline{w}(f_1) = 0.1$

For any other item a, we define
$$\overline{w}(a) = \begin{cases} 0 & \text{if } a \le \frac{1}{6}, \\ \frac{3}{5}(a - \frac{1}{6}) & \text{if } a \in \left[\frac{1}{6}, \frac{1}{3}\right], \\ 0.1 & \text{if } a \in \left[\frac{1}{3}, \frac{1}{2}\right], \\ 0.4 & \text{if } a > \frac{1}{2}. \end{cases}$$

For every item a we define $\overline{\overline{w}}(a) = \frac{6}{5}a$ *and* $w(a) = \overline{\overline{w}}(a) + \overline{w}(a)$.
For a set of items A and a set of bins $\mathcal{A}$, let $w(A)$ and $w(\mathcal{A})$ denote the total weight of all items in A or $\mathcal{A}$; similarly for $\overline{\overline{w}}$ and $\overline{w}$. Furthermore, let $W = w(I)$ be the total weight of all items of the instance I.

Note that H-items are exactly the items with bonus greater than 0.1.

In the previous definition, the function $\overline{w}$ is continuous on the case boundaries, except for a jump at 0.4. Furthermore, if we have a set A of k items with size in $[\frac{1}{6}, \frac{1}{3}]$ and $d_0 \notin A$, then the definition implies that the bonus of A is exactly $\overline{w}(A) = \frac{3}{5}\left(s(A) - \frac{k}{6}\right)$. More generally, if A contains at least k items and no H-item, then we get an upper bound $\overline{w}(A) \le \frac{3}{5}\left(s(A) - \frac{k}{6}\right)$.

The analysis of Opt-bins and big, dedicated and huge-first BF-bins in the next two lemmata is easy.

Lemma 4.7. *For every optimal bin A its weight $w(A)$ can be bounded as follows:*
(i) *$w(A) \le 1.7$.*
(ii) *If A contains no H-item, then $w(A) \le 1.5$.* □

Lemma 4.8. (i) *The total weight of the big bins is $w(\mathcal{B}) \ge \overline{\overline{w}}(\mathcal{B}) \ge \beta$.*
(ii) *The total weight of the dedicated and huge-first bins is $w(\mathcal{D} \cup \mathcal{H}) \ge \delta + \eta$.* □

The analysis of the common bins is significantly harder. Typically we prove that their weight is at least $\gamma - 0.2$ which easily implies that BF $\le 1.7 \cdot$ Opt$+0.1$. Due to the integrality of BF and Opt, this implies our main result whenever Opt $\not\equiv 7 \pmod{10}$. To tighten the bound by the remaining 0.1 and to analyze the freaky case, we need to reserve the bonus of some of the items in the common bins instead of using it for amortization; this is possible if we still have two items in each regular bins whose bonus we can use. Now we define a notion of an exceptional set E, which contains these items with reserved bonus. In the freaky case, $g_0 \in E$, as its bonus is always needed to amortize for F. Other items are added to E only if Opt $\equiv 7 \pmod{10}$, depending on various cases.

Definition 4.9. *A set of items E is called an* **exceptional set** *if*

(i) *for each $i = 2, \ldots, \rho$, the bin C_i contains at least two items $c, c' > \frac{1}{6}$ that are not in E;*

(ii) *if* $\textsc{Opt} \not\equiv 7 \pmod{10}$ *then $E = \emptyset$ in the regular case and $E = \{g_0\}$ in the freaky case; and*

(iii) *if* $\textsc{Opt} \equiv 7 \pmod{10}$ *then E has at most two items and $g_0 \in E$ in the freaky case.*

The next lemma modifies the amortization lemma for the presence of the exceptional set.

Lemma 4.10. (i) *Let $i = 2, \ldots, \rho$ and $s(C_{i-1}) \geq 2/3$. Then $\overline{\overline{w}}(C_{i-1}) + \overline{w}(C_i \setminus E) \geq 1$.*

(ii) *In the freaky case, if $s(F) \geq 2/3$ then $\overline{\overline{w}}(F) + \overline{w}(f_1) + \overline{w}(g_0) \geq 1$.*

Proof. **(i):** Let $c, c' > \frac{1}{6}$ be two items in $C_i \setminus E$; their existence is guaranteed by the definition of the exceptional set. By Lemma 4.2(i), $c, c' > 1 - s(C_{i-1})$. The claim follows by Lemma 2.2 (which applies even to the modified weights, as we noted before).

(ii): Lemma 4.3(i) implies $g_0 > 1 - s(F)$. Trivially, $f_1 > 1/2 > 1 - s(F)$. Thus we can apply Lemma 2.2 with $c = g_0$ and $c' = f_1$ and the claim follows. □

Analyzing the Common Bins

The following proposition is relatively straightforward if $s(C_\rho) \geq 2/3$, otherwise it needs a delicate argument. It implies easily our upper bound with the additive term 0.1.

Proposition 4.11. *Let* $\textsc{Opt} \geq 8$, $\mathrm{BF} > 1.7 \cdot \textsc{Opt}$, *and E be an exceptional set. Then:*

(i) *$w(\mathcal{R}) - \overline{w}(E) \geq \rho - 0.2$.*

(ii) *If C_ρ has rank at least 3 then $w(\mathcal{R}) - \overline{w}(E) \geq \rho$.*

(iii) *In the freaky case, if $E = \{g_0\}$, and $G = C_g \neq C_\rho$ then we have $w(\mathcal{R}) - \overline{w}(E) - \overline{\overline{w}}(C_g) - \overline{w}(C_{g+1}) \geq \rho - 1.2$.* □

Proposition 4.12. *For any instance of bin packing with* $\textsc{Opt} \geq 8$, *we have* $W > \mathrm{BF} - 0.2$ *and* $W \leq 1.7 \cdot \textsc{Opt}$. *Thus also* $\mathrm{BF} \leq 1.7 \cdot \textsc{Opt} + 0.1$.

Proof. Suppose that $\mathrm{BF} > 1.7 \cdot \textsc{Opt}$. First we show that $w(\mathcal{C}) \geq \gamma - 0.2$, distinguishing three cases.

In the regular case we set $E = \emptyset$ and Proposition 4.11(i) gives $w(\mathcal{C}) \geq \gamma - 0.2$.

In the freaky case, if $s(F) \geq 2/3$, we set $E = \{g_0\}$, then sum Lemma 4.10(ii) and Proposition 4.11(i) to obtain $w(\mathcal{C}) = w(\mathcal{R}) - \overline{w}(E) + \overline{w}(g_0) + w(F) > \rho - 0.2 + 1 = \gamma - 0.2$.

In the freaky case, if $s(F) < 2/3$, then Lemma 4.3(ii) implies that F opens before C_2 and $G = C_1$. Each C_j, $j \geq 2$, contains two items larger than $1/3$, thus $w(C_j) > 1$. Finally, $f_1 < 2/3$, thus the level of C_1 when F opens is greater than $1/3$. Using Lemma 4.3(i) we have $s(F) + g_0 > 1$, thus also $g_0 > 1/3$ and $\overline{w}(g_0) \geq$

0.1. Thus $w(G)+w(F) \geq \overline{\overline{w}}(G)+\overline{\overline{w}}(F)+\overline{w}(g_0)+\overline{w}(f_1) \geq \frac{6}{5}(\frac{1}{3}+1)+0.1+0.1 = 1.8$. Summing this with $w(C_j) > 1$ for $j \geq 2$ we obtain $w(\mathcal{C}) > \gamma - 0.2$ as well.

Together with Lemma 4.8, $w(\mathcal{C}) > \gamma - 0.2$ implies $W = w(\mathcal{B})+w(\mathcal{D})+w(\mathcal{H})+w(\mathcal{C}) > \beta+\eta+\delta+(\gamma-0.2) = \mathrm{BF}-0.2$. By Lemma 4.7(i) we have $W \leq 1.7 \cdot \textsc{Opt}$. Thus $\mathrm{BF} - 0.2 < W \leq 1.7 \cdot \textsc{Opt}$. Since BF and Opt are integers the theorem follows. □

Now after having proved $\mathrm{BF} \leq 1.7 \cdot \textsc{Opt} + 0.1$, we are going to prove our main result.

Theorem 4.13. *For any instance of bin packing we have* $\mathrm{BF} \leq 1.7 \cdot \textsc{Opt}$.

Proof. Suppose the theorem does not hold. Then Proposition 4.12 implies $\mathrm{BF} = 1.7 \cdot \textsc{Opt} + 0.1$ and integrality of Opt and BF then gives $\textsc{Opt} \equiv 7 \pmod{10}$, in particular Opt is odd.

In general, we try to save 0.1 in the analysis of the common bins, i.e., to prove $w(\mathcal{C}) > \gamma - 0.1$. In some of the subcases we need to use some additional weight of other bins and we then show $W > \mathrm{BF} - 0.1$. In both cases we then get $\mathrm{BF} - 0.1 < W \leq 1.7 \cdot \textsc{Opt}$ and the theorem follows by integrality of BF and Opt. In a few remaining cases we derive a contradiction directly.

The proof splits into three significantly different cases, $\textsc{Opt} = 7$, the regular case, and the freaky case. We give only a sketch of the proof in the freaky case. The next lemma enables the parity argument we mentioned before; it is thus also needed in the regular case.

Lemma 4.14. *Suppose that every* Opt*-bin contains an H-item. Then no* Opt*-bin contains two* 2*-items* c_1 *and* c_2. □

After excluding some easy subcases of the freaky case, we in particular know that every Opt-bin contains an H-item. The general idea of the proof in the freaky case is that we try to find an item c different from f_1 such that the bonus of $\{g_0, c\}$ is sufficient and can be used to pay for the freaky bin F. If we find such c, we save the bonus 0.1 of f_1 and use it to tighten Proposition 4.11 by the necessary 0.1. We have three subcases.

Case 1: F opens after C_2. Thus F contains no small item by Lemma 4.2(iii); since f_1 is huge and $s(F) < 5/6$, it follows that F is a 2-bin containing only f_1 and f_2.

The intuition is that we use the bonus of f_2 instead of f_1 to pay for F. However, in general, the bonus of $\{g_0, f_2\}$ is not sufficient to pay for F, if F is smaller than C_g. In that case, the bonus of $\{g_0, f_2\}$ is sufficient to pay for G_g and we use the bonus of the next common bin, G_{g+1} to pay for F. A further complication is that the bonus of $\{g_0, f_2\}$ smaller than necessary by a term proportional to Δ; this is compensated by the dedicated and huge-first bins.

Case 2: F opens before C_2 and some C_k, $k \geq 2$, has rank at least three. Let c be one of the medium items in this C_k and set $E = \{g_0, c\}$. Then E is a valid exceptional set. Furthermore, c does not fit into F.

If $s(F) \geq 2/3$, we have $\overline{\overline{w}}(F) + \overline{w}(E) \geq 1$ by Lemma 2.2. Using Proposition 4.11(i) we have $w(\mathcal{C}) \geq (w(\mathcal{R}) - \overline{w}(E)) + (\overline{\overline{w}}(F) + \overline{w}(E)) + \overline{w}(f_1) \geq \rho - 0.2 + 1 + 0.1 = \gamma - 0.1$.

If $s(F) < 2/3$, we have $\overline{w}(E) = 0.2$ and by Lemma 4.4(iv), C_ρ is a 2-bin such that $s(C_\rho) + s(F) > 4/3$. Thus $\overline{w}(E) + w(F) + \overline{\overline{w}}(C_\rho) > 0.2 + 0.1 + 1.6 = 1.9$. Adding all the inequalities $\overline{\overline{w}}(C_{i-1}) + \overline{w}(C_i \setminus E) \geq 1$, $i = 2, \ldots, \rho$ from Lemma 4.10(i), we get $w(\mathcal{C}) > \gamma - 0.1$.

Case 3: F opens before C_2 and each C_i, $i \geq 2$, has rank 2. Then all bins C_i, $i \geq 2$, are 2-bins and by Lemma 4.14, all items in these $\rho - 1$ bins are packed into different optimal bins. Thus there are at most $\text{OPT}/2$ such bins, and since OPT is odd (from $\text{OPT} \equiv 7 \pmod{10}$) we actually get $\rho \leq (\text{OPT} + 1)/2$. and thus $\gamma = \rho + 1 \leq \text{OPT}/2 + 3/2$. Instead of using the weights, here we get a contradiction by bounding the size of all the bins. □

Acknowledgements. We are grateful to anonymous referees for useful comments. G. Dósa supported by the Hungarian State and the European Union under the TAMOP-4.2.2.A-11/1/ KONV-2012-0072 and Chinese-Hungarian bilateral project TET-12-CN-1-2012-0028. J. Sgall partially supported by the Center of Excellence – Institute for Theoretical Computer Science, Prague (project P202/12/G061 of GA ČR).

References

1. Boyar, J., Dósa, G., Epstein, L.: On the absolute approximation ratio for First Fit and related results. Discrete Appl. Math. 160, 1914–1923 (2012)
2. Coffman, E.G., Garey, M.R., Johnson, D.S.: Approximation algorithms for bin packing: A survey. In: Hochbaum, D. (ed.) Approximation algorithms. PWS Publishing Company (1997)
3. Dósa, G., Sgall, J.: First Fit bin packing: A tight analysis. In: Proc. of the 30th Ann. Symp. on Theor. Aspects of Comput. Sci (STACS). LIPIcs, vol. 3, pp. 538–549. Schloss Dagstuhl (2013)
4. Garey, M.R., Graham, R.L., Johnson, D.S., Yao, A.C.-C.: Resource constrained scheduling as generalized bin packing. J. Combin. Theory Ser. A 21, 257–298 (1976)
5. Garey, M.R., Graham, R.L., Ullman, J.D.: Worst-case analysis of memory allocation algorithms. In: Proc. 4th Symp. Theory of Computing (STOC), pp. 143–150. ACM (1973)
6. Graham, R.L.: Bounds for certain multiprocessing anomalies. Bell System Technical J. 45, 1563–1581 (1966)
7. Graham, R.L.: Bounds on multiprocessing timing anomalies. SIAM J. Appl. Math. 17, 263–269 (1969)
8. Johnson, D.S.: Near-optimal bin packing algorithms. PhD thesis. MIT, Cambridge, MA (1973)
9. Johnson, D.S.: Fast algorithms for bin packing. J. Comput. Syst. Sci. 8, 272–314 (1974)
10. Johnson, D.S., Demers, A., Ullman, J.D., Garey, M.R., Graham, R.L.: Worst-case performance bounds for simple one-dimensional packing algorithms. SIAM J. Comput. 3, 256–278 (1974)

11. Németh, Z.: A first fit algoritmus abszolút hibájáról (in Hungarian). Eötvös Loránd Univ., Budapest, Hungary (2011)
12. Sgall, J.: A new analysis of best fit bin packing. In: Kranakis, E., Krizanc, D., Luccio, F. (eds.) FUN 2012. LNCS, vol. 7288, pp. 315–321. Springer, Heidelberg (2012)
13. Simchi-Levi, D.: New worst case results for the bin-packing problem. Naval Research Logistics 41, 579–585 (1994)
14. Ullman, J.D.: The performance of a memory allocation algorithm. Technical Report 100, Princeton Univ., Princeton, NJ (1971)
15. Williamson, D.P., Shmoys, D.B.: The Design of Approximation Algorithms. Cambridge University Press (2011)
16. Xia, B., Tan, Z.: Tighter bounds of the First Fit algorithm for the bin-packing problem. Discrete Appl. Math. 158, 1668–1675 (2010)

Light Spanners

Michael Elkin[1], Ofer Neiman[1,⋆], and Shay Solomon[2,⋆⋆]

[1] Department of Computer Science, Ben-Gurion University of the Negev,
Beer-Sheva, Israel
`elkinm,neimano@cs.bgu.ac.il`

[2] Department of Computer Science and Applied Mathematics,
The Weizmann Institute of Science, Rehovot 76100, Israel
`shay.solomon@weizmann.ac.il`

Abstract. A *t-spanner* of a weighted undirected graph $G = (V, E)$, is a subgraph H such that $d_H(u, v) \leq t \cdot d_G(u, v)$ for all $u, v \in V$. The sparseness of the spanner can be measured by its size (the number of edges) and weight (the sum of all edge weights), both being important measures of the spanner's quality – in this work we focus on the latter.

Specifically, it is shown that for any parameters $k \geq 1$ and $\varepsilon > 0$, any weighted graph G on n vertices admits a $(2k-1) \cdot (1+\varepsilon)$-stretch spanner of weight at most $w(MST(G)) \cdot O_\varepsilon(kn^{1/k}/\log k)$, where $w(MST(G))$ is the weight of a minimum spanning tree of G. Our result is obtained via a novel analysis of the classic greedy algorithm, and improves previous work by a factor of $O(\log k)$.

1 Introduction

Given a weighted connected graph $G = (V, E)$ with n vertices and m edges, let d_G be its shortest path metric. A t-spanner $H = (V, E')$ is a subgraph that preserves all distances up to a multiplicative factor t. That is, for all $u, v \in V$, $d_H(u, v) \leq t \cdot d_G(u, v)$. The parameter t is called the *stretch*. There are several parameters that have been studied in the literature that govern the quality of H, two of the most notable ones are the *size* of the spanner (the number of edges) and its total *weight* (the sum of weights of its edges).

There is a basic tradeoff between the stretch and the size of a spanner. For any graph on n vertices, there exists a $(2k-1)$-spanner with $O(n^{1+1/k})$ edges [ADD+93]. Furthermore, there is a simple greedy algorithm for constructing such a spanner, which we shall refer to as the *greedy spanner* (see Algorithm 1). The bound on the number of edges is known to be asymptotically tight for certain small values of k, and for all k assuming Erdős' girth conjecture.

In this paper we focus on the weight of a spanner. Light weight spanners are particularly useful for efficient broadcast protocols in the message-passing model of distributed computing [ABP90,ABP91], where efficiency is measured

⋆ Supported in part by ISF grant No. (523/12) and by the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n°303809.

⋆⋆ This work is supported by the Koshland Center for basic Research.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 442–452, 2014.

with respect to both the total communication cost (corresponding to the spanner's weight) and the speed of message delivery at all destinations (corresponding to the spanner's stretch). Additional applications of light weight spanners in distributed systems include network synchronization and computing global functions [ABP90,ABP91,Pel00]. Light weight spanners were also found useful for various data gathering and dissemination tasks in overlay networks [BKR$^+$02,KV02], in wireless and sensor networks [SS10], for network design [MP98,SCRS01], and routing [WCT02].

While a minimum spanning tree (MST) has the lowest weight among all possible connected spanners, its stretch can be quite large. Nevertheless, when measuring the weight of a spanner, we shall compare ourselves to the weight of an MST: The *lightness* of the spanner H is defined as $\frac{w(H)}{w(MST)}$ (here $w(H)$ is the total edge weight of H). It was shown by [ADD$^+$93] that the lightness of the greedy spanner is at most $O(n/k)$, and their result was improved by [CDNS92], who showed that for any $\varepsilon > 0$ the greedy $(2k-1)\cdot(1+\varepsilon)$-spanner has $O_\varepsilon(n^{1+1/k})$ edges and lightness $O(k \cdot n^{1/k}/\varepsilon^{1+1/k})$. A particularly interesting special case arises when $k \approx \log n$. Specifically, in this case the result of [CDNS92] provides stretch and lightness both bounded by $O(\log n)$. Another notable point on the tradeoff curve of [CDNS92] (obtained by setting $\varepsilon = \log n$ as well) is stretch $O(\log^2 n)$ and lightness $O(1)$.

These results of [CDNS92] remained the state-of-the-art for more than twenty years. In particular, prior to this work it was unknown if spanners with stretch $O(\log n)$ and lightness $o(\log n)$, or vice versa, exist. In this paper we answer this question in the affirmative, and show in fact something stronger – spanners with stretch and lightness both bounded by $o(\log n)$ exist. We provide a novel analysis of the classic greedy algorithm, which improves the tradeoff of [CDNS92] by a factor of $O(\log k)$. Specifically, we prove the following theorem.

Theorem 1. *For any weighted graph $G = (V, E)$ and parameters $k \geq 1$, $\varepsilon > 0$, there exists a $(2k-1)\cdot(1+\varepsilon)$-spanner H with $O(n^{1+1/k})$ edges*[1] *and lightness $O(n^{1/k} \cdot (1 + k/(\varepsilon^{1+1/k} \log k)))$.*

By substituting $k \approx \log n$ we obtain stretch $\log n$ and lightness $O(\log n/ \log\log n)$ (for fixed small ε). We also allow ε to be some large value. In particular, setting $\varepsilon = \log n/ \log\log n$ yields stretch $\log^2 n/ \log\log n$ and lightness $O(1)$. Also, by substituting $k = \log n/ \log\log\log n$ we can have both stretch and lightness bounded by $O(\log n/ \log\log\log n)$.

Our result shows that the potentially natural tradeoff between stretch $2k-1$ and lightness $O(k{\cdot}n^{1/k})$ is not the right one. This can also be seen as an indication that the right tradeoff is stretch $(2k-1)$ and lightness $O(n^{1/k})$. (Note that lightness $O(n^{1/k})$ is the weighted analogue of $O(n^{1+1/k})$ edges, and so it is asymptotically tight assuming Erdős' girth conjecture.)

[1] In fact for large ε a better bound can be obtained. Specifically, it is $O(n^{1+1/\lfloor\lceil(2k-1)\cdot(1+\varepsilon)\rceil/2\rfloor})$.

1.1 Proof Overview

The main idea in the analysis of the greedy algorithm by [CDNS92], is to partition the edges of the greedy spanner to scales according to their weight, and bound the contribution of edges in each scale separately. For each scale they create a graph from the edges selected by the greedy algorithm to the spanner, and argue that such a graph has high girth[2] and thus few edges. The main drawback is that when analyzing larger weight edges, this argument ignores the smaller weight edges that were already inserted into the spanner.

We show that one indeed can use information on lower weight edges when analyzing the contribution of higher scales. We create a different graph from edges added to the spanner, and argue that this graph has high girth. The new ingredient in our analysis is that we add multiple edges per spanner edge, proportionally to its weight. Specifically, these new edges form a *matching* between certain neighbors of the original edge's endpoints. Intuitively, a high weight edge enforces strong restrictions on the length of cycles containing it, so it leaves a lot of "room" for low weight edges in its neighborhood. The structure of the matching enables us to exploits this room, while maintaining high girth.

Unfortunately, with our current techniques we can only use edges of weight at most k times smaller than the weight of edges in the scale which is now under inspection. Hence this gives an improvement of $O(\log k)$ to the lightness of the greedy spanner. We hope that a refinement of our method, perhaps choosing the matching more carefully, will eventually lead to an optimal lightness of $O(n^{1/k})$.

1.2 Related Work

A significant amount of research attention was devoted to constructing light and sparse spanners for Euclidean and doubling metrics. A major result is that for any constant-dimensional Euclidean metric and any $\varepsilon > 0$, there exists a $(1+\varepsilon)$-spanner with lightness $O(1)$ [DHN93]. Since then there has been a flurry of work on improving the running time and other parameters. See, e.g., [CDNS92,ADM$^+$95,DES08,ES13,CLNS13], and the references therein. An important question still left open is whether the $O(1)$ lightness bound of [DHN93] for constant-dimensional Euclidean metrics can be extended to doubling metrics. Such a light spanner has implications for the running time of a PTAS for the traveling salesperson problem (TSP). Recently, [GS14] showed such a spanner exists for snowflakes[3] of doubling metrics.

Light spanners with $(1+\varepsilon)$ stretch have been sought for other graph families as well, with the application to TSP in mind. It has been conjectured that graphs excluding a fixed minor have such spanners. Currently, some of the known results are for planar graph [ADM$^+$95], bounded-genus graphs [Gri00], unit disk graphs [KPX08], and bounded pathwidth graphs [GH12].

[2] The girth of a graph is the minimal number of edges in a cycle.

[3] For $0 \leq \alpha \leq 1$, an α-snowflake of a metric is obtained by taking all distances to power α.

A lot of research focused on constructing sparse spanners efficiently, disregarding their lightness. Cohen [Coh93] devised a randomized algorithm for constructing $((2k-1)\cdot(1+\varepsilon))$-spanners with $O(k\cdot n^{1+1/k}\cdot(1/\varepsilon)\cdot\log n)$ edges. Her algorithm requires expected $O(m\cdot n^{1/k}\cdot k\cdot(1/\varepsilon)\cdot\log n))$ time. Baswana and Sen [BS03] improved Cohen's result, and devised an algorithm that constructs $(2k-1)$-spanners with expected $O(k\cdot n^{1+1/k})$ edges, in expected $O(k\cdot m)$ time. Roditty et al. [RTZ05] derandomized this algorithm, while maintaining the same parameters (including running time). Roditty and Zwick [RZ04] devised a deterministic algorithm for constructing $(2k-1)$-spanners with $O(n^{1+1/k})$ edges in $O(k\cdot n^{2+1/k})$ time.

2 Preliminaries

Let $G=(V,E)$ be a graph on n vertices with weights $w : E\to\mathbb{R}_+$, and let d_G be the shortest path metric induced by G. For simplicity of the presentation we shall assume that the edge weights are positive integers. (The extension of our proof to arbitrary weights is not difficult, requiring only a few minor adjustments.) For a subgraph $H=(V',E')$ define $w(H)=w(E')=\sum_{e\in E'}w(e)$. A subgraph $H=(V,E')$ is called a *t-spanner* if for all $u,v\in V$, $d_H(u,v)\le t\cdot d_G(u,v)$. Define the *lightness* of H as $\frac{w(H)}{w(MST(G))}$, where $MST(G)$ is a minimum spanning tree of G. The girth g of a graph is the minimal number of edges in a cycle of G. The following standard Lemma is implicit in [Bol78].

Lemma 1. *Let $g>1$ be an integer. A graph on n vertices and girth g has at most $O\left(n^{1+\frac{1}{\lfloor(g-1)/2\rfloor}}\right)$ edges.*

2.1 Greedy Algorithm

The natural greedy algorithm for constructing a spanner is described in Algorithm 1.

Algorithm 1. `Greedy`$(G=(V,E),t)$

```
$H=(V,\emptyset)$.
for each edge $\{u,v\}\in E$, in non-decreasing order of weight, do
    if $d_H(u,v)>t\cdot w(u,v)$ then
        Add the edge $\{u,v\}$ to $E(H)$.
    end if
end for
```

Note that whenever an edge $e\in E$ is inserted into $E(H)$, it cannot close a cycle with $t+1$ or less edges, because the edges other than e of such a cycle will form a path of length at most $t\cdot w(e)$ (all the existing edges are not longer than $w(e)$). This argument suggests that H (viewed as an unweighted graph) has girth $t+2$ (when t is an integer), and thus by Lemma 1

$$|E(H)| \leq O\left(n^{1+\frac{1}{\lfloor (t+1)/2 \rfloor}}\right) . \qquad (1)$$

We observe that the greedy algorithm must select all edges of an MST (because when inspected they connect different connected components in H). We will assume without loss of generality that the graph G has a unique MST, since any ties can be broken using lexicographic rules.

Observation 2. *If Z is the MST of G, then $Z \subseteq H$. Furthermore, each edge in the MST does not close a cycle in H when it is inspected.*

3 Proof of Main Result

Let H be the greedy spanner with parameter $t = (2k-1) \cdot (1+\varepsilon)$. Let Z be the MST of G, and order the vertices $v_1, v_2, \ldots, v_n$ according to the order they are visited in some preorder traversal of Z (with some fixed arbitrary root). Since every edge of Z is visited at most twice in such a tour,

$$L := \sum_{i=2}^{n} d_Z(v_{i-1}, v_i) \leq 2w(Z) .$$

Let $I = \lceil \log_k n \rceil$. For each $i \in [I]$, define $E_i = \{e \in E(H) \setminus E(Z) \mid w(e) \in (k^{i-1}, k^i] \cdot L/n\}$. We may assume the maximum weight of an edge in H is bounded by $w(Z)$ (in fact $w(Z)/t$, as heavier edges surely will not be selected for the spanner), so each edge in $H \setminus Z$ of weight greater than L/n is included in some E_i. The main technical theorem is the following.

Theorem 2. *For each $i \in [I]$ and any $\varepsilon > 0$,*

$$w(E_i) \leq O(L \cdot (n/k^{i-1})^{1/k}/\varepsilon^{1+1/k}) .$$

Given this, the proof of Theorem 1 quickly follows.

Proof (Proof of Theorem 1). Using that the stretch of the spanner is $t \geq 2k-1$, by (1) we have $|E(H)| \leq O(n^{1+1/k})$. The total weight of edges in H that have weight at most L/n can be bounded by $L/n \cdot |E(H)| \leq L/n \cdot O(n^{1+1/k}) = O(w(MST) \cdot n^{1/k})$. The contribution of the other (non-MST) edges to the weight of H, using Theorem 2, is at most

$$\begin{aligned}\sum_{i=1}^{I} O(L \cdot (n/k^{i-1})^{1/k}/\varepsilon^{1+1/k}) &\leq O(L \cdot n^{1/k}/\varepsilon^{1+1/k}) \sum_{i=0}^{\infty} e^{-(i \ln k)/k} \\ &= O(L \cdot n^{1/k}/\varepsilon^{1+1/k}) \cdot \frac{1}{1-e^{-(\ln k)/k}} \\ &= O(w(MST)) \cdot kn^{1/k}/(\varepsilon^{1+1/k} \ln k) .\end{aligned}$$

3.1 Proof of Theorem 2

Overview: Fix some $i \in [I]$. We shall construct a certain graph K from the edges of E_i, and argue that this graph has high girth, and therefore few edges. The main difference from [CDNS92] is that our construction combines into one scale edges whose weight may differ by a factor of k (in the construction of [CDNS92] all edges in a given scale are of the same weight, up to a factor of 2). In order to compensate for heavy edges, the weight of the edge determines how many edges are added to K. Specifically, if the edge $\{u, v\} \in E_i$ has weight $w \cdot k^{i-1} \cdot L/n$, we shall add (at least) $\lceil w \rceil$ edges to K that form a *matching* between vertices in some neighborhoods of u and v. In this way the weight of K dominates $w(E_i)$. To prove that K has high girth, we shall map a cycle in K to a closed tour in H of proportional length. The argument uses the fact that the new edges are close to the original edge, and that a potential cycle in K cannot exploit more than one such new edge, since these edges form a matching.

Construction of the Graph K: Let $P = (p_0, \ldots, p_L)$ be the unweighted path on $L + 1$ vertices, created from V by placing $v_1, \ldots, v_n$ in this order and adding Steiner vertices so that all consecutive distances are 1, and for all $2 \leq j \leq n$, $d_P(v_{j-1}, v_j) = d_Z(v_{j-1}, v_j)$. In particular, $p_0 = v_1$, $p_L = v_n$, and for every $1 \leq j < j' \leq n$,

$$d_P(v_j, v_{j'}) = \sum_{h=j+1}^{j'} d_Z(v_{h-1}, v_h) \ .$$

Note that $d_P(v_j, v_{j'}) \geq d_Z(v_j, v_{j'}) \geq d_G(v_j, v_{j'})$, and all the inequalities may be strict. In order to be able to map edges of K back to H, we shall also add corresponding Steiner points to the spanner H: For every Steiner point p_h that lies on P between v_{j-1} and v_j, add a Steiner point on the path in the MST Z that connects v_{j-1} to v_j at distance $d_P(v_{j-1}, p_h)$ from v_{j-1} (unless there is a point there already). By Observation 2 all MST edges are indeed in H, and one can simply subdivide the appropriate edge on the MST path. Note that distances in H do not change, as the new Steiner points have degree 2. Denote by $\hat{H}$ the modified spanner H, i.e., H with the Steiner points.

Let $a = k^{i-1} \cdot L/n$ be a lower bound on the weight of edges in E_i. Divide P into $s = 8L/(\varepsilon a)$ intervals $I_1, \ldots, I_s$, each of length $L/s = \frac{\varepsilon}{8} a$ (by appropriate scaling, we assume all these are integers). For $j \in [s]$, the interval I_j contains the points $p_{(j-1)L/s}, \ldots, p_{jL/s}$. In each interval I_j pick an arbitrary (interior) point r_j as a representative, and let R be the set of representatives. For each representative r_j and an integer $b \geq 0$ we define its neighborhood $N_b(j) = \{r_h \ : \ |j - h| \leq b\}$ to be the set of (at most) $2b + 1$ representatives that are at most b intervals away from I_j. (Note that the size of the neighborhood $N_b(j)$ can be smaller than $2b+1$ if r_j is too close to one of the endpoints of the path P.) Define an unweighted (multi) graph $K = (R, F)$ in the following manner. Let $e = \{u, v\} \in E_i$. Assume that $u \in I_h$ and $v \in I_j$ for some $h, j \in [s]$. Let $b = \lfloor w(e)/a \rfloor$, and let M be an arbitrary maximal matching between $N_b(h)$ and $N_b(j)$. Add all the edges of M to F, see Figure 1. For each of the edges $\{q, q'\} \in M$ added to F, we say

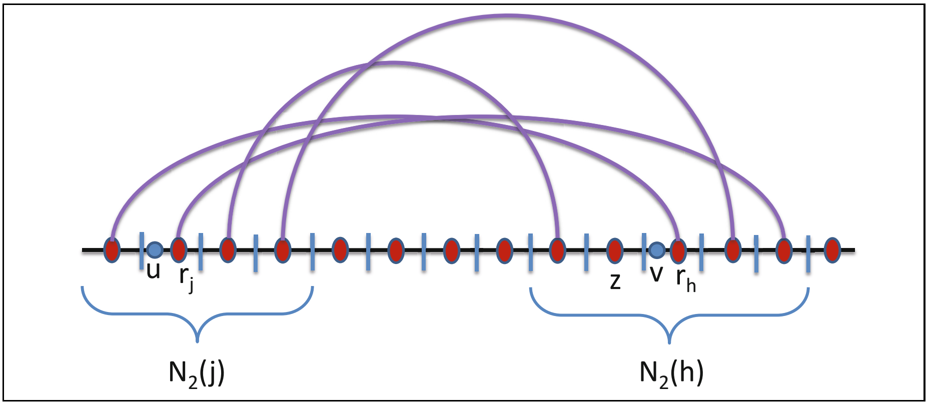


Fig. 1. *Construction of the graph K*: The oval vertices are R, the representatives. The edge $\{u, v\}$ is an edge of weight $2a$ selected for the spanner, and $u \in I_j$, $v \in I_h$ with representatives r_j, r_h. The depicted edges, that form a maximal matching between the neighborhoods of r_j and r_h, are added to K. (The vertex z in $N_2(h)$ does not participate in the matching, because r_j is too close to the left endpoint of the path P.)

that the edge $\{u, v\}$ is its *source* when $q \in N_b(h)$ and $q' \in N_b(j)$, and write $S(q, q') = (u, v)$. We will soon show (in Proposition 2 below) that each edge in K has a single source.

The following observation suggests that if all the edges of K were given weight a, then its total weight is greater than or equal to the weight of the edges in E_i.

Observation 3. *$|F| \cdot a \geq w(E_i)$.*

Proof. Note that always $|N_b(j)| \geq b + 1$, which means that we add at least $b + 1 \geq w(e)/a$ edges to K for each edge $e \in E_i$. Summing over all edges concludes the proof.

Mapping from K to $\hat{H}$: We shall map every edge $\{q, q'\} \in F$ to a tour $T(q, q')$ in the spanner $\hat{H}$ connecting q and q'. If $S(q, q') = (u, v)$, then $T(q, q')$ consists of the following paths:

- A path in Z connecting q to u.
- The edge $\{u, v\}$.
- A path in Z connecting v to q'.

The following proposition asserts that the length of the tour is not longer than the weight of the source edge, up to a $1 + \varepsilon/2$ factor.

Proposition 1. *If an edge $\{q, q'\} \in F$ has a source $S(q, q') = (u, v)$ of weight w, then $T(q, q')$ is a tour in $\hat{H}$ of length at most $(1 + \varepsilon/2)w$.*

Proof. First observe that the distance in P between any two points in intervals I_j and I_{j+b} is at most $(b+1)L/s$. Since $d_P \geq d_Z$ we also have that the distance in the MST Z between two such points is bounded by $(b+1)L/s$. (By definition, this holds for Steiner points as well.) Denote the representatives of u, v as r_j, r_h, respectively. For $b = \lfloor w/a \rfloor$, the set $N_b(j)$ contains representatives of at most b intervals away from I_j. As $u \in I_j$ we get that $d_Z(q, u) \leq (b+1)L/s$. Similarly $d_Z(q', v) \leq (b+1)L/s$, thus the total length of the tour is at most $w + 2(b+1)L/s = w + 2(\lfloor \frac{w}{a} \rfloor + 1)\frac{\varepsilon}{8}a \leq (1+\varepsilon/2)w$.

Our goal is to show that K is a simple graph of girth at least $2k+1$. As a warmup, let us first show that K does not have parallel edges.

Proposition 2. *The graph K does not have parallel edges.*

Proof. Seeking contradiction, assume there is an edge $\{q, q'\} \in F$ with two different sources $\{u, v\}, \{u', v'\} \in E_i$. Without loss of generality assume that $\{u, v\}$ is the heavier edge of the two, with weight w. Then $\{q, q'\}$ is mapped to two tours in $\hat{H}$ connecting q, q', whose total length, using Proposition 1, is at most $w(2+\varepsilon)$. Consider the tour $\hat{T} = u \to q \to u' \to v' \to q' \to v$ in $\hat{H}$ which has total length at most $w(2+\varepsilon) - w = w(1+\varepsilon)$. Since the Steiner points have degree 2, they can be removed from $\hat{T}$ without increasing its length, and thus there is in H a simple path T from u to v of length at most $w(1+\varepsilon)$.

We claim that T must exist at the time the edge $\{u, v\}$ is inspected by the greedy algorithm. The edge $\{u', v'\}$ exists because it is lighter. The MST edges exist since by Observation 2 they must connect different components when inspected, while if some of them are inserted after $\{u, v\}$, at least one of them will close the cycle $T \cup \{u, v\}$. As $w(1+\varepsilon) \leq w \cdot (2k-1)(1+\varepsilon)$, we conclude that the edge $\{u, v\}$ should not have been added to H, which is a contradiction.

Showing that K has large girth will follow similar lines, but is slightly more involved. The difficulty arises since we added multiple edges for each edge of H, thus a cycle in K may be mapped to a closed tour in H that uses the same edge $e \in E(H)$ more than once. In such a case, e may not be a part of any simple cycle contained in the closed tour, and we will not be able to derive a contradiction from the greedy choice of e to H. [4] To rule out such a possibility, we use the fact that the multiple edges whose source is e form a matching, and that the weights are different by a factor of at most k.

Lemma 4. *The graph $K = (R, F)$ has girth $2k+1$.*

Proof. It will be easier to prove a stronger statement, that for any $j \in [s]$ and any $r, r' \in N_k(j)$, every path in K between r and r' contains at least $2k+1$ edges. Once this is proven, we may use this with $r = r'$ to conclude that $girth(K) \geq 2k+1$.

Seeking contradiction, assume that there is a path Q in K from r to r' that contains at most $2k$ edges, and take the shortest such Q (over all possible choices of j and r, r'). Let $\{q, z\} \in F$ be the last edge added to Q, with source $S(q, z) =$

[4] In fact, this is the only reason our method improves the lightness by a factor of $\log k$ rather than the desired k.

(x, y) (so that $\{x, y\} \in E_i$ is the heaviest among all the sources of edges in Q). We claim that no other edge in Q has $\{x, y\}$ as a source. To see this, consider a case in which such an edge $\{q', z'\} \in F$ is also in Q with $S(q', z') = (x, y)$. We may assume w.l.o.g that $q \notin \{r, r'\}$ (since the path Q contains at least 2 edges), then by definition of the graph K, there exists some $j' \in [s]$ with $q, q' \in N_k(j')$ (recall that the neighborhood length b always satisfies $b \leq k$ by definition of E_i). But then the sub-path of Q from q to q' is strictly shorter than Q, and connects two points in the same k-neighborhood. Since the edges in K with $\{x, y\}$ as a source form a matching, we get that $q \neq q'$, and thus this path is not of length 0. This contradicts the minimality of Q. Next, we will show that $\{x, y\}$ should not have been chosen for H, because there is a short path connecting x to y.

By Proposition 1 every edge $e \in Q$ whose source $S(e) = e'$ has weight $w(e')$, is mapped to a tour $T(e)$ of length at most $(1+\varepsilon/2)w(e')$ in $\hat{H}$. Since $w(x, y)$ is the maximum weight source of all edges in Q, we conclude that the total length of tours connecting x to r and r' to y is at most $(2k-1)\cdot(1+\varepsilon/2)w(x, y)$. Note that r, r' are representatives in $N_k(j)$, which are at most $2k$ intervals apart. So their distance in the MST Z is at most $2k \cdot \varepsilon a/8 \leq k \cdot \varepsilon w(x, y)/4$. The total length of the tour $x \to r \to r' \to y$ in $\hat{H}$ is at most

$$(2k-1)\cdot(1+\varepsilon/2)w(x, y) + k \cdot \varepsilon w(x, y)/4 \leq (2k-1)(1+\varepsilon)\cdot w(x, y)\ .$$

When the algorithm considers the edge $\{x, y\}$, all the edges of the above tour exist in $\hat{H}$. (This follows since they are all MST edges or lighter than $w(x, y)$, similarly to the argument used in Proposition 2.) We conclude that there is a path between x and y in H of length at most $(2k-1)\cdot(1+\varepsilon)\cdot w(x, y)$, hence $\{x, y\}$ should not have been added to $E(H)$, which yields a contradiction.

Proof (Proof of Theorem 2). Recall that the graph K has s vertices. By Proposition 2 it is a simple graph, and Lemma 4 suggests it has girth at least $2k+1$, thus by using Lemma 1 it has at most $O(s^{1+1/k})$ edges. Using Observation 3,

$$\begin{aligned} w(E_i) &\leq |F| \cdot a \\ &\leq O(s^{1+1/k}) \cdot (L/n \cdot k^{i-1}) \\ &= \left(\frac{8 \cdot n}{\varepsilon k^{i-1}}\right)^{1+1/k} \cdot (O(L)/n \cdot k^{i-1}) \\ &\leq O(L \cdot (n/k^{i-1})^{1/k}/\varepsilon^{1+1/k})\ . \end{aligned}$$

4 Weighted Girth Conjecture

The girth of a graph is defined on unweighted graphs. Here we give an extension of the definition that generalizes to weighted graphs as well, and propose a conjecture on the extremal graph attaining a weighted girth.

Definition 1. *Let $G = (V, E)$ be a weighted graph with weights $w : E \to \mathbb{R}_+$, the* weighted girth *of G is the minimum over all cycles C of the weight of C*

divided by its heaviest edge, that is

$$\min_{C \text{ cycle in } G} \left\{ \frac{w(C)}{\max_{e \in C} w(e)} \right\} .$$

Note that this matches the standard definition of girth for unweighted graphs. Recall that the lightness of G is $\frac{w(G)}{w(MST)}$. For a given weighted girth value g and cardinality n, we ask what is the graph on n vertices with weighted girth g that maximizes the lightness?

Conjecture 1. For any integer $g \geq 3$, among all graphs with n vertices and weighted girth g, the maximal lightness is attained for an unweighted graph.

Recall that Erdős' girth conjecture asserts that there exists an (unweighted) graph with girth $g > 2k$ and $\Omega(n^{1+1/k})$ edges, that is, its lightness is $\Omega(n^{1/k})$. Observe that any graph of weighted girth larger than $2k + \varepsilon(2k-1)$ can be thought of as the output of Algorithm 1 with parameter $t = (2k-1) \cdot (1+\varepsilon)$. In particular, Theorem 1 implies that its lightness is at most $O_\varepsilon(kn^{1/k}/\log k)$. Thus (up to the term of $\varepsilon(2k-1)$ in the girth), there exists an unweighted graph which is at most $O(k/\log k) = O(g/\log g)$ lighter than the heaviest weighted graph.

The intuition behind this conjecture follows from our method of replacing high weight edges by many low weight edges. We believe that such replacement should hold when performed on all possible scales simultaneously. An immediate corollary of Conjecture 1, is that the lightness of a greedy $(2k-1)$-spanner of a weighted graph on n vertices is bounded by $O(n^{1/k})$. To see why this is true, note that the spanner's weighted girth must be strictly larger than $2k$, and $O(n^{1/k})$ is a bound on the lightness of an *unweighted* graph on n vertices with girth $2k+1$.

References

ABP90. Awerbuch, B., Baratz, A., Peleg, D.: Cost-sensitive analysis of communication protocols. In: Proc. of 9th PODC, pp. 177–187 (1990)

ABP91. Awerbuch, B., Baratz, A., Peleg, D.: Efficient broadcast and light-weight spanners (1991) (manuscript)

ADD$^+$93. Althöfer, I., Das, G., Dobkin, D.P., Joseph, D., Soares, J.: On sparse spanners of weighted graphs. Discrete & Computational Geometry 9, 81–100 (1993)

ADM$^+$95. Arya, S., Das, G., Mount, D.M., Salowe, J.S., Smid, M.H.M.: Euclidean spanners: short, thin, and lanky. In: Proc. of 27th STOC, pp. 489–498 (1995)

BKR$^+$02. Braynard, R., Kostic, D., Rodriguez, A., Chase, J., Vahdat, A.: Opus: an overlay peer utility service. In: Prof. of 5th OPENARCH (2002)

Bol78. Bollobás, B.: Extremal Graph Theory. Academic press, London (1978)

BS03. Baswana, S., Sen, S.: A simple linear time algorithm for computing a $(2k-1)$-spanner of $O(n^{1+1/k})$ size in weighted graphs. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 384–396. Springer, Heidelberg (2003)

CDNS92. Chandra, B., Das, G., Narasimhan, G., Soares, J.: New sparseness results on graph spanners. In: Proc. of 8th SOCG, pp. 192–201 (1992)

CLNS13. Chan, T.-H.H., Li, M., Ning, L., Solomon, S.: New doubling spanners: Better and simpler. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) ICALP 2013, Part I. LNCS, vol. 7965, pp. 315–327. Springer, Heidelberg (2013)

Coh93. Cohen, E.: Fast algorithms for constructing t-spanners and paths with stretch t. In: Proc. of 34th FOCS, pp. 648–658 (1993)

DES08. Dinitz, Y., Elkin, M., Solomon, S.: Shallow-low-light trees, and tight lower bounds for Euclidean spanners. In: Proc. of 49th FOCS, pp. 519–528 (2008)

DHN93. Das, G., Heffernan, P.J., Narasimhan, G.: Optimally sparse spanners in 3-dimensional Euclidean space. In: Proc. of 9th SOCG, pp. 53–62 (1993)

ES13. M. Elkin and S. Solomon. Optimal Euclidean spanners: really short, thin and lanky. In *STOC*, pages 645–654, 2013.

GH12. Grigni, M., Hung, H.-H.: Light spanners in bounded pathwidth graphs. In: Rovan, B., Sassone, V., Widmayer, P. (eds.) MFCS 2012. LNCS, vol. 7464, pp. 467–477. Springer, Heidelberg (2012)

Gri00. Grigni, M.: Approximate TSP in graphs with forbidden minors. In: Welzl, E., Montanari, U., Rolim, J.D.P. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 869–877. Springer, Heidelberg (2000)

GS14. Gottlieb, L., Solomon, S.: Light spanners for snowflake metrics. In: SoCG (to appear, 2014)

KPX08. Kanj, I.A., Perković, L., Xia, G.: Computing lightweight spanners locally. In: Taubenfeld, G. (ed.) DISC 2008. LNCS, vol. 5218, pp. 365–378. Springer, Heidelberg (2008)

KV02. Kostic, D., Vahdat, A.: Latency versus cost optimizations in hierarchical overlay networks. Technical report, Duke University, CS-2001-04 (2002)

MP98. Mansour, Y., Peleg, D.: An approximation algorithm for minimum-cost network design. Technical report, Weizmann Institute of Science, Rehovot (1998)

Pel00. Peleg, D.: Distributed Computing: A Locality-Sensitive Approach. SIAM, Philadelphia (2000)

RTZ05. Roditty, L., Thorup, M., Zwick, U.: Deterministic constructions of approximate distance oracles and spanners. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 261–272. Springer, Heidelberg (2005)

RZ04. Roditty, L., Zwick, U.: On dynamic shortest paths problems. In: Albers, S., Radzik, T. (eds.) ESA 2004. LNCS, vol. 3221, pp. 580–591. Springer, Heidelberg (2004)

SCRS01. Salman, F.S., Cheriyan, J., Ravi, R., Subramanian, S.: Approximating the single-sink link-installation problem in network design. SIAM Journal on Optimization 11(3), 595–610 (2001)

SS10. Shpungin, H., Segal, M.: Near optimal multicriteria spanner constructions in wireless ad-hoc networks. IEEE/ACM Transactions on Networking 18(6), 1963–1976 (2010)

WCT02. Wu, B.Y., Chao, K., Tang, C.Y.: Light graphs with small routing cost. Networks 39(3), 130–138 (2002)

Semi-Streaming Set Cover[⋆]
(Extended Abstract)

Yuval Emek[1] and Adi Rosén[2,⋆⋆]

[1] Technion, Israel
yemek@ie.technion.ac.il
[2] CNRS and Université Paris Diderot, France
adiro@liafa.univ-paris-diderot.fr

Abstract. This paper studies the set cover problem under the semi-streaming model. The underlying set system is formalized in terms of a hypergraph $G = (V, E)$ whose edges arrive one-by-one and the goal is to construct an edge cover $F \subseteq E$ with the objective of minimizing the cardinality (or cost in the weighted case) of F. We consider a parameterized relaxation of this problem, where given some $0 \leq \epsilon < 1$, the goal is to construct an edge $(1 - \epsilon)$-cover, namely, a subset of edges incident to all but an ϵ-fraction of the vertices (or their benefit in the weighted case). The key limitation imposed on the algorithm is that its space is limited to (poly)logarithmically many bits per vertex.

Our main result is an asymptotically tight trade-off between ϵ and the approximation ratio: We design a semi-streaming algorithm that on input graph G, constructs a succinct data structure $\mathcal{D}$ such that for every $0 \leq \epsilon < 1$, an edge $(1 - \epsilon)$-cover that approximates the optimal edge (1-)cover within a factor of $f(\epsilon, n)$ can be extracted from $\mathcal{D}$ (efficiently and with no additional space requirements), where

$$f(\epsilon, n) = \begin{cases} O(1/\epsilon), \text{ if } \epsilon > 1/\sqrt{n} \\ O(\sqrt{n}), \text{ otherwise} \end{cases} .$$

In particular for the traditional set cover problem we obtain an $O(\sqrt{n})$-approximation. This algorithm is proved to be best possible by establishing a family (parameterized by ϵ) of matching lower bounds.

1 Introduction

Given a *set system* consisting of a *universe* of items and a collection of item sets, the goal in the *set cover* problem is to construct a minimum cardinality subcollection of sets that covers the whole universe. This problem is fundamental to combinatorial optimization with applications ranging across many different domains. It is one of the 21 problems whose NP-hardness was established by Karp in [9] and its study has led to the development of various techniques in the field of approximation algorithms (see, e.g., [15]).

[⋆] A full version of this paper can be obtained from http://arxiv.org/abs/1404.6763.

[⋆⋆] Research supported in part by ANR project RDAM.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 453–464, 2014.

In this paper, we investigate the set cover problem under the *semi-streaming* model [5], where the sets arrive one-by-one and the algorithm's space is constrained to maintaining a small number of bits per item (cf. the *set-streaming* model of [14]). In particular, we are interested in the following two research questions: (1) What is the best approximation ratio for the set cover problem under such memory constraints? (2) How does the answer to (1) change if we relax the set cover notion so that the set subcollection is required to cover only a δ-fraction of the universe? On top of the theoretical interest in the aforementioned research questions, studying the set cover problem under the semi-streaming model is justified by several practical applications too (see [14]).

The Model. In order to fit our terminology to the graph theoretic terminology traditionally used in the semi-streaming literature (and also to ease up the presentation), we use an equivalent formulation for the set cover problem in terms of edge covers in hypergraphs: Consider some *hypergraph* $G = (V, E)$, where V is a set of n *vertices* and E is a (multi-)set of m *hyperedges* (henceforth *edges*), where each edge $e \in E$ is an arbitrary non-empty subset $e \subseteq V$. Assume hereafter that G does not admit any isolated vertices, namely, every vertex is incident to at least one edge. We say that an edge subset $F \subseteq E$ *covers* G if every vertex in V is incident to some edge in F. The goal of the *edge cover* problem is to construct a subset $F \subseteq E$ of edges that covers G, where the objective is to minimize the cardinality $|F|$.

A natural relaxation of the covering notion asks to cover some fraction of the vertices in V: Given some $0 < \delta \leq 1$, we say that an edge subset $F \subseteq E$ *δ-covers* G if at least δn vertices are incident to the edges in F, namely, $|V(F)| \geq \delta n$, where $V(F) = \{v \in V \mid \exists e \in F \text{ s.t. } v \in e\}$. Under this terminology, a cover of G is referred to as a 1-cover. This raises a bi-criteria optimization version of the set cover problem, where the goal is to construct an edge subset $F \subseteq E$ that δ-covers G with the objective of minimizing $|F|$ and maximizing δ. In this paper, we focus on approximation algorithms, where the cardinality of F is compared to that of an optimal edge (1-)cover of G.

In the *weighted* version of the edge cover problem, the hypergraph G is augmented with vertex *benefits* $b : V \to \mathbb{Q}_{>0}$ and edge *costs* $c : E \to \mathbb{Q}_{>0}$. The edge cover definition is generalized so that edge subset $F \subseteq E$ is said to *δ-cover* G if the benefit of the vertices incident to the edges in F is at least a δ-fraction of the total benefit, namely, $b(V(F)) \geq \delta \cdot b(V)$, where $b(U) = \sum_{v \in U} b(v)$ for every vertex subset $U \subseteq V$. The goal is then to construct an edge subset F that δ-covers $G = (V, E, b, c)$, where the objective is to maximize δ and minimize the cost of F, denoted $c(F) = \sum_{e \in F} c(e)$.

Under the *semi-streaming* model, the execution is partitioned into discrete time steps and the edges in E are presented one-by-one so that edge $e_t \in E$ is presented at time $t = 0, 1, \ldots, m - 1$, listing all vertices $v \in e_t$;[1] in the weighted version, the cost of e_t and the benefits of the vertices it contains are also

[1] With the exception of our related work discussion, all semi-streaming algorithms in this paper make a single (one way) pass over the input hypergraph.

listed. The key limitation imposed on the algorithm is that its space is limited; specifically, we allow the algorithm to maintain $\log^{O(1)} |G|$ bits per vertex, where $|G|$ denotes the number of bits in the standard binary encoding of G. Each edge $e \in E$ is associated with a unique *identifier* $\mathrm{id}(e)$ of size $O(\log m)$ bits, say, the time t at which edge e_t is presented. We may sometimes use the identifier $\mathrm{id}(e)$ when we actually refer to the edge e itself, e.g., replacing $c(e)$ with $c(\mathrm{id}(e))$; our intention will be clear from the context.

In contrast to the random access memory model of computation, where given a collection $\mathcal{I}$ of identifiers, one can easily determine which vertex in V is incident to which of the edges whose identifiers are in $\mathcal{I}$ simply by examining the input, under the semi-streaming model, the collection $\mathcal{I}$ by itself typically fails to provide this information. Therefore, instead of merely returning the identifiers of some edge δ-cover, we require that the algorithm outputs a *δ-cover certificate* χ for G which is a partial function from V to $\{\mathrm{id}(e) \mid e \in E\}$ with *domain* $\mathrm{Dom}(\chi) = \{v \in V \mid \chi \text{ is defined over } v\}$ and *image* $\mathrm{Im}(\chi) = \{\mathrm{id}(e) \mid \exists v \in \mathrm{Dom}(\chi) \text{ s.t. } \chi(v) = \mathrm{id}(e)\}$ that satisfies (1) if $v \in \mathrm{Dom}(\chi)$ and $\chi(v) = \mathrm{id}(e)$, then $v \in e$; and (2) $b(\mathrm{Dom}(\chi)) \geq \delta \cdot b(V)$. By definition, the image of χ consists of the identifiers of the edges in some edge δ-cover F of G and the quality of the δ-cover certificate χ is thus measured in terms of $c(\mathrm{Im}(\chi)) = c(F)$.

Our Contribution. Consider some unweighted hypergraph $G = (V, E)$ with optimal edge 1-cover $\mathtt{OPT}$. We design a deterministic semi-streaming algorithm, referred to as $\mathtt{SSSC}$ (acronym of the paper's title), for the edge (δ-)cover problem that given some $0 \leq \epsilon < 1$, outputs a $(1 - \epsilon)$-cover certificate χ_ϵ for G with image of cardinality $|\mathrm{Im}(\chi_\epsilon)| = O(\min\{1/\epsilon, \sqrt{n}\} \cdot |\mathtt{OPT}|)$.[2] This result is extended to the weighted case, where $G = (V, E, b, c)$, showing that $c(\mathrm{Im}(\chi_\epsilon)) = O(\min\{1/\epsilon, \sqrt{n}\} \cdot c(\mathtt{OPT}))$ (see Thm. 1 and additional theorems in the full version). In particular, for the edge (1-)cover problem, we obtain an $O(\sqrt{n})$-approximation for both the weighted and unweighted cases.

On the negative side, we prove that for every $\epsilon \geq 1/\sqrt{n}$, if a randomized semi-streaming algorithm for the set cover problem outputs a $(1 - \epsilon)$-cover certificate χ for G, then it cannot guarantee that $\mathbb{E}[|\mathrm{Im}(\chi)|] = o(|\mathtt{OPT}|/\epsilon)$ (see Thm. 3). This demonstrates that the approximation guarantee of our algorithm is asymptotically optimal for the whole range of parameter $0 \leq \epsilon < 1$ even for randomized algorithms.

Notice that $\mathtt{SSSC}$ has the attractive feature that the (near-linear size) data structure $\mathcal{D}$ it maintains is oblivious to the parameter ϵ. That is, the algorithm processes the stream of edges with no knowledge of ϵ, generating the data structure $\mathcal{D}$, and the promised $(1 - \epsilon)$-cover certificate χ_ϵ can be efficiently extracted from $\mathcal{D}$ (with no additional space requirements) for every $0 \leq \epsilon < 1$ (in fact several such covers for different values of ϵ can be extracted). From a bi-criteria optimization perspective, our lower bound implies that the parameterized collection $\{\chi_\epsilon\}_{0 \leq \epsilon < 1}$ encoded in $\mathcal{D}$ is an (asymptotically) optimal solution frontier (cf. Pareto optimality).

[2] Define $\min\{1/x, y\} = y$ when $x = 0$.

Using a simple adjustment of the randomized rounding technique for set cover (see, e.g., [15]), it is not difficult to show that a basic feasible solution to the linear program relaxation of a given set cover instance also serves as a compact data structure from which a $(1-\epsilon)$-cover certificate χ_ϵ can be extracted for every $0 \leq \epsilon < 1$. In fact, the approximation ratio obtained this way is better than ours, namely, $O(\log(1/\epsilon))$. However, our lower bound shows that this approach cannot be applied under the semi-streaming model.

Can our tight lower bound be an artifact of the requirement that the algorithm outputs a cover *certificate*? We nearly eliminate this possibility by proving that for every constant $c > 0$ and for every $\epsilon \geq n^{-1/2+c}$, even if the randomized algorithm only guarantees an "uncertified" output, i.e., only the identifiers of the edges in some edge $(1-\epsilon)$-cover F of G are returned, then the cardinality of F must still be large, specifically, $|F| = \Omega\left(\frac{\log\log n}{\log n} \cdot |\mathtt{OPT}|/\epsilon\right)$, where $\mathtt{OPT}$ in this case is proportional to $\epsilon^2 n$ (the proof of this lower bound and further discussion of negative results for set cover under the semi-streaming model are deferred to the full version).

Related Work. The work most closely related to the present paper is probably the one presented in Saha and Getoor's paper [14] that also considers the set cover problem under the semi-streaming model (referred to as *set-streaming* in [14]) formulated as the edge cover problem in hypergraphs. Saha and Getoor design a 4-approximation semi-streaming algorithm for the *maximum coverage* problem that given a hypergraph $G = (V, E)$ and a parameter k, looks for k edges that cover as many vertices as possible. Based on that, they observe that an $O(\log n)$-approximation for the optimal set cover can be obtained in $O(\log n)$ passes over the input. Using the terminology of the present paper, Saha and Getoor's maximum coverage algorithm is very efficient for obtaining edge $(1-\epsilon)$-covers as long as ϵ is large, but it does not provide any (single pass) guarantees for $\epsilon < 3/4$. In contrast, our algorithm has asymptotically optimal (single pass) guarantees for any $0 \leq \epsilon < 1$. Another paper that considers semi-streaming algorithms in hypergraphs is that of Halldórsson et al. [8] that studies the independent set problem.

The semi-streaming model was introduced by Feigenbaum et al. [5] for graph theoretic problems, where the edges of an n vertex input graph arrive sequentially and the algorithm is allowed to maintain only $\log^{O(1)} n$ bits of memory per vertex. Since the number of bits required to encode an n vertex graph is $n^{O(1)}$, the space-per-vertex bound used in the present paper can be viewed as a generalization of that of Feigenbaum et al. from graphs to hypergraphs. In any case, concerns regarding the comparison between the space bound used in the present paper and that of [5] can be lifted by restricting attention to hypergraphs with $m \leq 2^{\log^{O(1)} n}$ edges.

Various graph theoretic problems have been treated under the semi-streaming model. These include matching [13, 4, 11], diameter and shortest path [5, 6], min-cut and sparsification [1, 10], graph spanners [6], and independent set [8, 3].

Several variants of the set cover problem, all different than the problem studied in the present paper, have been investigated under the model of online computation. Alon et al. [2] focus on the online problem in which some master set system is known in advance and an unknown subset of its items arrive online; the goal is to cover the arriving items, minimizing the number of sets used for that purpose. Another online variant of the set cover problem is studied by Fraigniaud et al. [7], where the sets arrive online, but not all items have to be covered. Here, each item is associated with a penalty and the cost of the algorithm is the sum of the total cost of the sets chosen for the partial cover and the total penalty of the uncovered items.

2 A Semi-Streaming Algorithm

Our goal in this section is to design a semi-streaming algorithm for the edge (δ-)cover problem in hypergraphs. The algorithm, referred to as `SSSC`, is presented in Sec. 2.1 and its approximation ratio is analyzed in Sec. 2.2. In this extended abstract, we make the simplifying assumption that all numerical values (vertex benefits and edge costs) are encoded using $O(\log n)$ bits. Under this assumption, the space bounds of `SSSC` are quite trivial and the analysis in Sec. 2.2 yields Thm. 1. The assumption on the numerical values is lifted in the full version, where we implement `SSSC` so that it uses $O(\log |G|)$ memory bits per vertex for arbitrary input hypergraphs G as required by our model (recall that $|G|$ denotes the number of bits in a standard binary representation of G).

Theorem 1. *On a weighted input hypergraph $G = (V, E, b, c)$ with numerical values encoded using $O(\log n)$ bits, our algorithm uses $O(n \log(n + m))$ space, processes each input edge $e_t \in E$ in $O(|e_t| \log |e_t|)$ time, and produces a data structure $\mathcal{D}$ with the following guarantee: For every $0 \leq \epsilon < 1$, a $(1 - \epsilon)$-cover certificate χ_ϵ for G such that*

$$c(\mathrm{Im}(\chi_\epsilon)) = O\left(\min\left\{1/\epsilon, \sqrt{n}\right\} \cdot c(\mathtt{OPT})\right)$$

can be extracted from $\mathcal{D}$ in time $O(n \log n)$ with no additional space requirements, where `OPT` stands for an optimal edge (1-)cover of G.

2.1 The Algorithm

In what follows we consider some weighted hypergraph $G = (V, E, b, c)$ with optimal edge (1-)cover `OPT`. The main building block of algorithm `SSSC` is a procedure referred to as `COVER`. This procedure processes the stream of edges and outputs for every node $v \in V$, an identifier of an edge e that covers it, together with an integer variable that intuitively captures the quality of edge e in covering v. Algorithm `SSSC` uses two parallel invocations of `COVER`, one on the input graph G and one on some modification of G, and upon termination of the input stream, extracts the desired cover certificate from the output of these two invocations.

Procedure `COVER`**.** We maintain for each vertex $v \in V$, the following variables:

- $\mathrm{eid}(v)$ = an identifier $\mathrm{id}(e)$ of some edge $e \in E$; and
- $\mathrm{eff}(v)$ = a (not necessarily positive) integer called the *effectiveness* of v.

We denote by $\mathrm{eid}_t(v)$ and $\mathrm{eff}_t(v)$ the values of $\mathrm{eid}(v)$ and $\mathrm{eff}(v)$, respectively, at time t (i.e., just before e_t is processed). Procedure `COVER` that relies on the following definition is presented in Algorithm 1.

Definition 1 (Level, Effectiveness). *Consider edge e_t presented at time t and some subset $T \subseteq e_t$. The* level *of T at time t, denoted $\mathrm{lev}_t(T)$, is defined as $\mathrm{lev}_t(T) = \left\lceil \log_2 \frac{b(T)}{c(e_t)} \right\rceil$. Subset T is said to be* effective *at time t if for every $v \in T$, it holds that $\mathrm{lev}_t(T) > \mathrm{eff}_t(v)$.*

Note that $\emptyset$ is always vacuously effective.

Algorithm 1. `COVER`$(G = (V, E, b, c))$

```
Initialization $\forall v \in V$: $\mathrm{eid}(v) \leftarrow$ NULL and $\mathrm{eff}(v) \leftarrow -\infty$
for $t = 0, 1, \ldots$ do
    Read edge $e_t \in E$ from the stream
    Compute an effective subset $T \subseteq e_t$ of largest benefit $b(T)$
    for all $v \in T$ do
        $\mathrm{eid}(v) \leftarrow \mathrm{id}(e_t)$
        $\mathrm{eff}(v) \leftarrow \mathrm{lev}_t(T)$
    end for
end for
return $\mathrm{eid}(\cdot)$ and $\mathrm{eff}(\cdot)$
```

Algorithm `SSSC`**.** We are now ready to present our algorithm `SSSC`. On input weighted graph $G = (V, E, b, c)$, algorithm `SSSC` runs in parallel the following procedures that process the stream of edges:

P1: $(\mathrm{eid}_\infty(\cdot), \mathrm{eff}_\infty(\cdot)) \leftarrow$ `COVER`$(G = (V, E, b, c))$.

P2: $(\mathrm{eid}^{\mathbf{1}}_\infty(\cdot), \mathrm{eff}^{\mathbf{1}}_\infty(\cdot)) \leftarrow$ `COVER`$(G = (V, E, \mathbf{1}, c))$, where $\mathbf{1}$ stands for the function that assigns a unit benefit to all vertices $v \in V$.

P3: A procedure that maintains for every vertex $v \in V$, a variable $\mathrm{emin}(v)$ that stores the identifier of the minimum cost edge that covers v, seen so far.

P4: A procedure that stores for every vertex $v \in V$, its benefit $b(v)$.

Upon termination of the input stream, `SSSC` takes some parameter $0 \leq \epsilon < 1$ and extracts the desired $(1-\epsilon)$-cover certificate for G from the variables returned by procedures P1–P4. We distinguish between the following two cases.

- Case $\epsilon \geq 1/\sqrt{n}$:
  The algorithm looks for the largest integer r^* such that $b(I(\leq r^*)) \leq \epsilon b(V)$, where $I(\leq r^*) = \{v \in V : \mathrm{eff}_\infty(v) \leq r^*\}$, and returns the partial function $\chi : V \to \mathrm{id}(E)$ that maps every vertex $v \in V - I(\leq r^*)$ to $\mathrm{eid}_\infty(v)$.

– Case $\epsilon < 1/\sqrt{n}$:
  The algorithm looks for the largest integer r^* such that $|I^{\mathbf{1}}(\leq r^*)| \leq \sqrt{n}$, where $I^{\mathbf{1}}(\leq r^*) = \{v \in V : \mathrm{eff}^{\mathbf{1}}_{\infty}(v) \leq r^*\}$, and sets χ' to be the partial function $\chi' : V \to \mathrm{id}(E)$ that maps every vertex $v \in V - I^{\mathbf{1}}(\leq r^*)$ to $\mathrm{eid}^{\mathbf{1}}_{\infty}(v)$. Then, it returns the (complete) function $\chi'' : V \to \mathrm{id}(E)$ extended from χ' by mapping every vertex $v \in I^{\mathbf{1}}(\leq r^*)$ to $\mathrm{emin}(v)$.

Notice that the unweighted case is much simpler: If $G = (V, E)$, then procedure P2 is identical to procedure P1; moreover, procedures P3 and P4 are redundant since all vertices/edges admit a unit benefit/cost. Further note that procedures P1–P4 are oblivious to ϵ. Upon termination of the input stream, the algorithm extracts, for the given $0 \leq \epsilon < 1$, the desired $(1 - \epsilon)$-cover certificate for G from the variables returned by procedures P1–P4. In fact, several such cover certificates can be extracted for different values of ϵ.

2.2 Analysis

We begin our analysis with some observations regarding procedure `COVER`.

Observation 1. *If $T \subseteq e_t$ is effective at time t and $v \in T$, then $T \cup \{u\}$ is effective at time t for every $u \in e_t$ such that $\mathrm{eff}_t(u) \leq \mathrm{eff}_t(v)$.*

Notice that `COVER`'s updating rule guarantees that the effectiveness $\mathrm{eff}(v)$ is non-decreasing throughout the course of the execution. Employing Obs. 1, we can now derive Obs. 2 and 3 (the former follows by sorting the vertices $v \in e_t$ in non-decreasing order of the value of the effectiveness $\mathrm{eff}(v)$).

Observation 2. *The run-time of `COVER` on edge e_t is $O(|e_t| \log |e_t|)$.*

Observation 3. *If $T \subseteq e_t$ is effective at time t, then for every $v \in T$, it holds that $\mathrm{eff}_{t+1}(v) \geq \mathrm{lev}_t(T)$.*

We are now ready to establish the following lemma.

Lemma 1. *Consider some integer r. Procedure `COVER` guarantees that $b\left(\{v \in e_t \mid \mathrm{eff}_{t+1}(v) \leq r\}\right) < 2^{r+1} \cdot c(e_t)$.*

Proof. Assume by contradiction that there exists a subset $R \subseteq e_t$, $b(R) \geq 2^{r+1} \cdot c(e_t)$, such that $\mathrm{eff}_{t+1}(v) \leq r$ for every $v \in R$. Since the effectiveness is non-decreasing, it follows that $\mathrm{eff}_t(v) \leq r$ for every $v \in R$, hence the assumption that $b(R) \geq 2^{r+1} \cdot c(e_t)$ ensures that R is effective at time t. But by Obs. 3, the effectiveness $\mathrm{eff}_{t+1}(v)$ should have been at least $r + 1$ for every $v \in R$, in contradiction to the choice of R. □

Let $\mathrm{eff}_{\infty}(v)$ denote the value of the variable $\mathrm{eff}(v)$ upon termination of the input stream. Given some integer r, define

$$I(r) = \{v \in V \mid \mathrm{eff}_{\infty}(v) = r\} \text{ and } S(r) = \{e \in E \mid \exists v \in I(r) \text{ s.t. } \mathrm{eid}(v) = \mathrm{id}(e)\}$$

in accordance with the notation defined in Sec. 2.1. We extend these two definitions to intervals of integers in the natural way and denote the intervals $(-\infty, r]$ and (r, ∞) in this context by $\leq r$ and $> r$, respectively.

Lemma 2. *Consider some integer r. Procedure* `COVER` *guarantees that $b(I(\leq r)) < 2^{r+1} \cdot c(\texttt{OPT})$.*

Proof. Since the effectiveness is non-decreasing, Lem. 1 ensures that for every edge $e \in E$, it holds that $b\left(\{v \in e \mid \mathrm{eff}_\infty(v) \leq r\}\right) < 2^{r+1} \cdot c(e)$. The assertion is established by observing that

$$b(I(\leq r)) \leq \sum_{e \in \texttt{OPT}} b\left(\{v \in e \mid \mathrm{eff}_\infty(v) \leq r\}\right) < \sum_{e \in \texttt{OPT}} 2^{r+1} \cdot c(e) = 2^{r+1} \cdot c(\texttt{OPT}) \, ,$$

where the first inequality is due to the fact that `OPT` is an edge cover of G. □

Lem. 2 will be used to bound from above the benefit of the vertices that are not covered by the edges returned by our algorithm. We now turn to bound from above the cost of these edges.

Lemma 3. *Consider some integer r. The edge collection $S(r)$ satisfies $c(S(r)) < b(V)/2^{r-1}$.*

Proof. If $e_t \in S(r)$, then there exists some subset $R = R(e_t) \subseteq e_t$ with $\mathrm{lev}_t(R) = r$ such that for every vertex $v \in R$, we have (1) $\mathrm{eff}_t(v) < r$; and (2) $\mathrm{eff}_{t+1}(v) = r$. By definition, the fact that $\mathrm{lev}_t(R) = r$ implies that $c(e_t) < b(R)/2^{r-1}$. Since the variable $\mathrm{eid}(v)$ is updated only when $\mathrm{eff}(v)$ increases and since $\mathrm{eff}(v)$ is non-decreasing, it follows that if $e_t, e_{t'} \in S(r)$, $e_t \neq e_{t'}$, then the subsets $R(e_t)$ and $R(e_{t'})$ are disjoint. Therefore,

$$\sum_{e_t \in S(r)} c(e_t) < \frac{1}{2^{r-1}} \sum_{e_t \in S(r)} b(R(e_t)) \leq b(V)/2^{r-1} \, .$$

□

The following corollary is obtained by applying Lem. 3 to the integers $r + 1, r + 2, \ldots$

Corollary 1. *Consider some integer r. The edge collection $S(> r)$ satisfies $c(S(> r)) < b(V)/2^{r-1}$.*

The following lemma shows that we can extract from the variables returned by `COVER` an edge subset of low total cost which covers much of the items.

Lemma 4. *Consider some $0 < \epsilon < 1$ and let r^* be the largest integer such that $b(I(\leq r^*)) \leq \epsilon \cdot b(V)$. The edge collection $S(> r^*)$ satisfies $c(S(> r^*)) < 8 \cdot c(\texttt{OPT})/\epsilon$.*

Proof. Let r be an integer such that $2^{r+1} < \epsilon \cdot \frac{b(V)}{c(\texttt{OPT})} \leq 2^{r+2}$. Lem. 2 guarantees that $b(I(\leq r)) < 2^{r+1} \cdot c(\texttt{OPT}) < \epsilon \cdot b(V)$, hence $r \leq r^*$. It follows by Cor. 1 that $c(S(> r^*)) \leq c(S(> r)) < b(V)/2^{r-1} \leq 8 \cdot c(\texttt{OPT})/\epsilon$. □

We are now ready to establish the approximation guarantees of algorithm `SSSC`. Thm. 1 (stated under the assumption that all vertex benefits and edge costs are encoded using $O(\log n)$ bits) follows immediately from Thm. 2.

Theorem 2. *For any* $0 \leq \epsilon < 1$, *our algorithm outputs a* $(1-\epsilon)$*-cover certificate for* G *whose image has cost* $O\left(\min\left\{\frac{1}{\epsilon}, \sqrt{n}\right\} \cdot c(\mathtt{OPT})\right)$.

Proof. If $\epsilon \geq 1/\sqrt{n}$, then the assertion follows immediately from Lem. 4, so it remains to consider the case of $\epsilon < 1/\sqrt{n}$. We show that χ'' is a 1-cover certificates for G such that $c(\mathrm{Im}(\chi'')) = O(\sqrt{n} \cdot c(\mathtt{OPT}))$. Observe first that since $\mathtt{OPT}$ covers all vertices in V, it is also an optimal edge 1-cover of $G^{\mathbf{1}}$. Thus, Lem. 4 guarantees that $c(\mathrm{Im}(\chi')) < 8\sqrt{n} \cdot c(\mathtt{OPT})$. The vertices $v \in V - \mathrm{Dom}(\chi')$ are mapped under χ'' to $\mathrm{emin}(v)$. Since $|V - \mathrm{Dom}(\chi')| \leq \sqrt{n}$ and since $c(\mathrm{emin}(v)) \leq c(\mathtt{OPT})$ for every $v \in V$, it follows that

$$c(\mathrm{Im}(\chi'')) < 8\sqrt{n} \cdot c(\mathtt{OPT}) + |V - \mathrm{Dom}(\chi')| \cdot c(\mathtt{OPT}) \leq 9\sqrt{n} \cdot c(\mathtt{OPT}),$$

which completes the proof. □

3 Lower Bounds

A *randomized* semi-streaming algorithm $\mathtt{ALG}$ for the edge cover problem in hypergraphs is said to be an (n, s, ϵ, ρ)*-algorithm* (resp., an *uncertified* (n, s, ϵ, ρ)*-algorithm*) if given any n-vertex unweighted hypergraph G, $\mathtt{ALG}$ is guaranteed to maintain a memory of size at most s bits and to output a $(1-\epsilon)$-cover certificate for G with image of expected cardinality at most $\rho \cdot |\mathtt{OPT}|$ (resp., to output the identifiers of an edge $(1-\epsilon)$-cover of G whose expected cardinality is at most $\rho \cdot |\mathtt{OPT}|$), where $\mathtt{OPT}$ is an optimal edge cover of G. We are now ready to state the main theorems of this section. Thm. 3 is proved in Sec. 3.1, whereas due to space limitations, the proof of Thm. 4 is deferred to the full version. Observe that the constructions that lie at the heart of Thm. 3 and 4 are based on hypergraphs whose number of vertices and number of edges are polynomially related, that is, $m = n^{\Theta(1)}$.

Theorem 3. *For every integer* n_0, *there exists an integer* $n \geq n_0$ *such that for every* $\epsilon = \Omega(1/\sqrt{n})$, *the existence of an* $(n, o(n^{3/2}), \epsilon, \rho)$*-algorithm implies that* $\rho = \Omega(1/\epsilon)$.

Theorem 4. *Fix some constant real* $\alpha > 0$. *For every integer* n_0, *there exists an integer* $n \geq n_0$ *such that for every* $\epsilon \geq n^{-1/2+\alpha}$, *the existence of an uncertified* $(n, o(n^{1+\alpha}), \epsilon, \rho)$*-algorithm implies that* $\rho = \Omega\left(\frac{\log\log n}{\log n} \frac{1}{\epsilon}\right)$.

3.1 The Certified Case

We establish Thm. 3 by introducing a probability distribution $\mathcal{G}$ over n-vertex hypergraphs that satisfy the following two properties: (1) Every hypergraph in the support of $\mathcal{G}$ admits an edge cover of cardinality $O(\epsilon\sqrt{n})$. (2) For every *deterministic* semi-streaming algorithm $\mathtt{ALG}$ that given an n-vertex hypergraph G, maintains a memory of size $o(n^{3/2})$ and outputs a $(1-\epsilon)$-cover certificate χ for G, when $\mathtt{ALG}$ is invoked on a hypergraph chosen according to $\mathcal{G}$, the expected cardinality of $\mathrm{Im}(\chi)$ is $\Omega(\sqrt{n})$. The theorem than follows by Yao's principle.

The Construction of $\mathcal{G}$. Let q be a large prime power. Our construction relies on the *affine plane* $\mathcal{A} = (P, L)$, where P is a set of q^2 *points* and $L \subseteq 2^P$ is a set of $q(q+1)$ *lines* satisfying the following properties:
(1) every line contains q points;
(2) every point is contained in $q+1$ lines;
(3) every two distinct points are contained (together) in exactly one line; and
(4) every two distinct lines intersect in at most one point.
Two lines with an empty intersection are called *parallel*. The line set L can be partitioned into $q+1$ clusters $A_1, \dots, A_{q+1}$ referred to as *angles*, where $A_i = \{\ell_i^1, \dots, \ell_i^q\}$ for $i = 1, \dots, q+1$, such that two distinct lines are parallel if and only if they belong to the same angel. Refer to [12] for an explicit construction of such a combinatorial structure.

Consider some $\frac{1}{3q} \leq \epsilon \leq \frac{1}{66} - \frac{1}{3q}$ and let $r = \lceil 3\epsilon q \rceil$. We construct a random hypergraph $G = (V, E)$ based on the affine plane $\mathcal{A} = (P, L)$ as follows. Fix $V = P$. Randomly partition each line $\ell \in L$ into 2 edges $e_1(\ell) \cup e_2(\ell) = \ell$ by assigning each point in L to one of the 2 edges u.a.r. (and independently of all other random choices). It will be convenient to denote the set of edges corresponding to the lines in angle A_i by $E_i = \{e_1(\ell), e_2(\ell) \mid \ell \in A_i\}$. Let $e^* = P - \bigcup_{t=1}^{r} \ell_i^{j(t)}$, where i is an index chosen u.a.r. (and independently) from $[q+1]$ and $1 \leq j(1) < \cdots < j(r) \leq q$ are r distinct indices chosen u.a.r. (and independently) from $[q]$. In other words, e^* is constructed by randomly choosing an angle A_i and then randomly choosing r distinct lines $\ell_i^{j(1)}, \dots, \ell_i^{j(r)}$ from A_i; the edge consists of all points except those contained in these r lines.

Fix $E = E_1 \cup \cdots \cup E_{q+1} \cup \{e^*\}$. Observe that $n = |P| = q^2$ and $m = 1 + 2 \cdot |L| = 1 + 2 \cdot q(q+1)$. The execution is divided into two stages, where in the first stage, the edges in $E_1 \cup \cdots \cup E_{q+1}$ are presented in an arbitrary order and in the second stage, edge e^* is presented.

Analysis. We start the analysis by observing that G can be covered by the edge e^* and the edges in $\{e_1(\ell_i^{j(t)}), e_2(\ell_i^{j(t)}) \mid 1 \leq t \leq r\}$. Therefore,

$$|\mathtt{OPT}| \leq 2r + 1 = O(\epsilon q), \tag{1}$$

where the equation follows from the definition of $r = \lceil 3\epsilon q \rceil$ due to the requirement that $\epsilon \geq \frac{1}{3q}$.

Let s be the space of the deterministic semi-streaming algorithm $\mathtt{ALG}$. Thm. 3 is established by combining (1) with the following lemma (that ensures an $\Omega(q)$ expected image cardinality whenever $s = o(n^{3/2})$).

Lemma 5. *If $s \leq q^2(q+1)/48$, then w.p. $\geq 1/8$, the $(1-\epsilon)$-cover certificate returned by $\mathtt{ALG}$ has image of cardinality at least $q/3$.*

Bounding the Expected Entropy. The proof of Lem. 5 is based on information theoretic arguments that require the following definitions. Let X_i^j be a random variable that depicts the partition $(e_1(\ell_i^j), e_2(\ell_i^j))$ of line $\ell_i^j = e_1(\ell_i^j) \cup e_2(\ell_i^j)$ for every $i \in [q+1]$ and $j \in [q]$. Let $X_i = (X_i^1, \ldots, X_i^q)$ and $X = (X_1, \ldots, X_{q+1})$. The independent random choices in the construction of the hypergraph G guarantee that $H(X_i^j) = q$, $H(X_i) = q^2$, and $H(X) = q^2(q+1)$, where $H(\cdot)$ denotes the binary entropy function.

Let M be a random variable that depicts the memory image of ALG upon completion of the first stage of the execution. Since M is fully determined by X, it follows that $H(X, M) = H(X)$, hence $H(X \mid M) = H(X) - H(M)$. Recalling that M is described by s bits, we conclude that $H(M) \leq s \leq q^2(q+1)/48$, thus $H(X \mid M) \geq \frac{47}{48} \cdot q^2(q+1) = \frac{47}{48} \cdot H(X)$. The following lemma can now be established (proof deferred to the full version).

Lemma 6. *Our construction guarantees that* $\mathbb{P}_{i,j(1),\ldots,j(r)}\left(H\left(X_i^{j(1)}, \ldots, X_i^{j(r)} \mid M\right) \geq \frac{5}{6} \cdot rq\right) \geq 1/4$*, where* $i \in [q+1]$ *and* $1 \leq j(1) < \cdots < j(r) \leq q$ *are the random indices chosen during the construction of edge* e^**.*

Introducing the Random Variable Z. Let μ be the actual memory image of ALG upon completion of the first stage of the execution and recall that μ is some instance of the random variable M. Let Z be a real valued random variable that maps the event $M = \mu$ to the entropy in the joint random variable $X_i^{j(1)}, \ldots, X_i^{j(r)}$ given $M = \mu$. Observe that by the definition of conditional entropy, we have $\mathbb{E}[Z] = H(X_i^{j(1)}, \ldots, X_i^{j(r)} \mid M)$. If the event described in Lem. 6 occurs, then $\mathbb{E}[Z] \geq \frac{5}{6} \cdot rq$ and since Z is never larger than rq, we can apply Markov's inequality to conclude that $H\left(X_i^{j(1)}, \ldots, X_i^{j(r)} \mid M = \mu\right) \geq \frac{2}{3} \cdot rq$ w.p. $\geq 1/2$. The following corollary is established since the event described in Lem. 6 holds w.p. $\geq 1/4$.

Corollary 2. *W.p.* $\geq 1/8$*, the entropy that remains in* $X_i^{j(1)}, \ldots, X_i^{j(r)}$ *after* e^* *is exposed to* ALG *given that* $M = \mu$ *is at least* $\frac{2}{3} \cdot rq$ *bits.*

High Entropy Implies a Large Edge Cover. Condition hereafter on the event described in Cor. 2. Consider the $(1-\epsilon)$-cover certificate χ returned by ALG and let $P' = \bigcup_{t=1}^{r} \ell_i^{j(t)} = P - e^*$ be the set of points not covered by e^*. Let $R = \{p \in P' \mid p \in \mathrm{Dom}(\chi) \wedge \chi(p) \in E_i\}$ be the set of points not covered by e^* that are mapped under χ to some edge in E_i, where recall that E_i is the set of edges corresponding to the lines in angle A_i (the angle chosen in the random construction of e^*). Using Cor. 2, we establish the following lemma (proof deferred to the full version).

Lemma 7. *Our construction guarantees that* $|R| \leq rq/3$*.*

The cardinality of $\mathrm{Dom}(\chi)$ is at least $|\mathrm{Dom}(\chi)| \geq (1-\epsilon)q^2$. The choice of $r = \lceil 3\epsilon q \rceil$ ensures that $\epsilon q^2 \leq rq/3$, thus $|\mathrm{Dom}(\chi)| \geq q^2 - rq/3$. The key observation now is that even if all these $rq/3$ missing points from $\mathrm{Dom}(\chi)$ are in P', it still leaves us with $|\mathrm{Dom}(\chi) \cap (P' - R)| \geq rq/3$ by Lem. 7.

Every point in $\mathrm{Dom}(\chi) \cap (P' - R)$ is covered by some edge $e \in E_j$, $j \neq i$. The properties of the affine plane guarantee that each such edge e covers at most one point in line $\ell_i^{j(t)}$, which sums up to at most r points in P'. Thus, the image of χ must contain (the identifiers of) at least $q/3$ different edges. Lem. 5 follows.

References

[1] Ahn, K.J., Guha, S.: Graph sparsification in the semi-streaming model. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009, Part II. LNCS, vol. 5556, pp. 328–338. Springer, Heidelberg (2009)

[2] Alon, N., Awerbuch, B., Azar, Y., Buchbinder, N., Naor, J.: The online set cover problem. SIAM J. Comput. 39(2), 361–370 (2009)

[3] Emek, Y., Halldórsson, M.M., Rosén, A.: Space-constrained interval selection. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012, Part I. LNCS, vol. 7391, pp. 302–313. Springer, Heidelberg (2012)

[4] Epstein, L., Levin, A., Mestre, J., Segev, D.: Improved approximation guarantees for weighted matching in the semi-streaming model. In: STACS, pp. 347–358 (2010)

[5] Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., Zhang, J.: On graph problems in a semi-streaming model. Theor. Comput. Sci. 348, 207–216 (2005)

[6] Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., Zhang, J.: Graph distances in the data-stream model. SIAM J. Comput. 38(5), 1709–1727 (2008)

[7] Fraigniaud, P., Halldórsson, M.M., Patt-Shamir, B., Rawitz, D., Rosén, A.: Shrinking maxima, decreasing costs: New online packing and covering problems. In: Raghavendra, P., Raskhodnikova, S., Jansen, K., Rolim, J.D.P. (eds.) RANDOM 2013 and APPROX 2013. LNCS, vol. 8096, pp. 158–172. Springer, Heidelberg (2013)

[8] Halldórsson, B.V., Halldórsson, M.M., Losievskaja, E., Szegedy, M.: Streaming algorithms for independent sets. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6198, pp. 641–652. Springer, Heidelberg (2010)

[9] Karp, R.M.: Reducibility Among Combinatorial Problems. In: Miller, R.E., Thatcher, J.W. (eds.) Complexity of Computer Computations, pp. 85–103. Plenum Press (1972)

[10] Kelner, J.A., Levin, A.: Spectral sparsification in the semi-streaming setting. Theory Comput. Syst. 53(2), 243–262 (2013)

[11] Konrad, C., Magniez, F., Mathieu, C.: Maximum matching in semi-streaming with few passes. In: Gupta, A., Jansen, K., Rolim, J., Servedio, R. (eds.) APPROX 2012 and RANDOM 2012. LNCS, vol. 7408, pp. 231–242. Springer, Heidelberg (2012)

[12] Lindner, C.C., Rodger, C.A.: Design Theory, 2nd edn. Discrete Mathematics and its Applications. CRC Press (2011)

[13] McGregor, A.: Finding graph matchings in data streams. In: Chekuri, C., Jansen, K., Rolim, J.D.P., Trevisan, L. (eds.) APPROX 2005 and RANDOM 2005. LNCS, vol. 3624, pp. 170–181. Springer, Heidelberg (2005)

[14] Saha, B., Getoor, L.: On maximum coverage in the streaming model & application to multi-topic blog-watch. In: SDM 2009, pp. 697–708 (2009)

[15] Vazirani, V.V.: Approximation algorithms. Springer-Verlag New York, Inc., New York (2001)

Online Stochastic Reordering Buffer Scheduling

Hossein Esfandiari[1,⋆], MohammadTaghi Hajiaghayi[1,⋆], Mohammad Reza Khani[1,⋆], Vahid Liaghat[1,⋆], Hamid Mahini[1,⋆], and Harald Räcke[2]

[1] Computer Science Department, University of Maryland, College Park, MD 20742
[2] Institut für Informatik, Technische Universität München, München, Germany

Abstract. In this paper we consider online buffer scheduling problems in which an online stream of n items (jobs) with different colors (types) has to be processed by a machine with a buffer of size k. In the *standard* model initially introduced by Räcke, Sohler, and Westermann [31], the machine chooses an *active color* and processes items whose color matches that color until no item in the buffer has the active color (note that the buffer is refilled in each step). In the *block-operation* model, the machine chooses an active color and can–in each step–process all items of that color in the buffer. Motivated by practical applications in real-world, we assume we have prior stochastic information about the input. In particular, we assume that the colors of items are drawn i.i.d. from a possibly *unknown distribution*, or more generally, the items are coming in a *random order*. In the random order setting, an adversary determines the color of each item in advance, but then the items arrive in a random order in the input stream. To the best of our knowledge, this is the first work which considers the reordering buffer problem in stochastic settings.

Our main result is demonstrating constant competitive online algorithms for both the standard model and the block operation model in the unknown distribution setting and more generally in the random order setting. This provides a major improvement of the competitiveness of algorithms in stochastic settings; the best competitive ratio in the adversarial setting is $\Theta(\log\log k)$ for both the standard and the block-operation models by Avigdor-Elgrabli and Rabani [8] and Adamaszek et al. [3]. Along the way, we also show that in the random order setting, designing competitive algorithms with the same competitive ratios (up to constant factors) in both the block operation model and the standard model are equivalent. To the best of our knowledge this is the first result of this type which relates an algorithm for the standard model to an algorithm for the block-operation model. Last but not least, we show in the *uniform distribution* setting, in which the probabilities of appearances of all colors are the same, a simple greedy algorithm is the *best* online algorithm in both models.

1 Introduction

We consider a scheduling problem where a stream of items (jobs) with different colors (types) arrives online at a processing machine and each item needs to be processed on the machine. The cost of processing items with the same color is negligible in comparison to the cost of switching from an item to another item with a different color. In fact, the main source of the cost is the context switching cost. In order to decrease the

[⋆] Supported in part by NSF CAREER award 1053605, NSF grant CCF-1161626, ONR YIP award N000141110662, DARPA/AFOSR grant FA9550-12-1-0423.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 465–476, 2014.

context switching cost, a reordering buffer of size k is used to store the arrived items. The items inside the buffer could be reordered at any time to minimize the switching cost. We consider two buffer management models in this paper:

The Standard Model. In this model the machine has an *active color* which is the color of the most recently processed item. At each time, the machine can select an item e from the buffer to process. The item is removed from the buffer and the next item in the input stream takes its place. In order to process e, the machine has to change its active color to the color of item e. It incurs no cost if color of item e is the same as the active color and incurs a unit cost otherwise. This model is well studied in the literature [1, 2, 6–8, 19, 31].

The Block-operation Model. In this model, items are processed in blocks. At each point in time, the machine can select a block (subset) of items in the buffer with the same color and process all of them in a single step. The total cost is exactly the number of block operations to process all input items. This model was introduced recently in [3].

There are many applications for this versatile framework in real world problems, *e.g.*, computer graphics and rendering [27], information retrieval [13], production line management [32], disk scheduling [33]. For more details about the applications see [2, 3, 6–8, 19, 31].

The online buffer management problem has been studied for the adversary setting [1–3, 6, 8, 19, 31]. In the adversary setting items which have been chosen by an adversary, arrive online and the machine should process the input without any assumption about the rest of the input. The cost is compared to that of an optimal offline algorithm that knows the input stream in advance. The goal is to design an online algorithm which performs well in the worst case. A *competitive ratio* is a criterion used for analyzing online algorithms. The competitive ratio of an online algorithm is the worst case ratio of its cost to the cost of optimal offline algorithm.

We consider the online buffer management problem in the stochastic setting and propose the first constant competitive online algorithm for the problem. The competitive ratio that we consider in this paper is the ratio of the expected cost of the online algorithm to the expected cost of an optimum offline algorithm over all input sequences. In the stochastic setting, we have a prior knowledge about the input sequence. Our goal is to design an online algorithm with a constant competitive ratio. The three different stochastic models in the order of generality are as follows:

i.i.d. with known distribution: In this setting, the color of each item is determined independently based on a known distribution. The distribution does not change over time and is known. More precisely, the color of each item will be i with probability p_i. The algorithm designer knows probability p_i in advance.

i.i.d. with unknown distribution: This is exactly the same as the previous model except the algorithm designer does not know the distribution in advance.

random order: In this setting, the adversary determines the color of each item in advance. Then, items arrive in a random order. The algorithm designer only knows that the items arrive in a random order and does not know the color of each item in advance. Let n_i be the number of items with color i and let $p_i = \frac{n_i}{n}$. Note that the algorithm designer does not know these values. More precisely, the adversary chooses the color of each item in advance. Then, the input will be a random permutation of items where each permutation is chosen uniformly at random.

Since an online algorithm has more information in the known i.i.d. setting, designing a competitive online algorithm for the unknown i.i.d. setting is harder than the known i.i.d. setting. On the other hand, the following proposition shows that the random order setting is even more general than the unknown i.i.d. setting (see [26] for the proof idea). We consider the buffer scheduling problem for the most general case of the random order setting in this paper.

Proposition 1 ([26]). *Every α-competitive online algorithm in the random order setting is also an α-competitive online algorithm for the unknown i.i.d. setting.*

1.1 Related Work

The online buffer scheduling problem is well studied for the standard model in the adversary setting. It was first considered by Räcke et al. [31] where an online algorithm with an $O(\log^2 k)$-competitive ratio has been presented in the adversary setting. Englert and Westermann [19] enhanced the competitive ratio and designed an online algorithm with an $O(\log k)$-competitive ratio. Their proof consists of two parts. They first design an online algorithm with a buffer of size k which is 4-competitive against an optimum offline algorithm with a buffer of size $k/4$. Then, they prove the cost of an optimum offline algorithm with a buffer of size $k/4$ is within an $O(\log k)$ factor of the cost of an optimum offline algorithm with a buffer of size k. It has been shown there is a gap of $\Omega(\log k)$ between an optimum offline algorithm with a buffer of size $k/4$ and an optimum offline algorithm with a buffer of size k [1]. We show that the gap between two optimum offline algorithms with different buffer sizes is constant in all stochastic settings (w.h.p.). The next improvement over the competitive ratio is by Avigdor-Elgrabli and Rabani [6]. They provide a randomized online algorithm based on linear programming with competitive ratio $O(\log k/ \log\log k)$. Adamaszek et al. [2] enhance the previous result and design an online algorithm with competitive ratio $O(\sqrt{\log k})$ for the problem. They also prove that there is no online algorithm with competitive ratio $\Omega(\sqrt{\log k/ \log\log k})$ and $\Omega(\log\log k)$ for deterministic and randomized strategies, respectively. Avigdor-Elgrabli and Rabani [8] show that the bound is tight and design another randomized online algorithm based on linear programming with competitive ratio $O(\log\log k)$. However, we design an online algorithm with a constant competitive ratio for the problem in all stochastic settings. It is worth mentioning that designing an optimum algorithm for the offline buffer scheduling problem is NP-hard [5, 15]. Avigdor-Elgrabli and Rabani [7] recently designed a constant factor approximation algorithm for the problem in the offline setting.

The block-operation model was introduced by Adamaszek et al. [3] in the adversary setting. They provide an online algorithm with competitive ratio $O(\log\log k)$. They prove the bound is asymptotically tight in the block-operation model as well. Their online primal-dual algorithm is inspired by a work on the online algorithm design for the weighted caching problem [11]. In this paper, we design a constant competitive algorithm for the block-operation model in the stochastic setting as well.

Designing an online algorithm in the stochastic setting is well-studied in the computer science literature. There are many problems such as the mechanism design problem [14, 24], the bipartite matching problem [10, 20, 28, 26, 22, 29], the adwords problem [4, 16, 17], the oblivious routing problem [25, 23], and the secretary problem [9, 12, 21] which have been studied in different stochastic settings. We mention these works in three groups based on their stochastic settings. The first group is the set of works which design an online algorithm for the known i.i.d. setting *e.g.* [10, 20, 29] for the bipartite matching problem and [25, 23] for the oblivious routing problem. The second group consider their problems in the unknown i.i.d. setting and their goal is to design an online algorithm with a good competitive ratio in this setting *e.g.* for the bipartite matching [26] and for the adword problem [16]. The last group study their problems in the general random order setting. This model introduced by Dynkin [18] for the problem of finding a maximum number in a random order setting and was used recently for the mechanism design problem [24], the bipartite matching problem [22, 28], the adwords problem [4, 17], and the secretary problem [9, 12, 21] as well. Karande and Mehta [26] also show that every online algorithm which is α-competitive in the random order setting is α-competitive in the unknown i.i.d. setting as well. Note that we consider the buffer scheduling problem in the general random order setting in this paper.

1.2 Our Results

We focus on the online buffer management problem in the stochastic setting. In particular, we look for an online algorithm with two main properties. First, the algorithm should be simple and easy to implement. Second, it should perform well in expectation over all input sequences. Indeed, analyzing the algorithms under stochastic settings are often sophisticated and tricky. In this paper, we present simple algorithms for the online buffer management problem with constant competitive ratios in both the standard model and the block-operation model. The main result of this paper is the following theorem.

Theorem 1. *There is an online algorithm with a constant competitive ratio for both the standard model and the block-operation model in all stochastic settings.*

We first demonstrate a constant competitive online algorithm for the block-operation model and then we provide an online algorithm with a constant competitive ratio for the standard model. One obstacle throughout all the proofs arises from the various dependencies between the colors of different items. Throughout our analysis we often deal with the correlations by carefully choosing the proper random variables and showing negative correlations between them. Thus we are able to use the tail-bounds first proved by Panconesi and Srinivasan [30]. We discuss the required tools formally in Section 2.

In order to prove Theorem 1, we need a few definitions and theorems. Consider an instance of buffer scheduling problem with n items and define m as the number of colors with non zero probability. Let ALG_k^M be an algorithm with a buffer of size k for model M, where M will be equal to S or B for the standard model and the block-operation model respectively. Note that ALG_k^M can be an online or offline algorithm. Define $\mathrm{ALG}_k^M(\sigma)$ to be the cost of algorithm ALG_k^M on the input sequence

σ, and $\mathbf{E}[\mathrm{ALG}_k^M(\sigma)]$ to be expected cost of algorithm ALG_k^M over all possible input sequences, where all possible input sequences are defined based on the stochastic setting. In fact, in the known i.i.d. and the unknown i.i.d. settings with m different colors there are at most m^n possible input sequences and in the random order setting there are at most $n!$ possible input sequences. Similarly, define $\mathrm{ALG}_{k,i}^M(\sigma)$ and $\mathbf{E}[\mathrm{ALG}_{k,i}^M(\sigma)]$ to be the cost and the expected cost of algorithm ALG_k^M for the color i; so $\mathrm{ALG}_k^M(\sigma) = \sum_i \mathrm{ALG}_{k,i}^M(\sigma)$.

Let OPT_k^M be an optimum offline algorithm for the buffer scheduling problem with a buffer of size k for model M. We say algorithm ALG_k^M is α-competitive for the stochastic setting if its expected cost is not more than α times the expected cost of optimum, *i.e.*,

$$\mathbf{E}[\mathrm{ALG}_k^M(\sigma)] \leq \alpha\,\mathbf{E}[\mathrm{OPT}_k^M(\sigma)] + f_{-\sigma},$$

where the expectation is over all possible input sequences based on the stochastic setting. Here $f_{-\sigma}$ is a function which does not depend on the input sequence. First, we need an extension of a theorem of Englert and Westermann [19].

Theorem 2. *There is an online algorithm for both the standard model and the block-operation model with a buffer of size k which is 4-competitive against an optimum offline algorithm with a buffer of size $\frac{k}{4}$ in the stochastic setting.*

Indeed, Theorem 2 has been proved for the standard model with the following algorithm. A penalty is defined for every color i. At each time, the penalty for a color i increases by the number of items with color i in the buffer when the online algorithm processes a block of items with another color $j \neq i$. When the buffer becomes full the algorithm processes a block of items with the highest penalty color and resets the penalty of that color to zero. They prove the proposed algorithm with a buffer of size k is 4-competitive against an optimum offline algorithm with a buffer of size $\frac{k}{4}$ for the standard model in the adversary setting [19]. Every online algorithm for the standard model in the adversary setting with competitive ratio z works for the standard model in the stochastic setting with an expected competitive ratio at most z. Thus, the algorithm proposed in [19] works for the standard model in the stochastic setting. Indeed, it can be shown that the proposed algorithm works for the block-operation model as well; although minor changes are needed in the analysis (see the full version of the paper for a formal discussion).

We overcome the problem of bounding the cost of an optimum offline algorithm with a buffer of size $\frac{k}{4}$ based on the following theorem. This is the first result in the buffer scheduling problem which shows the expected cost of an offline optimum increases by a constant factor if the buffer becomes half. Note that this result is in contrast with a result of [1] which shows there is a gap of size $\Omega(\log k)$ between the cost of an optimum offline algorithm with a buffer of size $\frac{k}{4}$ and that of size k in the adversary setting.

Theorem 3. *Consider the buffer scheduling problem in the random order setting. We have*

$$\mathbf{E}[\mathrm{OPT}_{k/2}^B(\sigma)] = O(\mathbf{E}[\mathrm{OPT}_k^B(\sigma)]) + O(k^2).$$

We study the effect of a buffer size on the performance of an optimum offline algorithm in Section 2 and demonstrate a formal proof for the above theorem. In our analysis, we focus on showing the constant factor dependency without optimizing the constants that are incurred due to the concentration bounds. The current analysis leads to the constant factor (roughly) 120. Here, we give the intuition about the proof. Indeed, proving the above theorem directly seems challenging since we do not know anything about the behavior of an optimum offline algorithm[1]. Therefore, we first introduce a family of algorithms for the block-operation model called *threshold-based algorithms* which are simple and easy to implement.

Definition 1. *For a vector $\tau = (\tau_1, \tau_2, \cdots, \tau_m)$ of m integers, a* threshold-based algorithm (TALG$^\tau$) with respect to τ *is an algorithm with a buffer of size at least $\sum_{i=1}^{m} \tau_i + 1$ such that in each step if for any color i the number of items of color i in the buffer exceeds τ_i, it processes color i immediately.*

Our proof for Theorem 3 has two steps. First we prove for every input instance, there exists an offline threshold-based algorithm which approximates the cost of an optimum offline within a constant factor. We define a novel potential function for proving this fact. Intuitively, the new potential function increases when an algorithm keeps an item in the buffer and a new item arrives. Potential functions have been defined for analyzing an online algorithm in the adversary setting. The basic idea behind most of them is to increase the potential function when an algorithm processes a block of items with color i and the color of item e is not equal to i [19, 2]. There is a difference between our potential function and the previous ones. Our potential function changes upon item arrivals but the previous ones change upon a color change in the algorithm. As our potential function fluctuates based on item arrivals, it is a better tool to analyze our algorithm in the stochastic setting. We believe previous potential functions are mainly suitable to analyze an algorithm in the adversary setting. We analyze the expected value of our potential function for an optimum offline algorithm for every input and find a lower bound for this value based on distributional information of input. This result helps us to show there is an offline threshold-based algorithm which approximates an optimum offline solution. Next, we show for each offline threshold-based algorithm with a buffer of size k there exists a constant competitive offline threshold-based algorithm with a buffer of size $\frac{k}{2}$.

By combining Theorems 2 and 3, we prove Theorem 1 for the block-operation model. In the following theorem we also present an online algorithm with a constant competitive ratio for the standard model by converting an online algorithm for the block-operation model to an online algorithm for the standard model. We study the problem in the standard model in Section 3.

Theorem 4. *There is an online algorithm with a constant competitive ratio for the standard model in the stochastic setting.*

The intuition to prove the above theorem is to predict the most frequent color d in the input stream in a learning phase and then change the processing color to color d after processing each block of item. In fact, we want to make sure when an item with color

[1] Note that designing an optimum offline algorithm for this problem is NP-hard [5, 15].

d arrives, we process it online and do not store it in the buffer. We use the buffer to manage items with all colors except color d. We prove this strategy does not increase the cost by more than a constant factor. We propose an online algorithm for the block-operation model to process all colors except color d which is constant competitive to an offline optimum algorithm of the block-operation model. In order to prove Theorem 4, we need the following lemma which relates an optimum offline algorithm for the standard model to an optimum offline algorithm for the block-operation model. To the best of our knowledge, this is the first result of this type in the context.

Lemma 1. *Let* ALG_k^S *be an algorithm for the standard model with a buffer of size* k *and expected cost* $\mathbf{E}[\mathrm{ALG}_{k,i}^S(\sigma)]$ *for each color* i*. There is an algorithm for the block-operation model with a buffer of size* $2k$ *whose expected cost for a color* i *is at most* $O(\frac{1}{1-p_i})\,\mathbf{E}[\mathrm{ALG}_{k,i}^S(\sigma)]$*, where* p_i *is the probability of color* i*.*

The idea is to design an algorithm for the block-operation model, ALG_{2k}^B, to simulate the algorithm for the standard model, ALG_k^S, by increasing the buffer size. We divide the buffer of the algorithm for the block-operation model into two parts of size k. The first part is used to simulate the buffer of ALG_k^S at each time. In fact, the set of items stored in this part of the buffer is supposed to be a subset of items stored in the buffer of ALG_k^S at each moment. Second part is used to store items which ALG_k^S processes without storing them in the buffer. If ALG_k^S processes the items after storing them in the buffer, ALG_{2k}^B could easily simulate it by only using the first part of its buffer. However, the difficulty arises when the ALG_k^S processes many items without storing them in the buffer. In this case the algorithm for the block-operation model should store them and try to process them together. Thus, in order to prove Lemma 1 we should bound the number of block operations needed to process items in the second part of the buffer. We demonstrate a method to bound this number based on a concentration bound. Finally, observe that if for color i, p_i is close to one, Lemma 1 does not establish a constant gap between the expected cost of evicting i in the two models. However, if $p_i > 0.5$ for a color i, we can ignore the items with that color by losing factor 2 in the competitive ratio. This can be done by immediately evicting color i as soon as the buffer becomes full and there is at least a ball with color i in the buffer. We refer the reader to the full version of the paper for the formal proofs.

We also consider the problem in the *uniform distribution* setting. The uniform distribution setting can be thought of as a specific case of the unknown i.i.d. setting where the probabilities of all colors are the same.

Definition 2. *The greedy algorithm processes the color with maximum number of items in the buffer when the buffer is full.*

We demonstrate that the greedy algorithm is the best online algorithm for both the standard model and the block-operation model in the uniform distribution setting.

Theorem 5. *In the uniform distribution setting, the greedy algorithm is an optimum online deterministic algorithm, i.e., it has the best expected cost in both the standard model and the block-operation model.*

Roughly speaking, we describe a state of an online algorithm by its buffer content, number of remaining items, and the active color. Then we assign a value to each state of an algorithm which denotes the expected cost of the algorithm in the uniform distribution setting. For a fixed number of remaining items, we first show that the active color in a state does not change the expectation of the state. Second, we demonstrate that evicting a color with the most number of items in the buffer minimizes the expected cost. We prove this fact based on a coupling technique which is presented in the full version of the paper.

2 Block-Operation with Half the Buffer

In this section we study the structure of the (offline) optimum solution in the block-operation model. We show that in the random order model, w.h.p. the cost of the optimum solution increases by at most a constant factor, when halving the size of the buffer. This is in contrast to the adversarial model where the cost may increase by a logarithmic factor [1]. We exploit the properties of the random order model by introducing a novel class of algorithms, the *threshold-based algorithms* (Def. 1). An important property of algorithms in this class is that they can be adapted to only use half the buffer while increasing the cost by only a constant factor. As an intermediate result, we show that w.h.p. there exists an algorithm in this class which gives a constant factor approximation. This directly implies that w.h.p. for any instance $\mathrm{OPT}_{k/2}(\sigma) \leq O(1) \cdot \mathrm{OPT}_k(\sigma)$.

First, we need to provide the required tools for handling the dependencies among the colors of items. The negatively correlated variables we use in our analysis can be categorized in the following two theorems. The proofs, particularly that of Theorem 7 which handles the dependency between different colors, are relatively involved and presented in the full version of the paper. We believe these results might be of independent interest.

Theorem 6. *Lets fix color i and consider a random selection of b items without replacement from a set of n items where n_i of them have color i. For every $j \in [b]$, let boolean random variable X_j indicates whether the j-th picked item has color i. The variables $X_1, \dots, X_b$ are negatively correlated.*

Theorem 7. *Consider a random selection of r packs of b items without replacement from a set of items where there are n_j items with color j. Let n be the total number of items, $c_1, c_2, \dots, c_r$ be an arbitrary sequence of non-negative integers, x_i be the number of items with color j in the i-th pack of the selected items, and X_i be the boolean random variable indicating whether $x_i \geq c_i$. The random variables $X_1, \dots, X_r$ are negatively correlated.*

Recall that n denotes the total number of items in the input. In the remainder of this section we assume that $n \geq 200k^4$ and $k \geq 2$. We note that these assumptions do not change the asymptotical analysis of the approximation ratio since for $n < 200k^4$ the cost is bounded by $O(k^4)$. In order to derive a bound on the cost of the optimum solution we first analyze the structure of a (close-to-)optimum solution.

We say a color i is *rare* if $n_i \leq \frac{n}{k^2}$. Intuitively, if a color has few items, we can ignore these items by evicting them right away without increasing the cost by much. The following lemma formalizes this argument. Due to lack of space, we refer the reader to the full version of the paper for most of the proofs in this section.

Lemma 2. *An algorithm* ALG *can be transformed into an algorithm* ALG' *such that* ALG' *does not buffer items with rare colors and with probability at least* $1-e^{-k}$ *fulfills* $\mathrm{ALG}'(\sigma) \le 17\mathrm{ALG}(\sigma)$.

In the remainder of the section we assume that algorithms do not buffer rare items. In order to analyze the cost of an algorithm we introduce the following *potential function*. For an algorithm ALG, we define $\textsc{Pot}_i^{\mathrm{ALG}}(\sigma,t)$ to denote the number of color i items stored in the buffer of ALG when the t-th item arrives. We further define *the potential of color i for algorithm* ALG by $\textsc{Pot}_i^{\mathrm{ALG}}(\sigma) = \sum_t \textsc{Pot}_i^{\mathrm{ALG}}(\sigma,t)$ and the *potential of algorithm* ALG by $\textsc{Pot}^{\mathrm{ALG}}(\sigma) = \sum_i \textsc{Pot}_i^{\mathrm{ALG}}(\sigma)$. Clearly, $\textsc{Pot}^{\mathrm{ALG}}(\sigma) \le nk$ for any algorithm. We will drop the superscript if the algorithm in question is clear from the context.

Given an input sequence σ an algorithm has to decide how many color-changes to do for each color and when to do these color-changes. Let for an algorithm ALG, T_i be the number of times ALG decides to process color i during input sequence σ. We say ALG processes a color *non-frequently* if $T_i \le \frac{n_i}{104}$. Otherwise we say that the color i is processed *frequently*. Note that by our assumption all rare colors are processed frequently.

Lemma 3. *With probability at least* $1-e^{-k}$ *all non-frequent colors fulfill*

$$\textsc{Pot}_i^{\mathrm{ALG}}(\sigma) = \Omega\Big(\frac{n \cdot n_i}{T_i}\Big) \ , \tag{1}$$

for every algorithm ALG.

Proof. We can view the potential of an item as the number of time-steps that it stays in the buffer. We show that w.h.p. no matter where the algorithm performs the T_i color-changes for color i, the items of this color will on average stay in the buffer for $\Omega(\frac{n}{T_i})$ time-steps. We split σ into $21T_i$ intervals of length $\ell = \frac{n}{21T_i}$. Let for a color i and an interval I, $Y_i(I)$ denote the number of color i items in interval I. We say that an interval I is *rich for color i* if $Y_i(I) \ge \frac{n_i}{126T_i}$. Note that $\mathbf{E}[Y_i(I)] = \ell\frac{n_i}{n} = \frac{n_i}{21T_i}$.

We will show that a large number of rich intervals will lead to a large potential for color i. Therefore we first show that there exists a large number of rich intervals. Note that we need to show this for every possible value of T_i.

Claim. For a given color i and value T the probability that there exist at least $3T_i$ rich intervals for every $T_i \in [T, 2T]$ is at least $1-e^{-2k}$.

We only have k^2 non-frequent colors. Also, we can cover the possible range of $T_i \in [\frac{n_i}{k}, \frac{n_i}{104}]$ by at most $\log k$ intervals of the form $[T, 2T]$. Applying a union bound we obtain that with probability $1-(k^2\log(k))e^{-2k} \ge 1-e^{-k}$ every color has at least $3T_i$ rich intervals.

Suppose that for some color i we have a rich interval I such that neither I nor the following interval contains a color-change to i (See Figure 1). Then the at least $\frac{n_i}{126T_i}$ items in I stay in the buffer for at least $\frac{n}{21T_i}$ steps and, hence, contribute $\frac{n_i}{126T_i}\cdot\frac{n}{21T_i} = \Omega(\frac{n \cdot n_i}{T_i^2})$ to the potential. If we have $3T_i$ rich intervals there must exist at least T_i intervals that fulfill the above property which gives that the generated potential

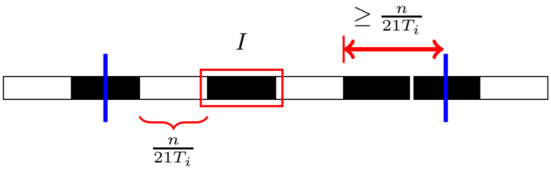


Fig. 1. This figure shows the input sequence partitioned into $21T_i$ intervals for a color i. The black intervals denote the rich intervals and a vertical line indicates a color-change where the algorithm evicts the items of color i. The highlighted rich interval I is at least $\frac{n}{21T_i}$ far from the next color-change since neither I nor the next interval contains a color-change to i.

for color i is at least

$$\textsc{Pot}_i^{\text{ALG}}(\sigma) = \Omega\Big(\frac{n \cdot n_i}{T_i}\Big) \ .$$

as desired. □

The following lemma shows that for any input with high probability there exists a threshold-based algorithm with a cost no more than a constant ratio of the optimum offline algorithm on the same input.

Lemma 4. *With probability at least $1 - 2e^{-k}$ over the sampled input σ, there exists a vector of m integers $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_m)$ such that $\sum_{i=1}^m \tau_i + 1 \leq k$ and*

$$\text{TALG}_k^{\boldsymbol{\tau}}(\sigma) \leq O(1)\text{OPT}_k(\sigma) \tag{2}$$

By Lemma 4, we can now prove the following theorem to show that by using a constant fraction of the buffer, the cost may increase by at most a constant factor. Finally, this completes the proof of Theorem 3.

Theorem 8. *With probability at least $1 - 2e^{-k}$ over the sampled input σ we have*

$$\text{OPT}_{\frac{k}{2}}(\sigma) \leq O(1)\text{OPT}_k(\sigma)$$

Proof (of Theorem 3). Indeed Theorem 8 leads to a stronger result where the *expected ratio* of the optimum cost with the buffer size $\frac{k}{2}$ to the optimum cost with the buffer size k is bounded. For a sampled sequence σ,

$$\text{OPT}_{\frac{k}{2}}(\sigma) \leq \begin{cases} O(1)\text{OPT}_k(\sigma) & \text{by Theorem 8 w.h.p.} \\ k\text{OPT}_k(\sigma) & \text{w.p.} \leq 2e^{-k} \end{cases}$$

Therefore

$$\mathbf{E}\left[\frac{\text{OPT}_{\frac{k}{2}}(\sigma)}{\text{OPT}_k(\sigma)}\right] = O(1) \ .$$

3 An Online Algorithm for the Standard Model

The main result of this section is to design an online algorithm for the standard model with a constant expected competitive ratio. The idea to design an online algorithm for the standard model is to simulate an online algorithm for the block-operation model for all the colors except the most frequent one. However, the algorithm does not know the most frequent color in advance. We propose a method to predict the most frequent color (d). The algorithm changes the active color to color d, when the online algorithm for the block-operation model does not output any color.

In order to show that the proposed online algorithm is a constant competitive algorithm for the standard model, we first prove that it is constant competitive against an optimum offline algorithm for the block-operation model. We then bound the cost of an optimum offline algorithm in the block-operation model with the cost of an optimum offline algorithm in the standard model in Lemma 1. In fact, we transform each offline algorithm for the standard model to an offline algorithm for the block-operation model by increasing the buffer size. The buffer of the proposed algorithm for the block-operation model is divided into two parts. The first part simulates the buffer of the algorithm for the standard model. The second part is used for storing items which the algorithm for the standard model outputs them with no cost without storing them in the buffer.

Let ALG^S_k be an offline algorithm for the standard model with a buffer of size k. We design an offline algorithm ALG^B_{2k} in Lemma 1 and show for every color i with np_i items in the input $\mathbf{E}[\mathrm{ALG}^B_{2k,i}] \leq O(\frac{1}{1-p_i}) \times \mathbf{E}[\mathrm{ALG}^S_{k,i}]$. In order to prove Lemma 1 we need Lemma 5.

Lemma 5. *We toss a biased coin* $\lceil \frac{4}{q} \rceil k$ *times where* $0 \leq q < 1$ *is the probability of head and* X *be the random variable indicating the number of heads. We can prove that* $(k+2)\,\mathbf{Pr}[X \leq k] \leq \mathbf{Pr}[k < X < \lceil \frac{4}{q} \rceil k]$

The proofs of Lemmas 1 and 5 are given in the full version of the paper, which ultimately leads to the proof of Theorem 4.

References

1. Aboud, A.: Correlation clustering with penalties and approximating the reordering buffer management problem. Master's thesis (2008)
2. Adamaszek, A., Czumaj, A., Englert, M., Räcke, H.: Almost tight bounds for reordering buffer management. In: STOC (2011)
3. Adamaszek, A., Czumaj, A., Englert, M., Räcke, H.: Optimal online buffer scheduling for block devices. In: STOC (2012)
4. Agrawal, S., Wang, Z., Ye, Y.: A dynamic near-optimal algorithm for online linear programming. CoRR (2009)
5. Asahiro, Y., Kawahara, K., Miyano, E.: NP-hardness of the sorting buffer problem on the uniform metric. Discrete Appl. Math. 160(10-11) (2012)
6. Avigdor-Elgrabli, N., Rabani, Y.: An improved competitive algorithm for reordering buffer management. In: SODA (2010)
7. Avigdor-Elgrabli, N., Rabani, Y.: A constant factor approximation algorithm for reordering buffer management. In: SODA (2013)

8. Avigdor-Elgrabli, N., Rabani, Y.: An optimal randomized online algorithm for reordering buffer management. In: FOCS (2013)
9. Babaioff, M., Immorlica, N., Kleinberg, R.: Matroids, secretary problems, and online mechanisms. In: SODA (2007)
10. Bahmani, B., Kapralov, M.: Improved Bounds for Online Stochastic Matching. In: de Berg, M., Meyer, U. (eds.) ESA 2010, Part I. LNCS, vol. 6346, pp. 170–181. Springer, Heidelberg (2010)
11. Bansal, N., Buchbinder, N., Naor, J(S.): A primal-dual randomized algorithm for weighted paging. In: FOCS (2007)
12. Bateni, M., Hajiaghayi, M., Zadimoghaddam, M.: Submodular secretary problem and extensions. In: Serna, M., Shaltiel, R., Jansen, K., Rolim, J. (eds.) APPROX 2010. LNCS, vol. 6302, pp. 39–52. Springer, Heidelberg (2010)
13. Blandford, D., Blelloch, G.: Index compression through document reordering. In: DCC (2002)
14. Cai, Y., Daskalakis, C., Weinberg, S.M.: On optimal multidimensional mechanism design. SIGecom Exch. 10(2), 29–33 (2011)
15. Chan, H.-L., Megow, N., van Stee, R., Sitters, R.: The sorting buffer problem is NP-hard. CoRR (2010)
16. Devanur, N.R., Jain, K., Sivan, B., Wilkens, C.A.: Near optimal online algorithms and fast approximation algorithms for resource allocation problems. In: EC (2011)
17. Devenur, N.R., Hayes, T.P.: The adwords problem: online keyword matching with budgeted bidders under random permutations. In: EC (2009)
18. Dynkin, E.B.: The optimum choice of the instant for stopping a markov process. Soviet Math. Dokl 4 (1963)
19. Englert, M., Westermann, M.: Reordering buffer management for non-uniform cost models. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 627–638. Springer, Heidelberg (2005)
20. Feldman, J., Mehta, A., Mirrokni, V., Muthukrishnan, S.: Online stochastic matching: beating $1 - 1/e$. In: FOCS (2009)
21. Freeman, P.R.: The secretary problem and its extensions: a review. Inter. Statistical Review 51(2), 189–206 (2011)
22. Goel, G., Mehta, A.: Online budgeted matching in random input models with applications to adwords. In: SODA (2008)
23. Hajiaghayi, M.T., Kleinberg, R.D., Leighton, T., Räcke, H.: New lower bounds for oblivious routing in undirected graphs. In: SODA (2006)
24. Hajiaghayi, M.T., Kleinberg, R., Parkes, D.C.: Adaptive limited-supply online auctions. In: EC (2004)
25. Hajiaghayi, M., Kim, J.H., Leighton, T., Räcke, H.: Oblivious routing in directed graphs with random demands. In: STOC (2005)
26. Karande, C., Mehta, A., Tripathi, P.: Online bipartite matching with unknown distributions. In: STOC (2011)
27. Krokowski, J., Räcke, H., Sohler, C., Westermann, M.: Reducing state changes with a pipeline buffer. In: VMV (2004)
28. Mahdian, M., Yan, Q.: Online bipartite matching with random arrivals: an approach based on strongly factor-revealing LPs. In: STOC (2011)
29. Manshadi, V.H., Gharan, S.O., Saberi, A.: Online stochastic matching: online actions based on offline statistics. In: SODA (2011)
30. Panconesi, A., Srinivasan, A.: Randomized distributed edge coloring via an extension of the chernoff–hoeffding bounds. SIAM Journal on Computing 26(2), 350–368 (1997)
31. Räcke, H., Sohler, C., Westermann, M.: Online scheduling for sorting buffers. In: Möhring, R.H., Raman, R. (eds.) ESA 2002. LNCS, vol. 2461, p. 820. Springer, Heidelberg (2002)
32. Spieckermann, S., Gutenschwager, K., VoB̈, S.: A sequential ordering problem in automotive paint shops. International journal of production research 42(9), 1865–1878 (2004)
33. Teorey, T.J., PinkertonA, T.B.: comparative analysis of disk scheduling policies. Communications of the ACM 15(3), 177–184 (1972)

Demand Queries with Preprocessing

Uriel Feige and Shlomo Jozeph

Department of Computer Science and Applied Mathematics,
The Weizmann Institute of Science, Rehovot, Israel
{uriel.feige,shlomo.jozeph}@weizmann.ac.il

Abstract. Given a set of items and a submodular set-function f that determines the value of every subset of items, a demand query assigns prices to the items, and the desired answer is a set S of items that maximizes the profit, namely, the value of S minus its price. The use of demand queries is well motivated in the context of combinatorial auctions. However, answering a demand query (even approximately) is NP-hard. We consider the question of whether exponential time preprocessing of f prior to receiving the demand query can help in later answering demand queries in polynomial time. We design a preprocessing algorithm that leads to approximation ratios that are NP-hard to achieve without preprocessing. We also prove that there are limitations to the approximation ratios achievable after preprocessing, unless NP $\subset$ P/poly.

1 Introduction

Given a universe U of n items, a *valuation function* is a set-function f that assigns nonnegative integer values to every subset of items, and satisfies the following three properties:

- *Normalization*: $f(\emptyset) = 0$.
- *Monotonicity*: for every two sets $S \subseteq T \subseteq U$, $f(S) \leq f(T)$.
- *M-bounded*: there is some fixed constant c independent of n such that $f(U)$ (the value assigned to the whole universe, which we denote by M) satisfies $\log M \leq n^c$. Hence every single value of the function f can be represented by a number of bits that is polynomial in n.

Valuation functions are used in order to represent the internal preferences of bidders in combinatorial auctions, where the items of U are for sale. The *maximum welfare* allocation problem associated with a combinatorial auction is the following: given the valuation functions of the bidders (f_i for bidder i), one needs to give each bidder i one *bundle* $B_i \subseteq U$ of items (a bundle is simply a set, which may also be empty), and these bundles need to be disjoint ($B_i \bigcap B_j = \emptyset$ for $i \neq j$). The objective is to do so while maximizing $\sum_i f_i(B_i)$, which is referred to as the welfare of the allocation.

Ideally, one would like to compute the maximum welfare allocation in time polynomial in n. However, there are two obstacles to overcome.

- The communication bottleneck. An explicit representation of a valuation function (as a table) might take space $2^n \log M$, and hence bidders might not be able to communicate their valuation functions to the seller.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 477–488, 2014.

- The computation bottleneck. Even if valuation functions have *succinct representations* of polynomial size (for example, the case of *single minded bidders* where f is determined by a single bundle SB, with $f(S) = 1$ if $SB \subseteq S$, and $f(S) = 0$ otherwise), the problem of computing the maximum welfare allocation is NP-hard, and also at least as hard to approximate as the notorious *set packing* problem.

Remark 1. Another difficulty associated with combinatorial auctions is how to provide incentives to the bidders to reveal their true valuation function to the seller, but this game-theoretic aspect is beyond the scope of our current work.

One way of handling the communication bottleneck is by allowing the seller to ask the bidder *queries* regarding the nature of his valuation function f. One wishes to design a polynomial time allocation algorithm in which the seller makes only polynomially many queries to each bidder. One natural class of queries is that of *value queries*: the query is a bundle S and the reply is its value $f(S)$. The class of queries that is the focus of this work is called *demand queries* (see [15] for several types of queries commonly used). For demand queries, one assumes that the utilities for bidders can be separated into two components: the value of the bundle received, and the payment that the bidder pays. Namely, if the bidder received bundle B and pays for it a price P, then the utility derived by bidder is $f(B) - P$. A demand query is a vector $\bar{p}$ of nonnegative integer prices to items ($\bar{p}(j)$ for item j) and its answer is the most preferable bundle for the bidder under these prices, namely, a bundle S that maximizes $f(S) - \sum_{j \in S} \bar{p}(j)$, together with the value $f(S)$. The ability to answer demand queries appears to be a natural requirement from a bidder, as without this ability, if the bidder comes to a market in which the items have prices, he himself would not know what he prefers to buy. Moreover, the assumption that bidders can answer demand queries turns out to be very beneficial for algorithms that approximate the maximum welfare problem. (Demand queries implement a separation oracle for the dual of the *configuration LP*, and solving this configuration LP is a first step in many of the approximation algorithms for maximum welfare.)

Despite their attractiveness, demand queries are problematic in the sense that even if a succinct representation of a valuation function is given (which allows efficient replies to value queries), answering demand queries is in general NP-hard (and also hard to approximate). This NP-hardness allows for the situation that for certain classes of valuation functions (such as submodular functions, to be defined shortly), the approximation ratio achieved for the maximum welfare problem if demand queries are allowed is strictly better (unless P=NP) than without demand queries (but with succinct representations of the valuation functions). See Section 1.1.

In the current work, we investigate a certain approach for reconciling the NP-hardness of demand queries with the desirability of being able to answer them. A valuation function with a succinct representation would in general have more than one such representation. Could it be that NP-hardness of answering demand queries is a consequence of the choice of representation, but under a different representation answering demand queries is easy? We model this question using the notion of *preprocessing*.

We envision the following situation. First, an arbitrary valuation function f is given. One is allowed to preprocess f for arbitrary amount of time, even more than exponential. The outcome of this preprocessing phase is a polynomial size advice string $A(f)$.

Thereafter, upon being given a vector $\bar{p}$ of prices as a demand query, the demand query is answered in polynomial time based only on $\bar{p}$, $A(f)$, and at most polynomially many value queries. (Typically, the original representation of f is succinct and allows efficient replies to value queries. In these cases this succinct representation can be made part of $A(f)$. However, some functions (in fact, almost all functions) do not have a succinct representation, and for them we assume access to a *value oracle* that can answer value queries.) The focus of our work is that of approximate answers to demand queries. The quality of the answering algorithm is measured by the *approximation ratio*: the ratio between $f(S) - \sum_{j \in S} p_j$ and $f(T) - \sum_{j \in T} p_j$, where S is the optimal solution and T is the solution returned by the algorithm.

Our results focus on the well studied class of *submodular* valuation functions (see Section 1.2 for definitions). Though this is considered to be a relatively simple class, answering demand queries is very difficult even for this class. The following proposition is well known.

Proposition 1. *For every $\epsilon > 0$, given a succinct representation of a submodular valuation function (without preprocessing) and a demand query, returning an answer with approximation ratio at most $n^{1-\epsilon}$ is NP-hard.*

Our main result shows that preprocessing helps.

Theorem 1. *For every submodular valuation function f (even with no succinct representation), there is a polynomial size advice string $A(f)$, such that given any demand query $\bar{p}$, the answer can be approximated in polynomial time within a ratio of* $\mathrm{O}(n^{3/4})$, *based only on $\bar{p}$, $A(f)$ and value queries.*

However, there is a limit to the effectiveness of preprocessing.

Theorem 2. *For some $\delta > 0$, there are submodular valuation functions with succinct representations, for which regardless of the polynomial size advice string given, demand queries cannot be approximated within a ratio better than $\Omega(n^{\delta})$ in polynomial time, unless NP has polynomial size circuits (namely, unless NP $\subset$ P/poly).*

We also consider in our work a natural subclasses of submodular functions (that we refer to as NH, *negative hyperedges*, see section 3) and show that for this class demand queries can be approximately answered (after preprocessing) within a ratio of $\mathrm{O}(n^{1/2})$. An even more restricted but very natural setting is that of MWIS (*maximum weight independent set*) demand queries, which is addressed in Section 2. The negative results of Theorem 2 apply already to a further restriction that we refer to as MIS (*maximum independent set*) demand queries.

1.1 Related Work

Submodular set-functions is a well studied class of functions and surveying all the literature about it is beyond the scope of this paper. We just mention a few facts that help put our work in perspective. There have been studies of trying to approximately learn (in the sense of approximately answering future *value queries*) submodular functions by making polynomially many value queries to the function (which is a weaker notion than preprocessing). Approximation ratios of $\tilde{\Theta}(\sqrt{n})$ are achievable [11]. Our Theorem 2 constructs submodular functions that have succinct representations that allow one to efficiently answer value queries exactly, and still they have no succinct representation that allows one to even approximately answer demand queries (unless NP $\subset$ P/poly). We remark that the ability to answer value queries is very powerful in the context of submodular functions. For example, it allows one to efficiently find the minimum value of the function [4], and to approximate the maximum within a factor of $1/2$ if the function is nonnegative [3].

There is much work on the maximum welfare problem, and we mention some of it. The focus on submodular valuation functions was initiated by [15]. The configuration LP and demand queries were introduced by [5]. With only value queries, the maximum welfare problem with submodular valuation functions can be approximated within a ratio of $1 - 1/e$ [6], and doing better is NP-hard, given succinct representations of the valuation functions [13]. However, with demand queries an approximation ratio better than $1 - 1/e$ is achievable [10].

Preprocessing is a natural and well studied notion in several different contexts, some of which are beyond the scope of our paper (e.g., quickly answering database queries, quickly breaking cryptographic schemes). The direction most relevant to the current work is that of preprocessing of NP-hard problems. This direction received much attention in the context of coding theory [2] and lattice problems [9,14]. In our negative results (Theorem 2) we build on earlier work of the authors [8] that considered preprocessing for constraint satisfaction problems, and introduced the notion of *universal factor graphs* as a method for establishing limitations on what preprocessing can achieve.

1.2 Preliminaries

Submodular set-functions can be defined in several equivalent ways. We shall use the following definition.

Definition 1. *A set-function* $f : 2^{[n]} \to \mathbb{R}$ *is* submodular *if it has* decreasing marginal values*, that is, for* $x \notin A \supseteq B$*,* $f(A \bigcup \{x\}) - f(A) \leq f(B \bigcup \{x\}) - f(B)$*.*

Submodular functions need not be monotone. However, from now on we shall always assume that functions are normalized, namely, $f(\emptyset) = 0$. Let us mention some easily verifiable properties of submodular functions. Every linear function is submodular. The sum of two submodular functions (and hence also the difference between a submodular function and a linear function) is submodular. Submodular functions are *subadditive*: for a submodular function f and pairwise disjoint $\{S_i\}$, $f(\bigcup S_i) \leq \sum f(S_i)$.

The submodular functions considered in our work will typically either be a submodular valuation function (thus being integer valued, monotone, and M-bounded) or the difference between a submodular valuation function and a nonnegative integer linear function (the price vector). In the latter case, the resulting submodular function will still be M-bounded, but not necessarily monotone. A rounding down aspect used in Section 4 might result in functions having noninteger values, but these values have simple representations.

Definition 2. *Given a submodular function* $f : 2^U \to \mathbb{R}$*, an* optimal set *with respect to* f *is a set* $S \subseteq U$ *that maximizes* $f(S)$*.*

Definition 3. *Given a set-function* $f : 2^U \to \mathbb{R}$*, the* hypergraph representation *of* f *is the unique function* $w_f : 2^U \to \mathbb{R}$ *such that for every* $A \subseteq U$*,*

$$f(A) = \sum_{B \subseteq A} w_f(B)$$

Note that an explicit formula for $w_f(A)$ is $w_f(A) = f(A) - \sum_{B \subset A} w_f(B)$ (where $\subset$ denotes strict containment). This allows one to determine w_f by an inductive process, starting with $w_f(\emptyset) = f(\emptyset) = 0$ and progressing to larger sets.

Definition 4. *The* linear part L_f *of a set function* f *is defined as*

$$L_f(A) = \sum_{\{x\} \subseteq A} w_f(\{x\})$$

The high order part H_f *of a set function* f *is defined as*

$$H_f(A) = f(A) - L_f(A)$$

The linear part of f is completely determined by the value of the hypergraph representation on individual items, and hence on vertices of the corresponding hypergraph, whereas the high order part is determined by the value of the hypergraph representation on larger sets of items, and hence on hyperedges of the corresponding hypergraph.

A submodular valuation function f can be decomposed into its linear part L_f and into its high order part H_f, with $f = L_f + H_f$. A demand query is a vector $\bar{p}$ of non-negative integer prices for the items, with the interpretation that the price of a bundle is the sum of the prices of the items that it contains. Hence $\bar{p}$ is a representation of a linear set-function $\bar{p}(S) = \sum_{i \in S} p(i)$. The desired answer for the demand query is thus an optimal set with respect to a new function $g = f - \bar{p}$. Observe that g differs from f only in its linear part. Hence equivalently, we may write $g = H_f + (L_f - \bar{p})$. Let us denote (the vector of coefficients of the linear function) $L_f - \bar{p}$ by $\bar{q}$. Observe that we may assume that the vector $\bar{q}$ is nonnegative. This can be explained as follows. g is a submodular function, and submodular functions have decreasing marginal costs. When trying to maximize the value of g we can always ignore any item x with $g(\{x\}) \leq 0$, because that item cannot possibly add positive value if included in the solution. Hence without loss of generality we may assume that such an item would not be part of the answer to the demand query. Hence all coordinates in which $\bar{q}$ is negative can be rounded up to 0 without affecting the answer to the demand query.

As a consequence of the above discussion, the preprocessing done by our algorithms will depend only on H_f and not on L_f. Then, given a demand query $\bar{p}$, we translate it as above to the corresponding $\bar{q} = L_f - \bar{p}$ (rounded up to 0), and attempt to find the set S maximizing $g(S) = H_f(S) + \bar{q}(S)$. Under this view, it is convenient to think of $\bar{q}$ rather than $\bar{p}$ as the query, and then L_f can effectively be ignored. Moreover, since H_f is not necessarily monotone, our positive results regarding preprocessing do not require f to be monotone – they hold with no change even if f is not monotone.

2 Approximately Answering MWIS Demand Queries

Given a graph $G(V, E)$ with n vertices, a *maximum weight independent set* (MWIS) query is an n dimensional vector $\bar{q}$ of nonnegative integers, where for every $1 \leq i \leq n$ entry $\bar{q}(i)$ is interpreted as the weight given to vertex i. Given G and $\bar{q}$, the goal is to output a maximum weight independent set with respect to these weights. The special case in which $\bar{q} \in \{0, 1\}^n$ is referred to as a *maximum independent set* (MIS) query. MWIS queries can easily be seen to be a special case of demand queries with respect to submodular valuation function. Let W be an upper bound on the possible weight a vertex might be assigned in a query $\bar{q}$. Then G can be thought of as a hypergraph representation of the submodular valuation function f_G, with $w_f(e) = -W$ for every edge $e \in E$, and $w_f(i) = nW$ for every vertex $i \in V$. Then the MWIS query $\bar{q}$ is equivalent to a demand query $\bar{p}$ with $\bar{p}(i) = nW - \bar{q}(i)$ for every $i \in V$, because the optimal answer to the demand query will never be a set of vertices that induces any edges (due to their large negative weight). In particular, taking $\bar{q}$ to be the all 1 vector shows that the problem of finding a maximum independent set in G can be formulated as a demand query. This observation coupled with the known $\Omega(n^{1-\epsilon})$ NP-hardness of approximation results for the maximum independent set problem [12,16] proves Proposition 1.

As a simple introduction to our proof of Theorem 1, we show how preprocessing G helps in improving the approximation ratio for the special case of MWIS queries.

Preprocessing. Given a graph $G = (V, E)$, consider the following collection of sets defined inductively. Let $J_j = \bigcup_{i=1}^{j} I_i$ ($J_0 = \emptyset$), where I_i (for $i \geq 1$) is a maximum independent set in $G_i = (V \setminus J_{i-1}, E_i)$ (where E_i denotes the the set of those edges induced by $V \setminus J_{i-1}$). The sets I_i are referred to as the *advice sets*.

Answering a MWIS Query. Given a query vector $\bar{q}$ assigning nonnegative weights to the vertices, return the advice set with highest sum of vertex weights. Namely, the advice set I_i that maximizes $\sum_{j \in I_i} \bar{q}(j)$.

Theorem 3. *The answer to the query is a $\sqrt{2n}$ approximation to the maximum weight independent set in the weighted graph $G' = (V, E, \bar{q})$.*

Proof. The following claim shows that the coloring defined by the sets $\{I_i\}$ has the property that any independent set of G is colored by at most $\sqrt{2n}$ colors.

Claim. Given an independent set I, $|\{i | I_i \bigcap I \neq \emptyset\}| < \sqrt{2n}$.

Proof. Suppose otherwise. Let $s_j = \left| I \setminus \bigcup_{i=1}^{j-1} I_i \right|$. Let $j_1, \cdots, j_{\sqrt{2n}}$ be the last $\sqrt{2n}$ indices for which $I_{j_i} \bigcap I \neq \emptyset$, given in reverse order (that is, advice set I_{j_i} was generated in the preprocessing phase after advice set $I_{j_{i+1}}$). Hence $s_{j_1} \geq 1$, and $s_{j_i} < s_{j_{i+1}}$. Induction establishes that $s_{j_i} \geq i$. Due to the maximality of I_i, $s_i \leq |I_i|$ (otherwise, I_i is not the maximum independent set in G after removing $\bigcup_{i=1}^{j-1} I_i$). Since $I_{j_1}, \cdots, I_{j_{\sqrt{2n}}}$ are disjoint, and $|I_{j_i}| \geq i$, the union of the advice sets would contain more than n vertices, which is a contradiction. □

Let I be a maximum weight independent set in G'. I is also an independent set in G. From the claim, I intersects at most $\sqrt{2n}$ of the I_i's. Hence for some j the weight of

$I' = I \bigcap I_j$ is at least $\frac{1}{\sqrt{2n}}$ of the weight of I. This advice set I_j is an independent set and contains I', so its weight must be at least the weight of I'. Thus, the I_i of highest weight is a $\sqrt{2n}$ approximation to the weight of I. □

3 Negative Hyperedges Set-Functions

Definition 5. *A set-function $f : 2^U \to \mathbb{R}$ is said to be a* negative hyperedges *(NH) function if its hypergraph representation satisfies $w_f(S) \leq 0$ whenever $|S| > 1$.*

Observe that every NH set-function is necessarily submodular. However, a submodular set function need not be NH. For example, the function $f(S) = 1$ for all nonempty S is submodular, and its hypergraph representation is $w_f(S) = (-1)^{|S|+1}$.

As an intermediate step towards proving Theorem 1 and strictly generalizing the notion of MWIS queries considered in Section 2, we consider demand queries with respect to NH functions. We first define our building block for the preprocessing stage.

Definition 6. *Given a submodular function $f : 2^U \to \mathbb{R}$, the* greedy optimal sets for f *is a collection of sets $\{S_i\}$, defined inductively: S_1 is the optimal set with respect to f, and S_{i+1} is the optimal set with respect to f among those sets in $U \setminus \bigcup_{j=1}^{i} S_j$.*

Preprocessing. Given an NH valuation function $f : 2^U \to \mathbb{N}$ that is M-bounded and a nonnegative integer k, define the functions $f_k(A) = 2^k |A| + H_f(A)$. Namely, f_k maintains the high order part of f, and makes the linear part equal to 2^k for every item.

Denote the greedy optimal sets for f_k by $\left\{S_i^k\right\}$. These sets, for all integer k in the range $0 \leq k \leq \log M$, will be referred to as the *advice sets*.

Answering a Demand Query. Recall from Section 1.2, that the demand query can be thought of as a nonnegative integer vector $\bar{q}$, and the goal is to find an optimal set with respect to $H_f + \bar{q}$. Let $\tilde{y} = \mathrm{argmax}_x \{q_x\}$ be the highest valued item in U according to $\bar{q}$. Let $T_k = \left\{x \in U | 2^k \leq q_x < 2^{k+1}\right\}$.

Answer with the highest valued set according to $H_f + \bar{q}$ from $\left\{\{\tilde{y}\}, T_k \bigcap S_i^k\right\}$, where S_i^k ranges over all advice sets.

Theorem 4. *The answer to the query is an $\mathrm{O}(\sqrt{n})$ approximation to the optimal set.*

Proof. To prove this theorem, we will use the following two lemmas.

Lemma 1. *For $|U| = n$, let $g : 2^U \to \mathbb{R}$ be a submodular function such that $g(\{x\}) = c > 0$ for all $x \in U$, and let $\{S_i\}$ be the greedy optimal sets for g. Then, for every $A \subset U$ with $g(A) > 0$ there is an i such that $g(A \bigcap S_i) \geq \frac{g(A)^2}{2cn}$.*

Proof. Subadditivity of g implies that $g(A) \leq \sum g(A \cap S_i)$. Suppose for the sake of contradiction that $g(A \bigcap S_i) < \frac{g(A)^2}{2cn}$ for all i. Then there must be at least $\frac{2cn}{g(A)}$ greedy sets. Let $s_j = g\left(A \setminus \bigcup_{i=1}^{j-1} S_i\right)$. Note that $s_j - s_{j+1} < \frac{g(A)^2}{2cn}$ (otherwise, $g(A \bigcap S_j) \geq \frac{g(A)^2}{2cn}$ by the subadditivity of g). Observe further that $g(S_j) \geq s_j$ (otherwise, S_j is not the optimal after removing $\bigcup_{i=1}^{j-1} S_i$). Since g gives a value of at most c to each element, $|S_j| \geq \frac{g(S_j)}{c} \geq \frac{s_j}{c}$.

Using these facts, we lower bound the number of items in $\biguplus_{i=1}^{2cn/g(A)} S_i$. Observe that $|S_1| \geq \frac{s_1}{c} = \frac{g(A)}{c}$. Since $s_j - s_{j+1} < \frac{g(A)^2}{2cn}$, we have that $s_j \geq g\left(A\right)\left(1 - \frac{(j-1)g(A)}{2cn}\right)$, and $|S_j| > \frac{g(A)}{c}\left(1 - \frac{(j-1)g(A)}{2cn}\right)$. The sum of the $\frac{2cn}{g(A)}$ terms is thus larger than $n = |U|$, a contradiction. □

Lemma 2. *Given an NH function $g : 2^U \rightarrow \mathbb{N}$, let $\tilde{g}$ be the function obtained from g by keeping the high order part of g, and rounding down each value in the linear part to the closest power of 2. Then* $\max_{S \subseteq U}[\tilde{g}(S)] \geq \frac{1}{4}\max_{T \subseteq U}[g(T)]$.

Proof. Let T be the optimal set for g. Select $S \subseteq T$ by including each item of T in S independently with probability $1/2$. We show that in expectation (all expectations taken over choice of S), $E[\tilde{g}(S)] \geq \frac{1}{4}[g(T)]$, and hence there must be an S satisfying the lemma.

Observe that $E[L_{\tilde{g}}(S)] \geq \frac{1}{2}E[L_g(S)] = \frac{1}{4}L_g(T)$, where the inequality is because rounding down loses at most a factor of 2, and the equality is because each item is included with probability $1/2$. Observe also that $E[H_g(S)] \geq \frac{1}{4}H_g(T)$, because every hyperedge (of size at least 2) in the hyperedge representation of g is negative, and is included into S (if included in T) with probability at most $1/4$. Hence $E[\tilde{g}(S)] = E[L_{\tilde{g}}(S)] + E[H_{\tilde{g}}(S)] = E[L_{\tilde{g}}(S)] + E[H_g(S)] \geq \frac{1}{4}(L_g(T) + H_g(T)) = \frac{g(T)}{4}$ □

A consequence of Lemma 2 is that given a query $\bar{q}$, we may round down each entry of $\bar{q}$ to the nearest power of 2, and lose at most a factor of 4 in the value of the optimal set. Hence we assume from now on that all entries in $\bar{q}$ are powers of 2. In particular, we can update the definition of T_k to $T_k = \left\{x \in U | q_x = 2^k\right\}$.

We use $\tilde{f}$ to denote $H_f + \bar{q}$. Recall that $\tilde{y}$ is the highest valued item. Let $\tilde{w} = q_{\tilde{y}}$. If $\tilde{w}$ is a $4\sqrt{n}$ approximation for the value of the optimal set for $\tilde{f}$, then by returning $\{\tilde{y}\}$ the theorem is proved.

Otherwise, let $A \subseteq U$ be the optimal set for $\tilde{f}$, with $\tilde{f}\left(A\right) \geq 4\tilde{w}\sqrt{n}$. Let r be such that $\tilde{w} = 2^r$. Hence, $r \geq k$ for any non-empty T_k. Using the subadditivity of $\tilde{f}$, there is a $k \leq r$ such that $\tilde{f}\left(A \bigcap T_k\right) \geq \tilde{f}\left(A\right)/2^{(r-k)/2+2}$ (otherwise, $\tilde{f}\left(A\right) \leq \sum_{k \leq r} \tilde{f}\left(A \bigcap T_k\right) < \tilde{f}\left(A\right) \sum_{k \leq r} 2^{(k-r)/2-2} < \tilde{f}\left(A\right)$). For this k Lemma 1 guarantees an i such that $\tilde{f}\left(A \bigcap T_k \bigcap S_i^k\right) \geq \tilde{f}\left(A \bigcap T_k\right)^2/2^{k+1}n \geq \tilde{f}\left(A\right)^2/2^{r+5}n$. Since S_i^k is maximal, using the decreasing marginal cost definition for submodular functions, $\tilde{f}\left(T_k \bigcap S_i^k\right) \geq \tilde{f}\left(A \bigcap T_k \bigcap S_i^k\right) \geq \tilde{f}\left(A\right)^2/2^{r+5}n \geq \tilde{f}\left(A\right)/8\sqrt{n}$. □

4 Approximately Answering Submodular Demand Queries

In this section we prove Theorem 1. The proof of Theorem 4 does not apply to some submodular valuation functions, because Lemma 2 need not hold. Consider for example a submodular function g with $g(A) = |A|+2$ for every nonempty A. Its maximum value is $n+2$. The rounded down version of it rounds down the linear part from 3 to 2, giving $\tilde{g}(A) = |A| + 2 - |A| = 2$, and the maximum drops to 2. Our solution is to work at a finer scale than powers of 2. A factor of 2 is broken to $n^{1/4}$ intermediate scales, and this will cost another factor of $n^{1/4}$ (beyond $n^{1/2}$) in the approximation ratio.

Preprocessing. Given a submodular valuation function $f : 2^U \to \mathbb{N}$ that is M-bounded and a nonnegative integer k, define the functions $f_k(A) = \left(1 + n^{-1/4}\right)^k |A| + H_f(A)$. Namely, f_k makes the linear part equal to $\left(1 + n^{-1/4}\right)^k$ for every item.

Denote the greedy optimal sets for f_k by $\left\{S_i^k\right\}$. These sets, for all nonnegative integer k satisfying $\left(1 + n^{-1/4}\right)^k \leq M$, will be referred to as the *advice sets*.

Answering a Demand Query. Given a query vector $\bar{q}$, let $\tilde{y} = \operatorname{argmax}_x \{q_x\}$ be the highest valued item in U according to $\bar{q}$.
Let $T_k = \left\{x \in U | \left(1 + n^{-1/4}\right)^k \leq q_x < \left(1 + n^{-1/4}\right)^{k+1}\right\}$.

Answer with the highest valued set according to $H_f + \bar{q}$ from $\left\{\{\tilde{y}\}, T_k \bigcap S_i^k\right\}$.

We now prove Theorem 1 by showing that the answer to the query is an $\mathrm{O}\left(n^{3/4}\right)$ approximation to the optimal set with respect to $H_f + \bar{q}$.

Proof. Define $\tilde{f}_1(C) = \sum_{\{x\}\subseteq C} \rho(q_x)$, where $\rho(0) = 0$ and $\rho(z)$ rounds positive integer z down to the nearest power of $1 + n^{-1/4}$. Let $\tilde{f} = \tilde{f}_1 + H_f$ and $\tilde{w} = w_{\tilde{f}}(\{\tilde{y}\})$. If $\{\tilde{y}\}$ is a $8n^{3/4}$ approximation to $H_f + \bar{q}$, we are done. Otherwise, there is a set A such that $f(A) \geq 8n^{3/4}\tilde{w}$.

Claim. If $f(A) \geq 2n^{3/4}\tilde{w}$, then $\tilde{f}(A) \geq f(A)/2$.

Proof. Note that $H_f(A) \leq 0$, otherwise f is not submodular. Let α be such that $H_f(A) = -(1 - \alpha) L_f(A)$. Then $f(A) = \alpha L_f(A)$. $L_f(A) \leq n\tilde{w}$, so $\alpha \geq 2n^{-1/4}$. $\left(1 + n^{-1/4}\right)\tilde{f}_1(A) \geq L_f(A)$, so $\tilde{f}(A) > \left(1 - n^{-1/4}\right) L_f(A) - (1 - \alpha) L_f(A) \geq (1 - \alpha/2) L_f(A) - (1 - \alpha) L_f(A) = f(A)/2$. □

Since $\tilde{f} \leq f$, we only lose a factor 2 when approximating $\tilde{f}$ instead of f.

Let A be the set of maximum value. $\tilde{f}(A) \geq 4\tilde{w}n^{3/4}$. Recall that
$T_k = \left\{x \in S | w_{\tilde{f}}(\{x\}) = \left(1 + n^{-1/4}\right)^k\right\}$. Let r be such that $\left(1 + n^{-1/4}\right)^r = \tilde{w}$.
$r \geq k$ for any non-empty T_k.

Using subadditivity, $\tilde{f}(A \bigcap T_k) \geq \frac{1}{4}n^{-1/4}\tilde{f}(A)\left(1 + n^{-1/4}\right)^{(k-r)/2}$ for some k (otherwise, $\tilde{f}(A) \leq \sum_{k\leq r} \tilde{f}(A \bigcap T_k) < \frac{1}{4}n^{-1/4}\tilde{f}(A)\sum_{k\leq r}\left(1 + n^{-1/4}\right)^{(k-r)/2} < \frac{1}{2}n^{-1/4}\tilde{f}(A)\sum_{k\leq r}\left(1 + n^{-1/4}\right)^{(k-r)} < \tilde{f}(A)$). The value of $A \bigcap T_k$ approximates the value of A within $4n^{1/4}\left(1 + n^{-1/4}\right)^{(k-r)/2}$. Using Lemma 1, $\tilde{f}\left(A \bigcap T_k \bigcap S_i^k\right) \geq \tilde{f}(A \bigcap T_k)^2 \left(1 + n^{-1/4}\right)^{-k}/2n \geq \tilde{f}(A)^2\left(1 + n^{-1/4}\right)^{-r}/32n^{3/2} \geq \tilde{f}(A)/8n^{3/4}$ for some i. Therefore, the value of $A \bigcap T_k \bigcap S_i^k$ is a $8n^{3/4}$ approximation to the value of A. Since S_i^k is maximal, the marginal value of every item is non-negative (otherwise, the set without this item has higher value). Using the decreasing marginal value definition for submodular functions, every item has non-negative marginal value for every subset of S_i^k. Hence, $\tilde{f}\left(T_k \bigcap S_i^k\right) \geq \tilde{f}\left(A \bigcap T_k \bigcap S_i^k\right)$. This proves that the answer to the query is a set of higher or equal value to a set that is an $\mathrm{O}\left(n^{3/4}\right)$ approximation to the optimal set with respect to $H_f + \bar{q}$. □

5 Hardness for Approximately Answering MIS Queries

In this section we revisit the setting of MIS queries introduced in Section 2. Given an input graph $G(V, E)$, one is allowed to preprocess this graph for arbitrary time and record a polynomial size advice string $A(G)$. Thereafter a subset $U \subset V$ is given as a query, and one is required to approximate the maximum independent set in the subgraph induced on U, and do so in polynomial time. We shall show that for some $\delta > 0$, even after preprocessing, MIS queries cannot be approximately answered within a ratio better than $\Omega(n^\delta)$, unless $NP \subset P/poly$. Given that MIS queries are a special case of demand queries with respect to a submodular function, this will thus prove Theorem 2.

We shall use Theorem 5, taken from [8]. Recall the problem max-3SAT: given a 3CNF formula with n variables and m clauses (each containing three literals), find an assignment that satisfies the maximum number of clauses. A *factor graph* is a template for a 3CNF formula that specifies the variables in each clause, but leaves the polarities of the variables unspecified. A family of factor graphs includes for each sufficiently large value of n one template. (The need to consider an infinite family is a complexity theory technicality. The reader may fix one n of interest and think of just one factor graph in this case.) Given a factor graph, a *3SAT query* is an assignment of polarities to the occurrences of variables in the template, and one is asked to solve the resulting max-3SAT instance in polynomial time. Preprocessing the factor graph before receiving the query is allowed.

Theorem 5. *For some $\rho < 1$, there is a family of factor graphs such that even after preprocessing, one cannot distinguish between satisfiable 3SAT queries, and those that are at most ρ-satisfiable, unless $NP \subset P/poly$.*

The value of ρ in Theorem 5 can be taken to be roughly $77/80$. A factor graph from the family of Theorem 5 is referred to as a *universal factor graph* (UFG). Proposition 2 is a first step towards proving Theorem 2.

Proposition 2. *For some $\rho < 1$, even after preprocessing, MIS queries cannot be answered with an approximation ratio better than $1/\rho$, unless $NP \subset P/poly$.*

Proof. Given a UFG F that is a template for 3CNF formulas with n variables and m clauses, use the following variation on the FGLSS reduction [7] to obtain a graph $G(V, E)$ on $8m$ vertices. Every clause $v \in F$ is associated with a cluster of eight vertices $v_1, \cdots, v_8 \in V$, one for each possible assignment to the three variables in the clause. There is an edge in E between two vertices iff their corresponding assignments disagree on some variable.

A 3SAT query to F can be cast as a MIS query to G. Setting the polarities to a clause v of F is equivalent to discarding the unique member of v's cluster in G that corresponds to an assignment not satisfying the clause, and keeping the remaining vertices of the cluster. The MIS query is the set U of vertices that remains. The size of the maximum independent set in the subgraph $U(G)$ induced on U is exactly the maximum number of satisfiable clauses in the 3SAT query. The proposition now follows from Theorem 5. □

Before we continue, we review the notion of *derandomized graph products* [1]. For a desired value of ϵ, we say that a d-regular graph is an *ϵ-expander* if the eigenvalues $d = \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ of its adjacency matrix satisfy $\max[|\lambda_2|, |\lambda_n|] \leq \epsilon\lambda_1$. It is known that for every $\epsilon > 0$ there is sufficiently large d (one would need $d > \Omega(1/\epsilon^2)$), such that there are ϵ-expanders of size n for all sufficiently large n. Moreover, it is known how to construct such ϵ-expanders.

Given a graph $G(V, E)$ its derandomized graph product $DG^k = (U, E_U)$ uses an arbitrary auxiliary d-regular ϵ-expander $F(V, E_F)$ defined on the same set of vertices V (and a set E_F edges unrelated to E). U consists of all walks with $k-1$ steps in F (hence $|U| = |V|d^{k-1}$), and there is an edge in DG^k between $(v_1, \cdots, v_k)$ and $(u_1, \cdots, u_k)$ iff there are i, j such that $(v_i, u_j) \in E$. Let $\alpha(G)$ denote the size of the maximum independent set in graph G. The following theorem is from [1].

Theorem 6. *For derandomized graph products as defined above and using an ϵ-expander with $\epsilon < \frac{\alpha(G)}{|V|}$:*

$$\alpha(G)d^{k-1}\left(\frac{\alpha(G)}{|V|} - \epsilon\right)^{k-1} \leq \alpha(DG^k) \leq \alpha(G)d^{k-1}\left(\frac{\alpha(G)}{|V|} + \epsilon\right)^{k-1}$$

In order to use Theorem 6 it will be convenient for us to use a special class of expanders. Given a graph $F = (V, E)$ on m vertices and a positive integer k, define the graph $F_k = (V \times k, E_k)$ as follows: every vertex of F is replaced by an independent set of size k, and every edge of F is replaced by a complete bipartite graph between the corresponding independent sets.

Proposition 3. *If F is an ϵ-expander, then so is F_k for every k.*

Proof. The adjacency matrix of F_k is a tensor product of two matrices: the adjacency matrix of F, and a k by k all 1 matrix (whose eigenvalues are k and 0 with multiplicity $k-1$). The eigenvalues of the tensor are all products of the eigenvalues of its factors. □

We can now prove Theorem 2.

Proof. Let $G(V, E)$ be a graph with $8m$ vertices, the outcome of Proposition 2. Recall that its vertices are arranged in m clusters of size 8. Pick ϵ sufficiently small (e.g., $\epsilon = \frac{1-\rho}{20}$ for ρ as in Proposition 2). Let F be an arbitrary d-regular ϵ-expander on m vertices. F_8 is an $8d$-regular graph on $8m$ vertices. Match vertices of G with those in F_8, with each cluster of G mapped to a cluster of F_8. For $k = \Theta(\log m)$, consider the derandomized graph product DG^k with respect to F_8. The number of its vertices is polynomial in m.

Given a MIS query U to $G(V, E)$ for which one wants to distinguish between the case that the optimal answer is m and the case that it is at most ρm, transform it into a MIS query U' to DG^k, where a vertex of DG^k belongs to U' iff the k vertices of that walk that it corresponds to are all in U. Observe that the set of vertices in U' is precisely what one would get by taking a k-fold derandomized graph product of the subgraph $U(G)$ with respect to F_7 rather than F_8. Proposition 3 implies that F_7 is an ϵ-expander. Theorem 6 (and some straightforward calculations that are omitted for lack of space) implies that the ratio between the m versus ρm cases has been amplified to some polynomial N^{δ}, where N is the total number of vertices in DG^k, and $\delta > 0$. □

Acknowledgements. Work supported in part by the Israel Science Foundation (grant No. 621/12) and by the I-CORE Program of the Planning and Budgeting Committee and The Israel Science Foundation (grant No. 4/11). We thank an anonymous ICALP reviewer for detailed comments.

References

1. Alon, N., Feige, U., Wigderson, A., Zuckerman, D.: Derandomized graph products. Computational Complexity 5(1), 60–75 (1995)
2. Bruck, J., Naor, M.: The hardness of decoding linear codes with preprocessing. IEEE Transactions on Information Theory 36(2), 381–385 (1990)
3. Buchbinder, N., Feldman, M., Naor, J., Schwartz, R.: A tight linear time (1/2)-approximation for unconstrained submodular maximization. In: FOCS. pp. 649–658 (2012)
4. Cunningham, W.H.: On submodular function minimization. Combinatorica 5(3), 185–192 (1985)
5. Dobzinski, S., Nisan, N., Schapira, M.: Approximation algorithms for combinatorial auctions with complement-free bidders. In: STOC, pp. 610–618 (2005)
6. Dobzinski, S., Schapira, M.: An improved approximation algorithm for combinatorial auctions with submodular bidders. In: SODA, pp. 1064–1073 (2006)
7. Feige, U., Goldwasser, S., Lovász, L., Safra, S., Szegedy, M.: Interactive proofs and the hardness of approximating cliques. J. ACM 43(2), 268–292 (1996)
8. Feige, U., Jozeph, S.: Universal factor graphs. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012, Part I. LNCS, vol. 7391, pp. 339–350. Springer, Heidelberg (2012)
9. Feige, U., Micciancio, D.: The inapproximability of lattice and coding problems with preprocessing. J. Comput. Syst. Sci. 69(1), 45–67 (2004)
10. Feige, U., Vondrák, J.: The submodular welfare problem with demand queries. Theory of Computing 6(1), 247–290 (2010)
11. Goemans, M.X., Harvey, N.J.A., Iwata, S., Mirrokni, V.S.: Approximating submodular functions everywhere. In: SODA, pp. 535–544 (2009)
12. Håstad, J.: Clique is hard to approximate withinn $n^{1-\epsilon}$. Acta Mathematica 182(1), 105–142 (1999)
13. Khot, S., Lipton, R.J., Markakis, E., Mehta, A.: Inapproximability results for combinatorial auctions with submodular utility functions. Algorithmica 52(1), 3–18 (2008)
14. Khot, S., Popat, P., Vishnoi, N.K.: $2^{\log^{1-\epsilon} n}$ hardness for the closest vector problem with preprocessing. In: STOC, pp. 277–288 (2012)
15. Lehmann, B., Lehmann, D.J., Nisan, N.: Combinatorial auctions with decreasing marginal utilities. Games and Economic Behavior 55(2), 270–296 (2006)
16. Zuckerman, D.: Linear degree extractors and the inapproximability of max clique and chromatic number. In: STOC, pp. 681–690 (2006)

Algorithmic Aspects of Regular Graph Covers with Applications to Planar Graphs*

Jiří Fiala[1], Pavel Klavík[2,**], Jan Kratochvíl[1], and Roman Nedela[3]

[1] Department of Applied Mathematics, Faculty of Mathematics and Physics,
Charles University, Malostranské náměstí 25, 118 00 Prague, Czech Republic
{fiala,honza}@kam.mff.cuni.cz

[2] Computer Science Institute, Faculty of Mathematics and Physics,
Charles University, Malostranské náměstí 25, 118 00 Prague, Czech Republic
klavik@kam.mff.cuni.cz

[3] Institute of Mathematics and Computer Science SAS and Matej Bel University,
Ďumbierska 1, 974 11 Banská Bystrica, Slovak Republic
nedela@savbb.sk

Abstract. A graph G *covers* a graph H if there exists a locally bijective homomorphism from G to H. We deal with *regular covers* in which this locally bijective homomorphism is prescribed by an action of a subgroup of $\mathrm{Aut}(G)$. Regular covers have many applications in constructions and studies of big objects all over mathematics and computer science.

We study *computational aspects* of regular covers that have not been addressed before. The decision problem RegularCover asks for two given graphs G and H whether G regularly covers H. When $|H| = 1$, this problem becomes Cayley graph recognition for which the complexity is still unresolved. Another special case arises for $|G| = |H|$ when it becomes the graph isomorphism problem. Therefore, we restrict ourselves to graph classes with polynomially solvable graph isomorphism.

Inspired by Negami, we apply the structural results used by Babai in the 1970's to study automorphism groups of graphs. Our main result is an FPT algorithm solving RegularCover for planar input G in time $\mathcal{O}^*(2^{e(H)/2})$ where $e(H)$ denotes the number of the edges of H. In comparison, testing general graph covers is known to be NP-complete for planar inputs G even for small fixed graphs H such as K_4 or K_5. Most of our results also apply to general graphs, in particular the complete structural understanding of regular covers for 2-cuts.

1 Introduction

The notion of *covering* originates in topology as a notion of local similarity of two topological surfaces. Suppose that we have two graphs: a big graph G and

* For the full version of this paper, see arXiv:1402.3774. The first three authors are supported by the ESF Eurogiga project GraDR as GAČR GIG/11/E023, the fourth author by the ESF Eurogiga project GReGAS as the APVV project ESF-EC-0009-10. The first author is also supported by the project Kontakt LH12095. The second and the third authors are also supported by Charles University as GAUK 196213.

** Also supported by GAČR 14-14179S.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 489–501, 2014.

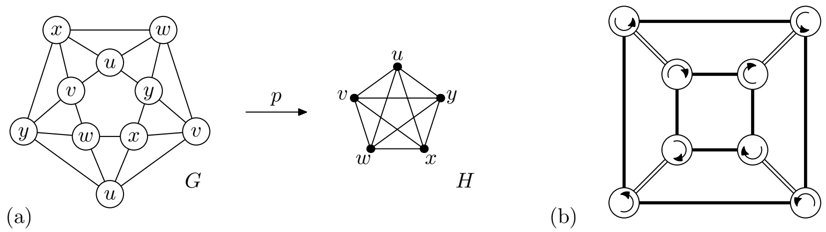


Fig. 1. (a) A covering projection p from a graph G to a graph H. (b) The Cayley graph of the dihedral group $\mathbb{D}_4$ generated by the 90° rotations (in black) and the reflection around the x-axis (in white).

a small graph H. We say that G *covers* H if there exists a mapping called a *covering projection* $p : G \to H$ which locally preserves the structure of G. The existence of a covering projection ensures that G looks locally the same as H; see Figure 1a. In this paper, we study algebraically restricted coverings called *regular coverings*; see Section 2 for the formal definition.

1.1 Applications of Graph Coverings

Suppose that G covers H and we have some information about one of the objects. How much knowledge does translate to the other object? It turns out that quite a lot, and this makes covering a powerful technique with many diverse applications.

Powerful Constructions. The reverse of covering called *lifting* can be applied to small objects in order to construct large objects of desired properties. For instance, the well-known Cayley graphs are large objects which can be described easily by a few elements of a group. See Figure 1b for an example. Cayley graphs were originally invented to study the structure of groups [6].

In the language of coverings, every Cayley graph G can be described as a lift of a one vertex graph H. Regular covers can be viewed as a generalization of Cayley graphs where the small graph H may contain more then one vertex. For example, the famous Petersen graph can be constructed as a lift of a two vertex graph H, see Figure 2a. Figure 2b shows a simple construction [17] of the Hoffman-Singleton graph [11] which is a 7-regular graph with 50 vertices.

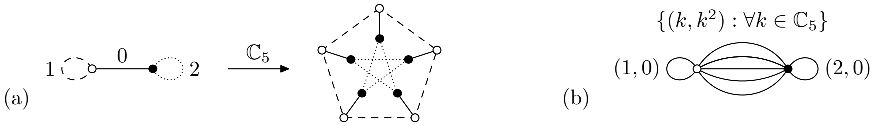


Fig. 2. (a) A construction of the Petersen graph by lifting with the group $\mathbb{C}_5$. (b) By lifting the described graph with the group $\mathbb{C}_5^2$, we get the Hoffman-Singleton graph. The five parallel edges are labeled $(0,0)$, $(1,1)$, $(2,4)$, $(3,4)$ and $(4,1)$.

The Petersen and the Hoffman-Singleton graphs are extremal graphs for the *degree-diameter* problem: Given integers d and k, find a maximal graph G with diameter d and degree k. In general, the size of G is not known. Many currently best constructions are obtained using the covering techniques [18].

Models of Local Computation. These and similar constructions have many practical applications in designing highly efficient computer networks [7,20] since these networks can be efficiently described/constructed and have many strong properties. In particular, networks based on covers of simple graphs allow fast parallelization of computation as described e.g. in [5,1].

1.2 Complexity Aspects

In the constructions we described, the covers are regular and satisfy additional algebraic properties. The advantage of regular covers is that they are easier to describe. In this paper, we initiate the study of the computational complexity of regular covering.

Problem: RegularCover
Input: Connected graphs G and H.
Output: Does G regularly cover H?

Relations to Covers. This problem is closely related to the complexity of general covering which was widely studied before. We try to understand how much the additional algebraic structure changes the computational complexity. Study of the complexity of general covers was pioneered by Bodlaender [5] in the context of networks of processors in parallel computing, and for fixed target graph was first asked by Abello et al. [9]. The problem H-Cover asks for an input graph G whether it covers a fixed graph H. The general complexity is still unresolved but papers [14,10] show that it is NP-complete for every r-regular graph H where $r \geq 3$.

The complexity results concerning graph covers are mostly NP-complete unless the graph H has very simple structure. The additional algebraic structure of regular graph covers makes sometimes the problem easier, as shown by the following two contrasting results. The problem H-Cover remains NP-complete for several small fixed graphs H (such as K_4, K_5) even for planar inputs G [4]. On the other hand, our main result is that the problem RegularCover is fixed-parameter tractable in the number of edges of H for every planar graph G and for every H.

Relations to Cayley Graphs and Graph Isomorphism. The notion of regular covers builds a bridge between two seemingly different problems. If the graph H consists of a single vertex, it corresponds to recognizing Cayley graphs which is still open; a polynomial-time algorithm is known only for circulant graphs [8]. When both graphs G and H have the same size, we get graph isomorphism testing. Our results are far from this but we believe that better understanding of regular covering can also shed new light on these famous problems.

Theoretical motivation for studying graph isomorphism is very similar to RegularCover. For practical instances, one can solve isomorphism very efficiently using heuristics. But a polynomial-time algorithm working for all graph is not known and it is very desirable to understand the complexity of graph isomorphism. The notion of isomorphism is widely used in mathematics to show that two seemingly different mathematical structures are the same. One proceeds by guessing a mapping and proving that this mapping is an isomorphism. The natural complexity question is whether there is a better way in which one algorithmically derives an isomorphism. Similarly, regular covering is a well-known mathematical notion which is algorithmically interesting and not understood.

Further, a regular covering is described by a semiregular subgroup of the automorphism group $\mathrm{Aut}(G)$. Therefore it seems to be closely related to computation of $\mathrm{Aut}(G)$ since one should have a good understanding of this group first to solve the regular covering problem. The problem of computing automorphism groups is known to be closely related to graph isomorphism.

Finding Lifts and Quotients. The input of RegularCover gives two graphs G and H. There are two other variants in which the input specifies only one of the two graphs and ask for existence of the other graph of, say, a given size. If only H is given, the problem is lifting and the answer is always positive. The theory of covering describes a technique called voltage assignment which can be applied to generate all k-fold covers G. Questions like efficient enumeration of all non-isomorphic lifts, or finding a lift of additional properties are nevertheless highly non-trivial and very interesting; e.g., the conjecture of Negami [19].

The other variant gives only G and asks for existence of a quotient H which is regularly covered by G and $|G| = k|H|$. This problem is NP-complete even for fixed $k = 2$, proved in a different language by Lubiw [15]. Lubiw shows that testing existence of a fixed-point free involutory automorphism is NP-complete which is equivalent to existence of a half-quotient H. This reduction does not imply NP-completeness for the RegularCover problem. Since the input gives also a graph H, one can decode the assignment of the variables from it, and thus this reduction does not work. We actually conjecture that for a fixed k the RegularCover problem is not NP-complete. Also, the reduction of Lubiw cannot be modified for planar inputs G. Our algorithmic and structural insights allow an efficient enumeration of all quotients H of a given planar graph G.

1.3 Our Results

As already stated, we show that the regular cover problem is for planar graphs easier than testing general covering. We use the asymptotic notation $f = \mathcal{O}^*(g)$ which omits polynomial factors. Our main result is the following FPT algorithm:

Theorem 1.1. *There is an* FPT *algorithm for* RegularCover *for planar inputs G in the parameter $e(H)$, running in time $\mathcal{O}^*(2^{e(H)/2})$ where $e(H)$ is the number of edges of H.*

It is important that most of our results apply to general graphs. We wanted to generalize the result of Babai [2] which states that it is sufficient to solve graph isomorphism for 3-connected graphs. Our main goal was to understand how regular covering behaves with respect to vertex 1-cuts and 2-cuts. For 1-cuts, regular covering behaves non-trivially only on the central block of G, so they are easy to deal with. But regular covering can behave highly complex on 2-cuts. From structural point of view, we give a complete description of this behaviour. Algorithmically, we succeeded partially and we need several other assumptions to get an efficient algorithm; see the full version for details.

Planar graphs are very important and also well studied in connection to covering. Negami's Theorem [19] dealing with regular covers of planar graphs is one of the oldest results in topological graph theory; therefore we decided to start the study of computational complexity of RegularCover for planar graphs. In particular, our theory applies to planar graphs since they satisfy these additional assumptions.

Our Approach. We quickly sketch our approach. The algorithm proceeds by a series of *reductions* replacing parts of the graphs by edges. These reductions are inspired by the approach of Negami [19] and turn out to follow the same lines as the reductions introduced by Babai for studying automorphism groups of planar graphs [2,3]. Since the key properties of the automorphism groups are preserved by the reductions, computation of automorphism groups can be reduced to solving it for 3-connected graphs [2]. In [13,12], this is used to compute automorphism groups of planar graphs since the automorphism groups of 3-connected planar graphs are the automorphism groups of tilings of the sphere, and are well-understood.

The RegularCover problem is more complicated, and we use the following novel approach. When the reductions reach a 3-connected graph, the natural next step is to compute all its quotients; there are polynomially many of them. What remains is the most difficult part, to test for each quotient whether it corresponds to H after unrolling the reductions. This process is called *expanding* and the issue here is that there may be exponentially many different ways to expand the graph, so we have to test in a clever way whether it is possible to reach H. Our algorithm consists of several subroutines, most of which we indeed can perform in polynomial time. Only one subroutine (finding a certain "generalized matching") we have not been able to solve in polynomial time.

This slow subroutine can be avoided in some cases:

Corollary 1.2. *If the planar graph G is 3-connected or if $k = |G|/|H|$ is odd, then the algorithm of Theorem 1.1 can be modified to run in polynomial time.*

2 Definitions and Preliminaries

A multigraph G is a pair $(V(G), E(G))$ where $V(G)$ is a set of vertices and $E(G)$ is a multiset of edges. We denote $|V(G)|$ by $v(G)$ and $|E(G)|$ by $e(G)$. The graph can possibly contain parallel edges and loops. Further, we consider graphs with colored edges and also with three different edge types (directed edges, undirected edges and a special type called halvable edges). We consider such general objects, because they naturally arise during reductions.

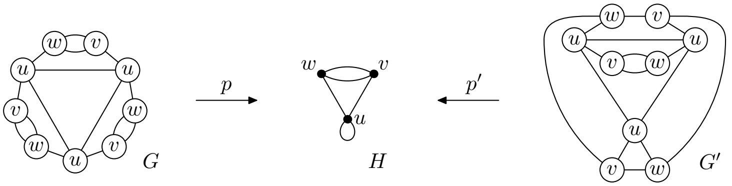


Fig. 3. Two covers of H. The projections p_v and p'_v are written inside of the vertices, and the projections p_e and p'_e are omitted. Notice that each loop is realized by having two neighbors labeled the same, and parallel edges are realized by having multiple neighbors labeled the same. Also covering projections preserve degrees.

Automorphism Groups. Each element π of the automorphism group $\mathrm{Aut}(G)$ acts on G, permutes its vertices and edges while it preserves incidences between the edges and the vertices. A subgroup of $\mathrm{Aut}(G)$ is called *semiregular* if its non-identity permutations do not fix any vertex and any *non-halvable edge.*

Coverings. A graph G *covers* a graph H (or G is a *cover* of H) if there exists a locally bijective homomorphism p called a *covering projection*. A homomorphism means that p consists of two mappings $p_v : V(G) \to V(H)$ and $p_e : E(G) \to E(H)$ such that $p_e(uv) = p_v(u)p_v(v)$ for every $uv \in E(G)$. The local bijectivness condition states that for every vertex $u \in V(G)$ the mapping p_e restricted to the edges incident with u is a bijection. Figure 3 contains two examples of graph covers. Again, we mostly omit subscripts and just write $p(u)$ or $p(e)$.

A *fiber* of a vertex $v \in V(H)$ is the set $p^{-1}(v)$, i.e., the set of all vertices $V(G)$ that are mapped to v. We adopt the standard assumption that both G and H are connected. Then all fibers of p are of the same size. In other words, $|G| = k|H|$ for some $k \in \mathbb{N}$ which is the size of each fiber, and we say that G is a *k-fold cover* of H.

Regular Coverings. A regular covering projection p is defined by a semiregular subgroup Γ of $\mathrm{Aut}(G)$. We get $p : G \to G/\Gamma$ where the graph G/Γ, called a *quotient* of G, is defined as follows: The vertices of G/Γ are the orbits of the action Γ on $V(G)$, the edges of G/Γ are the orbits of the action Γ on $E(G)$. A vertex-orbit $[v] = \{\pi(v) : \pi \in \Gamma\}$ is incident with an edge-orbit $[e] = \{\pi(e) : \pi \in \Gamma\}$ if and only if the vertices of $[v]$ are incident with the edges of $[e]$. We naturally construct $p : G \to G/\Gamma$ by mapping the vertices to their vertex-orbits and the edges to their edge-orbits. Since Γ acts semiregularly on G, one can prove that p is a $|\Gamma|$-fold regular covering. For Figure 3, the covering projection p is regular since we get $H \cong G/\Gamma$ for $\Gamma \cong \mathbb{C}_3$, and p' is not regular.

We note that when Γ fixes some halvable edge e, then $p(e)$ is a half-edge of G/Γ. In the theory of regular coverings, it is natural to decompose each edge into two half-edges, each attached to one vertex. The reason is that such graphs are closed under taking quotients, see [16] for more details. For alternative definition of regular covering and more details, see the full version. Since the size of Γ is polynomial in the size of G and H, we can give it as a certificate, and we get:

Lemma 2.1. *The problem* RegularCover *is in* NP.

3 Structural Properties of Atoms

In this section, we establish structural theory used in the algorithm. We introduce special inclusion-minimal subgraphs of G called atoms. We show that they behave nicely with respect to the regular covers and replace them by edges in the reduction.

Block-Trees. The *block-tree* T of G is a tree defined as follows. Consider all articulations in G and all maximal 2-connected subgraphs which we call *blocks* (with bridge-edges also counted as blocks). The block-tree T is the incidence graph of the articulations and the blocks. It is well-known that each automorphism of G induces an automorphism of T, and that the automorphism group preserves the *centrum* of T which is always either the central articulation, or the central block. The central articulation is fixed by every automorphism. So if G has a non-trivial semiregular automorphism, it necessarily has a central block.

In the following, we shall assume that T contains a central block. We orient edges of the block-tree T towards the central block; so the block-tree becomes rooted. A *subtree* of the block-tree is defined by any vertex different from the central block acting as *root* and by all its descendants.

Definition of Atoms. Let u and v be two distinct vertices of degree at least three joined by at least two parallel edges. Then the subgraph induced by u and v is called a *dipole*. Let B be a block of G, so B is a 2-connected graph. Two vertices u and v form a *2-cut* $U = \{u, v\}$ if $B \setminus U$ is disconnected. We say that a 2-cut U is *non-trivial* if $\deg(u) \geq 3$ and $\deg(v) \geq 3$.

We first define a set $\mathcal{P}$ of subgraphs of G which we call *parts*:

- A *block part* is a subgraph non-isomorphic to K_2 induced by the blocks of a subtree of the block-tree.
- A *proper part* is a subgraph S defined by a non-trivial 2-cut U of a block B not containing the central block. It consists of a connected component C of $G \setminus U$ together with u and v and all edges between $\{u, v\}$ and C.
- A *dipole part* is any dipole.

The inclusion-minimal elements of $\mathcal{P}$ are called *atoms*. We distinguish *block atoms*, *proper atoms* and *dipoles* according to the type of the defining part. Block atoms are either pendant stars, or pendant blocks possibly with single pendant edges attached to it. Also each proper atom or dipole is a subgraph of a block. For an example, see Figure 4.

We use topological notation to denote the *boundary* ∂A and the *interior* $\mathring{A}$ of an atom A. If A is a dipole, we set $\partial A = V(A)$. If A is a proper or block atom, we set ∂A equal to the set of vertices of A which are incident with an edge

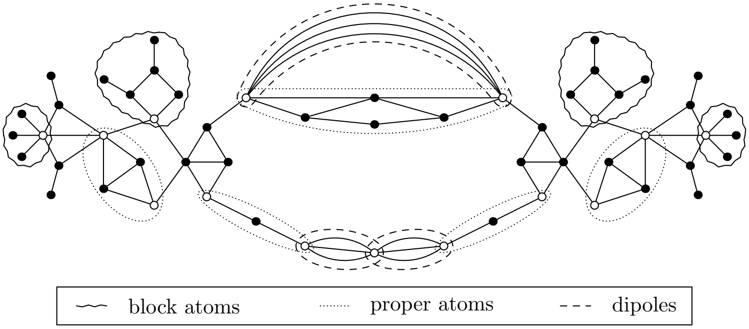


Fig. 4. An example of a graph with indicated atoms. The white vertices belong to the boundery of some atom, possibly several of them.

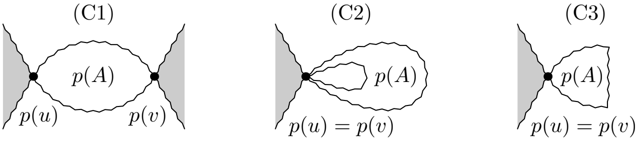


Fig. 5. How can Q look in G_i/Γ_i, depending on the cases (C1), (C2) and (C3)

not contained in A. For the interior, we use the standard definition $\mathring{A} = A \setminus \partial A$ where we only remove the vertices ∂A, the edges adjacent to ∂A are kept.

Lemma 3.1. *Let A and A' be two atoms. Then $A \cap A' = \partial A \cap \partial A'$.*

We call a graph *essentially 3-connected* if it is a 3-connected graph with possibly single pendant edges attached to it. The automorphism group of an essentially 3-connected graph is a subgroup of the automorphism group when pendant edges are removed. Every block atom is essentially 3-connected. A proper atom A with $\partial A = \{u, v\}$ becomes essentially 3-connected after adding the edge uv, and we denote this modification by A^+.

Quotionts of Atoms. Let Γ be a semiregular subgroup of $\mathrm{Aut}(G)$, which defines a regular covering projection $p : G \to G/\Gamma$. Negami [19, p. 166] investigated possible projections of proper atoms, and we further pursue this question.

Let A be an atom with $\partial A = \{u, v\}$ and let $Q = p(A)$.

(C1) *An edge-quotient Q.* The atom A is preserved in G/Γ, meaning $p(A) \cong A$.
(C2) *A loop-quotient Q.* The interior of the atom A is preserved and the vertices u and v are identified, i.e., $p(\mathring{A}) \cong \mathring{A}$ and $p(u) = p(v)$.
(C3) *A half-quotient Q.* The covering projection p is a $2k$-fold cover. There exists an involutory permutation π in Γ exchanging u and v which preserves A.

See Figure 5.

Lemma 3.2. *For every atom A and every semiregular subgroup Γ defining covering projection p, one of the cases (C1), (C2) and (C3) happens. Moreover, for a block atom we have exclusively the case (C1).*

It is easy to see that the edge/loop-quotient of an atom is unique. For half-quotients uniqueness is not true. Each half-quotient is determined by an involutory permutation. For planar graphs, there are at most polynomially many half-quotients of a proper atom. For a dipole, we get generally at most $2^{\lfloor e(A)/2 \rfloor}$ different half-quotients and this bound is tight; see Figure 6. The bound plays the key role for the complexity of our algorithm of Section 4; in one subroutine, we iterate over all half-quotients of a dipole.

The Reduction. The reduction starts with a graph G and produces a sequence of graphs $G = G_0, G_1, \ldots, G_r$. To produce G_{i+1} from G_i, we find all atoms of G_i. We replace a block atom A by a pendant edge of some color based at u where $\partial A = \{u\}$. We replace each proper atom or dipole A with $\partial A = \{u, v\}$ by a new edge uv of some color and of one of the three edge types according to the type of A. According to Lemma 3.1, the replaced parts for the atoms of $\mathcal{A}$ are pairwise disjoint, so the reduction is well defined.

We stop at step r when G_r is *primitive* which means that it contains no further atoms. A primitive graph is either K_2, or C_n or a 3-connected graph, together with possibly single pendant edges attached. We call this sequence of graphs starting with G and ending with a primitive graph G_r the *reduction series* of G.

We distinguish three symmetry types of atoms which describe how symmetric each atom is. When such an atom is reduced, we replace it by an edge carrying the type. Let A be a proper atom or dipole with $\partial A = \{u, v\}$. The atom is *halvable* if it has some half-quotient. If it is not halvable, we call it *symmetric* if it has some automorphism transposing u and v, and *asymmetric* otherwise. A block atom is by definition symmetric.

We replace halvable atoms by *halvable edges*, symmetric atoms by *undirected edges* and assymetric atoms by *directed edges.* We consider only the automorphisms which preserve these edge types and indeed the orientation of directed edges. For planar graphs, the type of each atom can be determined in polynomial time.

Further, the edges introduced in G_i are colored and their colors encode isomorphism classes of the atoms of G_{i-1}. We say that two graphs G and G' are isomorphic if there exists an isomorphism which preserves all colors and edge types. All established results transfer to colored graphs and colored atoms without any

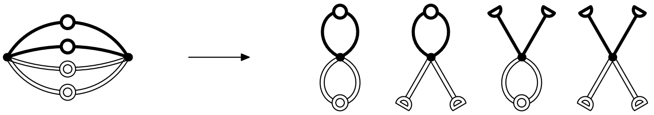


Fig. 6. An example of a dipole with four non-isomorphic half-quotients. This example can easily be generalized to exponentially many non-isomorphic quotients.

problems. Two atoms A and A' are isomorphic if there exists an isomorphism which maps ∂A to $\partial A'$. We obtain isomorphism classes for the set of all atoms $\mathcal{A}$ such that A and A' belong to the same class if and only if $A \cong A'$. To each color class, we assign one new color not yet used in the graph. When we replace the atoms of $\mathcal{A}$ by edges, we color the edges according to the colors assigned to the isomorphism classes. For an asymmetric atom, we choose an arbitrary orientation, but consistently for the entire isomorphism class.

Quotient Reduction and Expansion. Suppose that Γ_i is a semiregular action of G_i. Then it corresponds to the unique action Γ_{i+1} of G_{i+1}. If $H_i = G_i/\Gamma_i$ and $H_{i+1} = G_{i+1}/\Gamma_{i+1}$, then we can construct H_{i+1} from H_i as follows. Let A be an atom of G_i and let Q be its quotient in H_i. Then Q is replaced in H_{i+1} by a colored edge, loop or half-edge depending whether Q is an edge-, a loop-, or a half-quotient.

On the other hand, suppose that H_{i+1} is given. We study in how many possible ways can we *expand* it to H_i. Here H_i is not uniquely determined anymore. But we get the following characterization useful for the algorithm:

Proposition 3.3. *Every quotient H_i of G_i can be constructed from some quotient H_{i+1} of G_{i+1} by replacing each edge, loop and half-edge corresponding to an atom of G_i by an edge-, loop-, or half-quotient, respectively. Moreover, for different choices of H_{i+1} and of half-quotients we get different labeled graphs H_i.*

Note that the block structure is preserved by these expansions, only new pendant blocks are attached. Also, edge-quotients of block atoms, and loop- and half-edge quotients of proper atoms and dipoles correspond to block parts of H_0.

4 Algorithm for Planar Graphs

We establish Theorem 1.1. Let $k = |G|/|H|$. The algorithm proceeds as follows:

1. We construct the reduction series for $G = G_0, \dots, G_r$ ending with the unique primitive graph G_r.
2. We construct all semiregular subgroups Γ_r of $\mathrm{Aut}(G_r)$ of the order k. The number of subgroups in the list is polynomial for planar graphs.
3. For each Γ_r, we compute $H_r = G_r/\Gamma_r$. We say that a graph H_r is *expandable* if there exists a sequence of extensions repeatedly applying Proposition 3.3 which constructs H_0 isomorphic to H. We test the expandability of H_r using dynamic programming.

The fundamental difficulty is in the third step since there might be exponentially many possible expansions of H_r. So we approach the problem from the other side, and try to reduce the graph H to H_r, by replacing atoms in H with edges. Consider a pendant block of H. The issue here that there is that is no way to know whether this block is an edge-quotient of a block in G, or a loop-quotient or a half-quotient of a proper atom or dipole; see Figure 7. So without exploiting some additional information from H, there is no way to know what

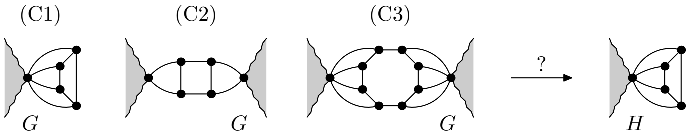


Fig. 7. For a pendant block of H, there are three possible preimages in G. It could be a block atom mapped by (C1), or a proper atom mapped by (C2), or another proper atom mapped by (C3).

is the preimage of this pendant block. Our approach is not to decide everything in one stage, instead we just remember a list of possibilities. We use dynamic programming to deal with these lists and compute further lists. The following diagram shows the overview of the algorithm:

$$\begin{tikzcd}
G_0 \arrow[r, "\text{red.}"] \arrow[d, "\mathit{\Gamma}_0"'] & G_1 \arrow[r, "\text{red.}"] \arrow[d, "\mathit{\Gamma}_1"'] & \cdots \arrow[r, "\text{red.}"] & G_r \arrow[d, "\mathit{\Gamma}_r"'] & & & \\
H_0 & H_1 \arrow[l, "\text{exp.}"'] & \cdots \arrow[l, "\text{exp.}"'] & H_r \arrow[l, "\text{exp.}"'] \arrow[r, bend left, "\text{red.}"] & H_{r+1} \arrow[l, bend left, "\text{exp.}"] \arrow[r, bend left, "\text{red.}"] & \cdots \arrow[l, bend left, "\text{exp.}"] \arrow[r, bend left, "\text{red.}"] & H_s \arrow[l, bend left, "\text{exp.}"] \arrow[d, hook, "?"] \\
\mathcal{R}_0 \arrow[r, "\text{red.}"] & \mathcal{R}_1 \arrow[r, "\text{red.}"] & \mathcal{R}_2 \arrow[r, "\text{red.}"] & \mathcal{R}_3 \arrow[r, "\text{red.}"] & \mathcal{R}_4 \arrow[r, "\text{red.}"] & \cdots \arrow[r, "\text{red.}"] & \mathcal{R}_t
\end{tikzcd} \qquad (1)$$

Atoms in Quotients. We choose one arbitrary block/articulation called the *core* in H_r. The core plays the role of the central block in the definition of parts and atoms for the quotients. We consider half-edges and loops as pendant edges. We first reduce H_r to a primitive quotient graph H_s, where $s \geq r$. Notice that all atoms in $H_r, \dots, H_{s-1}$ are necessarily block atoms.

Now, the graph H_s consists of the core together with some pendant edges, loops and half-edges. Then the core in H_0 has to correspond to some block or articulation of H, and we iterate over all possible positions of the core in H. In what follows, we have the core fixed in H as well.

Reducing in Quotients. Our goal is to apply a reduction series on H defining $\mathcal{H}_0, \dots, \mathcal{H}_t$. As already discussed above, we do not know which parts of G project to different parts of H. Therefore each $\mathcal{H}_i$ represents a set of graphs, and $\mathcal{H}_t$ represents a set of primitive graphs. We then determine expandability of H_r by testing whether $H_s \in \mathcal{H}_t$.

We represent each $\mathcal{H}_i$ by one graph $\mathcal{R}_i$ with so-called *pendant elements* attached to some vertices. Each pendant element corresponds to a block part in H. Further each pendant element has a polynomially large *list* of possible realizations by colored edges, loops and half-edges. Each list may contain at most one edge and at most one loop, but it may contain many half-edges. A pendant element contains an edge/loop/half-edge in its list if and only if it is possible to expand this edge/loop/half-edge to a graph isomorphic to the corresponding block part of H. Each graph of $\mathcal{H}_i$ is created from $\mathcal{R}_i$ by replacing the pendant elements by some edges, loops and half-edges from the respective lists.

Testing whether $H_s \in \mathcal{H}_t$ can be done for planar graphs, see the full version for details. Here, we code the colors of pendant edges, loops and half-edges by the color of the vertices of H_s. And we encode the lists of the pendant elements by lists of colors in $\mathcal{R}_t$. It remains to argue that using dynamic programming, we can compute $\mathcal{R}_{i+1}$ from $\mathcal{R}_i$.

One Step of the Reduction. To compute $\mathcal{R}_{i+1}$ from $\mathcal{R}_i$, we first find all atoms in $\mathcal{R}_i$. We replace all dipoles and proper atoms by edges of the corresponding colors. We replace block atoms by pendant elements for which we need to compute their lists. Let A be a block atom in $\mathcal{R}_i$. A *star atom* is a block atom consisting of an articulation with attached pendant edges, loops and pendant elements. For a non-star atom, we just iterate over all quotient and test using (P3) whether each quotient matches A. If A is a star-atom, then it corresponds to a dipole, or a star atom, we again iterate over all of them.

For simplicity, we just describe testing for a dipole D. We iterate over all of at most $2^{e(D)/2}$ possible quotients of D. Each such quotient is a vertex with several loops and half-edges attached, and each of these has to one-to-one correspond to a pendant element attached to A. We just test existence of a perfect matching in a bipartite graph: Here, one part are loops and half-edges attached in D, the other part are pendant elements, and edges are according to which loops and half-edges are in the list. We add D to the list if and only if a perfect matching exists.

Proof (Theorem 1.1, Sketch). The algorithm can be implemented in FPT time $\mathcal{O}^*(2^{e(H)/2})$, and it is correct since we compute all possible ways how H can be reached. So if G regularly covers H, there has to be some way in which the algorithm succeeds. □

5 Concluding Remarks

We just state open problems.

Problem 5.1. Is the problem RegularCover GI-complete?

As possible next direction of research, we suggest to attack classes of graphs close to planar graphs, for instance projective planar graphs or toroidal graphs. To do so, it seems that new techniques need to be built. Also the automorphism groups of projective planar graphs and toroidal graphs are not well understood. Lastly, we indeed ask whether the slow subroutine for dipole expansion of the algorithm can be solved in polynomial time; see the full version for details.

Problem 5.2. Can the complexity of the algorithm of Theorem 1.1 be improved to polynomial?

References

1. Angluin, D.: Local and global properties in networks of processors. In: ACM Symposium on Theory of Computing, pp. 82–93. ACM (1980)
2. Babai, L.: Automorphism groups of planar graphs II. In: Infinite and finite sets, pp. 29–84. North-Holland, Bolyai (1975); Proc. Conf. Keszthely, Hungary (1973)
3. Babai, L.: Automorphism groups, isomorphism, reconstruction. In: Handbook of combinatorics, vol. 2, pp. 1447–1540. MIT Press (1996)
4. Bílka, O., Jirásek, J., Klavík, P., Tancer, M., Volec, J.: On the complexity of planar covering of small graphs. In: Kolman, P., Kratochvíl, J. (eds.) WG 2011. LNCS, vol. 6986, pp. 83–94. Springer, Heidelberg (2011)
5. Bodlaender, H.L.: The classification of coverings of processor networks. Journal of Parallel and Distributed Computing 6(1), 166–182 (1989)
6. Cayley, A.: The theory of groups: Graphical representation. Amer. J. Math. 1, 174–176 (1878)
7. Erdös, P., Fajtlowicz, S., Hoffman, A.J.: Maximum degree in graphs of diameter 2. Networks 10(1), 87–90 (1980)
8. Evdokimov, S., Ponomarenko, I.: Circulant graphs: recognizing and isomorphism testing in polynomial time. St. Petersburg Mathematical Journal 15(6), 813–835 (2004)
9. Fellows, M.R., Stillweil, J.C.: On the complexity and combinatorics of covering finite complexes. Australasian Journal of Combinatorics 4, 103–112 (1991)
10. Fiala, J.: Note on the computational complexity of covering regular graphs. In: 9th Annual Conference of Doctoral Students, WDS 2000, pp. 89–90. Matfyzpress (2000)
11. Hoffman, A.J., Singleton, R.R.: On moore graphs with diameters 2 and 3. IBM Journal of Research and Development 4(5), 497–504 (1960)
12. Hopcroft, J.E., Tarjan, R.E.: Isomorphism of planar graphs. In: Complexity of computer computations, pp. 131–152. Springer (1972)
13. Hopcroft, J.E., Tarjan, R.E.: Dividing a graph into triconnected components. SIAM Journal on Computing 2(3), 135–158 (1973)
14. Kratochvíl, J., Proskurowski, A., Telle, J.A.: Covering regular graphs. J. Comb. Theory Ser. B 71(1), 1–16 (1997)
15. Lubiw, A.: Some NP-complete problems similar to graph isomorphism. SIAM Journal on Computing 10(1), 11–21 (1981)
16. Malnič, A., Nedela, R., Škoviera, M.: Lifting graph automorphisms by voltage assignments. European Journal of Combinatorics 21(7), 927–947 (2000)
17. McKay, B.D., Miller, M., Širáň, J.: A note on large graphs of diameter two and given maximum degree. J. Combin. Theory Ser. B 74(1), 110–118 (1998)
18. Miller, M., Širáň, J.: Moore graphs and beyond: A survey of the degree/diameter problem. Electronic Journal of Combinatorics 61, 1–63 (2005)
19. Negami, S.: The spherical genus and virtually planar graphs. Discrete Mathematics 70(2), 159–168 (1988)
20. Zhou, S.: A class of arc-transitive Cayley graphs as models for interconnection networks. SIAM Journal on Discrete Mathematics 23(2), 694–714 (2009)

Public vs Private Coin in Bounded-Round Information[*]

Mark Braverman[**] and Ankit Garg

Princeton University

Abstract. We precisely characterize the role of private randomness in the ability of Alice to send a message to Bob while minimizing the amount of information revealed to him. We give an example of a (randomized) message which can be transmitted while revealing only I bits of information using private randomness, but requires Alice to reveal $I+\log I-O(1)$ bits of information if only public coins are allowed. This gives the first example of an $\omega(1)$ additive separation between these two models. Our example also shows that the one-round compression construction of Harsha et al. [HJMR07] cannot be improved.

Moreover, we show that our example is tight up to an additive $O(1)$ factor: We show that if using private randomness a message can be transmitted while revealing I bits of information, the transmission can be simulated without private coins using $I + \log I + O(1)$ bits of information. This improves over an earlier result by Brody et al. [BBK$^+$12].

1 Introduction

In this paper we investigate the role of private randomness in the ability of two parties to communicate while revealing as little information as possible to each other – i.e. to communicate at low information cost. More specifically, Alice and Bob are given possibly correlated inputs X and Y and need to perform a task T by means of a communication protocol π. Alice and Bob share a public random string R; in addition they have access to private random strings R_A and R_B, respectively. The *internal information cost* of π with respect to a distribution $(X, Y) \sim \mu$ is the quantity

$$\mathsf{IC}_\mu(\pi) := I(\Pi; Y|XRR_A) + I(\Pi; X|YRR_B),$$

where $\Pi = \Pi(X, Y, R, R_A, R_B)$ is the random variable representing the transcript of the protocol.

It is not hard to see that if the goal is to solve a task T while minimizing the information cost of the protocol, we can always avoid using the public randomness string R: to simulate public randomness, before the beginning of the

[*] Full version of the paper can be found at `http://eccc.hpi-web.de/report/2013/130/`

[**] Research supported in part by an Alfred P. Sloan Fellowship, an NSF CAREER award CCF-1149888, and a Turing Centenary Fellowship.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 502–513, 2014.

protocol's execution, Alice can send a portion of R_A, which will be used as R for the remainder of the protocol. This modification increases the communication cost of the protocol, but it is not hard to see that it does not change its information cost. Therefore, in the context of information complexity, private randomness is at least as good as public randomness. Is the converse true? In other words, can any protocol π that uses private randomness be simulated by a protocol π' which uses only public randomness so that $\mathsf{IC}_\mu(\pi') \leq \mathsf{IC}_\mu(\pi)$? The naïve "solution" to this problem would be to simulate π by using the public randomness to simulate private randomness. The following simple example shows why this approach fails. Consider the protocol π in which $X \in \{0,1\}^n$. Alice samples a uniformly random string $R_A \in_U \{0,1\}^n$, and sends the bitwise XOR $M := X \oplus R_A$ to Bob. This protocol conveys 0 information to Bob about X. However, if the public randomness R were to be used to produce R_A, then Bob would also know R_A, and thus the message M reveals $X = M \oplus R_A$ to Bob – drastically increasing the information cost of the protocol. This, of course, does not mean that a more sophisticated simulation scheme cannot work.

It is instructive to compare this question to the public-vs-private randomness question in randomized *communication complexity.* In the context of communication complexity the situation is somewhat reversed: it is obvious that public randomness can be used to simulate private randomness: the parties can always designate part of their public randomness as "private randomness". This will not affect the communication cost of the protocol (although, as seen above, it may affect its information cost). In the reverse direction, Newman [New91] showed that $R_{\epsilon+\delta}(f) \leq R_\epsilon^{\text{pub}}(f) + O(\log(\frac{n}{\delta}))$. Thus, up to an additive $\log n$, private randomness replaces public randomness in communication complexity. Does a "reverse Newman theorem" hold for information complexity? Can private randomness be replaced with public randomness at a small cost?

This question has been considered by Brody et al. in [BBK$^+$12], which showed a version of the private-by-public simulation for one-round protocols. In the one-round setting, Alice wishes to send Bob her message – a random variable $M = M(X, R_A)$. Obviously, the information cost of this task is just $I(M; X|Y)$. If Bob receives no input, then it is just $I(M; X)$. In this paper we prove tight bounds on the one round private-by-public simulation. Specifically, we give a family of examples in which the cost of simulating a message M of information cost I without the use of private randomness goes up to $I + \log(I) - O(1)$, and the lower bound is in fact tight in some cases. Previously, [BBK$^+$12] showed an upper bound on private-by-public simulation of a message M of information cost I to information cost of at most $I + O(\log n)$, where $n = \max(\log|\mathcal{X}|, \log|\mathcal{Y}|)$ – the log of the sizes of the domains of X and Y. We also improve their bound to $I + \log(I) + O(1)$ and provide a simpler proof. Note that it is always the case that $I \leq H(X) \leq \log|\mathcal{X}| \leq n$, and therefore $\log I \leq \log n$.

It is interesting to consider the connection between the problem of simulating a protocol without private randomness, and the problem of compressing communication protocols. The general protocol compression problem [BBCR10,Bra12] is the problem of simulating a protocol π with communication cost C and information cost I with a protocol π' of communication cost C' that is as close to I as possible. The problem of compressing interactive communication is essentially equivalent to the direct sum problem for randomized communication complexity [BR11]. The best known general compression results gives $C' = \tilde{O}(\sqrt{I \cdot C})$, and it is wide open whether $C' = O(I \cdot (\log C)^{O(1)})$ is possible. It has been shown in [BBK+12] (and independently in [Pan12]) that if a protocol π does not use private randomness, then it can be compressed to $O(I \cdot (\log C)^{O(1)})$. Thus a way to replace private randomness with public randomness for unbounded-round protocols would imply a substantial improvement in the state-of-the-art on protocol compression. Moreover, separating private information complexity from public information complexity is one possible approach of separating (private) information complexity from communication complexity (if a separation exists). Our small (albeit tight) separation is the first $\omega(1)$ separation between these two models, and is a step in that direction.

Another interesting connection between removing private randomness and compression is in the context of one-message protocol s. In the setting where Bob has no input Y, the information cost of sending a message M is just $I := I(M; X)$. Harsha et al. [HJMR07] showed how to simulate such a transmission using $I + O(\log I)$ bits of (expected) *communication* (with access to public randomness). Their work left open the interesting question of whether the additive $O(\log I)$ is necessary. As noted above, a communication protocol with communication C can always be simulated by a protocol with same communication and only public randomness. As information cost is bounded from above by communication cost, a compression scheme is in particular a private-by-public scheme. Thus our lower bound gives an example showing that the $O(\log I)$ additive overhead in [HJMR07] is necessary.

Results and Techniques

Our main result gives an upper and lower bound on simulating private randomness by public randomness for one-message protocols.

Theorem 1. *Let X, Y be inputs to Alice and Bob respectively distributed according to a distribution μ. Alice and Bob have access to public randomness R', and Alice has access to private randomness R_A. Let π be a protocol where Alice sends a message $M = M(X, R', R_A)$ to Bob, so that the information cost of π is $I := I(X; M|YR')$. Then*

1. *for each I, there is an example with no Y (i.e. Bob has no "private" knowledge), and no R', such that if $I := I(X; M)$, then any public-coin protocol π' simulating the transmission of M must have information cost of at least $I + \log I - O(1)$.*
2. *π can be simulated by a one-message public-coin protocol π' such that $\mathsf{IC}_\mu(\pi') \le I + \log I + O(1)$.*

Thus, up to an additive constant, our bounds are tight. Note that while the upper bound holds under the most general conditions, for the lower bound it is sufficient to consider protocols without Y (this is the type of protocols considered, for example, in [HJMR07]).

Both the upper and lower bound require some careful analysis. For the upper bound, a natural variant of the one-round compression scheme of Braverman and Rao [BR11] is used. The main challenge is in analyzing the information cost of the resulting public randomness protocol: we need to prove that Bob does not learn too much about X from Alice's message. Suppose that given X and the public randomness R of the simulating protocol, Alice's message in the simulating protocol is $S = S(X, R)$. Observe that in this case

$$I(S; X|YR) = H(S|YR) - H(S|XYR) = H(S|YR).$$

To establish an upper bound on $H(S|YR)$, we show how, someone knowing X, Y and R, can describe S to Bob using a message M' (i.e. $H(S|M'YR) = 0$) such that

$$H(M') \leq I + \log(I) + O(1)$$

Noting that this expression is an upper bound for $H(S|YR)$, completes the proof.

To prove the lower bound, we give a family of specific examples whose information cost necessarily increases by $\log I - O(1)$ when private randomness is replaced with public randomness. Details of the construction are given in Section 3, here we only give the high level idea for why the information cost increases in lieu of private randomness. Consider the following example: Alice knows a secret random string $PASS$ of 128 bits (which we can think of as her password). She wants to send Bob a message M such that $M = PASS$ with probability 1/2 and $M = RANDOM$ with probability 1/2 – that is, half of the time she sends her password and half the time she sends a random 128-bit string. The message M reveals approximately 63 bits of information about $PASS$. To see this, note that given M the posterior distribution of $PASS$ puts mass 1/2 on M and mass 1/2 on the remaining $2^{128} - 1$ strings. The entropy of this distribution is $\approx \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 129 = 65$, down from the prior entropy of 128. Thus $I(M; PASS) \approx 128 - 65 = 63$ bits. One might have expected this number to be 64 bits. Indeed, if Alice had told Bob which of the two cases has occurred, M would reveal $\frac{1}{2} \cdot 128 + \frac{1}{2} \cdot 0 = 64$ bits of information. However, not knowing whether Alice's message is the password or a random string "saves" one bit in information cost. Now suppose Alice was not allowed to use private randomness. Then, intuitively, the public random string R should reveal to Bob whether $M = PASS$ or $M = RANDOM$. Therefore, the information cost of a public-randomness protocol increases to 64 bits. Generalizing from this example, we construct a situation where Alice sends a binary message M of length n and information cost $I \approx n/2 - \log n$, so that any public randomness simulation of M requires information cost of $\geq n/2 - O(1) = I + \log I - O(1)$ – demonstrating the desired gap.

Let us have a look at another example. Suppose that Alice gets a bit $X \sim B_{\frac{1}{2}}$ and she wants to transmit this bit to Bob with error $\frac{1}{2} - \epsilon$. Consider a private-coin protocol in which Alice samples a $B_{\frac{1}{2}+\epsilon}$ bit R. She sends X if $R = 1$ and a $\neg X$ if $R = 0$. Clearly the protocol performs the task of transmitting the bit with error $\frac{1}{2} - \epsilon$. Let Π denote the random variable for Alice's message. The information cost of this protocol is

$$\begin{aligned} I(\Pi; X) &= \frac{1}{2}D(\Pi_0||\Pi) + \frac{1}{2}D(\Pi_1||\Pi) = \frac{1}{2}D(1/2 - \epsilon||1/2) + \frac{1}{2}D(1/2 + \epsilon||1/2) \\ &= \frac{2}{\ln 2}\epsilon^2 \pm o(\epsilon^2) \end{aligned}$$

However if we don't allow private coins, then the information complexity of this task is $\geq 2\epsilon$. To see this consider a public-coin protocol that transmits X with error probability $\leq \frac{1}{2} - \epsilon$. It is basically a function $f : \{0,1\} \times \mathcal{R} \rightarrow \{0,1\}$ (in case Alice sends a longer message and then Bob applies a deterministic function to that, f could be the composition of those two functions) such that $\mathbb{E}_{r \sim \mathcal{R}}[f(0,r)] = \frac{1}{2} - \epsilon$ and $\mathbb{E}_{r \sim \mathcal{R}}[f(1,r)] = \frac{1}{2} + \epsilon$. Then $Pr_{r \sim \mathcal{R}}[f(1,r) = 1, f(0,r) = 0] \geq 2\epsilon$. Hence

$$I(f(X,R); X|R) = H(f(X,R)|R) = \mathbb{E}_{r \sim \mathcal{R}} H(f(X,r)) \geq 2\epsilon$$

since if $f(1,r) = 1, f(0,r) = 0$, then $H(f(X,r)) = 1$. This example, in some sense, highlights the information-cost advantage one gains from having access to private randomness. It will be interesting to see if this advantage can be amplified over multiple rounds to get a separation between unbounded round private-coin information complexity and public-coin information complexity, and thus in particular between information complexity and communication complexity.

Open Problems

Our lower bound example is really about the simulation of a protocol and not about solving a boolean function. So it will be nice to get a 1-round gap for a boolean function. Also it would be nice to get a bigger separation between r-round public-coin information complexity and private-coin information complexity, where r is a constant. Note that using the 1-round example, we can also construct a 2-round example by requiring both Alice and Bob to perform the 1-round task.

1. Does there exist a boolean function f for which 0-error private-coin information complexity is I but 0-error public-coin information complexity is $\geq I + \log(I) - O(1)$?
2. Does there exist a (family of) 3-round private-coin protocol(s) π such that information cost of π is I but any 3-round public-coin protocol simulating π has information cost $\geq I + 3\log(I) - O(1)$?

2 Preliminaries

2.1 Communication Complexity

In the two-party communication model, the parties, traditionally called Alice and Bob, are trying to collaboratively compute a known Boolean function $f : \mathcal{X} \times \mathcal{Y}$.

Each party is computationally unbounded; however, Alice is only given input $x \in \mathcal{X}$ and Bob is only given $y \in \mathcal{Y}$. In order to compute $f(x, y)$, Alice and Bob communicate in accordance with an agreed-upon communication protocol π. Protocol π specifies as a function of transmitted bits only whether the communication is over and, if not, who sends the next bit. Moreover, π specifies as a function of the transmitted bits and x the value of the next bit to be sent by Alice. Similarly for Bob. The communication is over when *both parties* know the value of $f(x, y)$. The cost of the protocol π is the number of bits exchanged on the worst input. *The transcript* of a protocol is a concatenation of all the bits exchanged during the execution of the protocol.

There are several ways in which the deterministic communication model can be extended to include randomness. In the *public-coin model*, Alice and Bob have access to a shared random string r chosen according to some probability distribution. The only difference in the definition of a protocol is that now the protocol π specifies the next bit to be sent by Alice as a function of x, the already transmitted bits, and a random string r. Similarly for Bob. This process can also be viewed as the two players having an agreed-upon distribution on deterministic protocols. Then the players jointly sample a protocol from this distribution. In the *private-coin model*, Alice has access to a random string r_A hidden from Bob, and Bob has access to a random string r_B hidden from Alice.

Definition 1 (Randomized Communication Complexity). *For a function $f : \mathcal{X} \times \mathcal{Y} \to Z$ and a parameter $\epsilon > 0$, $R_\epsilon(f)$ denotes the communication cost of the best randomized private-coin protocol for computing f with error at most ϵ on* every *input. Similarly $R_\epsilon^{pub}(f)$ denotes the cost of the best randomized public-coin protocol for computing f with error at most ϵ on* every *input.*

Definition 2. *We will say that a (randomized) protocol ϕ* simulates *a protocol π if there is a deterministic function g such that $g(\Phi(x, y, R^\phi, R_A^\phi, R_B^\phi))$ is equal in distribution to $\Pi(x, y, R^\pi, R_A^\pi, R_B^\pi)$, $\forall x, y$. Here $R^\phi, R_A^\phi, R_B^\phi$ are the public and private randomness of protocol ϕ and Φ is the random variable for the transcript. Similarly for π.*

For the pre-1997 results on communication complexity, see the excellent book by Kushilevitz and Nisan [KN97].

2.2 Information Theory

In this section we briefly provide the essential information-theoretic concepts required to understand the rest of the paper. For a thorough introduction to the area of information theory, the reader should consult the classical book by Cover and Thomas [CT91]. Unless stated otherwise, all log's in this paper are base-2.

Definition 3. *Let μ be a probability distribution on sample space Ω.* Shannon entropy *(or just* entropy*) of μ, denoted by $H(\mu)$, is defined as $H(\mu) := \sum_{\omega \in \Omega} \mu(\omega) \log \frac{1}{\mu(\omega)}$.*

For a random variable A we shall write $H(A)$ to denote the entropy of the induced distribution on the range of A. The same also holds for other information-theoretic quantities appearing later in this section.

Definition 4. Conditional entropy *of a random variable A conditioned on B is defined as*

$$H(A|B) = \mathbb{E}_b(H(A|B=b)).$$

Fact 2. $H(AB) = H(A) + H(B|A)$.

Definition 5. *The* mutual information *between two random variable A and B, denoted by $I(A;B)$ is defined as*

$$I(A;B) := H(A) - H(A|B) = H(B) - H(B|A).$$

The conditional mutual information *between A and B given C, denoted by $I(A;B|C)$, is defined as*

$$I(A;B|C) := H(A|C) - H(A|BC) = H(B|C) - H(B|AC).$$

Fact 3 (Chain Rule). *Let A_1, A_2, B, C be random variables. Then*

$$I(A_1A_2;B|C) = I(A_1;B|C) + I(A_2;B|A_1C).$$

2.3 Information Complexity

A much more detailed discussion of information complexity and its applications can be found in [CSWY01,BYJKS04,BBCR10,BGPW13] and references therein.

Definition 6. *The* internal information cost *of a protocol π with respect to a distribution μ on inputs from $\mathcal{X} \times \mathcal{Y}$ is defined as*

$$\mathrm{IC}_\mu(\pi) := I(\Pi;X|YRR_B) + I(\Pi;Y|XRR_A).$$

where $\Pi = \Pi(X,Y,R,R_A,R_B)$ is the random variable denoting the transcript of the protocol, R is the public randomness and R_A and R_B are the private random strings of Alice and Bob, respectively. In the previous works, it is defined in a different way (without the conditioning on private random strings), but both the definitions are in fact equivalent.

The *information complexity* of f with respect to μ is

$$\mathrm{IC}_\mu(f,\epsilon) := \inf_\pi \mathrm{IC}_\mu(\pi),$$

where the infimum ranges over all (randomized) protocols π solving f with error at most ϵ when inputs are sampled according to μ.

3 Lower Bound

We describe our lower bound in this section. Alice is given a uniformly random string $x \in_R \{0,1\}^n$. Let $M(x,i)$ denote a message distributed according to $x_1, \ldots, x_{i-1}, \bar{x_i}, b_{i+1}, \ldots, b_n$, where b_j's are random bits $\sim B_{1/2}$ and $\bar{x_i}$ denotes the flip of bit x_i.

Given x, Alice's task, T, is to transmit a message distributed according to $M(x,I)$, where $I \in_R \{1,2,\ldots,n\}$. Note that Bob has no input in this task.

First let us bound the private-coin information complexity of this task. Given x, Alice can privately sample I and send $M \sim M(x,I)$. Then the information cost of this protocol is $I(M;X) = H(M) - H(M|X)$. It is clear that $H(M) = n$.

$$H(M|X) = \mathbb{E}_x[H(M|X=x)]$$

Denote $M|X = x$ by $M|_x$. For strings $x, y \in \{0,1\}^n$ with $x \neq y$, let $j(x,y)$ denote the first index of disagreement between x and y i.e. index j s.t. $x_j \neq y_j$. Then

$$Pr[M|_x = y] = \frac{1}{n} \cdot \frac{1}{2^{n-j(x,y)}}$$

if $x \neq y$ and 0 if $x = y$.

$$\begin{aligned}
H(M|_x) &= \sum_y Pr[M|_x = y] \log\left(\frac{1}{Pr[M|_x = y]}\right) \\
&= \sum_{j=1}^{n} 2^{n-j} \cdot \frac{1}{n} \cdot \frac{1}{2^{n-j}} \log(n \cdot 2^{n-j}) + 0 \\
&= \log(n) + \frac{1}{n} \sum_{j=1}^{n} (n-j) \\
&= n/2 + \log(n) - 1/2
\end{aligned}$$

The second equality follows from the fact that there are 2^{n-j} strings y with $j(x,y) = j$, when $j \in \{1, \ldots, n\}$. This gives

$$I(M;X) = n/2 - \log(n) + 1/2$$

The following lemma lower bounds the information complexity of a public round protocol for the task T. Note that the strategy of sampling I publicly would have an information cost $\approx n/2$.

Lemma 1. *Let Π be a one round public-coin protocol (using public randomness R) such that there is a deterministic function g such that $g(\Pi_x, R)$ is distributed according to $M(x,I)$. Then $I(\Pi; X|R) \geq n/2 - O(1)$.*

Proof. Since Π is a deterministic function of X and R,

$$I(\Pi; X|R) = H(\Pi|R) - H(\Pi|X,R) = H(\Pi|R)$$

Let J be a random variable that denotes the first index of disagreement between $g(\Pi, R)$ and X (Note that J is well defined because of the distribution of M). Fix a value of $R = r$. Let $p_j = Pr[J = j|R = r]$. Note that the probability is just over random X. Let μ denote the distribution of $\Pi|R = r$ and let μ_j be the distribution of $\Pi|R = r, J = j$. Note that p_j, μ and μ_j depend on r but we will slightly abuse notation and suppress the dependence on r since r will be fixed throughout. It holds that

$$\mu = \sum_{j=1}^{n} p_j \cdot \mu_j$$

Let us analyze the distribution μ_j. Let $S_r(j)$ be the set of x's which lead to $J = j$ i.e.

$$S_r(j) = \{x \in \{0,1\}^n : j(x, g(\Pi(x,r), r)) = j\}$$

Note that $|S_r(j)| = p_j \cdot 2^n$. Fixing $\Pi = t$ and $R = r$ fixes $g(\Pi, R) = g(t, r)$. Then

$$Pr[\Pi = t|R = r, J = j] \leq \frac{|\{x \in S_r(j) : j(x, g(t,r) = j)\}|}{|S_r(j)|} \leq \frac{2^{n-j}}{p_j \cdot 2^n} = \frac{1}{p_j \cdot 2^j}$$

The first inequality is because if $R = r, J = j$ are fixed, the event $\Pi = t$ implies that $j(x, g(t,r)) = j$. The second inequality follows from the fact that there are 2^{n-j} x's with $j(x, g(t,r)) = j$.

Claim. $H(\mu) \geq \sum_{j=1}^{n} j \cdot p_j - O(1)$.

Given the claim, we can bound $H(\Pi|R)$ as follows :

$$\begin{aligned} H(\Pi|R) &= \mathbb{E}_{r\sim R}[H(\Pi|R = r)] \\ &\geq \mathbb{E}_{r\sim R} \sum_{j=1}^{n} j \cdot p_j - O(1) \\ &= \sum_{j=1}^{n} j \cdot \frac{1}{n} - O(1) \\ &= n/2 - O(1) \end{aligned}$$

The inequality follows from the claim. The second equality follows from the fact that $\mathbb{E}_{r\sim R} Pr[J = j|R = r] = Pr[J = j] = \frac{1}{n}$.

Proof. (Of Claim 3) Increasing a larger probability and decreasing a smaller probability by the same amount always lowers the entropy of a distribution

$$\left(p \log\left(\frac{1}{p}\right)\right)' - \left(q \log\left(\frac{1}{q}\right)\right)' = \log\left(\frac{q}{p}\right) < 0 \text{ if } q < p$$

We are given a μ_j where the mass of every entry $\mu_j(z)$ does not exceed $2^{-j}/p_j$. Therefore, we can replace μ_j with a uniform distribution on a set $\mathcal{L}_j$ of L_j entries, where $L_j = \max(1, \lfloor p_j \cdot 2^j \rfloor)$ (given any z_1, z_2 with $0 < \mu_j(z_1), \mu_j(z_2) < 1/L_j$

we can make sure that one of them becomes 0 or that one of them becomes $1/L_j$ without increasing the entropy). Note that it is always the case that $L_j > p_j \cdot 2^{j-1}$.

Therefore, we can assume wlog that each μ_j is uniform on a set $\mathcal{L}_j$ of size L_j. Consider the process of selecting an index K according to the distribution p_j, and then $Z \sim \mu_K$. Our goal is to show that $H(Z) \geq \sum_{j=1}^{n} j \cdot p_j - O(1)$. We have

$$H(KZ) = H(K) + H(Z|K) = \sum_{j=1}^{n} p_j \log(L_j/p_j) > \sum_{j=1}^{n} p_j \log(p_j \cdot 2^{j-1}/p_j)$$
$$= \sum_{j=1}^{n} j \cdot p_j - 1,$$

and $H(Z) = H(KZ) - H(K|Z)$. Therefore, it suffices to show that $H(K|Z) = O(1)$.

We define a subset S of j's for which p_j is "small":

$$S := \{j : p_j < 2^{-j}\}.$$

Note that for $j \notin S$ we have $p_j \cdot 2^j \geq 1$, and therefore $L_j = \lfloor p_j \cdot 2^j \rfloor$, and $p_j \cdot 2^{j-1} < L_j \leq p_j \cdot 2^j$. Denote by χ_S the indicator random variable for the event $K \in S$. We have

$$\begin{aligned} H(K|Z) &\leq H(K, \chi_S|Z) = H(\chi_S|Z) + H(K|\chi_S Z) \\ &\leq 1 + \Pr[K \in S] H(K|Z, K \in S) + \Pr[K \notin S] H(K|Z, K \notin S). \end{aligned}$$

The second inequality is because χ_S is a boolean random variable. We bound the two terms separately. Assuming $S \neq \emptyset$, denote $p_S := \sum_{j \in S} p_j$.

$$\Pr[K \in S] H(K|Z, K \in S) \leq \Pr[K \in S] H(K|K \in S) = p_S \cdot \sum_{j \in S} \frac{p_j}{p_S} \log \frac{p_S}{p_j}$$
$$\leq \sum_{j \in S} p_j \log \frac{1}{p_j} < 1 + \sum_{j \geq 2, j \in S} p_j \log \frac{1}{p_j} \leq 1 + \sum_{j=2}^{n} 2^{-j} \log \frac{1}{2^{-j}} = O(1).$$

The last inequality is because the function $x \log 1/x$ is monotone increasing on the interval $(0, 1/e)$, and we have $0 < p_j < 2^{-j} < 1/e$ for $j \in S, j \geq 2$.

Finally, we need to show $\Pr[K \notin S] H(K|Z, K \notin S) = O(1)$. We will in fact show that $H(K|Z, K \notin S) = O(1)$. We have

$$H(K|Z, K \notin S) = \mathbb{E}_{z \sim Z|_{K \notin S}} H(K|Z = z, K \notin S). \tag{1}$$

Fix any value of z such that $\Pr[K \notin S|Z = z] > 0$. We can precisely describe the distribution q of $K|Z = z, K \notin S$. Denote $T_z := \{j : j \notin S, z \in \mathcal{L}_j\}$. Order the elements of T_z in increasing order, and index them: $T_z = \{j_1 < j_2 < \ldots < j_k\}$. Then the distribution q puts weight $q_r := \frac{p_{j_r}/L_{j_r}}{q}$ on j_r, where

$q := \sum_{r=1}^{k} p_{j_r}/L_{j_r}$. We have for each r:

$$q_r \leq \frac{p_{j_r}/L_{j_r}}{p_{j_1}/L_{j_1}} < \frac{p_{j_r}/(p_{j_r} \cdot 2^{j_r - 1})}{p_{j_1}/(p_{j_1} \cdot 2^{j_1})} = 2^{j_1 - j_r + 1} \leq 2^{2-r}.$$

The second inequality follows from $L_{j_r} > p_{j_r} \cdot 2^{j_r - 1}$ and $L_{j_1} \leq p_{j_1} \cdot 2^{j_1}$(since $j_1 \notin S$) . $q_r \leq 2^{2-r}$ implies that $H(q) = O(1)$. Therefore we have $H(K|Z = z, K \notin S) = O(1)$ for each z, and by (1) this implies $H(K|Z, K \notin S) = O(1)$, and completes the proof.

4 Upper Bound

Theorem 4. *Let X, Y be inputs to Alice and Bob respectively distributed according to a distribution μ. Alice and Bob have access to public randomness R', and Alice has access to private randomness R_A. Let π be a protocol where Alice sends a message $M = M(X, R', R_A)$ to Bob, so that the information cost of π is $I := I(X; M|YR')$. Then π can be simulated by a one-message public-coin protocol π' such that $\mathsf{IC}_\mu(\pi') \leq I + \log I + O(1)$.*

Proof. We can assume wlog that R' is a part of M, since $I(X; M|YR') = I(X; MR'|Y)$. Let $\mathcal{U}$ be the message space of the message M. Consider the protocol π' defined in Figure 1.

1. Using public randomness, Alice and Bob get samples $\{(u_i, p_i)\}_{i \geq 1}$, where (u_i, p_i) uniformly sampled from $\mathcal{U} \times [0, 1]$.
2. Let P denote the distribution $M_x = M|_{X=x}$ and Q denote the distribution $M_y = M|_{Y=y}$. Alice sends Bob the index of the first sample, s, such that $p_s < P(u_s)$. Bob decodes this message as being u_s

Protocol 1: Protocol π'

It turns out this protocol has information cost $\leq I + \log I + O(1)$. We leave the proof to the full version of the paper.

We also mention a few easy corollaries to multi-round private-by-public simulation and one round compression in the full version.

Acknowledgments. We would like to thank Omri Weinstein for the helpful comments on earlier versions of this paper. We would like to thank Anup Rao and Makrand Sinha for pointing out a minor bug in the previous version of this paper. We also thank anonymous referees for useful comments for improving the exposition of this paper.

References

BBCR10. Barak, B., Braverman, M., Chen, X., Rao, A.: How to compress interactive communication. In: Proceedings of the 42nd ACM symposium on Theory of computing, STOC 2010, pp. 67–76. ACM, New York (2010)

BBK$^+$12. Brody, J., Buhrman, H., Koucký, M., Loff, B., Speelman, F., Vereshchagin, N.: Towards a reverse newman's theorem in interactive information complexity. In: Electronic Colloquium on Computational Complexity (ECCC), vol. 19, p. 179 (2012)

BGPW13. Braverman, M., Garg, A., Pankratov, D., Weinstein, O.: From information to exact communication. In: Proceedings of the 45th Annual ACM Symposium on Theory of Computing, pp. 151–160. ACM, Palo Alto (2013)

BR11. Braverman, M., Rao, A.: Information equals amortized communication. In: FOCS, pp. 748–757 (2011)

Bra12. Braverman, M.: Interactive information complexity. In: Proceedings of the 44th Symposium on Theory of Computing, STOC 2012, pp. 505–524. ACM, New York (2012)

BRWY13. Braverman, M., Rao, A., Weinstein, O., Yehudayoff Direct, A.: product via round-preserving compression. ECCC 20(35) (2013)

BYJKS04. Bar-Yossef, Z., Jayram, T.S., Kumar, R., Sivakumar, D.: An information statistics approach to data stream and communication complexity. Journal of Computer and System Sciences 68(4), 702–732 (2004)

CSWY01. Chakrabarti, A., Shi, Y., Wirth, A., Yao, A.: Informational complexity and the direct sum problem for simultaneous message complexity. In: Werner, B. (ed.) Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science, Los Alamitos, CA, October 14-17, pp. 270–278. IEEE Computer Society Press (2001)

CT91. Cover, T.M., Thomas, J.A.: Elements of information theory. Wiley-Interscience, New York (1991)

HJMR07. Harsha, P., Jain, R., McAllester, D.A., Radhakrishnan, J.: The communication complexity of correlation. In: IEEE Conference on Computational Complexity, pp. 10–23. IEEE Computer Society (2007)

KN97. Kushilevitz, E., Nisan, N.: Communication complexity. Cambridge University Press, Cambridge (1997)

New91. Newman, I.: Private vs. common random bits in communication complexity. Information Processing Letters 39(2), 67–71 (1991)

Pan12. Pankratov, D.: Direct sum questions in classical communication complexity. PhD thesis, Masters thesis, University of Chicago (2012)

En Route to the Log-Rank Conjecture: New Reductions and Equivalent Formulations

Dmitry Gavinsky[1,⋆] and Shachar Lovett[2,⋆⋆]

[1] Institute of Mathematics, Academy of Sciences, Praha, Czech Republic
[2] Department of Computer Science and Engineering,
University of California, San Diego, USA

Abstract. We prove that several measures in communication complexity are equivalent, up to polynomial factors in the logarithm of the rank of the associated matrix: deterministic communication complexity, randomized communication complexity, information cost and zero-communication cost. This shows that in order to prove the log-rank conjecture, it suffices to show that low-rank matrices have efficient protocols in any of the aforementioned measures.

Furthermore, we show that the notion of zero-communication complexity is equivalent to an extension of the common discrepancy bound. Linial et al. [Combinatorica, 2007] showed that the discrepancy of a sign matrix is lower-bounded by an inverse polynomial in the logarithm of the associated matrix. We show that if these results can be generalized to the extended discrepancy, this will imply the log-rank conjecture.

1 Introduction

The *log-rank conjecture* proposed by Lovász and Saks [8] suggested that for any function $f : X \times Y \to \{0,1\}$ its deterministic communication complexity $\mathrm{CC}^{\mathrm{det}}(f)$ is polynomially related to the logarithm of the rank (over the field $\mathbb{R}$) of the associated matrix $M_f \overset{\mathrm{def}}{=} (f(x,y))_{x,y}$. Validity of this conjecture is one of the most fundamental open problems in communication complexity. Very little progress has been made towards resolving it. The best known bounds are

$$\Omega\Big(\log^{\log_3 6} \mathrm{rank}(M_f)\Big) \le \mathrm{CC}^{\mathrm{det}}(f) \le \log(4/3)\,\mathrm{rank}(M_f), \qquad (1)$$

where the lower bound is due to Kushilevitz (unpublished, cf. [10]) and the upper bound is due to Kotlov [4]. Recently a conditional improvement has been made by Ben-Sasson et al. [1], who showed that the polynomial Freiman-Ruzsa conjecture from additive combinatorics implied $\mathrm{CC}^{\mathrm{det}}(f) \le O(\mathrm{rank}(M_f)/\log \mathrm{rank}(M_f))$.

In this work we study the relation of the log-rank conjecture to several communication complexity measures. Besides those ones that directly correspond to

[⋆] Partially funded by the grant P202/12/G061 of GA ČR and by RVO: 67985840.

[⋆⋆] Supported by NSF CAREER award 1350481.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 514–524, 2014.

natural communication models (e.g., *randomized communication cost* or *nondeterministic communication cost*), there is a number of "auxiliary" complexity measures that are mostly used as technical tools for studying communication complexity. Probably, the best known and the most useful one is *discrepancy.* More recently a number of other measures have been introduced, including *partition bound*, *rectangle bound*, *smooth discrepancy*, *smooth rectangle bound*, *relaxed partition bound*, γ_2 *norm*, *information cost*, etc. (cf. [2,3]). Most of the currently known structural separations and concrete lower bound proofs in communication complexity can be viewed as analyzing one of the auxiliary complexity measures with respect to a specific communication problem.

1.1 Our Contribution

We show that several conjectures, seemingly weaker than the log-rank conjecture, are in fact equivalent to it. We do so by showing that several natural communication complexity measures are equivalent, up to a polynomial in the logarithm of the rank of the associated matrix.

Theorem 1 (Main Result, Informal). *Let $f : X \times Y \to \{0, 1\}$ be a function and M_f its associated matrix. The following communication complexity costs are equivalent, up to* $\mathrm{poly}(\log \mathrm{rank}(M_f))$ *factors:*

- *The deterministic communication cost of f*
- *The randomized communication cost of f (with public randomness)*
- *The information cost of f*
- *The zero-communication cost of f (a slight variant of the relaxed partition bound)*

Regardless of $\mathrm{rank}(M_f)$, the last three measures are small whenever a short deterministic protocol exists. On the other hand, an assumption that any of those measures (or even all three of them) is small does not, in general, imply existence of a efficient deterministic protocol. We have the following immediate corollary.

Corollary 1. *To prove the log-rank conjecture, it suffices to show that any low-rank function has an efficient randomized protocol, or a protocol with low information cost, or a protocol with low zero-communication complexity.*

In the second part of this work we investigate the weakest notion of communication complexity mentioned in Theorem 1, namely that of zero-communication cost. We use linear-programming duality to show that it is equivalent to the following notion, which extends the usual definition of discrepancy: Let $F : X \times Y \to \{\pm 1\}$ be a sign matrix, then for $0 < \alpha < 1/3$ the *α-extended discrepancy* of F is

$$\mathrm{disc}_\alpha(F) \overset{\mathrm{def}}{=} \max K,$$

$$\textit{subject to:} \quad \sigma \text{ is a distribution on } X \times Y;$$

$$\forall R \in \mathcal{R} : \left| \sum_{(x,y)\in R} \sigma(x, y) \cdot F(x, y) \right| \leq \frac{1}{K} + \alpha \cdot \sigma(R).$$

The case of $\alpha = 0$ corresponds to the usual discrepancy bound. We show that for $\alpha > \Omega(1)$, the α-extended discrepancy is equivalent to zero-communication cost.

Corollary 2. *To prove the log-rank conjecture, it suffices to show that* $\mathrm{disc}_\alpha(F) \leq \mathrm{poly}(\log(\mathit{rank}(F)))$ *for any* $\alpha \geq \Omega(1)$.

Interestingly, Linial et al. [6,7] showed that $\mathrm{disc}(F) \leq O(\sqrt{\mathrm{rank}(F)})$, which allows us to conclude that certain "slightly relaxed" version of our equivalent formulation in terms of extended discrepancy is already known to hold.

Finally, if we set $\alpha = O(1/\sqrt{\mathrm{rank}(F)})$ it immediately implies that $\mathrm{disc}_\alpha(F) \leq O(\sqrt{\mathrm{rank}(F)})$. This allows us to derive another interesting corollary.

Corollary 3. *Let F be a sign matrix with* $\mathrm{rank}(F) = r$ *and* $\mathrm{disc}(F) = d$. *Then F has a deterministic protocol of complexity* $O(d^2 \log(d) \log^2(r))$.

In particular, if for a matrix F one has a discrepancy bound that is better than the one guaranteed by [6,7] by only a poly-logarithmic factor, that already implies existence of a shorter deterministic protocol than what is guaranteed by (1).

Subsequent works. In a subsequent work [9], the second author showed that for any Boolean function of rank r, its deterministic communication complexity is bounded by $\sqrt{r} \log r$.

2 Preliminaries

Let $f : X \times Y \to \{0, 1\}$ be a total[1] Boolean function, where X and Y are finite sets. The *rank* of f is the rank of its associated $\{0, 1\}$-matrix. We review standard definitions in communication complexity (see, e.g., [5] for more definitions and discussion).

Unless stated otherwise, we let a randomized communication protocol use *shared randomness*. We will say that a protocol *computes f with respect to the input distribution μ* if it produces the right answer to $f(X, Y)$ with probability at least 2/3 when $(X, Y) \sim \mu$. We will also say that a protocol *computes f* if it produces the right answer to $f(X, Y)$ for every $(X, Y) \in X \times Y$ with probability at least 2/3. The *deterministic communication cost* of f, denoted $\mathrm{CC}^{\mathrm{det}}(f)$, is the maximal number of bits sent by an optimal deterministic protocol that computes f. The *randomized communication cost* of f, denoted $\mathrm{CC}^{\mathrm{rand}}(f)$, is the maximal number of bits sent by an optimal randomized protocol computing f. The *information cost* of a function, denoted $\mathrm{IC}(f)$, is the infimum of the total amount of information revealed by a randomized protocol computing f to each player about the other player's input, maximized over all choices of the input distribution.

[1] We always assume that the communication task is a total function, as that is all we need in the context of the log-rank conjecture.

Let $\mathcal{R} \stackrel{\text{def}}{=} \{A \times B \mid A \subset X, B \subset Y\}$, and call the elements of $\mathcal{R}$ *rectangles.* A labeled rectangle is a pair (R, z) with $R \in \mathcal{R}$ and $z \in \{0, 1\}$. We will also need a somewhat less common notion of *zero-communication cost* of a function, which we define next.

Definition 1 (Zero-communication cost). *The zero communication cost of f with error ε, denoted $\mathrm{CC}_{\varepsilon}^{\mathrm{zero}}(f)$, is the minimal c such that the following holds. There exists a distribution ρ on labeled rectangles (R, z) such that for any $(x, y) \in X \times Y$,*

1. $\Pr_{(R,z)\sim\rho}[(x, y) \in R] \geq 2^{-c}$.
2. $\Pr_{(R,z)\sim\rho}[f(x, y) = z|(x, y) \in R] \geq 1 - \varepsilon$.

We abbreviate $\mathrm{CC}^{\mathrm{zero}}(f) = \mathrm{CC}_{1/3}^{\mathrm{zero}}(f)$.

For all the notions of communication cost that we have defined, a protocol is considered *efficient* with respect to that specific cost if the latter is bounded by $\mathrm{poly}(\log n)$.

Claim 2. *For any function f,*

$$\mathrm{CC}^{\mathrm{det}}(f) \geq \mathrm{CC}^{\mathrm{rand}}(f) \geq \mathrm{IC}(f) \geq \Omega(\mathrm{CC}^{\mathrm{zero}}(f)).$$

Proof. The first two inequalities follow immediately from the definitions. The fact that $\mathrm{CC}^{\mathrm{zero}}(f) \leq O(\mathrm{IC}(f))$ has been established recently by Kerenidis et al. [3] (Theorem 1.1).[2] □

3 Zero-Error Protocols Reduce to Deterministic Protocols for Low-Rank Functions

We prove the following theorem in this section.

Theorem 3. *Let $f : X \times Y \to \{0, 1\}$ be a Boolean function. Then* $\mathrm{CC}^{\mathrm{det}}(f) \leq O\big(\mathrm{CC}^{\mathrm{zero}}(f) \cdot (\log(\mathrm{rank}(f)))^2\big)$.

We fix the function f and prove Theorem 3 in the remainder of this section. A rectangle $R \subset X \times Y$ is called *monochromatic* if the value of f is constant on R (i.e., all zero or all one). We will use the following theorem of Nisan and Wigderson [10] which shows that to establish that a low-rank function has small deterministic communication cost, it suffices to show that any rectangle contains a large monochromatic sub-rectangle. We denote by $|R|$ the number of elements in a rectangle.

[2] The definition of zero-communication protocol given in [3] (or more accurately, that of a relaxed partition bound) is somewhat different than the definition we are using here. Our definition is less restricting (and probably, somewhat more natural), and therefore $\mathrm{CC}^{\mathrm{zero}}(f) \leq O(\mathrm{IC}(f))$ holds. The main reason for choosing our definition is that it allows more straightforward error reduction by repetition, which we will need later.

Lemma 1 ([10]). *Let $f : X \times Y \to \{0,1\}$ be a Boolean function. Assume that for any rectangle $R_1 \subset X \times Y$ there exists a sub-rectangle $R_2 \subset R_1$ such that R_2 is monochromatic and $|R_2| \geq \delta |R_1|$. Then $\mathrm{CC}^{\mathrm{det}}(f) \leq O\big(\log(1/\delta) \cdot \log \mathrm{rank}(f) + (\log \mathrm{rank}(f))^2\big)$.*

Thus, we reduced proving the theorem to the task of showing that any rectangle contains a large monochromatic sub-rectangle. We next show that this follows given a zero-communication protocol with small enough error.

Lemma 2. *Let $f : X \times Y \to \{0,1\}$ be a Boolean function with $\mathrm{rank}(f) = r$. Set $\varepsilon = 1/8r$ and assume that $\mathrm{CC}_{\varepsilon}^{\mathrm{zero}}(f) = c$. Then any rectangle $R_1 \subset X \times Y$ contains a monochromatic sub-rectangle $R_2 \subset R_1$ with $|R_2| \geq (1/16)2^{-c}|R_1|$.*

Proof. We first establish that there exists a sub-rectangle $R' \subset R_1$ which is nearly monochromatic, and then use the low-rank property of f to establish the existence of a monochromatic rectangle $R_2 \subset R'$.

Let ρ be the distribution on labeled rectangles guaranteed by the zero-communication cost assumption, and let $(R,z) \sim \rho$ and $R' = R \cap R_1$. Let $\mathcal{E} = \{(x,y) \in R_1 : f(x,y) \neq z\}$ be the set of inputs whose value disagrees with z. We will show that $|R'| \geq (1/2)2^{-c}|R_1|$ and $|R' \cap \mathcal{E}| \leq 2\varepsilon |R'|$ with non-zero probability.

Note that we can reformulate the definition of a zero-communication protocol as

$$\Pr[(x,y) \in R] \geq 2^{-c}; \quad \Pr[(x,y) \in R \text{ and } f(x,y) \neq z] \leq \varepsilon \Pr[(x,y) \in R],$$

where the probabilities are taken over $(R,z) \sim \rho$. So,

$$\Pr[(x,y) \in R] - (1/2\varepsilon) \Pr[(x,y) \in R \text{ and } f(x,y) \neq z] \geq (1/2)2^{-c}.$$

Summing over all $(x,y) \in R_1$ gives

$$\mathbb{E}[|R'| - 1/(2\varepsilon) \cdot |R' \cap \mathcal{E}|] \geq (1/2)2^{-c}|R_1|.$$

Thus, there must exist a choice for R' exceeding the average. In particular, it satisfies $|R'| \geq (1/2)2^{-c}|R_1|$ and $|R' \cap \mathcal{E}| \leq 2\varepsilon |R'|$.

Next we establish existence of a large monochromatic rectangle $R_2 \subset R'$. Assume that $R' = A' \times B'$. Let $A'' \subset A'$ be the set of rows having at most 4ε-fraction of elements disagreeing with z,

$$A'' = \{x \in A' : |\{y \in B' : f(x,y) \neq z\}| \leq 4\varepsilon |B'|\}.$$

By Markov's inequality we have $|A''| \geq |A'|/2$. We now apply the low-rank property of f. Consider the matrix generated by restricting f to $A'' \times B'$. Its rank is also at most $r = \mathrm{rank}(f)$. Hence, there exist r elements $x_1, \ldots, x_r \in A''$ whose corresponding rows span all rows in A''. Define for $1 \leq i \leq r$ the sets of inputs taking the "wrong" value on row x_i,

$$B_i = \{y \in B' : f(x_i, y) \neq z\}.$$

We know by assumption that $|B_i| \leq 4\varepsilon|B'|$. Let $B'' = B' \setminus \cup_{i=1}^{r} B_i$. Then $|B''| \geq |B'|(1 - 4\varepsilon r) \geq |B'|/2$. Consider now the matrix restricted to $A'' \times B''$. It is spanned by rows which are all equal to z, hence all of its rows are constant! We take R_2 to be the set of rows taking the majority value. To conclude, we found a monochromatic rectangle $R_2 \subset R_1$ of size

$$|R_2| \geq (1/2)|A''||B''| \geq (1/8)|A'||B'| = (1/16)2^{-c}|R_1|,$$

as required.

□

We are nearly done, it remains to argue that the error in zero-communication protocols can be reduced efficiently.

Claim 4. *For any* $1/2 > \lambda > \varepsilon > 0$,

$$\mathrm{CC}^{\mathrm{zero}}_{\varepsilon}(f) \leq O\left(\mathrm{CC}^{\mathrm{zero}}_{\lambda}(f) \cdot \frac{\log(1/\varepsilon)}{(1/2 - \lambda)^2}\right).$$

In particular, $\mathrm{CC}^{\mathrm{zero}}_{\varepsilon}(f) \leq O(\mathrm{CC}^{\mathrm{zero}}(f) \cdot \log(1/\varepsilon))$.

Proof. Denote $c \overset{\mathrm{def}}{=} \mathrm{CC}^{\mathrm{zero}}_{\lambda}(f)$, and let ρ be a distribution over labeled rectangles (R, z) satisfying

1. $\Pr_{(R,z)\sim\rho}[(x, y) \in R] \geq 2^{-c}$.
2. $\Pr_{(R,z)\sim\rho}[f(x, y) = z|(x, y) \in R] \geq 1 - \lambda$.

Let $t = O\left(\frac{\log(1/\varepsilon)}{(1/2-\lambda)^2}\right)$, and sample $(R_1, z_1), \ldots, (R_t, z_t) \sim \rho$ independently. We define R^* to be the intersection of $R_1, \ldots, R_t$ and z^* to be the majority value of $z_1, \ldots, z_t$. We claim that the resulting distribution of (R^*, z^*) gives a zero-communication protocol with error ε and cost ct.

In order to see this, fix $(x, y) \in X \times Y$. First, we verify that $(x, y) \in R^*$ frequently enough,

$$\Pr[(x, y) \in R^*] = \Pr[(x, y) \in R_1, \ldots, (x, y) \in R_t] = \prod_{i=1}^{t} \Pr[(x, y) \in R_i] \geq 2^{-ct}.$$

It remains to verify that $\Pr[f(x, y) = z^*|(x, y) \in R^*] \geq 1 - \varepsilon$. Let $p \overset{\mathrm{def}}{=} \Pr[f(x, y) = z|(x, y) \in R]$; for any $v_1, \ldots, v_t \in \{0, 1\}$ such that $|\{i \,|\, v_i = f(x, y)\}| = m$,

$$\Pr[z_1 = v_1, \ldots, z_t = v_t|(x, y) \in R^*] = \prod_{i=1}^{r} \Pr[z_i = v_i|(x, y) \in R_i] = p^m(1-p)^{t-m}.$$

So, the probability that $z^* = f(x, y)$ conditioned on $(x, y) \in R^*$ is given by summing over all values $v_1, \ldots, v_t$ whose majority is equal to $f(x, y)$. This equals the probability that a binomial distribution with t trials and success probability $p \geq 1-\lambda$ has at least $t/2$ successes:

$$\Pr[f(x,y)=z^*|(x,y)\in R^*] \geq \Pr[\mathrm{Bin}(t,1-\lambda)\geq t/2] \geq 1-\alpha^{(1/2-\lambda)^2\cdot t}$$

for some constant $0<\alpha<1$. Choosing $t=O\left(\frac{\log(1/\varepsilon)}{(1/2-\lambda)^2}\right)$ large enough gives the required bound. □

4 Equivalent Formulations

Recall the original log-rank conjecture of Lovász and Saks [8]:

*Conjecture 1 (***log-rank***, [8]).* For every $\{0,1\}$-valued matrix M,

$$\mathrm{CC}^{\mathrm{det}}(M) \leq \mathrm{poly}(\log(\mathrm{rank}(M))).$$

We will present several equivalent formulations of the conjecture.

First, we give a version that "looks weaker" in the following sense: While the original conjecture can be phrased as "low rank implies an efficient deterministic protocol", this formulation only requires existence of an efficient protocol of one of several more powerful types.

Theorem 5 (Log-Rank Conjecture, Equivalent Formulations). *The following statements are equivalent:*

1. *The log-rank conjecture (Conjecture 1).*
2. *For every $\{0,1\}$-valued M, $\mathrm{CC}^{\mathrm{rand}}(M) \leq \mathrm{poly}(\log(\mathrm{rank}(M)))$.*
3. *For every $\{0,1\}$-valued M, $\mathrm{IC}(M) \leq \mathrm{poly}(\log(\mathrm{rank}(M)))$.*
4. *For every $\{0,1\}$-valued M, $\mathrm{CC}^{\mathrm{zero}}(M) \leq \mathrm{poly}(\log(\mathrm{rank}(M)))$.*

Proof. From Claim 2 and Theorem 3 it follows that $\mathrm{CC}^{\mathrm{det}}(M)$, $\mathrm{CC}^{\mathrm{rand}}(M)$, $\mathrm{IC}(M)$ and $\mathrm{CC}^{\mathrm{zero}}(M)$ are equal, up to the factor of $\mathrm{poly}(\log(\mathrm{rank}(M)))$. □

4.1 Extended Discrepancy: Extrapolating between Discrepancy and Zero-Communication Cost

Let us denote $F(x,y) \overset{\mathrm{def}}{=} (-1)^{M_{x,y}}$ and $\mathrm{CC}_{\varepsilon}^{\mathrm{zero}}(F)=\mathrm{CC}_{\varepsilon}^{\mathrm{zero}}(M)$. Define for every $\{\pm1\}$-valued matrix F its *α-extended discrepancy* as

$$\mathrm{disc}_{\alpha}(F) \overset{\mathrm{def}}{=} \max K,$$

subject to: σ is a distribution on $X\times Y$;

$$\forall R\in\mathcal{R}: \left|\sum_{(x,y)\in R}\sigma(x,y)\cdot F(x,y)\right| \leq \frac{1}{K}+\alpha\cdot\sigma(R).$$

If we choose $\alpha=0$ then $\mathrm{disc}_{\alpha}(F)$ equals the *discrepancy* of F, which is one of the most commonly used tools for proving lower bounds on $\mathrm{CC}^{\mathrm{rand}}(M)$.

Claim 6. *For every $\{\pm 1\}$-valued matrix F and every constant $0 < \alpha < \frac{1}{3}$,*

$$2^{\mathrm{CC}^{\mathrm{zero}}(F)} = \Theta(\mathrm{disc}_\alpha(F)).$$

Proof. Let us express $\mathrm{CC}^{\mathrm{zero}}(M)$ as the optimal value of a linear program,

$$\begin{aligned} 2^{\mathrm{CC}^{\mathrm{zero}}_{\varepsilon}(M)} &= \min \sum_{R \in \mathcal{R}} \left(w_{R,0} + w_{R,1}\right), \\ \textit{subject to:}\quad & \forall R \in \mathcal{R} : w_{R,0} \geq 0, w_{R,1} \geq 0; \\ & \forall (x,y) \in X \times Y : \sum_{R:(x,y)\in R} \left(w_{R,0} + w_{R,1}\right) \geq 1; \\ & \forall (x,y) \in X \times Y : \sum_{R:(x,y)\in R} \left(w_{R,M_{x,y}} - \frac{1-\varepsilon}{\varepsilon} \cdot w_{R,1-M_{x,y}}\right) \geq 0. \end{aligned}$$

Its dual can be written as

$$\begin{aligned} 2^{\mathrm{CC}^{\mathrm{zero}}_{\varepsilon}(M)} &= \max K, \\ \textit{subject to:}\quad & \forall (x,y) \in X \times Y : C_{x,y} \geq 0;\ \mu \text{ is a distribution on } X \times Y; \\ & \forall R {\in} \mathcal{R}, z \in \{0,1\} : \mu(R) \leq \frac{1}{K} + \frac{1-\varepsilon}{\varepsilon} \cdot \sum_{\substack{(x,y)\in R \\ M_{x,y}=z}} C_{x,y} - \sum_{\substack{(x,y)\in R \\ M_{x,y}=1-z}} C_{x,y}. \end{aligned}$$

We can rewrite it as

$$\begin{aligned} 2^{\mathrm{CC}^{\mathrm{zero}}_{\varepsilon}(F)} &= \max K, \\ \textit{subject to:}\quad & \sigma : X \times Y \to \mathbb{R}^{+};\ \mu \text{ is a distribution on } X \times Y; \\ & \forall R \in \mathcal{R} : \left| \sum_{(x,y)\in R} \sigma(x,y)F(x,y) \right| - (1-2\varepsilon)\cdot\sigma(R) + \mu(R) \leq \frac{1}{K}. \end{aligned} \tag{2}$$

First, we show that $2^{\mathrm{CC}^{\mathrm{zero}}(F)} \geq \Omega(\mathrm{disc}_\alpha(F))$. For $\alpha < \frac{1}{3}$, let μ be a distribution on $X \times Y$, such that $\forall R \in \mathcal{R}$:

$$\left| \sum_{(x,y)\in R} \mu(x,y)F(x,y) \right| \leq \frac{1}{\mathrm{disc}_\alpha(F)} + \alpha\mu(R).$$

Then for $t \overset{\mathrm{def}}{=} \frac{3\alpha}{1-3\alpha}$,

$$\frac{t}{\alpha} \cdot \left| \sum_{R} \mu(x,y)F(x,y) \right| - (t+1)\cdot\mu(R) + \mu(R) \leq \frac{t}{\alpha \mathrm{disc}_\alpha(F)},$$

and therefore,

$$2^{\mathrm{CC}^{\mathrm{zero}}(F)} \geq \frac{1-3\alpha}{3} \cdot \mathrm{disc}_\alpha(F).$$

For the other direction of our proof, let $\sigma : X \times Y \to \mathbb{R}^+$ and μ be a distribution on $X \times Y$, such that $\forall R \in \mathcal{R}$:

$$\left| \sum_{(x,y)\in R} \sigma(x,y)F(x,y) \right| - (1-2\varepsilon) \cdot \sigma(R) + \mu(R) \leq \frac{1}{2^{\mathrm{CC}_\varepsilon^{\mathrm{zero}}(F)}}, \tag{3}$$

which implies

$$\left| \sum_{(x,y)\in R} \sigma(x,y)F(x,y) \right| \leq \frac{1}{2^{\mathrm{CC}_\varepsilon^{\mathrm{zero}}(F)}} + (1-2\varepsilon) \cdot \sigma(R).$$

For $R = X \times Y$, (3) implies

$$\sigma(X \times Y) \geq \left(1 - \frac{1}{2^{\mathrm{CC}_\varepsilon^{\mathrm{zero}}(F)}}\right) \cdot \frac{1}{1-2\varepsilon} \geq \frac{1}{2},$$

as long as $\mathrm{CC}_\varepsilon^{\mathrm{zero}}(F) \geq 1$. Then for the distribution $\sigma' \overset{\mathrm{def}}{=} \frac{\sigma}{\sigma(X \times Y)}$,

$$\left| \sum_{(x,y)\in R} \sigma'(x,y)F(x,y) \right| \leq \frac{2}{2^{\mathrm{CC}_\varepsilon^{\mathrm{zero}}(F)}} + (1-2\varepsilon) \cdot \sigma'(R).$$

Accordingly,

$$\mathrm{disc}_{1-2\varepsilon}(F) \geq \frac{2^{\mathrm{CC}_\varepsilon^{\mathrm{zero}}(F)}}{2}. \tag{4}$$

As long as $\alpha > 0$, we can use the error-reducing technique given by Claim 4, and therefore $2^{\mathrm{CC}^{\mathrm{zero}}(F)} \leq O(\mathrm{disc}_\alpha(F))$, as required. □

The following equivalent formulations of the log-rank conjecture is immediate from Theorem 5 and Claim 6.

Theorem 7 (log-rank conjecture, an equivalent formulation). *The log-rank conjecture (Conjecture 1) is true if and only if the following holds for some* $\alpha_0 \geq \Omega(1)$*: For every* $\{\pm 1\}$*-valued matrix* M *and probability distribution* μ *on* $X \times Y$*, there exists a rectangle* $R \in \mathcal{R}$ *such that*

$$\left| \sum_{(x,y)\in R} \sigma(x,y) \cdot M_{x,y} \right| \geq \frac{1}{\mathrm{qpoly}(\mathrm{rank}(M))} + \alpha_0 \cdot \sigma(R),$$

where $\mathrm{qpoly}(x) \overset{def}{=} \exp(\mathrm{poly}(\log x))$.

Interestingly, it was shown by Linial et al. [6,7] that $\mathrm{disc}(M) \leq O\left(\sqrt{\mathrm{rank}(M)}\right)$. In other words,

Lemma 8 ([6,7])**.** *For every $\{\pm 1\}$-valued matrix M and probability distribution μ on $X \times Y$, there exists a rectangle $R \in \mathcal{R}$ such that*

$$\left| \sum_{(x,y)\in R} \sigma(x,y) \cdot M_{x,y} \right| \geq \Omega\left(\frac{1}{\sqrt{\mathrm{rank}(M)}} \right).$$

Note that the above statement can be viewed as a version of the equivalent formulation given in Theorem 7, relaxed by letting "$\alpha_0 = 1 \big/ \sqrt{\mathrm{rank}(M)}$".

Finally, our techniques can be used to derive a polynomial upper bound on $\mathrm{CC}^{\mathrm{det}}(F)$ in terms of $\mathrm{disc}(F)$ and $\log(\mathrm{rank}(F))$.

Claim 9. *For a $\{\pm 1\}$-valued matrix F, let $d = \mathrm{disc}(F)$. Then*

$$\mathrm{CC}^{\mathrm{det}}(F) \leq O\big(d^2 \cdot (\log(\mathrm{rank}(F)))^2 \cdot \log d\big).$$

Proof. It is immediate from the definition that $\frac{1}{\mathrm{disc}(F)}$-extended discrepancy is equivalent to $\mathrm{disc}(F)$ up to a constant multiplicative factor, and therefore

$$\mathrm{disc}_{1/d}(F) \leq O(d).$$

In the proof of Claim 6 we have shown (cf. (4)) that

$$2^{\mathrm{CC}_{\varepsilon}^{\mathrm{zero}}(F)} \leq O(\mathrm{disc}_{1-2\varepsilon}(F))$$

holds for every $\varepsilon > 0$, and therefore

$$\mathrm{CC}^{\mathrm{zero}}_{\frac{1}{2}-\frac{1}{2d}}(F) \leq \log d + O(1).$$

By Claim 4,

$$\mathrm{CC}^{\mathrm{zero}}(F) \leq O\big(d^2 \log d\big),$$

and the result follows by Theorem 3. □

References

1. Ben-Sasson, E., Lovett, S., Ron-Zewi, N.: An Additive Combinatorics Approach Relating Rank to Communication Complexity. In: Proceedings of the 53rd Annual Symposium on Foundations of Computer Science, pp. 177–186 (2012)
2. Jain, R., Klauck, H.: The Partition Bound for Classical Communication Complexity and Query Complexity. In: Proceedings of the 25th IEEE Conference on Computational Complexity, pp. 247–258 (2010)
3. Kerenidis, I., Laplante, S., Lerays, V., Roland, J., Xiao, D.: Lower Bounds on Information Complexity via Zero-Communication Protocols and Applications. In: Proceedings of the 53rd Annual Symposium on Foundations of Computer Science, pp. 500–509 (2012)
4. Kotlov, A.: Rank and Chromatic Number of a Graph. Journal of Graph Theory 26(1), 1–8 (1997)

5. Kushilevitz, E., Nisan, N.: Communication Complexity. Cambridge University Press (1997)
6. Linial, N., Mendelson, S., Schechtman, G., Schraibman, A.: Complexity Measures of Sign Matrices. Combinatorica 27(4), 439–463 (2007)
7. Linial, N., Schraibman, A.: Learning Complexity vs. Communication Complexity. Combinatorics, Probability & Computing 18(1-2), 227–245 (2009)
8. Lovász, L., Saks, M.: Lattices, Möbius Functions and Communication Complexity. In: Annual Symposium on Foundations of Computer Science, pp. 81–90 (1988)
9. Lovett, S.: Communication is Bounded by Root of Rank. In: Proceedings of the 46th Symposium on Theory of Computing (2014)
10. Nisan, N., Wigderson, A.: On Rank vs. Communication Complexity. In: Proceedings of the 35th Annual Symposium on Foundations of Computer Science, pp. 831–836 (1994)

Improved Submatrix Maximum Queries in Monge Matrices*

Paweł Gawrychowski[1], Shay Mozes[2,**], and Oren Weimann[3,**]

[1] MPII
gawry@mpi-inf.mpg.de
[2] IDC Herzliya
smozes@idc.ac.il
[3] University of Haifa
oren@cs.haifa.ac.il

Abstract. We present efficient data structures for submatrix maximum queries in Monge matrices and Monge partial matrices. For $n \times n$ Monge matrices, we give a data structure that requires $O(n)$ space and answers submatrix maximum queries in $O(\log n)$ time. The best previous data structure [Kaplan et al., SODA'12] required $O(n \log n)$ space and $O(\log^2 n)$ query time. We also give an alternative data structure with constant query-time and $O(n^{1+\varepsilon})$ construction time and space for any fixed $\varepsilon < 1$. For $n \times n$ *partial* Monge matrices we obtain a data structure with $O(n)$ space and $O(\log n \cdot \alpha(n))$ query time. The data structure of Kaplan et al. required $O(n \log n \cdot \alpha(n))$ space and $O(\log^2 n)$ query time. Our improvements are enabled by a technique for exploiting the structure of the upper envelope of Monge matrices to efficiently report column maxima in skewed rectangular Monge matrices. We hope this technique will be useful in obtaining faster search algorithms in Monge partial matrices. In addition, we give a linear upper bound on the number of breakpoints in the upper envelope of a Monge partial matrix. This shows that the inverse Ackermann $\alpha(n)$ factor in the analysis of the data structure of Kaplan et. al is superfluous.

1 Introduction

A matrix M is a *Monge* matrix if for any pair of rows $i < j$ and columns $k < \ell$ we have that $M_{ik} + M_{j\ell} \geq M_{i\ell} + M_{jk}$. Monge matrices have many applications in combinatorial optimization and computational geometry. For example, they arise in problems involving distances in the plane [20,23,25,27], and in problems on convex n-gons [2,3]. See [9] for a survey on Monge matrices and their uses in combinatorial optimization.

In this paper we consider the following problem: Given an $n \times n$ Monge matrix M, construct a data structure that can report the maximum entry in any query submatrix (defined by a set of consecutive rows and a set of consecutive columns). Recently, Kaplan, Mozes, Nussbaum and Sharir [21] presented an $\tilde{O}(n)$ space[1]

[*] A full version of this paper can be found as Arxiv preprint [18].

[**] Mozes and Weimann supported in part by Israel Science Foundation grant 794/13.

[1] The $\tilde{O}(\cdot)$ notation hides polylogarithmic factors in n.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 525–537, 2014.

data structure with $\tilde{O}(n)$ construction time and $O(\log^2 n)$ query time. They also described an extension of the data structure to handle *partial* Monge matrices (where some of the entries of M are undefined, but the defined entries in each row and in each column are contiguous). The extended data structure incurs larger polylogarithmic factors in the space and construction time. Both the original and the extended data structures have various important applications. They are used in algorithms that efficiently find the largest empty rectangle containing a query point, in dynamic distance oracles for planar graphs, and in algorithms for maximum flow in planar graphs [6]. See [21] for more details on the history of this problem and its applications.

Note that, even though explicitly representing the input matrix requires $N = \Theta(n^2)$ space, the additional space required by the submatrix maximum data structure of [21] is only $\tilde{O}(n)$. In many applications (in particular [6,21]), the matrix M is not stored explicitly but any entry of M can be computed when needed in $O(1)$ time. The space required by the application is therefore dominated by the size of the submatrix maximum data structure. With the increasing size of problem instances, and with current memory and cache architectures, space often becomes the most significant resource.

For general (i.e., not Monge) matrices, a long line of research over the last three decades including [5,13,14,17,28] achieved $\tilde{O}(N)$ space and $\tilde{O}(1)$ query data structures, culminating with the $O(N)$-space $O(1)$-query data structure of Yuan and Atallah [28]. Here $N = n^2$ denotes the total number of entries in the matrix. It is also known [8] that reducing the space to $O(N/c)$ incurs an $\Omega(c)$ query-time. Tradeoffs requiring $O(N/c)$ additional space and $\tilde{O}(c)$ query-time were given in [7,8]. When the matrix has only $N = o(n^2)$ nonzero entries, the problem is known in computational geometry as the *orthogonal range searching* problem on the $n \times n$ grid. In this case as well, various tradeoffs with $\tilde{O}(N)$-space and $\tilde{O}(1)$-query appear in a long history of results including [4,10,11,15,17]. In particular, a linear $O(N)$-space data structure was given by Chazelle [11] at the cost of an $O(\log^{\varepsilon} n)$ query time. See [24] for a survey on orthogonal range search.

Contribution. Our first contribution is in designing $O(n)$-space $O(\log n)$-query data structures for submatrix maximum queries in Monge matrices and in partial Monge matrices (see Section 3). Our data structures improve upon the data structures of Kaplan et al. in both space and query time. Consequently, using our data structures for finding the largest empty rectangle containing a query point improves the space and query time by logarithmic factors.

In the full version of this paper [18], we further provide alternative data structures with faster query-time; We achieve $O(1)$ query-time at the cost of $O(n^{1+\varepsilon})$ construction time and space for an arbitrarily small constant $0 < \varepsilon < 1$.

Our results are achieved by devising a data structure for reporting column maxima in $m \times n$ Monge matrices with many more columns than rows ($n >> m$). We refer to this data structure as the *micro* data structure. The space required by the micro data structure is linear in m, and independent of n. Its construction-time depends only logarithmically on n. The query-time is $O(\log\log n)$, the time required for a predecessor query in a set of integers bounded by n. We use the micro data structure in the design of our submatrix maximum query data structures, exploiting its sublinear dependency on n, and an ability to trade off construction and query times.

For partial Monge matrices, we provide a tight $O(m)$ upper bound on the complexity of the upper envelope (see Section 4). The best previously known bound [26] was $m\alpha(m)$, where $\alpha(m)$ is the inverse Ackermann function. This upper bound immediately implies that the $\alpha(m)$ factor stated in the space and construction time of the data structures of Kaplan et al. is superfluous.

Notice that the upper envelope of a *full* $m \times n$ Monge matrix also has complexity $O(m)$. The famous SMAWK algorithm [2] can find all column maxima in $O(n + m)$ time. However, this is not the case for partial Monge matrices. Even for simple partial Monge matrices such as triangular, or staircase matrices, where it has been known for a long time that the complexity of the upper envelope is linear, the fastest known algorithm for finding all column maxima is the $O(n\alpha(m) + m)$ time algorithm of Klawe and Kleitman [22]. We hope that our micro data structure will prove useful for obtaining a linear-time algorithm. The known algorithms, including the $(n\alpha(m) + m)$-time algorithm of Klawe and Kleitman [22], partition the matrix into skewed rectangular matrices, and use the SMAWK algorithm. It is plausible that our micro data structure will yield a speed up since it is adapted to skewed matrices.

2 Preliminaries and Our Results

In this section we overview the data structures of [21] and highlight our results.

A matrix M is a *Monge* matrix if for any pair of rows $i < j$ and columns $k < \ell$ we have that $M_{ik} + M_{j\ell} \geq M_{i\ell} + M_{jk}$. A matrix M is *totally monotone in columns* if for any pair of rows $i < j$ and columns $k < \ell$ we have that if $M_{ik} \leq M_{jk}$ then $M_{i\ell} \leq M_{j\ell}$. Similarly, M is *totally monotone in rows* if for any pair of rows $i < j$ and columns $k < \ell$ we have that if $M_{ik} \leq M_{i\ell}$ then $M_{jk} \leq M_{j\ell}$. Notice that the Monge property implies total monotonicity (in columns and in rows) but the converse is not true. When we simply say *totally monotone* (or TM) we mean totally monotone *in columns* (our results symmetrically apply to totally monotone *in rows*).

A matrix M is a *partial* matrix if some entries of M are undefined, but the defined entries in each row and in each column are contiguous. We assume w.l.o.g. that every row has at least one defined element and that the defined elements form a single connected component (i.e., the defined column intervals in each pair of consecutive rows overlap). If this is not the case then only minor changes are needed in our algorithms. A partial TM (resp., Monge) matrix is a partial matrix whose defined entries satisfy the TM (resp., Monge) condition.

We consider $m \times n$ matrices, but for simplicity we sometimes state the results for $n \times n$ matrices. For a Monge matrix M, denote $r(j) = i$ if the maximum element in column j lies in row i. (We assume this maximum element is unique. It is simple to break ties by, say, taking the highest index.) The *upper envelope* $\mathcal{E}$ of all the rows of M consists of the n values $r(1), \ldots, r(n)$. Since M is Monge we have that $r(1) \leq r(2) \leq \ldots \leq r(n)$ and so $\mathcal{E}$ can be implicitly represented in $O(m)$ space by keeping only the $r(j)$s of $O(m)$ columns called *breakpoints.* Breakpoints are the columns j where $r(j) \neq r(j+1)$. The maximum element $r(\pi)$ of any column π can then be retrieved in $O(\log m)$ time by a binary search for the first breakpoint-column j after π, and setting $r(\pi) = r(j)$.

The first data structure of [21] is a balanced binary tree T_h over the rows of M. A node u whose subtree contains k leaves (i.e., k rows) stores the $O(k)$ breakpoints of the $k \times n$ matrix M^u defined by these k rows and all columns of M. A leaf represents a single row and requires no computation. An internal node u obtains its breakpoints by merging the breakpoints of its two children: its left child u_1 and its right u_2. By the Monge property, the list of breakpoints of u starts with a prefix of breakpoints of u_1 and ends with a suffix of breakpoints of u_2. Between these there is possibly one new breakpoint j. The prefix and suffix parts can be found easily in $O(k)$ time by linearly comparing the lists of breakpoints of u_1 and u_2. The new breakpoint j can then be found in additional $O(\log n)$ time via binary search. Summing $O(k+\log n)$ over all nodes of T_h gives $O(m(\log m + \log n))$ time. The total size of T_h is $O(m \log m)$.

Note that the above holds even if M is not Monge but only TM. This gives rise to a data structure that answers subcolumn (as opposed to submatrix) queries:

Subcolumn Queries in TM Matrices [21]. *Given a $n \times n$ TM matrix, one can construct, in $O(n \log n)$ time, a data structure of size $O(n \log n)$ that reports the maximum in a query column and a contiguous range of rows in $O(\log n)$ time.*

The maximum entry in a query column π and a contiguous range of rows R is found using T_h by identifying $O(\log m)$ *canonical nodes* of T_h. A node u is canonical if u's set of rows is contained in R but the set of rows of u's parent is not. For each such canonical node u, we find in $O(\log m)$ time the maximum element in column π amongst all the rows of u. The output is the largest of these and the total query time is $O(\log^2 m)$. The query time can be reduced to $O(\log m)$ by using fractional cascading [12].

The first results of our paper improve the above subcolumn query data structure of [21], as indicated in Table 1 under subcolumn query in TM matrices. The next data structure of [21] extends the queries from subcolumn to submatrix (specified by ranges R of consecutive rows, and C of consecutive columns.)

Submatrix Queries in Monge Matrices [21]. *Given a $n \times n$ Monge matrix, one can construct, in $O(n \log n)$ time, a data structure of size $O(n \log n)$ that reports the maximum entry in a query submatrix in $O(\log^2 n))$ time.*

To obtain $O(\log^2 n) = O(\log m(\log m + \log n))$ query time, note that R is the disjoint union of $O(\log m)$ canonical nodes of T_h. For each such canonical node u, we use u's list of breakpoints $\{j_1, j_2, \ldots, j_k\}$ to find in $O(\log m + \log n)$ time the maximum element in all rows of u and the range of columns C. This is done as follows: we first identify in $O(\log m)$ time the set $\mathcal{I} = \{j_a, j_{a+1}, \ldots, j_b\}$ of u's breakpoints that are fully contained in C. The columns of C that are to the left of j_a all have their maximum element in row $r(j_a)$. To find the maximum of these we construct, in addition to T_h, a symmetric binary tree $\mathcal{B}$ that can report in $O(\log n)$ time the maximum entry in a query *row* and a contiguous range of *columns*. $\mathcal{B}$ is built in $O(n(\log m + \log n))$ time and $O(n \log n)$ space using the subcolumn query data structure on the transpose of M. This is possible since M is Monge.[2] Similarly, we find in $O(\log n)$ time the maximum in all columns of C that are to the right of j_b.

To find the maximum in all columns between j_a and j_b, let $m(j_i)$ denote the maximum element in the columns interval $(j_{i-1}, j_i]$ (note it must be in row $r(j_i)$). We wish to find $\max\{m(j_{a+1}), \ldots, m(j_b)\}$ which corresponds to a Range Maximum Query in the array $A^u = \{m(j_1), \ldots, m(j_k)\}$. We compute the array A^u (along with a naive RMQ data structure with logarithmic query time) of every node u during the construction of T_h. Most of the entries of A^u are simply copied from u's children arrays A^{u_1} and A^{u_2}. The only new $m(\cdot)$ value that u needs to compute is for the single new breakpoint j (that is between the prefix from u_1 and the suffix from u_2). Since $m(j)$ must be in row $r(j)$ it can be computed in $O(\log n)$ time by a single query to $\mathcal{B}$.

Overall, we get a query time of $O(\log m + \log n)$ per canonical node u for a total of $O(\log m(\log m + \log n))$. Building T_h (along with all the RMQ arrays A^u) and $\mathcal{B}$ takes total $O((m+n)(\log m + \log n))$ time and $O(m \log m + n \log n)$ space. Our two improvements to this bound of [21] are stated in Table 1 under submatrix queries in Monge matrices.

The next data structures of [21] extend the above subcolumn and submatrix data structures from full to *partial* TM matrices. The construction is very similar. Merging the breakpoints of the two children u_1, u_2 of a node u of T_h is slightly more involved now, since the envelopes may cross each other multiple times. The number of breakpoints of any subset of consecutive k rows is $O(k \cdot \alpha(k))$ [26], and so there are $O(m \log m \cdot \alpha(m))$ breakpoints in total over all nodes of T_h (as opposed to $O(m)$ in full matrices). This implies the following:

Subcolumn Queries in Partial TM Matrices [21]. *Given a partial TM $n \times n$ matrix, one can construct, in $O(n \log^2 n \cdot \alpha(n))$ time, a data structure of size $O(n \log n \cdot \alpha(n))$ that reports the maximum entry in a query column and a contiguous range of rows in $O(\log n)$ time.*

We improve this data structure to the same bounds we get for full matrices. i.e, we show that our bounds for full matrices also apply to partial matrices. This is stated in Table 1 under subcolumn query in Partial TM matrices. Finally, [21] extended their submatrix data structure from full to partial Monge matrices. It uses a similar construction of T_h and $\mathcal{B}$ as in the case of full matrices, but again requires the additional $O(\log m \cdot \alpha(m) + \log n \cdot \alpha(n))$ multiplicative factor to store the breakpoints of all nodes of T_h and $\mathcal{B}$.

Submatrix Queries in Partial Monge Matrices [21]. *Given a $n \times n$ partial Monge matrix, one can construct, in $O(n\alpha(n) \log^2 n)$ time, a data structure of size $O(n\alpha(n) \log n)$ that reports the maximum entry in a query submatrix in $O(\log^2 n)$ time.*

[2] In fact it suffices that M is a TM matrix whose transpose is also TM.

Table 1. Our results compared to [21]

property	query type	space	construction time	query time	
TM	subcolumn	$O(n \log n)$	$O(n \log n)$	$O(\log n)$	Lemma 3.1 [21]
TM	subcolumn	$O(n)$	$O(n \log n/ \log\log n)$	$O(\log n)$	Lemma 2
TM	subcolumn	$O(n^{1+\varepsilon})$	$O(n^{1+\varepsilon})$	$O(1)$	Full version [18]
Monge	submatrix	$O(n \log n)$	$O(n \log n)$	$O(\log^2 n)$	Theorem 3.2 [21]
Monge	submatrix	$O(n)$	$O(n \log n)$	$O(\log n)$	Theorem 1
Monge	submatrix	$O(n)$	$O(n \log n/ \log\log n)$	$O(\log^{1+\varepsilon} n)$	Corollary 1
Monge	submatrix	$O(n^{1+\varepsilon})$	$O(n^{1+\varepsilon})$	$O(1)$	Full version [18]
Partial TM	subcolumn	$O(n \log n \cdot \alpha(n))$	$O(n \log^2 n \cdot \alpha(n))$	$O(\log n)$	Lemma 3.3 [21]
Partial TM	subcolumn	$O(n)$	$O(n \log n/ \log\log n)$	$O(\log n)$	Full version [18]
Partial TM	subcolumn	$O(n^{1+\varepsilon})$	$O(n^{1+\varepsilon})$	$O(1)$	Full version [18]
Partial Monge	submatrix	$O(n \log n \cdot \alpha(n))$	$O(n \log^2 n \cdot \alpha(n))$	$O(\log^2 n)$	Theorem 3.4 [21]
Partial Monge	submatrix	$O(n)$	$O(n \log n)$	$O(\log n \cdot \alpha(n))$	Theorem 2 here
Partial Monge	submatrix	$O(n)$	$O(n \log n/ \log\log n)$	$O(\log^{1+\varepsilon} n \cdot \alpha(n))$	Corollary 2

We remove the $O(\log n \cdot \alpha(n))$ multiplicative factor and obtain the bounds stated in the bottom of Table 1. The $\alpha(n)$ factor is removed by showing that the number of breakpoints in the upper envelope of a partial Monge matrix is linear.

3 Linear-Space Data Structures

In this section we present our data structures that improve the space to $O(n)$ and the query time to $O(\log n)$. We begin by introducing a new data structure for the case where a query is composed of an *entire* column (as opposed to a range of rows). This new data structure (which we call the micro data structure) is designed to work well when the number of rows in the matrix is much smaller than the number of columns. We denote by $pred(x, n) = O(\min\{\log x, \log\log n\})$ the time to query a predecessor data structure with x elements from $\{1, \ldots, n\}$.

Lemma 1 (The Micro Data Structure). *Given a $x \times n$ TM matrix and $r > 0$, one can construct in $O(x \log n/ \log r)$ time, a data structure of size $O(x)$ that given a query column can report the maximum entry in the entire column in $O(r + pred(x, n))$ time.*

Proof. Out of all n columns of the input matrix M, we will designate $O(x)$ columns as *special* columns. For each of these special columns we will eventually compute its maximum element. The first x special columns of M are columns $1, n/x, 2n/x, 3n/x, \ldots, n$ and are denoted $j_1, \ldots, j_x$.

Let X denote the $x \times x$ submatrix obtained by taking all x rows but only the x special columns $j_1, \ldots, j_x$. It is easy to verify that X is TM. We can therefore run the SMAWK algorithm [2] on X in $O(x)$ time and obtain the column maxima of all special columns. Let $r(j)$ denote the row containing the maximum element in column j. Since M is TM, the $r(j)$ values are monotonically non-decreasing. Consequently, $r(j)$ of a non-special column j must be between $r(j_i)$ and $r(j_{i+1})$ where $j_i < j$ and $j_{i+1} > j$ are the two special columns bracketing j.

For every i, let $x_i = r(j_{i+1}) - r(j_i)$. If $x_i \le r$ then *no* column between j_i and j_{i+1} will ever be a special column. When we will query such a column j we can simply check (at query-time) the r elements of j between rows $r(j_i)$ and $r(j_{i+1})$ in $O(r)$ time. If, however, $x_i > r$, then we designate more special columns between j_i and j_{i+1}. This is done recursively on the $x_i \times (n/x)$ matrix M_i composed of rows $r(j_i), \ldots, r(j_{i+1})$ and columns $j_i, \ldots, j_{i+1}$. That is, we mark x_i evenly-spread columns of M_i as special columns, and run SMAWK in $O(x_i)$ time on the $x_i \times x_i$ submatrix X_i obtained by taking all x_i rows but only these x_i special columns. We continue recursively until either $x_i \leq r$ or the number of columns in M_i is at most r. In the latter case, before terminating, the recursive call runs SMAWK in $O(x_i + r) = O(x_i)$ time on the $x_i \times r$ submatrix X_i obtained by taking the x_i rows and *all* columns of M_i (i.e., all columns of M_i will become special).

After the recursion terminates, every column j of M is either special (in which case we computed its maximum), or its maximum is known to be in one of at most r rows (these rows are specified by the $r(\cdot)$ values of the two special columns bracketing j). Let s denote the total number of columns that are marked as special. We claim that $s = O(x \log n / \log r)$. To see this, notice that the number of columns in every recursive call decreases by a factor of at least r and so the recursion depth is $O(\log_r n) = O(\log n / \log r)$. In every recursive level, the number of added special columns is $\sum x_i$ over all $x_i's$ in this level that are at least r. In every recursive level, this sum is bounded by $2x$ because each one of the x rows of M can appear in at most two M_i's (as the last row of one and the first row of the other). Overall, we get $2x \cdot O(\log n / \log r) = O(x \log n / \log r)$.

Notice that $s = O(x \log n / \log r)$ implies that the total time complexity of the above procedure is also $O(x \log n / \log r)$. This is because whenever we run SMAWK on a $y \times y$ matrix it takes $O(y)$ time and y new columns are marked as special. To complete the construction, we go over the s special columns from left to right in $O(s)$ time and throw away (mark as non-special) any column whose $r(\cdot)$ value is the same as that of the preceding special column. This way we are left with only $O(x)$ special columns, and the difference in $r(\cdot)$ between consecutive special columns is at least 1 and at most r. In fact, it is easy to maintain $O(x)$ (and not $O(s)$) space *during* the construction by only recursing on sub matrices M_i where $x_i > 1$. We note that when $r = 1$, the eventual special columns are exactly the set of breakpoints of the input matrix M.

The final data structure is a predecessor data structure that holds the $O(x)$ special columns and their associated $r(\cdot)$ values. Upon query of some column j, we search in $\mathit{pred}(x, n)$ time for the predecessor and successor of j and obtain the two $r(\cdot)$ values. We then search for the maximum of column j by explicitly checking all the (at most r) relevant rows of column j. The query time is therefore $O(r + \mathit{pred}(x, n))$ and the space $O(x)$. □

A Linear-Space Subcolumn Data Structure

Lemma 2. *Given a $m \times n$ TM matrix, one can construct, in $O(m(\log n + \log m)/\log\log m)$ time, a data structure of size $O(m)$ that can report the maximum entry in a query column and a contiguous range of rows in $O(\log m)$ time.*

Proof. Given an $m \times n$ input matrix M we partition it into m/x matrices $M^1, M^2, \ldots, M^{m/x}$ where $x = \log m$. Every M^i is an $x \times n$ matrix composed of x consecutive rows of M. We construct the micro data structure of Lemma 1 for each M^i separately choosing $r = x^\varepsilon$ for any constant $0 < \varepsilon < 1$. This requires $O(x \log n / \log r) = O(x \log n / \log x)$ construction time per M^i for a total of $O(m \log n / \log \log m)$ time. We obtain a (micro) data structure of total size $O(m)$ that upon query (i, j) can report in $O(x^\varepsilon + \mathit{pred}(x, n)) = O(\log^\varepsilon m)$ time the maximum entry in column j of M^i.

Now, consider the $(m/x) \times n$ matrix M', where M'_{ij} is the maximum entry in column j of M^i. We cannot afford to store M' explicitly, however, using the micro data structure we can retrieve any entry M'_{ij} in $O(\log^\varepsilon m)$ time. We next show that M' is also TM.

For any pair of rows $i < j$ and any pair of columns $k < \ell$ we need to show that if $M'_{ik} \le M'_{jk}$ then $M'_{i\ell} \le M'_{j\ell}$. Suppose that $M'_{ik}, M'_{jk}, M'_{i\ell}$, and $M'_{j\ell}$ correspond to entries $M_{ak}, M_{bk}, M_{c\ell}$, and $M_{d\ell}$ respectively. We assume that $M_{ak} \le M_{bk}$ and we need to show that $M_{c\ell} \le M_{d\ell}$. Notice that $M_{ck} \le M_{ak}$ because M_{ak} is the maximal entry in column k of M^i and M_{ck} is also an entry in column k of M^i. Since $M_{ck} \le M_{ak}$ and $M_{ak} \le M_{bk}$ we have that $M_{ck} \le M_{bk}$. Since $M_{ck} \le M_{bk}$, from the total monotonicity of M, we have that $M_{c\ell} \le M_{b\ell}$. Finally, we have $M_{b\ell} \le M_{d\ell}$ because $M_{d\ell}$ is the maximal entry in column ℓ of M^j and $M_{b\ell}$ is also an entry in column ℓ of M^j. We conclude that $M_{c\ell} \le M_{d\ell}$.

Now that we have established that the matrix M' is TM, we can use the subcolumn data structure of [21] (see previous section) on M'. Whenever an entry M'_{ij} is desired, we can retrieve it using the micro data structure. This gives us the macro data structure: it is of size $O(m/x \cdot \log(m/x)) = O(m)$ and can report in $O(\log m)$ time the maximum entry of M' in a query column and a contiguous range of rows. It is built in $O(m/x \cdot (\log(m/x) + \log n) \cdot x^\varepsilon)$ time which is $O(m(\log n + \log m)/ \log \log m)$ for any choice of $\varepsilon < 1$.

To complete the proof of Lemma 2 we need to show how to answer a general query in $O(\log m)$ time. Recall that a query is composed of a column of M and a contiguous range of rows. If the range is smaller than $\log m$ we can simply check all elements explicitly in $O(\log m)$ time and return the maximum one. Otherwise, the range is composed of three parts: a prefix part of length at most $\log m$, an infix part that corresponds to a range in M', and a suffix part of length at most $\log m$. The prefix and suffix are computed explicitly in $O(\log m)$ time. The infix is computed by querying the macro data structure in $O(\log m)$ time. □

A Linear-Space Submatrix Data Structure

Theorem 1. *Given a $m \times n$ Monge matrix, one can construct, in $O((m + n)(\log n + \log m))$ time, a data structure of size $O(m + n)$ that can report the maximum entry in a query submatrix in $O(\log m + \log n)$ time.*

Proof. Recall from Section 2 that the submatrix data structure of [21] is composed of the tree T_h over the rows of M and the tree $\mathcal{B}$ over the columns of M. Every node $u \in T_h$ stores its breakpoints along with the RMQ array A^u (where $A^u[j]$ holds the value of the maximum element between the $(j-1)$'th and the

j'th breakpoints of u). If u has k breakpoints then they are computed along with A^u in $O(k + \log n)$ time: $O(k)$ to copy from the children of u and $O(\log n)$ to find the new breakpoint and to query $\mathcal{B}$. As opposed to [21], we don't use a naive RMQ data structure but instead one of the existing linear-construction constant-query RMQ data structures such as [19].

To prove Theorem 1 we begin with two changes to the above. First, we build T_h on the rows of the $(m/x) \times n$ matrix M' instead of the $m \times n$ matrix M (again, when an entry M'_{ij} is desired, we retrieve it using the micro data structure in $O(x^{\varepsilon})$ time). Second, for $\mathcal{B}$ we use the data structure of Lemma 2 applied to the transpose of M. $\mathcal{B}$'s construction requires $O(n(\log m + \log n)/\log\log n)$ time and $O(n)$ space. After this, constructing T_h (along with the A_u arrays) on M' requires $O(m/x \cdot \log(m/x)) = O(m)$ space and $O((m/x)(\log(m/x) + \log n) \cdot x^{\varepsilon}) = O(m(\log m + \log n)/\log\log m)$ time by choosing $x = \log m$ and any $\varepsilon < 1$.

Finally, we construct a data structure T_v that is symmetric to T_h but applied to the transpose of M. Notice that T_v is built on the columns of an $m \times (n/\log n)$ matrix M'' instead of the $m \times n$ matrix M. The construction of T_v, from a symmetric argument to the previous paragraph, also takes $O((m + n)(\log n + \log m)/\log\log m)$ time and $O(m + n)$ space.

We now describe how to answer a submatrix query with row range R and column range C. Let R' be the set of consecutive rows of M' whose corresponding rows in M are entirely contained in R. Let R_p be the prefix of $O(\log m)$ rows of R that do not correspond to rows of R'. Let R_s be the suffix of $O(\log m)$ rows of R that do not correspond to rows of R'. We define the subranges C', C_p, C_s similarly (with respect to columns and to M''). The submatrix query (R, C) can be covered by the following: (1) a submatrix query (R', C) in M', (2) a submatrix query (R, C') in M'', and (3) four small $O(\log m) \times O(\log n)$ submatrix queries in M for the ranges (R_i, C_j), $i, j \in \{p, s\}$. We find the maximum in each of these six ranges and return the maximum of the six values.

We find the maximum of each of the small $O(\log m) \times O(\log n)$ ranges of M in $O(\log m + \log n)$ time using the SMAWK algorithm. The maximum in the submatrix of M' is found using T_h as follows (the maximum in the submatrix of M'' is found similarly using T_v). Notice that R' is the disjoint union of $O(\log m)$ canonical nodes of T_h. For each such canonical node u, we use binary-search on u's list of breakpoints $\{j_1, j_2, \ldots, j_k\}$ to find the set $\{j_a, j_{a+1}, \ldots, j_b\}$ of u's breakpoints that are fully contained in C. Although this binary-search can take $O(\log m)$ time for each canonical node, using fractional cascading, the searches on *all* canonical nodes take only $O(\log m)$ time and not $O(\log^2 m)$. The maximum in all rows of u and all columns between j_a and j_b is found by one query to the RMQ array A^u in $O(1)$ time. Over all canonical nodes this takes $O(\log m)$ time.

The columns of C that are to the left of j_a all have their maximum element in row $r(j_a)$ of M' (that is, in one of $O(\log m)$ rows of M) . Similarly, the columns of C that are to the right of j_b all have their maximum element in row $r(j_{b+1})$ of M'. This means we have two rows of M', $r(j_a)$ and $r(j_{b+1})$, where we need to search for the maximum. We do this only after we have handled all canonical nodes. That is, after we handle all canonical nodes we have a set $A = a_1, a_2, \ldots$

of $2\log m$ rows of M' in which we still need to find the maximum. We apply the same procedure on T_v which gives us a set $B = b_1, b_2, \ldots$ of $2\log n$ columns of M'' in which we still have to find the maximum. Note that we only need to find the maximum among the elements of M that lie in rows corresponding to a row in A and in columns corresponding to a column in B. This amounts to finding the maximum of the $O(\log m) \times O(\log n)$ matrix $\bar{M}$, with $\bar{M}_{ij}$ being the maximum among the elements of M in the intersection of the x rows corresponding to row a_i of M', and of the x columns corresponding to column b_j of M''.

An argument similar to the one in Lemma 2 shows that $\bar{M}$ is Monge. Therefore we can find its maximum element using the SMAWK algorithm. We claim that each element of $\bar{M}$ can be computed in $O(1)$ time, which implies that SMAWK finds the maximum of $\bar{M}$ in $O(x)$ time.

It remains to show how to compute an element of $\bar{M}$ in constant time. Recall from the proof of Lemma 2 that M is partitioned into x-by-n matrices M^i. During the preprocessing stage, for each M^i we compute and store its upper envelope, and an RMQ array over the maximum elements in each interval of the envelope (similar to the array A^u). Computing the upper envelope takes $O(x \log n)$ time by incrementally adding one row at a time and using binary search to locate the new breakpoint contributed by the newly added row. Finding the maximum within each interval of the upper envelope can be done in $O(x \log n)$ time using the tree $\mathcal{B}$. We store the upper envelope in an atomic heap [16], which supports predecessor searches in constant time provided x is $O(\log n)$. Overall the preprocessing time is $O(m \log n)$, and the space is $O(m)$. We repeat the same preprocessing on the transpose of M.

Now, given a row a_i of M' and column b_j of M'', let $[c_a, c_b]$ be the range of x columns of M that correspond to b_j. We search in constant time for the successor $c_{a'}$ of c_a and for the predecessor $c_{b'}$ of c_b in the upper envelope of M^{a_i}. We use the RMQ array to find in $O(1)$ time the maximum element y among elements in all rows of M corresponding to a_i and columns in the range $[c_{a'}, c_{b'})$. The maximum element in columns $[c_a, c_{a'})$ and $[c_{b'}, c_b]$ is contributed by two known rows r_1, r_2. We repeat the symmetric process for the transpose of M, obtaining a maximum element y', and two columns c_1, c_2. $\bar{M}_{a_i, b_j}$ is the maximum among six values: y, y' and the four elements $M_{r_1 c_1}, M_{r_1 c_2}, M_{r_2 c_1}, M_{r_2 c_2}$. □

Notice that in the above proof, in order to obtain an element of $\bar{M}$ in constant time, we loose the $O(\log \log m)$ speedup in the construction time. This is because we found the upper envelope of each M^i. To get the $O(\log \log m)$ speedup we can obtain an element of $\bar{M}$ in $O(x^{\varepsilon})$ time using the micro data structure.

Corollary 1. *Given a $m \times n$ Monge matrix, one can construct, in $O((m + n)(\log n + \log m)/\log \log m)$ time, a data structure of size $O(m + n)$ that reports the maximum entry in a query submatrix in $O((\log m + \log n)^{1+\varepsilon})$ time for any fixed $0 < \varepsilon < 1$.*

A Linear-Space Subcolumn Data Structure for Partial Matrices. We next claim that the bounds of Lemma 2 for TM matrices also apply to *partial* TM matrices. The reason is that we can efficiently turn any partial TM matrix

M into a full TM matrix by implicitly filling appropriate constants instead of the blank entries (see full version of this paper for a simple formal proof). We can then apply the data structure of Lemma 2. However, the maximum element in a query (a column π and a range of rows R) might now appear in one of the previously-blank entries. We overcome this by first restricting R to the defined entries in the column π and only then querying the data structure of Lemma 2.

A Linear-Space Submatrix Data Structure for Partial Matrices. Given a partial matrix M, the above simple trick of replacing appropriate constants instead of the blank entries does not work for submatrix queries because the defined (i.e., non-blank) entries in a submatrix do not necessarily form a submatrix. Instead, we need a more complicated construction, which yields the following theorem. The proof with the details of the construction appears in the full version of this paper.

Theorem 2. *Given a $m \times n$ partial Monge matrix, one can construct, in $O((m+n)\log(m+n))$ time, a data structure of size $O(m+n)$ that reports the maximum entry in a query submatrix in $O((\log m + \log n)\alpha(m+n))$ time.*

Finally, for the same reasons leading to Corollary 1 we can get a $\log\log m$ speedup in the construction-time with a $\log^{\varepsilon} n$ slowdown in the query-time.

Corollary 2. *Given a $m \times n$ partial Monge matrix, one can construct, in $O((m+n)\log(m+n)/\log\log m)$ time, a data structure of size $O(m+n)$ that reports the maximum entry in a query submatrix in $O((\log m + \log n)^{1+\varepsilon}\alpha(m+n))$ time for any fixed $0 < \varepsilon < 1$.*

4 The Complexity of the Upper Envelope of a Totally Monotone Partial Matrix

In this section we prove the following theorem, stating that the number of breakpoints of an $m \times n$ TM partial matrix is only $O(m)$.

Theorem 3. *Let M be a partial $m \times n$ matrix in which the defined entries in each row and in each column are contiguous. If M is TM (i.e., for all $i < j, k < \ell$ where $M_{ik}, M_{i\ell}, M_{jk}, M_{j\ell}$ are all defined, $M_{ik} \leq M_{jk} \implies M_{i\ell} \leq M_{j\ell}$), then the upper envelope has complexity $O(m)$.*

Proof. We decompose M into *staircase* matrices. A partial matrix is staircase if the defined entries in its rows either all begin in the first column or all end in the last column. It is well known (cf. [1]) that by cutting M along columns and rows, it can be decomposed into staircase matrices $\{M_i\}$ such that each row is covered by at most three matrices, and each column is covered by at most three matrices. We describe such a decomposition in the full version of this paper [18]. In [18], we also prove the fact that, if M is a TM staircase matrix with m rows, then the complexity of its upper envelope is $O(m)$.

Let $bp(M_i)$ denote the number of breakpoints in the upper envelope of M_i. Let m_i denote the number of rows in M_i. Since each row appears in at most three M_is, $\sum_i m_i = O(m)$. The total number of breakpoints in the envelopes of all of M_is is $O(m)$ since $\sum_i bp(M_i) = \sum_i O(m_i) = O(m)$.

Consider now a partition of M into rectangular blocks B_j defined by maximal sets of contiguous columns whose defined entries are at the same set of rows. There are $O(m)$ such blocks. The upper envelope of M is just the concatenation of the upper envelopes of all the B_j's. Hence, $bp(M) = \sum_j bp(B_j) + O(m)$ (the $O(m)$ term accounts for the possibility of a new breakpoint between every two consecutive blocks). Therefore, it suffices to bound $\sum_j bp(B_j)$.

Consider some block B_j. As we mentioned above, the columns of B_j appear in the same three row-disjoint staircase matrices M_1, M_2, M_3 in the decomposition of M. The column maxima of B_j are a subset of the column maxima of M_1, M_2, M_3. Assume wlog that the indices of rows covered by M_1 are smaller than those covered by M_2, which are smaller than those covered by M_3.

The breakpoints of the upper envelope of B_j are either breakpoints in the envelope of M_1, M_2, M_3, or breakpoints that occur when the maxima in consecutive columns of B_j originate in different M_i. However, since B_j is a (non-partial) TM matrix, its column maxima are monotone. So once a column maximum originates in M_i, no maximum in greater columns will ever originate in M_j for $j < i$. It follows that the number of breakpoints in B_j that are not breakpoints of M_1, M_2, M_3 is at most two. Since there are $O(m)$ blocks, $\sum_j bp(B_j) \leq \sum_i bp(M_i) + O(m) = O(m)$. □

References

1. Aggarwal, A., Klawe, M.: Applications of generalized matrix searching to geometric algorithms. Discrete Appl. Math. 27, 3–23 (1990)
2. Aggarwal, A., Klawe, M.M., Moran, S., Shor, P., Wilber, R.: Geometric applications of a matrix-searching algorithm. Algorithmica 2(1), 195–208 (1987)
3. Aggarwal, A., Park, J.: Notes on searching in multidimensional monotone arrays. In: 29th FOCS, pp. 497–512 (1988)
4. Alstrup, S., Brodal, G.S., Rauhe, T.: New data structures for orthogonal range searching. In: 41st FOCS, pp. 198–207 (2000)
5. Amir, A., Fischer, J., Lewenstein, M.: Two-dimensional range minimum queries. In: Ma, B., Zhang, K. (eds.) CPM 2007. LNCS, vol. 4580, pp. 286–294. Springer, Heidelberg (2007)
6. Borradaile, G., Klein, P.N., Mozes, S., Nussbaum, Y., Wulff-Nilsen, C.: Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time. In: 52nd FOCS, pp. 170–179 (2011)
7. Brodal, G.S., Davoodi, P., Lewenstein, M., Raman, R., Srinivasa Rao, S.: Two dimensional range minimum queries and fibonacci lattices. In: Epstein, L., Ferragina, P. (eds.) ESA 2012. LNCS, vol. 7501, pp. 217–228. Springer, Heidelberg (2012)
8. Brodal, G.S., Davoodi, P., Rao, S.S.: On space efficient two dimensional range minimum data structures. In: de Berg, M., Meyer, U. (eds.) ESA 2010, Part II. LNCS, vol. 6347, pp. 171–182. Springer, Heidelberg (2010)
9. Burkard, R.E., Klinz, B., Rudolf, R.: Perspectives of Monge properties in optimization. Discrete Appl. Math. 70, 95–161 (1996)

10. Chan, T.M., Larsen, K.G., Pătraşcu, M.: Orthogonal range searching on the RAM. In: 27th SOCG, pp. 354–363 (2011) (revisited)
11. Chazelle, B.: A functional approach to data structures and its use in multidimensional searching. SICOMP 17, 427–462 (1988)
12. Chazelle, B., Guibas, L.J.: Fractional cascading: I. A data structuring technique. Algorithmica 1, 133–162 (1986)
13. Chazelle, B., Rosenberg, B.: Computing partial sums in multidimensional arrays. In: 5th SOCG, pp. 131–139 (1989)
14. Demaine, E.D., Landau, G.M., Weimann, O.: On cartesian trees and range minimum queries. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009, Part I. LNCS, vol. 5555, pp. 341–353. Springer, Heidelberg (2009)
15. Farzan, A., Munro, J.I., Raman, R.: Succinct indices for range queries with applications to orthogonal range maxima. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012, Part I. LNCS, vol. 7391, pp. 327–338. Springer, Heidelberg (2012)
16. Fredman, M.L., Willard, D.E.: Trans-dichotomous algorithms for minimum spanning trees and shortest paths. J. Comput. Syst. Sci. 48(3), 533–551 (1994)
17. Gabow, H., Bentley, J.L., Tarjan, R.E.: Scaling and related techniques for geometry problems. In: 16th STOC, pp. 135–143 (1984)
18. Gawrychowski, P., Mozes, S., Weimann, O.: Improved submatrix maximum queries in Monge matrices. arXiv:1307.2313 (2013)
19. Harel, D., Tarjan, R.E.: Fast algorithms for finding nearest common ancestors. SICOMP 13(2), 338–355 (1984)
20. Hoffman, A.J.: On simple linear programming problems. In: Proc. Symp. Pure Math., vol. VII, pp. 317–327. Amer. Math. Soc. (1963)
21. Kaplan, H., Mozes, S., Nussbaum, Y., Sharir, M.: Submatrix maximum queries in monge matrices and monge partial matrices, and their applications. In: 23rd SODA, pp. 338–355 (2012)
22. Klawe, M.M., Kleitman, D.J.: An almost linear time algorithm for generalized matrix searching. SIAM J. Discrete Math. 3, 81–97 (1990)
23. Monge, G.: Mémoire sur la théorie des déblais et des remblais. In: Histoire de l'Académie Royale des Science, pp. 666–704 (1781)
24. Nekrich, Y.: Orthogonal range searching in linear and almost-linear space. Comput. Geom. 42(4), 342–351 (2009)
25. Park, J.K.: A special case of the n-vertex traveling-salesman problem that can be solved in $O(n)$ time. Inf. Process. Lett. 40(5), 247–254 (1991)
26. Sharir, M., Agarwal, P.K.: Davenport-Schinzel sequences and their geometric applications. Cambridge University Press, New York (1995)
27. Tiskin, A.: Fast distance multiplication of unit-monge matrices. In: 21st SODA, pp. 1287–1296 (2010)
28. Yuan, H., Atallah, M.J.: Data structures for range minimum queries in multidimensional arrays. In: 21st SODA, pp. 150–160 (2010)

For-All Sparse Recovery in Near-Optimal Time*

Anna C. Gilbert[1,**], Yi Li[2,***], Ely Porat[3], and Martin J. Strauss[4,†]

[1] Department of Mathematics, University of Michigan
annacg@umich.edu
[2] Max-Planck Institute for Informatics
yli@mpi-inf.mpg.de
[3] Department of Computer Science, Bar-Ilan University
porately@cs.biu.ac.il
[4] Department of Mathematics and Department of EECS, University of Michigan
martinjs@umich.edu

Abstract. An *approximate sparse recovery system* in ℓ_1 norm consists of parameters k, ϵ, N, an m-by-N measurement Φ, and a recovery algorithm, $\mathcal{R}$. Given a vector, $\mathbf{x}$, the system approximates x by $\widehat{\mathbf{x}} = \mathcal{R}(\Phi\mathbf{x})$, which must satisfy $\|\widehat{\mathbf{x}} - \mathbf{x}\|_1 \le (1+\epsilon)\|\mathbf{x} - \mathbf{x}_k\|_1$. We consider the "for all" model, in which a single matrix Φ is used for all signals $\mathbf{x}$. The best existing sublinear algorithm by Porat and Strauss (SODA'12) uses $O(\epsilon^{-3} k \log(N/k))$ measurements and runs in time $O(k^{1-\alpha}N^{\alpha})$ for any constant $\alpha > 0$.

In this paper, we improve the number of measurements to $O(\epsilon^{-2} k \log(N/k))$, matching the best existing upper bound (attained by super-linear algorithms), and the runtime to $O(k^{1+\beta}\operatorname{poly}(\log N, 1/\epsilon))$, with a modest restriction that $k \le N^{1-\alpha}$ and $\epsilon \le (\log k/\log N)^{\gamma}$, for any constants $\alpha, \beta, \gamma > 0$. With no restrictions on ϵ, we have an approximation recovery system with $m = O(k/\epsilon \log(N/k)((\log N/\log k)^{\gamma} + 1/\epsilon))$ measurements. The algorithmic innovation is a novel encoding procedure that is reminiscent of network coding and that reflects the structure of the hashing stages.

1 Introduction

Sparse signal recovery is a critical data-acquisition and processing problem that arises in many modern scientific and computational applications, including signal and image processing, machine learning, data networking, and medicine [6,15]. It is a method for acquiring linear measurements or observations of a signal with a measurement matrix Φ, and an algorithm $\mathcal{D}$, for recovering the significant components of the original signal. We model this problem mathematically by assuming that we *measure* a vector $\mathbf{x}$ and collect observation $\mathbf{y} = \Phi\mathbf{x}$, then we

[*] Omitted details and proofs can be found at arXiv:1402.1726 [cs.DS].

[**] Supported in part by DARPA/ONR N66001-08-1-2065.

[***] Supported in part by NSF CCF 0743372 when the author was at the University of Michigan.

[†] Supported in part by NSF CCF 0743372 and DARPA/ONR N66001-08-1-2065.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 538–550, 2014.

Table 1. Summary of the best previous results and the result obtained in this paper. Some constant factors are omitted for clarity. "LP" denotes (at least) the runtime for a linear program of size at least N. The column "A/E" indicates whether the algorithm works in the forall (A) model or the foreach (E) model. The column "noise" indicates whether the algorithm tolerates noisy measurements. The constants c could be different in different occurrences. The lower bound on number of measurements in table above is, in fact, the best upper bound attained by super-linear algorithms.

Paper	A/E	Number of Measurements	Column sparsity/ Update time	Decode time	Approx. error	Noise
[2]	E	$k \log^c N$	$\log^c N$	$N \log^c N$	$\ell_2 \le C\ell_2$	
[4]	E	$k \log^c N$	$\log^c N$	$k \log^c N$	$\ell_2 \le C\ell_2$	
[8]	E	$\epsilon^{-1}k\log(N/k)$	$\log^c N$	$\epsilon^{-1}k \log^c N$	$\ell_2 \le (1+\epsilon)\ell_2$	Y
[5,1]	A	$k\log(N/k)$	$k\log(N/k)$	LP	$\ell_2 \le (C/\sqrt{k})\ell_1$	Y
[10]	A	$\epsilon^{-2}k \log^c N$	$\epsilon^{-2}k \log^c N$	$\epsilon^{-4}k^2 \log^c N$	$\ell_2 \le (\epsilon/\sqrt{k})\ell_1$	Y
[9]	A	$k \log^c N$	$\log^c N$	$k \log^c N$	$\ell_1 \le (C \log N)\ell_1$	Y
[14]	A	$\epsilon^{-2}k\log(N/k)$	$\epsilon^{-1}\log(N/k)$	$N\log(N/k)$	$\ell_1 \le (1+\epsilon)\ell_1$	Y
[18] (any positive integer ℓ)	A	$\ell^c\epsilon^{-3}k\log(N/k)$	$\ell^c\epsilon^{-3}\log(N/k)\log k$	$\ell^c\epsilon^{-3}k(N/k)^{1/\ell}$	$\ell_1 \le (1+\epsilon)\ell_1$	Y
This paper (any $\beta > 0$, restrictions on ϵ apply)	A	$\epsilon^{-2}k\log(N/k)$	$\epsilon^{-1}\log(N/k)$	$k^{1+\beta}(\epsilon^{-1}\log N)^c$	$\ell_1 \le (1+\epsilon)\ell_1$	Y
Lower bound 'A'		$\epsilon^{-2}k\log(N/k)$	$\epsilon^{-1}\log(N/k)$	$\epsilon^{-2}k\log(N/k)$	$\ell_2 \le (\epsilon/\sqrt{k})\ell_1$	Y

run a *recovery algorithm* and produce an approximation $\widehat{\mathbf{x}} = \mathcal{D}(\Phi, \mathbf{y})$ to $\mathbf{x}$ with the guarantee that the approximation error $\|\widehat{\mathbf{x}} - \mathbf{x}\|$ is bounded above.

More quantitatively, let us denote the length of the vector $\mathbf{x}$ by N, the sparsity parameter k, and distortion parameter ϵ. Let $\mathbf{x}_{[k]}$ denote the best k-term approximation to $\mathbf{x}$, the "heavy hitters" of $\mathbf{x}$, *i.e.*, $\mathbf{x}$ with all but the k largest-magnitude terms zeroed out. There are many different ways to assess the error of the recovery algorithm and the quality of the measurement matrix, depending on the particular application. See Table 1 for an overview of all of problem variations. In this paper, we address the ℓ_1/ℓ_1-forall problem, formally defined below.

Definition 1. *An* (ℓ_1/ℓ_1) approximate sparse recovery system *consists of parameters N, k, ϵ, an m-by-N* measurement matrix $\boldsymbol{\Phi}$*, and a* decoding algorithm $\mathcal{D}$ *that satisfy the following property: for every vector* $\mathbf{x} \in \mathbb{R}^n$*, given* $\boldsymbol{\Phi}\mathbf{x}$*, the system approximates* $\mathbf{x}$ *by* $\widehat{\mathbf{x}} = \mathcal{D}(\boldsymbol{\Phi}\mathbf{x})$*, which satisfies* $\|\widehat{\mathbf{x}} - \mathbf{x}\|_1 \le (1+\epsilon)\|\mathbf{x}_{[k]} - \mathbf{x}\|_1$.

What makes this problem challenging is that we must simultaneously keep the number of measurements small, ensure the recovery algorithm is highly efficient, and achieve a good approximation for all input vectors. If we increase the number of measurements by factors of $\log N$, it is easy to optimize the runtime. Similarly, if we severely restrict the distortion parameter ϵ, we may also increase the number of measurements by factors of ϵ. In many applications, all three quantities are important; i.e., in medical imaging applications, the measurements reflect the time a patient is observed, the recovery time drives the effectiveness of real-time imaging systems, and the recovery accuracy determines the diagnostic effectiveness of the imaging system.

Related Work. There has been considerable work on this problem in a variety of parameter settings and we summarize the results in Table 1. A number of parameter values are incommensurate: we can achieve better approximation guarantees (in the ℓ_2/ℓ_2 norm) but only in the for-each model; in the for-all signal model, we can achieve ℓ_2/ℓ_1 error guarantees. A somewhat harder problem than the one we address in this paper is the mixed-norm (or ℓ_2/ℓ_1) for-all result. In this setting, the goal is to give Φ and $\mathcal{D}$, such that, for every $\mathbf{x}$, we have

$$\|\widehat{\mathbf{x}} - \mathbf{x}\|_2 \leq \frac{\epsilon}{\sqrt{k}} \|\mathbf{x}_{[k]} - \mathbf{x}\|_1. \tag{1}$$

It is known that if $(\Phi, \mathcal{D})$ solves the ℓ_2/ℓ_1 problem it also solves the ℓ_1/ℓ_1 problem [3]. In another direction, the ℓ_2/ℓ_2 for-each problem is to give *distribution* $\mathcal{F}$ on Φ and $\mathcal{D}$, such that, for any $\mathbf{x}$, if Φ is randomly chosen subject to $\mathcal{F}$, we have

$$\Pr_{\Phi \sim \mathcal{F}} \left\{ \|\widehat{\mathbf{x}} - \mathbf{x}\|_2 \leq (1+\epsilon)\|\mathbf{x}_{[k]} - \mathbf{x}\|_2 \right\} \geq 1 - O(1).$$

The ℓ_2/ℓ_2 for-each problem with constant failure probability was solved in [8], where the authors gave an algorithm with constant-factor-optimal runtime and number of measurements. The failure probability was recently improved to exponentially small [11], but the technique is not likely to give an ℓ_1/ℓ_1 for-all result without additional logarithmic factors in the number of measurements.

The first sublinear-time algorithm in the ℓ_1/ℓ_1 for-all setting was given in [18], though that algorithm had a number of limitations.

- The runtime, while sublinear, was $\sqrt{kN}$, or, more generally, of the form $k^{1-\alpha}N^{\alpha}$ for any constant $\alpha > 0$. That algorithm did not achieve runtime polynomial in $k\log(N)/\epsilon$.
- The algorithm required a precomputed table of size $Nk^{0.2}$.
- The result was far from optimal in its dependence of the number of measurements on ϵ.

Our Results. In this work, we rectify the above limitations, assuming the (modest) restriction that $\epsilon < \log k/\log N$ and $k \leq \sqrt{N}$. We also make the measurement dependence on ϵ optimal. The best lower bound for the ℓ_1/ℓ_1 for-all problem is $\Omega(k/\epsilon^2 + (k/\epsilon)\log(\epsilon N/k))$ [16], which is also the best lower bound for the ℓ_2/ℓ_1 for-all problem. Our algorithm uses $O(k/\epsilon^2 \log(N/k))$ measurements when $\epsilon < (\log k/\log N)^\gamma$ for any constant $\gamma > 0$, which is suboptimal only by a logarithmic factor. When $k \leq \log^c N$ for some $c > 0$, the runtime is reduced to $O(k\,\mathrm{poly}(\log N, 1/\epsilon))$.

Theorem 1 (Main Theorem). *Let $\beta, \gamma > 0$. There is an approximate sparse recovery system that uses $m = O\big(\frac{k}{\epsilon}(\log\frac{N}{k})((\frac{\log N}{\log k})^\gamma + \frac{1}{\epsilon})\big)$ measurements and runs in time $O(k^{1+\beta}\,\mathrm{poly}(\log N, 1/\epsilon))$, provided that $N = \Omega(\max\{k^2, k/\epsilon^2\})$. When $\epsilon = O\big((\frac{\log k}{\log N})^\gamma\big)$, the number of measurements $m = O(k/\epsilon^2\log(N/k))$.*

Overview of Techniques. Our overall approach builds on [18] and [11] with several critical innovations. In Fig. 1 is a framework which captures both the algorithm in [18] and the algorithm in this paper.

First, we describe the encoding procedure at a high level. Initially each $i \in [N]$ is associated with a unique message $\mathbf{m}_i$, which is encoded to a longer message $\mathbf{m}'_i$. In [18] this encoding is trivial, namely, $\mathbf{m}'_i = \mathbf{m}_i$; while in our work it is a more complicated procedure (see Fig. 3). The first hash assigns one of B buckets to each $i \in [N]$, while maintaining the original index i; the *aggregation* step sums each bucket. There are $\log(N/k)/(\epsilon \log(B/k))$ repetitions. The index i in each repetition is now associated with a chunk of $\mathbf{m}'_i$. In [18], the aggregated buckets are hashed into (k/ϵ) buckets and there are $\log(B/k)/\epsilon$ repetitions. Thus, altogether, there are $O(\epsilon^{-3} k \log(N/k))$ measurements. In our work, there are only $\log(B/k)$ repetitions, saving a factor of $1/\epsilon$, so the total number of measurements is $O(\epsilon^{-2} k \log(N/k))$.

The *identification* portion of the recovery algorithm is shown in Fig. 2. To recover the identity of heavy hitters, the algorithm reads off the measurements and recovers the message chunk associated with each bucket. This message chunk is supposed to be associated with the heavy hitter in the bucket. Then, all B buckets are examined exhaustively. The pre-image of each heavy bucket under the first hash is determined, in [18], from a look-up table and searched exhaustively. In our work, this is done by the decoding procedure illustrated in Fig. 4. We encode the "linking information" into the message chunks so that we can collect across the repetitions enough heavy buckets which contain the same heavy hitter i (whose actual value is unknown at this stage of the algorithm). Thus, we obtain a (small) fraction of $\mathbf{m}'_i$, which is sufficient for the Parvaresh-Vardy decoding algorithm to produce the exact $\mathbf{m}_i$, whence we recover the value of i immediately.

The *estimation* portion of the recovery algorithm estimates the coefficient at each of those candidate positions by reading the aggregated bucket value of the corresponding heavy buckets at the first hash level.

Putting these pieces together, we have a *weak recovery system*, which identifies all but $k/2$ of the heavy hitters. We then repeat with smaller (easier) sparsity parameter $k/2 < k$ and smaller (harder) distortion parameter $(3/4)\epsilon < \epsilon$, resulting in a number of measurements whose leading term is $(k/2)(4/3\epsilon)^2 = (8/9)k/\epsilon^2 < k/\epsilon^2$. Summing the geometric progression gives the result we need. Finally, we note that our algorithm works (deterministically) with any unbalanced expander having the appropriate properties.

Encoding and Decoding Details. See Fig. 3 and 4 for a detailed illustration of these steps. For each message $\mathbf{m}$, the Parvaresh-Vardy code encodes it into a longer message $\mathbf{m}'$, which automatically exhibits a chunk structure, so that if a few number of the chunks are correct, the original $\mathbf{m}$ will be recovered. Suppose there are D chunks. Now, choose a d-regular expander graph G (d is a constant) on D nodes such that after removing $O(D)$ nodes from G, the remaining graph still contains an expander of size $\Omega(D)$. For the i-th chunk of $\mathbf{m}'$, append to it the information of the neighbours of the i-th vertex in G. Then we apply Reed-Solomon to protect the appended chunks.

To decode, we first recover the appended message chunks. The two-layer hash guarantees that for the same heavy hitter, at most $O(D)$ of them will be wrong and the remaining ones are all correct. Now, consider a breadth-first search from a correct message chunk (whose "linking information" is therefore correct). By the special property of the expander graph G, we shall be able to visit all nodes (i.e., all corresponding message chunks) of a smaller expander graph of size $\Omega(D)$ in $\log D$ steps. This small fraction of good message chunks of $\mathbf{m}'$ will enable the PV code to recover the original message $\mathbf{m}$ successfully. Recall that d is a constant, the total number of vertices visited is $O(d^{\log D}) = O(\text{poly}(D)) = O(\text{poly}(\log N))$ for appropriate D. This enables a sublinear recovery time.

Our Contributions

- We give an algorithm for sparse recovery in the for-all setting, under a modest restriction on the distortion factor ϵ, having the number of measurements that matches the best upper bound, attained by super-linear algorithms; e.g., [14], and optimal in runtime up to a power.
- We conjecture that our algorithm can be extended from the 1-norm to the mixed norm guarantee and that the restriction on ϵ can be weakened or eliminated. Thus our algorithm may be a stepping stone to the final algorithm.
- Our work is not the first to consider list recovery. Indyk et al. introduces the idea in the context of combinatorial group testing [13]. The idea of list recovery is also used in [11], where the list decoding, however, would affect the hashing and the hashing was thus required to be sufficiently random. In our algorithm, the messages $\{\mathbf{m}_i\}$ are independent of the hashing, which enables us to obtain a better result.
- Finally, our encoding/decoding techniques are reminiscent of network coding and may have other contexts for soft-decoding or network coding.

Paper Organization. In Section 2 we review some properties of expanders. In Section 3, we show that provided with good identification results, unbalanced expanders with appropriate properties will give a weak system. Our construction of weak system culminates in Section 4, where we show how to achieve good identification via message encoding and decoding. Then we build the overall algorithm on the weak system in Section 5 and close with a short discussion in Section 6.

2 Preliminaries

Our main algorithm will be built on regular graph expanders and unbalanced bipartite expanders. In Let n, m, d, ℓ be positive integers and ϵ, κ be positive reals. The following two definitions are adapted from [12].

Definition 2 (Expander). *An (n, ℓ, κ)-expander is a graph $G(V, E)$, where $|V| = n$, such that for any set $S \subseteq V$ with $|S| \leq \ell$ it holds that $|\Gamma(S)| \geq \kappa|S|$.*

Definition 3 (Bipartite Expander). *An $(n, m, d, \ell, \epsilon)$-bipartite expander is a d-left-regular bipartite graph $G(L \cup R, E)$ where $|L| = n$ and $|R| = m$ such that for any $S \subseteq L$ with $|S| \leq \ell$ it holds that $|\Gamma(S)| \geq (1 - \epsilon)d|S|$, where $\Gamma(S)$ is the neighbour of S (in R).*

Theorem 2 ([7]). *For all sufficiently large n and even d, there exists a d-regular expander G such that $|V(G)| = n$ and $\lambda(G) \leq C\sqrt{d}$ for some absolute constant $C > 0$, where $\lambda(G)$ denote the second largest eigenvalue, in absolute value, of G.*

Theorem 3 ([19]). *Let G be a δ-regular expander of n nodes such that $\lambda(G) \leq C\sqrt{\delta}$, where δ is a (sufficiently large) constant. There exist absolute constants $\alpha, \zeta > 0$ and $\kappa > 1$ such that after removing an arbitrary set of at most ζn nodes from G, the remaining graph contains a subgraph G' such that $|V(G')| \geq \alpha n$ and G' is a $(|V(G')|, n/2, \kappa)$ graph expander.*

The rest of the section concerns hashing. Informally, we say an (N, B, d) *(one layer) hashing scheme*[1] is to hash N elements into B buckets and repeat d times independently. Each instance of such a hashing scheme induces a d-left-regular bipartite graph with Bd right nodes. An (N, B_1, d_1, B_2, d_2) (two-layer) hashing scheme[2] is to hash N elements into B_1 buckets and repeat d_1 times (those buckets will be referred to as first-layer buckets) and in each of the d_1 repetitions, hash B_1 elements into B_2 buckets and repeat d_2 times (those buckets will be referred to as second-layer buckets). Each instance of such a hashing scheme induces a $d_1 d_2$-left-regular bipartite graph with $B_2 d_1 d_2$ right nodes.

We note that bipartite expander graphs can be used as hashing schemes because of their isolation property.

Definition 4 (Isolation Property). *An $(n, m, d, \ell, \epsilon)$-bipartite expander G is said to satisfy the (ℓ, η, ζ)-isolation property if for any set $S \subset L(G)$ with $|S| \leq \ell$, there exists $S' \subset S$ with $|S'| \geq (1 - \eta)|S|$ such that for all $x \in S'$ it holds that $|\Gamma(x) \setminus \Gamma(S \setminus \{x\})| \geq (1 - \zeta)d$.*

3 Weak System

We decompose a signal $\mathbf{x}$ into two parts of disjoint support, $\mathbf{x} = \mathbf{y} + \mathbf{z}$, where $\mathbf{y}$ has small support and $\mathbf{z}$ small norm (by normalization we may assume that $\|\mathbf{z}\|_1 \leq 3/2$). We call $\mathbf{y}$ the *head* and $\mathbf{z}$ the *tail*. We aim to recover the elements in $\mathbf{y}$. Introduced in [18], a *weak system* takes an additional input, some set I of indices (called the candidate set), and tries to estimate $\mathbf{x}_i$ for $i \in I$, hoping to recover some head items with estimate error dependent on $\|\mathbf{z}\|_1$. It is shown in [18] that when I contains the entire head, we can always recover a good fraction of the head. In fact, we only need I to contain a good fraction of the head instead of the entire head, with a slight modification of the original proof.

Definition 5 (Weak System). *A* Weak system *consists of parameters N, s, η, ζ, an m-by-N* measurement matrix *$\boldsymbol{\Phi}$, and a* decoding algorithm *$\mathcal{D}$, that satisfy the following property: For any $\mathbf{x} \in \mathbb{R}^N$ that can be written as $\mathbf{x} = \mathbf{y} + \mathbf{z}$, where $|\operatorname{supp}(\mathbf{y})| \leq s$ and $\|\mathbf{z}\|_1 \leq 3/2$, given the measurements $\boldsymbol{\Phi}\mathbf{x}$ and a subset*

[1] When N is clear from the context, we simply write (B, d) hashing scheme.

[2] When N is clear from the context, we simply write (B_1, d_1, B_2, d_2) hashing scheme.

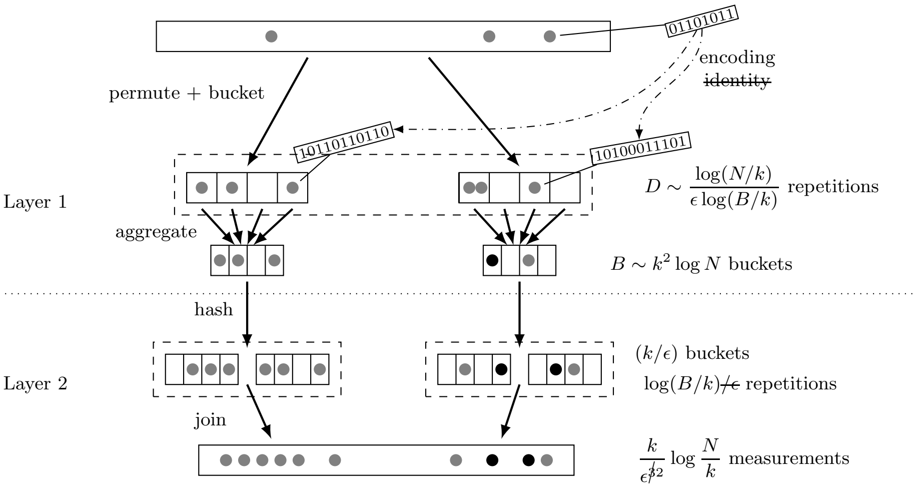

01101011
encoding
~~identity~~
permute + bucket
10110110110
10100011101
Layer 1
$D \sim \frac{\log(N/k)}{\epsilon\log(B/k)}$ repetitions
aggregate
$B \sim k^2 \log N$ buckets
hash
Layer 2
(k/ϵ) buckets
$\log(B/k)$~~/ε~~ repetitions
join
$\frac{k}{\epsilon^2} \log \frac{N}{k}$ measurements

Fig. 1. Algorithm to generate the measurements. Darker spots indicate a bigger value of the bucket/measurement. Strikethroughs are used to show where our approach or our object sizes differ from [18].

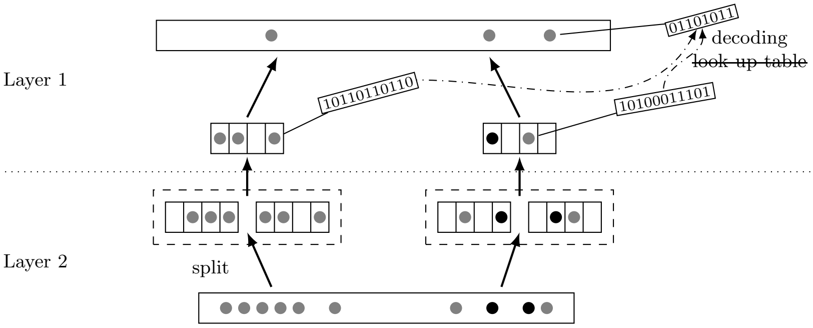


Fig. 2. Algorithm to recover from the measurements

$I \subseteq [N]$ such that $|I \cap \operatorname{supp}(\mathbf{y})| \geq (1 - \zeta/2)|\operatorname{supp}(\mathbf{y})|$, the decoding algorithm $\mathcal{D}$ returns $\widehat{\mathbf{x}}$, such that $\mathbf{x}$ admits the following decomposition: $\mathbf{x} = \widehat{\mathbf{x}} + \widehat{\mathbf{y}} + \widehat{\mathbf{z}}$, where $|\operatorname{supp}(\widehat{\mathbf{x}})| = O(s)$, $|\operatorname{supp}(\widehat{\mathbf{y}})| \leq \zeta s$, and $\|\widehat{\mathbf{z}}\|_1 \leq \|\mathbf{z}\|_1 + \eta$.

Theorem 4 (Weak). *Suppose that Φ is the adjacency matrix of an $(N, Bd, d, 4s, \eta)$-bipartite expander such that (a) $d = O(\frac{1}{\eta\zeta^2} \log \frac{N}{s})$ and $B = O(\frac{d}{\zeta\eta})$ and (b) it satisfies $(O(k/\epsilon), \epsilon, \zeta)$-isolation property. With appropriate instantiations of constants, Algorithm 1 yields a correct Weak system running in time $O(\frac{|I|}{\eta\zeta^2} \log \frac{N}{s})$.*

To complete the construction of a Weak system, it remains to show that a bipartite expander as required by Theorem 4 exists. Indeed, it can be attained

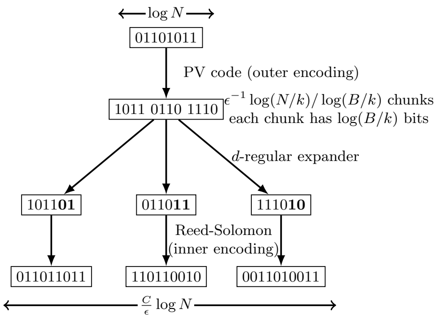


Fig. 3. Encoding scheme. The PV code exhibits a chunk structure. Suppose that there are D chunks. Choose a d-regular expander on D vertices as desired. For the i-th chunk of the PV code, append to it the information of the neighbours of the i-th vertex in the expander. Then apply Reed-Solomon to each appended message chunk. Recall that $k = O(\sqrt{N})$ so $\log(N/k) = \Theta(\log N)$.

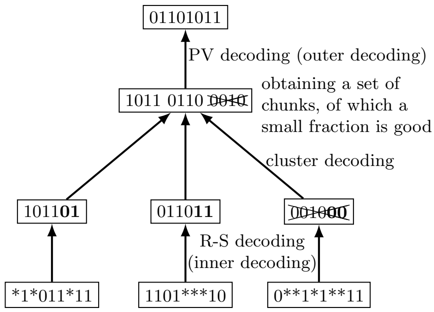


Fig. 4. Decoding scheme. The asterisks in the bottom layer indicates corrupted measurements (owing to collision or noise). The Reed-Solomon decoding either recovers the message chunk (with linking information) or produces a useless one (crossed out). Then the clustering procedure finds a set of chunks, of which a small fraction is good. This is sufficient for the Parvaresh-Vardy decoding to succeed.

Algorithm 1. Weak system

```
Input: $N$, $s$, $\boldsymbol{\Phi}$ (adjacency matrix of a $d$-left-regular expander $G$), $\boldsymbol{\Phi}\mathbf{x}$, and $I$
Output: $\widehat{\mathbf{x}}$
for $j \leftarrow 1$ to $d$ do
    for each $i \in I$ do
        $\mathbf{x}_i^{(j)} \leftarrow \operatorname{median}_{u \in \Gamma(\{i\})} \sum_{(u,v)\in E} \mathbf{x}_u$        /* each sum is an element of input $\boldsymbol{\Phi}\mathbf{x}$ */
for each $i \in I$ do
    $\mathbf{x}'_i \leftarrow \operatorname{median}_{1\le j\le d} \mathbf{x}_i^{(j)}$
$\widehat{\mathbf{x}} \leftarrow$ top $O(s)$ elements of $\mathbf{x}'$
return $\widehat{\mathbf{x}}$
```

by both one-layer and two-layer hashing schemes, with appropriate parameters. We state the two-layer result below as our construction will use it.

Lemma 1. *Let $\epsilon \in (0, \frac{1}{4})$, $\alpha > 1$, $k \geq 1$ and $N = \Omega\big(\max\{\frac{k}{\epsilon^2}, k^2\}\big)$. Consider a two-layer (B_1, d_1, B_2, d_2) hashing scheme with $B_1 = \Omega\big(\frac{k}{\zeta^\alpha \epsilon^{2\alpha}}\big)$, $d_1 = \Omega\big(\frac{\alpha}{\alpha-1} \cdot \frac{1}{\zeta\epsilon}\frac{\log(N/k)}{\log(B/k)}\big)$, $B_2 = \Omega\big(\frac{k}{\zeta\epsilon}\big)$ and $d_2 = \Omega\big(\frac{1}{\zeta}\log\frac{B_1}{k}\big)$. With probability $\geq 1 - N^{-\Omega(1)}$, such a two-layer hashing scheme gives an $(B_2 d_1 d_2, d_1 d_2, 4k, \epsilon)$ bipartite expander with the $(O(k/\epsilon), \epsilon, \zeta)$-isolation property.*

4 Identification of Heavy Hitters

In the previous section, we showed how to estimate all candidates in a candidate set I quickly. The main bottleneck in a highly efficient algorithm is finding a non-trivial set $I \subset [N]$ of candidates which we address in this section.

Algorithm 2. Encoding/Decoding paradigm.

```
// Encoding with $(B_1, d_1, B_2, d_2)$ hashing scheme
for $i = 1$ to $N$ do
    Break: Break the information of $i$ into $d_1$ chunks
    Outer encoding: Encode the chunks with cluster info (from a regular expander graph)
    and against errors, getting $\{\mathbf{m}_{i,j}\}_{j=1}^{d_1}$
for $j = 1$ to $d_1$ do
    Inner encoding: Encode $\mathbf{m}_{i,j}$, for $i \in [N]$
// Decoding with $(B_1, d_1, B_2, d_2)$ hashing scheme
for $j = 1$ to $d_1$ do
    // length $B_1$, $(B_2 d_2)$-measurement Sparse Recovery Channel
    Inner decoding: Recover $\widehat{\mathbf{m}}_j$ in the Weak List sense
    Record Side Info: Tag each element of $\widehat{\mathbf{m}}_j$ with $j$
Outer decoding: From $\widehat{\mathbf{m}} = \bigcup_j \widehat{\mathbf{m}}_j$'s, find chunk clusters and correct errors; produce $I$
```

The overall strategy is as follows. Using the two-layer hashing scheme (B_1, d_1, B_2, d_2), we expect that a heavy hitter dominates the first-layer buckets where it lands in $\Omega(d_1)$ repetitions. In each of these repetitions, it is a heavy hitter in a signal of length B_1, and we expect to recover it using the Weak algorithm applied to the signal of length B_1 with $I = [B_1]$. After finding the heavy buckets in each repetition, the remaining problem is to extract the position of a heavy hitter i from the $\Omega(d_1)$ repetitions that contain i. To do this, we encode the index i in such a way that if we recover the buckets containing i in enough repetitions we shall be able to reconstruct i. To that end, we introduce the following model of weak list recovery in the *sparse recovery channel.*

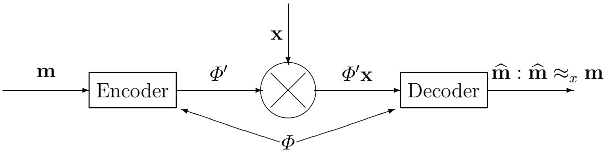
Definition 6. *The* (m, N, s) Sparse Recovery Channel *takes an* m*-by-*N *matrix* $\boldsymbol{\Phi}$ *as input, chooses a signal* $\mathbf{x}$ *with decomposition* $\mathbf{x} = \mathbf{y} + \mathbf{z}$ *with* $|\operatorname{supp}(\mathbf{y})| \leq s$ *and* $\|\mathbf{z}\|_1 \leq O(1)$*, and outputs* $\boldsymbol{\Phi}\mathbf{x}$.

Note that $\mathbf{x}$ may depend on $\boldsymbol{\Phi}$. Also note that *any* signal may be chosen by the channel and normalized so that $\|\mathbf{z}\|_1 \leq 3/2$. It will be convenient to assign the normalization at this point to match the Weak system (Defintion 5). Next, we define the *Weak Recovery Criterion* appropriate for this channel. See Fig. 5.

Definition 7 (Weak List Recovery Criterion). *Fix parameters* m, N, s, ϵ. *Let* $\mathbf{m}$ *be a vector of* β*-bit messages and* $i \in [N]$. *Suppose* $\widehat{\mathbf{m}}$ *is a list of possible index-message pairs. We say that* $\widehat{\mathbf{m}}$ *is* correct in the List Weak sense *if, for at least* $|\operatorname{supp}(\mathbf{y})| - s/8$ *indices* i *in* $\operatorname{supp}(\mathbf{y})$*, we have* $(i, \mathbf{m}_i) \in \widehat{\mathbf{m}}$.

The encoding/decoding scheme is given in Algorithm 2. We break each message $\mathbf{m}_i$ (which could be much longer than $\log N$ bits) associated with position i into d_1 chunks, $\mathbf{m}_{i,1}, \ldots, \mathbf{m}_{i,d_1}$. Now in the j-th repetition of the d_1 repetitions, we obtain a signal $\widetilde{\mathbf{x}}$ of length B. Each $\widetilde{\mathbf{x}}_\ell$ is associated with a message that can be viewed as a weighted sum of $\mathbf{m}_{i,j}$ for positions i hashed into bucket ℓ. If a heavy hitter i is isolated in bucket ℓ and the bucket noise is mild, this weighted sum would be approximately $\mathbf{m}_{i,j}$, and we expect to recover $\mathbf{m}_{i,j}$ from the second-layer hashing, with inner encoding and decoding.

The following lemma is a simple case to illustrate our idea of encoding, in which we show how to code $\beta = \log(B/k)$ bits in the length-B Sparse Recovery



x
m
Encoder
Φ'
$\Phi'\mathbf{x}$
Decoder
$\widehat{\mathbf{m}} : \widehat{\mathbf{m}} \approx_x \mathbf{m}$
Φ

Fig. 5. Sparse recovery channel. The encoder and decoder agree on some matrix $\boldsymbol{\Phi}$. The encoder takes messages $\mathbf{m}$ and produces a measurement matrix $\boldsymbol{\Phi}'$ based on $\mathbf{m}$ and $\boldsymbol{\Phi}$. The channel is fed with $\boldsymbol{\Phi}'$ and $\mathbf{x}$ and produces $\boldsymbol{\Phi}'\mathbf{x}$, from which the decoder tries to recover $\widehat{\mathbf{m}}$ in the sense of weak list recovery.

Channel and how to recover the messages associated with $\Omega(k)$ heavy hitters in the length B signal in time approximately B.

Lemma 2. *Let $B = \Omega(k)$ and $\beta = O(\log(B/k))$. There is a coding scheme for the length-B m-measurememt Sparse Recovery Channel for $m = O(k/\epsilon \log(B/k))$ in the weak list recovery sense in which decoding runs in time $O(B \log^3(B/k))$.*

The rest of this section is devoted to an expander-based coding scheme for the most difficult part of identification, that is, to match $\mathbf{m}_{i,j}$ with $\mathbf{m}_{i,j'}$ $(j \neq j')$ in order to find enough fraction of $\mathbf{m}_i$ in the end. We resolve this by embedding "linking information" in $\mathbf{m}_{i,j}$.

Parameters. We assume that the constants $\beta, \gamma > 0$ are fixed and the parameters B_1, d_1, B_2, d_2 are as in Lemma 1 such that $B_1 = \Omega\big((\frac{k}{\epsilon^2})^{1+\beta} \log \frac{N}{k}\big)$. Let G be a graph of d_1 nodes with constant degree δ that satisfies Theorem 2, and α, ζ, κ be constants provided by Theorem 3 when applied to G. Without loss generality we can assume that $\alpha \leq 1/2$. Let $c \leq m$ be positive integer constants, h a positive integer and $\epsilon = O\big((\frac{\alpha}{m})^{\frac{m}{m-c}} (\frac{\log(B_1/k)}{\log(N/k)})^{\gamma}\big)$. Adjust the hidden constants together with c, m and h appropriately (depending on β and γ) such that (a) $B_1 > d_1$; (b) $(h-1)m \log_{B_1} N < \alpha d_1$; (c) $(\alpha d_1 - (h-1)m \log_{B_1} N) \cdot h^m > d_1^c$ and (d) $c \geq \log \delta / \log \kappa$.

Encoding. We shall use Reed-Solomon for inner encoding. Next, we define our outer coding, which uses the Parvaresh-Vardy code [17]. Take N disconnected copies of G and call the union G_N, where each node is indexed by a pair $(i, r) \in [N] \times [d_1]$. Let $\mathbb{F}$ be a field such that $|\mathbb{F}| = \Theta(B_1)$ is a power of 2 and $E(x)$ be an irreducible monic polynomial over $\mathbb{F}$ such that $\deg E(x) = \log_{B_1} N$. View each $i \in [N]$ as a polynomial f_i over $\mathbb{F}$ with degree $\log_{B_1} N - 1$. For each $(i, r) \in G_N$, associate with it an element $p(i, r) \in \mathbb{F}^{m+1}$ defined by

$$p(i,r) = (x_{i,r}, f_i(x_{i,r}), (f_i^h \bmod E)(x_{i,r}), \ldots, (f_i^{h^{m-1}} \bmod E)(x_{i,r})),$$

where $x_{i,r} \in \mathbb{F}$ are distinct for all r. This is possible because of Property (a).

Attach to a node (i, r) a message $\mathbf{m}_{i,r}$ containing $p(i, r)$ as well as $H(i, v_1(r))$,..., $H(i, v_\delta(r))$, where $v_1(r), \ldots, v_\delta(r)$ are the neighbours of r in G and $H(i, j) \in [B_1]$ gives the bucket index where i lands in the j-th outer hashing repetition. It is clear that $\mathbf{m}_{i,r}$ has $\Theta(\log B_1) = O(d_2)$ bits and therefore we can encode it in d_2 hash repetitions, see Lemma 2.

Decoding. In each of the d_1 repetitions, we shall recover $O(k/\epsilon)$ heavy buckets and thus obtain $O(k/\epsilon)$ nodes with their messages. Even when the messages are recovered correctly, we only know that a message corresponds to $\mathbf{m}_{i,r}$ for some $i \in [N]$ and we do not know which i it is. As mentioned in the introduction, we wish to collect enough $p(i,r)$ for different values of r and the same i. To this end, we do clustering as follows.

Suppose that there are k heavy hitters at position $i_1, \dots, i_k$. Let $\tilde{G}$ be a graph of $d_1 \times O(k/\epsilon)$ nodes, arranged in a $d_1 \times O(k/\epsilon)$ grid. For now we assume an ideal situation where the messages are recovered correctly for each heavy hitter i in all d_1 repetitions (which implies no collisions and small bucket noise). Each message has the form $(p(i,r), h_1, \dots, h_\delta)$, where $h_j = H(i, v_j(r))$ $(j \in [\delta])$. Add an arc $(i,r) \to (h_j, v_j(r))$ for each $j \in [\delta]$.

Since the messages are recovered correctly, the graph $\tilde{G}$ will contain several disjoint copies of the expander graph G, say $G_{i_1}, \dots, G_{i_k}$. There will be arcs incoming to G_{i_j} from nodes not in any G_{i_j}, but there will be no outgoing arcs from G_{i_j}. In this case, we can recover each G_{i_j} perfectly, and collect the full set $\{\mathbf{m}_{i_j,r}\}_{r=1}^{d_1}$ and thus recover i_j. In this case, the columns $i_1, \dots, i_k$ are exact copies of the expander graph G.

The heavy hitters may not, however, be recovered in some repetitions and the messages could be seriously corrupted. Adding arcs introduces two kinds of errors: (i) We lose a node in G_{i_j} because the heavy hitter i_j is not recovered in that repetition; (ii) We connect a node in G_{i_j} to a node in $G_{i_{j'}}$ $(j \neq j')$, owing to errorous message. We know that for a heavy hitter i, only a few messages $\{\mathbf{m}_{i,r}\}_r$ are ruined and the i-th column of G_N will contain a large connected subgraph G' of G, by Theorem 3. Hence, if we start a breadth-first search at an appropriate node with depth $c \log_\delta d_1$, the whole G' will be visited. In other words, we shall obtain a large set of $\{p(i,r)\}$, a small number of which will be associated with the same i, but it is sufficient to extract f_i using a good error-correcting code such as the Parvaresh-Vardy code that allows us to recover the codeword from a large fraction of errors. Without identifying the "appropriate nodes", we perform a breadth-first search at every node in $\tilde{G}$.

Guarantee. We show that the system described above meets the aforementioned guarantee, using Property (b)–(d).

Theorem 5. *Let $\beta, \gamma > 0$. The encoding and decoding strategy above are correct in the sense of weak list recovery, against the channel described in that section. It uses $O(\epsilon^{-2} s \log(N/s))$ measurements and runs in time $O(s^{1+\beta} \operatorname{poly}(\log N, 1/\epsilon))$, provided that $N = \Omega(\max\{s^2, s/\epsilon^2\})$ and $\epsilon = O\big((\log s / \log N)^\gamma\big)$.*

5 Overall Algorithm

The construction of an approximate recovery system from a weak system is similar to existing works [8,18]. We use Theorem 5 for identification and Theorem 4 for estimation. Below we simply restate our main theorem result from Theorem 1 in a slightly different form.

Theorem 6. *Let $\beta, \gamma > 0$. There is an approximate recovery system that uses $O(\epsilon^{-2} k \log(N/k))$ measurements and runs in time $O(k^{1+\beta} \operatorname{poly}(\log N, 1/\epsilon))$, provided that $N = \Omega(\max\{k^2, k/\epsilon^2\})$ and $\epsilon = O((\log k / \log N)^\gamma)$.*

We remark that (a) the constants in big O-notations and the power in $\operatorname{poly}(\log N, 1/\epsilon)$ depend on β and γ; (b) the constraint that $k = O(\sqrt{N})$ could be relaxed to $k = O(N^{1-\alpha})$ for any $\alpha > 0$, the hidden constants will depend on α; (c) the factor $k^{1+\beta}$ in the runtime is due to our choice of $B_1 = \Omega((k/\epsilon^2)^{1+\beta} \log(N/k))$ such that $\log B_1 = O(\log(B_1/k)) = O(d_2)$. When $k \leq \operatorname{poly}(\log N)$, it suffices to choose $B_1 = \Theta(k \log(N/k)/\epsilon^{2(1+\beta)})$, leading to runtime $O(k \operatorname{poly}(\log N, 1/\epsilon))$; (d) for large ϵ we can take $d_1 = (\log(N/k)/\log(B_1/k))^{1+\alpha}$ for an arbitrary $\alpha > 0$, which gives an algorithm which uses more measurements $O(k \log^{1+\alpha}(N/k)/\epsilon^2)$ but suboptimal by only a logarithmic factor.

6 Discussions

At the core part of this paper lies the following list recovery problem: Suppose that there are $d_1 = \frac{1}{\epsilon} \cdot \frac{\log(N/k)}{\log(B/k)}$ lists $L_1, \ldots, L_{d_1}$ with $|L_i| = O(k/\epsilon)$ for all $i \in [d_1]$, we want to recover all possible codewords $c = (c_1, \ldots, c_{d_1})$ such that $c_i \in L_i$ for at least $\Omega(d_1)$ different i's in $[d_1]$. We used an expander structure to reduce the problem to kd_1/ϵ subproblems, each of which has a smaller number of nodes. It is natural to be tempted to apply Parvaresh-Vardy code directly without the expander structure. Indeed it works for some configurations of k and ϵ with a runtime of $O(k \operatorname{poly}(\log N, 1/\epsilon))$, but only for small k and ϵ. A direct application already fails even for $k = \exp(\sqrt{\log n})$. The runtime resulting from a direct application is also better for very small k, however, obtaining the precise range is difficult and beyond the scope of our work, as it relies on the precise complexity of factorizing a polynomial, which is not explicit in the literature. We also remark that the Parvaresh-Vardy code in the outer coding and the Reed-Solomon code in the inner coding could be replaced with other codes of similar parameters, or better parameters, which would lead to an improvement of the algorithm.

References

1. Candès, E., Romberg, J., Tao, T.: Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. IEEE T. Info. Theory 52(2), 489–509 (2006)
2. Charikar, M., Chen, K., Farach-Colton, M.: Finding frequent items in data streams. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 693–703. Springer, Heidelberg (2002)
3. Cohen, A., Dahmen, W., Devore, R.: Compressed sensing and best k-term approximation. J. Amer. Math. Soc., 211–231 (2009)

4. Cormode, G., Muthukrishnan, S.: Combinatorial algorithms for compressed sensing. In: SIROCCO, pp. 280–294 (2006)
5. Donoho, D.L.: Compressed sensing. IEEE T. Info. Theory 52(4), 1289–1306 (2006)
6. Duarte, M.F., Davenport, M.A., Takhar, D., Laska, J.N., Kelly, K.F., Baraniuk, R.G.: Single-pixel imaging via compressive sampling. IEEE Signal Processing Magazine 25(2), 83–91 (2008)
7. Friedman, J., Kahn, J., Szemerédi, E.: On the second eigenvalue of random regular graphs. In: STOC, pp. 587–598 (1989)
8. Gilbert, A., Li, Y., Porat, E., Strauss, M.: Approximate sparse recovery: Optimizing time and measurements. SIAM J. Comput. 41(2), 436–453 (2012)
9. Gilbert, A., Strauss, M., Tropp, J., Vershynin, R.: Algorithmic linear dimension reduction in the ℓ_1 norm for sparse vectors. In: Allerton (2006)
10. Gilbert, A., Strauss, M., Tropp, J., Vershynin, R.: One sketch for all: fast algorithms for compressed sensing. In: ACM STOC, pp. 237–246 (2007)
11. Gilbert, A.C., Ngo, H.Q., Porat, E., Rudra, A., Strauss, M.J.: ℓ_2/ℓ_2-foreach sparse recovery with low risk. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) ICALP 2013, Part I. LNCS, vol. 7965, pp. 461–472. Springer, Heidelberg (2013)
12. Guruswami, V., Umans, C., Vadhan, S.: Unbalanced expanders and randomness extractors from Parvaresh-Vardy codes. J. ACM 56(4), 20:1–20:34
13. Indyk, P., Ngo, H.Q., Rudra, A.: Efficiently decodable non-adaptive group testing. In: SODA, pp. 1126–1142 (2010)
14. Indyk, P., Ruzic, M.: Near-optimal sparse recovery in the ℓ_1 norm. In: FOCS, pp. 199–207 (2008)
15. Lustig, M., Donoho, D., Pauly, J.M.: Sparse MRI: The application of compressed sensing for rapid MR imaging. Magn. Reson. Med. 58(6), 1182–1195 (2007)
16. Nelson, J., Nguy˜ên, H.L., Woodruff, D.P.: On deterministic sketching and streaming for sparse recovery and norm estimation. In: Gupta, A., Jansen, K., Rolim, J., Servedio, R. (eds.) APPROX/RANDOM 2012. LNCS, vol. 7408, pp. 627–638. Springer, Heidelberg (2012)
17. Parvaresh, F., Vardy, A.: Correcting errors beyond the guruswami-sudan radius in polynomial time. In: FOCS, pp. 285–294 (2005)
18. Porat, E., Strauss, M.J.: Sublinear time, measurement-optimal, sparse recovery for all. In: SODA, pp. 1215–1227 (2012)
19. Upfal, E.: Tolerating linear number of faults in networks of bounded degree. In: PODC, pp. 83–89 (1992)

Families with Infants: A General Approach to Solve Hard Partition Problems*

Alexander Golovnev[1], Alexander S. Kulikov[2], and Ivan Mihajlin[2,3]

[1] New York University
[2] St. Petersburg Department of Steklov Institute of Mathematics
[3] St. Petersburg Academic University

Abstract. We introduce a general approach for solving partition problems where the goal is to represent a given set as a union (either disjoint or not) of subsets satisfying certain properties. Many NP-hard problems can be naturally stated as such partition problems. We show that if one can find a large enough system of so-called families with infants for a given problem, then this problem can be solved faster than by a straightforward algorithm. We use this approach to improve known bounds for several NP-hard problems (the traveling salesman problem, the graph coloring problem, the problem of counting perfect matchings) on graphs of bounded average degree, as well as to simplify the proofs of several known results.

1 Introduction

In this paper we consider algorithms for three classical hard problems: the traveling salesman problem, the chromatic number problem, and the problem of counting perfect matchings. $O^*(2^n)$ algorithms for the traveling salesman problem by Bellman [3] and Held and Karp [12] are known for more than 50 years already (n is the number of vertices of an input graph, O^* hides polynomial factors of input length). This bound is still the best known for the general version of the problem, but stronger bounds are known for special cases: Björklund [4] in 2010 proved an $O(1.66^n)$ bound for the symmetric case (i.e., undirected graphs), Cygan et al. [9] in 2013 proved an $O^*(1.89^n)$ bound for directed bipartite graphs. The current record upper bound $O^*(2^n)$ for the chromatic number problem is proved by Björklund et al. [8] in 2006. The number of perfect matchings in an n-vertex graph can be computed in time $O^*(2^{n/2})$ as shown by Björklund [5] in 2012 (this matches the bound of Ryser's algorithm [17] for bipartite graphs).

These problems (and many others) can be seen as partition problems. In the chromatic number problem the goal is to partition the vertices into independent sets. In counting perfect matchings the goal is to partition the vertices into adjacent pairs. The traveling salesman problem can be seen as a problem of partitioning the set of vertices into, say, three parts of roughly equal size such

[*] The full version of this paper is available at `http://arxiv.org/abs/1311.2456`.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 551–562, 2014.

that there exists a Hamiltonian path through the vertices of each part and these three paths can be connected into a Hamiltonian path in the original graph.

As shown by Björklund et al. [8] such partition problems can be solved in time $O^*(2^n)$ using a technique called fast subset convolution (FSC). In particular, FSC immediately implies an $O^*(2^n)$ upper bound for such problems as chromatic number, maximum k-cut, domatic number, bin packing. Cygan and Pilipczuk [10] later showed that an $O^*(2^n)$ bound can also be proved using fast Fourier transform (FFT). Lokshtanov and Nederlof [15] also used FFT in order to improve space requirements for exact solving of several NP-hard problems.

For all three problems mentioned above (chromatic number, traveling salesman, counting perfect matchings), improving the known bound for the general case is a major open problem in the field of algorithms for NP-hard problems. However in recent years it was shown that the bound can be improved for various special cases. In [6,7,11] better upper bounds are proved for graphs of bounded degree (these three problems are known to be NP-hard even on graphs of bounded degree).

We present a new approach to get bounds of the form $O^*((2-\epsilon)^n)$ in various special cases. Namely we show that such a bound follows almost immediately if the corresponding partition problem possesses a certain structure. Informally, this structure can be described as follows. Assume that a group of people is going to an excursion and our task is to seat them into buses with several constraints saying that a pair of people does not want to see each other in the same bus. This is the coloring problem and it can be solved in $O^*(2^n)$ time. Assume now that we have additional constraints: the group of people contains several infants and these infants should be accompanied by their relatives in a bus. Roughly, we prove that if the number of infants is linear then the problem can be solved in $O^*((2-\varepsilon)^n)$ time. The approach is based on efficient FFT multiplication of polynomials with bounded integer coefficients. There is an algorithm [18,21,20] which multiplies two polynomials of degree n using $n\,\mathrm{polylog}(n)$ arithmetic operations.

Using this approach we unify several known results of this kind. An additional advantage of the approach is that it is particularly easy to use it as a black box. Namely, all one needs to do is to reveal the corresponding structure of families with infants. This way, some of the known upper bounds for the above mentioned problems on graphs of bounded degree follow just in a few lines. By using additional combinatorial ideas we also prove the following new results.

For the chromatic number problem, Björklund et al. [7] presented an algorithm working in time $O^*((2-\varepsilon(\Delta))^n)$ on graphs of bounded maximum degree $\Delta = O(1)$. The algorithm is based on Yate's algorithm and Möbios inversion and thus uses exponential space. We extend this result to a wider class of bounded average degree graphs. This closes an open problem concerning the existence of such an algorithm stated by Cygan and Pilipczuk [11].

For the traveling salesman problem on graphs of maximum degree $\Delta = O(1)$, Björklund et al. [6] presented an algorithm working in time $O^*((2-\varepsilon(\Delta))^n)$ and exponential space. Cygan and Pilipczuk [11] extended the result to graphs of bounded average (instead of maximum) degree. Both algorithms are based on dynamic programming and the savings in the running time comes from an observation that in case of bounded degree graphs an algorithm does not need to go through all possible subsets of vertices (e.g., a disconnected subgraph does not have a Hamiltonian path for sure). It is also because of the dynamic programming technique that both mentioned algorithms use exponential space. We further improve these results by showing an algorithm working in time $O^*(W(2-\varepsilon(d))^n)$ and polynomial space on directed graphs of average degree d with integral weights bounded by W.

Cygan and Pilipczuk [11] developed an algorithm with running time $O^*((2-\varepsilon(d))^{n/2})$ and exponential space for counting perfect matching in graphs of average degree d. We present an algorithm solving this problem in $O^*((2-\varepsilon(d))^{n/2})$ time and polynomial space. Several bounds of this kind are already known for bipartite graphs [1,2,19,16,13,11].

2 Notation

Let $G = (V, E)$ be a simple undirected graph. Throughout the paper we implicitly assume that the set of vertices of a graph under consideration is $V = \{1, 2, \ldots, n\}$. For simplicity, we consider undirected graphs only (whether a graph is directed or not is only important for the traveling salesman problem; the presented algorithm works for both undirected and directed graphs).

By $d(G)$ and $\Delta(G)$ we denote the average and the maximum degree of G (we omit G if it is clear from the context). $N_G(v)$ is a *neighborhood* of v in G, i.e., all the neighbors of v in G and $N_G[v] = N_G(v) \cup \{v\}$ is its *closed neighborhood.* For $S \subseteq V$, by $G[S]$ we denote a subgraph of G induced by S. We use $G \setminus S$ as a shortcut for $G[V \setminus S]$.

The *square* of $G = (V, E)$ is a graph $G^2 = (V, E')$ where $E' \supseteq E$ is

$$E' = \{(u, v)\colon \text{ there is a path of length at most 2 from } u \text{ to } v \text{ in } G\}\,.$$

Note that $\Delta(G^2) \leq (\Delta(G))^2$ and hence one can easily find an independent set of size $\frac{n}{(\Delta(G))^2+1}$ in G^2.

Following [11], by $V_{>c}$ we denote a subset of vertices V of degree greater than c. $V_{<c}$, $V_{=c}$, $V_{\leq c}$, $V_{\geq c}$ are defined similarly.

By $\mathbb{Z}_{\geq c}$ we denote the set of all integers not smaller than c.

Throughout the paper by ε we denote a positive constant that does not depend on the size of a graph.

We often exploit the following simple fact: one can find in G an independent set of size at least $\frac{n}{\Delta(G)+1}$ in polynomial time (this is done by a straightforward greedy algorithm). We also use the following lemma proved by Cygan and Pilipczuk [11]. We prove and make use of a slightly more general version of the lemma in the full version of the paper.

Lemma 1 ([11], Lemma 3.4). *For any constants $\mu < 1, 0 < c < 1$ there exists $\beta > 0$ such that for any graph $G = (V, E)$ of average degree $d = O(1)$ one can find in polynomial time subsets $A, Y \subseteq V$ such that:*

1. *$A \cap Y = \emptyset$;*
2. *A is an independent set in $(G \setminus Y)^2$;*
3. *$2|Y| \le |A| \le cn$;*
4. *each vertex from A has at most $2d$ neighbors in $G \setminus Y$: $\forall v \in A$, $|\{u \in V \setminus Y \colon (u,v) \in E\}| \le 2d$;*
5. $$\binom{|A|}{|Y|}\mu^{|A|} < 2^{-\beta n}\,. \tag{1}$$

3 Solving Partition Problems

Definition 1. *Let $V = \{1,\dots,n\}$, $1 \le k \le n$ be an integer and $\mathcal{F} = \{\mathcal{F}_1,\dots,\mathcal{F}_k\}$, where each $\mathcal{F}_i \subseteq 2^V$ is a family of subsets of V. A $(V,k,\mathcal{F})$-*partition problem *is to represent V as a disjoint union of k sets from $\mathcal{F}_i$'s: $V = F_1 \sqcup \ldots \sqcup F_k$, where $F_i \in \mathcal{F}_i, \forall 1 \le i \le k$.*

This definition is similar to the one used by Björklund et al. [8] the only difference being that in the definition above the families $\mathcal{F}_i$'s are not necessarily equal.

The brute force search algorithm for this problem takes time $O^*(\max_{1\le i\le k}|\mathcal{F}_i|^k)$. Using FFT one can easily prove an upper bound $O^*(2^n)$ which beats the previously mentioned bound in many interesting cases.

Theorem 1. *A $(V,k,\mathcal{F})$-partition problem can be solved in $O^*(2^n)$ time and space.*

Below, we formally define a combinatorial structure called families with infants that allow to get an $O^*((2-\varepsilon)^n)$ upper bound for partition problems.

Definition 2. *$\mathcal{R} = ((R_1,r_1),\dots,(R_p,r_p))$ is called a (p,q)-*system of families with infants *for a $(V,k,\mathcal{F})$-partition problem if all of the following conditions are satisfied:*

1. *for all $i = 1,\dots,p$, $r_i \in R_i \subseteq V$; r_i is called* an infant *and all the elements of $R_i \setminus \{r_i\}$ are called* relatives *of r_i; the sets R_i are called* families*;*
2. *the size of each family R_i is at most q;*
3. *$pq \le n$;*
4. *all families R_i's are pairwise disjoint;*
5. *in any valid partition each infant is accompanied by at least one of its relatives:*

$$\forall 1 \le i \le p, 1 \le j \le k \text{ and } \forall F \in \mathcal{F}_j, \text{ if } r_i \in F \text{ then } |F \cap R_i| \ge 2. \tag{2}$$

The following theorem is the main technical result of the paper.

Theorem 2. *Let $\mathcal{R} = ((R_1,r_1),\dots,(R_p,r_p))$ be a (p,q)-system of families with infants for a $(V,k,\mathcal{F})$-partition problem. Then the problem can be solved in time and space*

$$O^*\left(2^n \cdot \left(\frac{2^q-1}{2^q}\right)^p \cdot 2^q\right)\,. \tag{3}$$

Roughly, the savings in the running time comes from the fact that while looking for a valid partition of V one can avoid the case $F \cap R_i = \{r_i\}$ (i.e., instead of considering all 2^q possibilities of $F \cap R_i$ one considers $2^q - 1$ of them).

We always use Theorem 2 with $q = O(1)$ and $p = \Omega(n)$, which makes the running time $O^*((2-\varepsilon)^n)$.

Corollary 1. *Let $\mathcal{R} = ((R_1, r_1), \ldots, (R_p, r_p))$ be a (p,q)-system of families with infants for a $(V, k, \mathcal{F})$-partition problem. If $q = O(1)$ and $p = \Omega(n)$, then the problem can be solved in time and space $O^*((2-\varepsilon)^n)$.*

As an illustration of using Theorem 2 we replicate a result from [7]. In the (decision version of) domatic number problem the question is to partition the set of vertices into k dominating sets.

Lemma 2. *The domatic number problem in a graph of maximum degree $\Delta = O(1)$ can be solved in time and space $O^*((2-\varepsilon(\Delta))^n)$.*

Proof. The domatic number problem is a $(V, k, \mathcal{F})$-problem where each $\mathcal{F}_i$ is just the set of all dominating sets of G. By definition, for any $v \in V$ and any dominating set $U \subseteq V$, $N_G[v] \cap U \neq \emptyset$. This gives a straightforward construction of families with infants.

Find greedily an independent set $I \subseteq V$ of size $p = \frac{n}{\Delta^2+2}$ in G^2. Assume w.l.o.g. that $I = \{1, \ldots, p\}$. For each $1 \leq i \leq p$, let $R_i = N_G[i]$. At this point we have at least $n - p(\Delta + 1) \geq p$ remaining vertices in $V \setminus \cup_{i=1}^{p} R_i$. So, we can extend each R_i by one vertex and declare this one additional vertex as the infant of R_i.

All R_i have size at most $q = \Delta + 2 = O(1)$, the total number of R_i's is $p = \frac{n}{\Delta^2+2} = \Omega(n)$. Clearly $pq \leq n$. The constructed sets satisfy the property (2) by an obvious reason: each R_i is a superset of $N_G[v]$ for some $v \in V$ and none of these elements is the infant of the family R_i. And $U \cap N_G[v] \neq \emptyset$ for any dominating set U and any vertex v, i.e., any dominating set always contains at least one relative of r_i (even if it does not contain r_i).

The upper bound now follows from Theorem 2 and Corollary 1. □

Another example is a $O^*((2-\varepsilon)^n)$ algorithm for finding a Hamiltonian cycle in a graph of bounded degree. This result was given in [6].

Lemma 3. *The Hamiltonian cycle problem on a graph of maximum degree $\Delta = O(1)$ can be solved in time and space $O^*((2-\varepsilon(\Delta))^n)$.*

Proof. Guess three vertices $v_0, v_1, v_2 \in V$. Let $\mathcal{F} = (\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_2)$ where $\mathcal{F}_i \subseteq 2^V$ consists of all subsets $S \subseteq V$ of size $|S| = n/3$ for which G contains a path P such that

1. P starts in v_i and ends in u_i such that $(u_i, v_{(i+1) \bmod 3}) \in E$;
2. P goes through all the vertices in S exactly once.

The family $\mathcal{F}$ can be computed in time $O^*(2^{H(1/3)n}) = O^*(1.99^n)$ (where $H(x) = -x \log_2 -(1-x)\log_2(1-x)$ is the binary entropy function) by dynamic programming.

We now need to construct a system of families with infants for the resulting $(V, 3, \mathcal{F})$-partition problem. We construct the required family for $p = \frac{n}{\Delta^2+1}$ and $q = \Delta + 1$. Find greedily an independent set I of size $p = \frac{n}{\Delta^2+1}$ in G^2. Assume that $I = \{1, \dots, p\}$ and let $R_i = N_G[i]$, $r_i = i$. Clearly, if $F \in \mathcal{F}_j$ contains an infant $r_i \in R_i$ then this infant is necessarily accompanied by one of its relatives since F contains a Hamiltonian path. □

The corresponding algorithms allow also to solve weighted partition problems. In such problems, each subset F of $\mathcal{F}_i$ is assigned a non-negative integer weight $w(F)$ and the goal is to find a partition of minimum total weight.

Theorem 3. *If in Theorems 1 and 2 one is given a weighted partition problem then the upper bounds on the running time and space are multiplied by W where W is the maximum weight of a subset.*

Also, one can turn the algorithm to use polynomial space by providing an algorithm that enumerates the sets $\mathcal{F}_i$.

Theorem 4. *Let $\mathcal{F} = (\mathcal{F}_1, \dots, \mathcal{F}_k)$ and assume that there exists an algorithm that for any $i = 1, \dots, k$ enumerates the set $\mathcal{F}_i$ in time T and polynomial space. Then one can turn algorithms from Theorems 1, 2, 3 into polynomial space algorithms at the cost of multiplying the running time by T.*

As a corollary we get a polynomial-space algorithm for the case when each $\mathcal{F}_i$ is of polynomial size.

Corollary 2. *Let $\mathcal{R}$ and $(V, k, \mathcal{F})$ be as in Theorem 4. If for all $i = 1, \dots, k$, $\mathcal{F}_i$ is enumerable in polynomial time (in particular, $|\mathcal{F}_i| = \text{poly}(n)$) then the corresponding algorithm uses polynomial space.*

We conclude the section by noting that the same bounds hold also for the case when instead of partition one looks for a cover of V by k subsets from $\mathcal{F}_1, \dots, \mathcal{F}_k$. We call the corresponding problem a $(V, k, \mathcal{F})$*-covering problem.*

Theorem 5. *Theorems 1, 2, 3, 4 hold for $(V, k, \mathcal{F})$-covering problems.*

3.1 Proofs

For a subset $U \subseteq V$, let $b(U) \in \{0,1\}^n$ denote the characteristic vector of the set U (i.e., $b(U)[i] = 1$ iff $i \in U$). In the analysis below we sometimes identify a bit vector $b(U)$ with a non-negative integer between 0 and $2^n - 1$ that it represents.

For a bit vector b, we denote the Hamming weight of b by $w(b)$, i.e., the number of 1's in b. Note the following simple fact: for any two non-negative integers a and b,

$$w(\text{bin}(a)) + w(\text{bin}(b)) \geq w(\text{bin}(a + b)) \tag{4}$$

and the equality holds iff there are no carries in $a + b$.

Proof (of Theorem 1). Consider the following polynomials for $1 \le i \le k$: $P_i(x,y) = \sum_{F \in \mathcal{F}_i} x^{|F|} y^{b(F)}$. We claim that there is a solution to a $(V,\mathcal{F},k)$-partition problem iff the monomial $x^n y^{b(V)}$ has a non-zero coefficient in $\prod_{i=1}^{k} P_i(x,y)$. One direction of this statement is straightforward: if there exist $F_1 \in \mathcal{F}_1, \ldots, F_k \in \mathcal{F}_k$ such that $F_1 \sqcup \ldots \sqcup F_k = V$ then clearly $\prod_{i=1}^{k} x^{|F_i|} y^{b(F_i)} = x^n y^{b(V)}$. For the reverse direction, assume that the product of the polynomials contains the monomial $x^n y^{b(V)}$. Because of the term x^n, there exist k subsets $F_1 \in \mathcal{F}_1, \ldots, F_k \in \mathcal{F}_k$ such that $|F_1| + \ldots + |F_k| = n$. In other words, the total number of 1's in all characteristic vectors of F_i's is exactly n. Moreover,

$$b(F_1) + \ldots + b(F_k) = b(V) \tag{5}$$

and the number of 1's in the characteristic vector of V is also n. Now, (4) implies that there are no carries in (5). This in turn implies that $\{F_1, \ldots, F_k\}$ is a partition of V.

Now it suffices to show that the coefficient of the monomial $x^n y^{b(V)}$ in $\prod_{i=1}^{k} P_i(x,y)$ can be found in time $O^*(2^n)$. Since $|F| \le n$ and $k \le n$, the degree of x in $\prod_{i=1}^{k} P_i(x,y)$ does not exceed n^2. Therefore, in order to obtain univariate polynomials we can use Kronecker substitution [14]. Namely, we replace y by x^{n^2+1}. Thus, for each $1 \le i \le k$ we consider a univariate polynomial $Q_i(x)$: $Q_i(x) = \sum_{F \in \mathcal{F}_i} x^{|F|} x^{(n^2+1)\cdot b(F)}$. It it easy to see that the coefficient of $x^{a_1 + (n^2+1)\cdot a_2}$ (where $a_1 \le n^2$) in $Q_i(x)$ equals the coeffient of $x^{a_1} y^{a_2}$ in $P_i(x)$, and vice versa. In other words, we associate an integer from $[0..(n^2+1)2^n]$ with each $F \in \mathcal{F}_i$. This integer is an encoding of the set F in $n + 2\log n$ bits, s.t. the first n bits indicate elements of F, the next $\log n$ bits are zeros, and the last $\log n$ bits are the binary expansion of $|F|$. We need to find the coefficient of $x^{n+(n^2+1)\cdot b(v)}$ in $Q = \prod_{i=1}^{k} Q_i(x)$, where $\deg(Q) = O^*(2^n), k \le n$. One can apply FFT $k-1$ times and get the desired upper bound on the running time. □

Remark 1. Note that the coefficient of the monomial $x^n y^{b(V)}$ in the theorem equals the number of valid k-partitions. So this theorem (and all the related theorems) in fact count the number of valid partitions.

Remark 2. Note that here and below all the coefficients are integers of length poly(n), so all arithmetic operations can be done in polynomial time.

Definition 3. *For a matrix* $M = (M[i,j])_{0 \le i \le p-1, 0 \le j \le q-1} \in \mathbb{Z}_{\ge 0}^{p \times q}$ *let*

$$\text{colweight}(M,j) = \sum_{i=0}^{p-1} M[i,j]\,, \text{weight}(M) = \sum_{i=0}^{p-1}\sum_{j=0}^{q-1} M[i,j] = \sum_{j=0}^{q-1} \text{colweight}(M,j)\,,$$

$$\text{rowcode}(M,i) = -M[i,0] + \sum_{j=1}^{q-1} 2^j \cdot M[i,j]\,, \text{rowsum}(M) = \sum_{i=0}^{p-1} \text{rowcode}(M,i)\,,$$

$$\text{code}(M) = \sum_{i=0}^{p-1} (2^q - 1)^i \cdot \text{rowcode}(M,i)\,.$$

Definition 4. *A matrix* $M \in \mathbb{Z}_{\geq 0}^{p \times q}$ *is called* row-normalized *if* $\mathrm{rowcode}(M, i) \geq 0$ *for all* $0 \leq i \leq p-1$.

Remark 3. In the analysis below we will need the following simple estimates. Let $E \in \{0,1\}^{p \times q}$. Then

1. $\mathrm{rowcode}(E, i) < 0$ iff $E[i] = [1, 0, 0, \ldots, 0]$,
2. $\mathrm{rowcode}(E, i) \leq 2^q - 2$.
3. $\mathrm{code}(E) \leq (2^q - 2) \cdot \sum_{i=0}^{p-1} (2^q - 1)^i < (2^q - 1)^p$ (assuming $q \geq 2$).

The following fact is well known so we state is without a proof.

Lemma 4. *The expansion of* $X \in \mathbb{Z}_{\geq 0}$ *in powers of* $b > 1$ *as* $X = \sum_{i=0}^{\infty} x_i \cdot b^i, x_i \geq 0$ *has the minimal value of the sum of digits* $\sum_{i=0}^{\infty} x_i$ *iff* $\forall i : 0 \leq x_i < b$ *(i.e.,* X *is written in the numeral system of base* b*).*

Lemma 5. *Let* $q \geq 2$ *and* $E \in \{0,1\}^{p \times q}$ *and* $M \in \mathbb{Z}_{\geq 0}^{p \times q}$ *be row-normalized matrices. If* $\mathrm{colweight}(M, 0) = \mathrm{colweight}(E, 0)$, $\mathrm{weight}(M) = \mathrm{weight}(E)$, $\mathrm{rowsum}(M) = \mathrm{rowsum}(E)$, $\mathrm{code}(M) = \mathrm{code}(E)$, *then* $M = E$.

Proof. The claim follows from Lemma 4. Since for all i, $\mathrm{rowcode}(E, i) \leq 2^q - 2$, $\mathrm{code}(E)$ has the minimal sum of digits in base $(2^q - 1)$ system. This in turn implies that for each i, $\mathrm{rowcode}(E, i) = \mathrm{rowcode}(M, i)$. Then the first columns of matrices E and M are equal modulo 2, because parities of rowcodes depend only on the first column. Since $\mathrm{colweight}(M, 0) = \mathrm{colweight}(E, 0)$ we conclude that the first columns of M and E are equal.

Now each $\mathrm{rowcode}(E, i)$ has the minimal sum of digits in the system of base 2, which means that $\mathrm{weight}(E)$ has the minimal possible value for these rowcodes. It follows from Lemma 4 that each $M[i, j]$ must be equal to $E[i, j]$. □

Definition 5. *An injective function* $\alpha \colon U \to \{0, \ldots, p-1\} \times \{0, \ldots, q-1\}$ *is called a* matrix representation *of a set* U*. For such* α *and* $S \subseteq U$*, a* characteristic matrix $M_\alpha(S) \in \{0,1\}^{p \times q}$ *is defined as follows:* $i \in U$ *iff* $M_\alpha(S)[\alpha(i)] = 1$.

Proof (of Theorem 2). Let $\mathcal{R} = ((R_1, r_1), \ldots, (R_p, r_p))$ be a (p, q)-system of families with infants for a $(V, k, \mathcal{F})$-partition problem. Append arbitrary elements from V to families so that the size of each family equals q and the families are still disjoint (this is possible since $pq \leq n$). Denote the union of families by R and the rest of V by L. For each family R_i, fix an order of its elements such that the 0th element is r_i. Now consider a matrix representation $\alpha \colon V \to \{0, \ldots, p-1\} \times \{0, \ldots, q-1\}$ defined as follows. If v is the jth element of R_i, then $\alpha(v) = (i, j)$. We encode each set $F \in \mathcal{F}_i$ by parts. We encode vertices from $F \cap L$ using the standard technique from Theorem 1. To encode vertices from $F \cap R$ we use the characteristic matrix $M_\alpha(F \cap R)$. Note that $M_\alpha(F \cap R)$ is a row-normalized matrix, because if F contains an infant r_i of a family R_i, then it must contain at least one other element from the same row. Consider the following polynomials for $1 \leq i \leq k$: $P_i(x, y, z, s, t, u)$ is equal to

$$\sum_{F \in \mathcal{F}_i} x^{|F \cap L|} \cdot y^{b(F \cap L)} \cdot z^{\mathrm{colweight}(M,0)} \cdot s^{\mathrm{weight}(M)} \cdot t^{\mathrm{rowsum}(M)} \cdot u^{\mathrm{code}(M)},$$

where $M = M_\alpha(F \cap R)$ and $b(F \cap L)$ is an integer from 0 to $2^{|L|} - 1$.

There is a solution to a $(V, \mathcal{F}, k)$-partition problem iff the monomial

$$x^{|L|} y^{b(L)} z^{\text{colweight}(R,0)} s^{\text{weight}(R)} t^{\text{rowsum}(R)} u^{\text{code}(R)}$$

has a non-zero coefficient in $\prod_{i=1}^{k} P_i(x, y, z, s, t, u)$. Indeed, as it was shown in Theorem 1, $x^{|L|}y^{b(L)}$ corresponds to partitions of L. Lemma 5 implies that only partitions of R may have the term $z^{\text{colweight}(R,0)} s^{\text{weight}(R)} t^{\text{rowsum}(R)} u^{\text{code}(R)}$. Note that the degrees of x, z, s in $\prod_{i=1}^{k} P_i(x, y, z, s, t, u)$ are bounded from above by n^2, the degree of y — by $n \cdot 2^{|L|}$, the degree of t — by $n^2 \cdot 2^q$, the degree of u — by $n \cdot (2^q - 1)^p$. Now we can apply Kronecker substitution: $y = x^{(n+1)^2}, z = x^{(n+1)^3 2^{|L|}}, s = x^{(n+1)^5 2^{|L|}}, t = x^{(n+1)^7 2^{|L|}}, u = x^{(n+1)^9 2^{|L|} 2^q}$. The running time of FFT is bounded by the degree of the resulting univariate polynomial, i.e.

$$O^*(\text{poly}(n) 2^{|L|} 2^q (2^q - 1)^p) = O^*(2^{n-pq}(2^q - 1)^p 2^q) = O^*\left(2^n \cdot \left(\frac{2^q - 1}{2^q}\right)^p \cdot 2^q\right).$$

□

Proof (of Theorem 3). Following the proofs of Theorem 1 and 2, we introduce polynomials corresponding to $\mathcal{F}_i$'s. But in the weighted partition problem we multiply each monomial by z^w, where z is a new variable and w is the weight of the corresponding set. For example, in Theorem 1 the new polynomials would look as follows: $P_i(x, y, z) = \sum_{F \in \mathcal{F}_i} x^{|F|} y^{b(F)} z^{w(F)}$, where $w(F)$ is the weight of F. Now it is clear that there exists a partition of total weight w iff the monomial $x^n y^{b(V)} z^w$ has a non-zero coefficient in $\prod_{i=1}^{k} P_i(x, y, z)$. We apply Kronecker substitution $y = x^{n^2+1}, z = x^{(n^2+1)n2^n}$ and use FFT to find all the coefficients of the product. Now we just need to find the smallest w, s.t. the coefficient of $x^{n+(n^2+1)\cdot b(V)+(n^2+1)n2^n w}$ does not equal 0. Since the degree of the polynomial is at most $O^*(2^n W)$, the running time of the algorithm is $O^*(2^n W)$. □

Proof (of Theorem 4). Assume that we need to find the coefficient of the monomial x^m in $P = \prod_{i=1}^{k} P_i(x)$, where $\deg(\Pi) \le d$. The theorem assumption claims that we can evaluate each P_i at any point in time $O^*(T)$. Note that in order to get one specific coefficient of $P(x)$, we just need to list all the Fourier coefficients of P. Indeed, the coefficient of x^m in P equals

$$\frac{1}{d} \sum_{i=0}^{d-1} \omega_d^{-im} P(\omega_d^k) = \frac{1}{d} \sum_{i=0}^{d-1} \omega_d^{-im} \prod_{i=1}^{k} P_i(\omega_d^k).$$

Since each $P_i(x)$ can be evaluated in time $O^*(T)$, we need $O^*(d \cdot T)$ steps and only polynomial space to find one coefficient of the product. □

Proof (of Theorem 5). The proofs are very similar to the proofs for partition problems, the only difference is that now we consider polynomials $P_i'(x, y) = \sum_{F \in \mathcal{F}_i} \prod_{v \in F} (1 + x \cdot y^{b(\{v\})})$. Let $\mathcal{F}_i' = \bigcup_{F \in \mathcal{F}_i} 2^F$. Clearly, a $(V, \mathcal{F}, k)$-covering problem has a solution iff a $(V, \mathcal{F}', k)$-partition problem has one. The polynomial $P_i'(x, y)$ corresponds to the polynomial $P_i(x, y)$ for $\mathcal{F}'$ from the proof of

Theorem 2. Namely, $x^m y^{b(U)}$ has a non-zero coefficient in $P'(x,y)$ iff $U \subseteq \mathcal{F}_i$ and $|U| = m$. Again, the degree of x in $\prod_{i=1}^{k} P'_i(x,y)$ is less than n^2+1, so we can apply Kronecker substitution $y = x^{n^2+1}$. Thus, there exists a valid cover iff the coefficient of the monomial $x^{n+(n^2+1)\cdot b(v)}$ does not equal 0. From now on we can follow the proofs of Theorems 1, 2, 3, 4. □

4 The Traveling Salesman Problem

Theorem 6. *The traveling salesman problem on graphs of average degree $d = O(1)$ with integer weights from $[1 \dots W]$ can be solved in time $O^*(W \cdot (2-\varepsilon(d))^n)$ and polynomial space $O(\mathrm{poly}(n) \log W)$.*

Proof. Let k be a parameter to be defined later. Guess k vertices $v_1, \dots, v_k \in V$ that split an optimal Hamiltonian cycle into k paths of length n/k. All such k-tuples can be enumerated in time n^k. Then the corresponding weighted $(V,k,\mathcal{F})$-partition problem is defined as follows. $\mathcal{F} = (\mathcal{F}_1, \dots, \mathcal{F}_k)$ where $\mathcal{F}_i$ consists of all subsets $S \subseteq V$ of size $|S| = \frac{n}{k}$ for which G contains a Hamiltonian path P such that P starts in v_i, goes through all vertices from S, and ends in u_i such that $(u_i, v_{(i \bmod k)+1}) \in E$. The weight $w(S)$ of such a set S is equal to the minimal possible weight of a path P (including the weight of the edge $(u_i, v_{(i \bmod k)+1})$). It is not difficult to see that solving the traveling salesman problem is equivalent to solving the $(V,k,\mathcal{F})$-partition problem if the vertices $v_1, \dots, v_k$ are guessed correctly.

Note that any $\mathcal{F}_i$ can be enumerated in time $\binom{n}{\frac{n}{k}} \cdot \left(\frac{n}{k}\right)!$ (the first term is for guessing the subset of vertices S, the second one is for guessing the order of these vertices in an optimal path P). Recall also that guessing the vertices $v_1, \dots, v_k$ requires $O(n^k)$ time. By choosing $k = \sqrt{n}$ we turn both these estimates into subexponential $2^{o(n)}$.

Now we turn to constructing the corresponding system of families with infants. Let $\mu < 1, 0 < c < 1$ be constants to be defined later. Let $A, Y \subseteq V$ be provided by Lemma 1. Consider an optimal Hamiltonian cycle C in the graph. Let $Y' \subseteq V$ be the successors of the vertices from Y in the cycle C and let $A' = A \setminus Y' \setminus \{v_1, \dots, v_k\}$. Note that $|A'| \geq \frac{|A|}{2} - k$ (since $|A| \geq 2|Y|$) and for each vertex $v \in A'$ its predecessor u in the cycle C belongs to $V \setminus Y$.

Let $A' = \{1, \dots, p\}$. Then for all $i = 1, \dots, p$, $R_i = N_{G \setminus Y}[i]$ and $r_i = i$. Clearly, $|R_i| \leq q = 2d+1$. By choosing $c < \frac{1}{2d+1}$ we can guarantee that $pq \leq n$. All R_i's are disjoint since A' is an independent set in $(G \setminus Y)^2$. Finally, if the set Y' is guessed correctly (i.e., Y' are indeed successors of Y in the optimal cycle C) then $((R_1, r_1), \dots, (R_p, r_p))$ is a (p,q)-system of families with infants. Indeed, if $r_i \in F$ for some $F \in \mathcal{F}_j$, then r_i's predecessor in C must lie in $V \setminus Y$, i.e., in R_i (and r_i must have a predecessor since $r_i \neq v_1, \dots, v_k$).

By Theorems 3 and 4 the total running time does not exceed

$$2^{o(n)} \cdot \binom{|A|}{|Y|} \cdot 2^n \cdot \left(\frac{2^q - 1}{2^q}\right)^p \cdot W.$$

Recall that $p = |A'| - k \geq |A|/2 - |A|/6 = |A|/3$ for large enough n (since $|A| = cn$ and $k = \sqrt{n}$). Choose $\mu = \left(\frac{2^{2d+1}-1}{2^{2d+1}}\right)^{1/3}$. Then (1) implies that $\binom{|A|}{|Y|} \cdot \left(\frac{2^q-1}{2^q}\right)^p < 2^{-\beta n}$ for a constant $\beta > 0$. Thus the total running time is $O^*(W \cdot (2-\varepsilon)^n)$. □

5 Counting Perfect Matchings and Coloring

Due to space restrictions the proofs of the following two theorems are provided in the full version of the paper only.

Theorem 7. *There is an algorithm checking whether for a given graph G of average degree $d = O(1)$ there exists a proper k-coloring in time $O^*((2-\varepsilon(d))^n)$ and exponential space.*

Remark 4. The algorithm actually solves a more general problem known as maximum k-cut. In this problem the goal is to partition the vertices into k parts such that the number of edges joining different parts is maximal possible.

The algorithm for counting perfect matchings uses many ideas from the algorithm for the traveling salesman problem presented in Theorem 6.

Theorem 8. *The number of perfect matchings in a graph G with $2n$ vertices of average degree $d = O(1)$ can be found in time $O^*((2-\varepsilon(d))^n)$ and polynomial space.*

Acknowledgments. Research is partially supported by the Government of the Russian Federation (grant 14.Z50.31.0030) and Grant of the President of Russian Federation (NSh-3856.2014.1).

We would like to thank the anonymous reviewers for their many valuable comments.

References

1. Arnborg, S., Lagergren, J., Seese, D.: Easy problems for tree-decomposable graphs. Journal of Algorithms 12(2), 308–340 (1991)
2. Bax, E., Franklin, J.: A permanent algorithm with $\exp[\Omega(n^{1/3}/2\ln n)]$ expected speedup for 0-1 matrices. Algorithmica 32(1), 157–162 (2002)
3. Bellman, R.: Dynamic programming treatment of the travelling salesman problem. J. ACM 9, 61–63 (1962)
4. Björklund, A.: Determinant sums for undirected hamiltonicity. In: Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science, FOCS 2010, pp. 173–182 (2010)
5. Björklund, A.: Counting perfect matchings as fast as Ryser. In: Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, pp. 914–921. SIAM (2012)

6. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: The travelling salesman problem in bounded degree graphs. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 198–209. Springer, Heidelberg (2008)
7. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Trimmed Moebius inversion and graphs of bounded degree. Theory of Computing Systems 47(3), 637–654 (2010)
8. Björklund, A., Husfeldt, T., Koivisto, M.: Set partitioning via inclusion-exclusion. SIAM Journal on Computing 39(2), 546–563 (2009)
9. Cygan, M., Kratsch, S., Nederlof, J.: Fast hamiltonicity checking via bases of perfect matchings. In: Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing, STOC 2013, pp. 301–310 (2013)
10. Cygan, M., Pilipczuk, M.: Exact and approximate bandwidth. Theoretical Computer Science 411(40-42), 3701–3713 (2010)
11. Cygan, M., Pilipczuk, M.: Faster exponential-time algorithms in graphs of bounded average degree. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) ICALP 2013, Part I. LNCS, vol. 7965, pp. 364–375. Springer, Heidelberg (2013)
12. Held, M., Karp, R.M.: A dynamic programming approach to sequencing problems. Journal of the Society for Industrial and Applied Mathematics 10(1), 196–210 (1962)
13. Izumi, T., Wadayama, T.: A new direction for counting perfect matchings. In: 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science (FOCS), pp. 591–598. IEEE (2012)
14. Kronecker, L.: Grundzüge einer arithmetischen theorie der algebraischen grössen. J. Reine Angew. Math. 92, 1–122 (1882)
15. Lokshtanov, D., Nederlof, J.: Saving space by algebraization. In: Proceedings of the 42nd ACM symposium on Theory of computing, STOC 2010, pp. 321–330. ACM (2010)
16. van Rooij, J.M.M., Bodlaender, H.L., Rossmanith, P.: Dynamic programming on tree decompositions using generalised fast subset convolution. In: Fiat, A., Sanders, P. (eds.) ESA 2009. LNCS, vol. 5757, pp. 566–577. Springer, Heidelberg (2009)
17. Ryser, H.J.: Combinatorial mathematics. Mathematical Association of America, Washington, DC (1963)
18. Schonhage, A., Strassen, V.: Schnelle multiplikation grosser zahlen. Computing 7(3-4), 281–292 (1971)
19. Servedio, R.A., Wan, A.: Computing sparse permanents faster. Information Processing Letters 96(3), 89–92 (2005)
20. Shoup, V.: A Computational Introduction to Number Theory and Algebra, 2nd edn. Cambridge University Press (2009)
21. Turk, J.: Fast arithmetic operations on numbers and polynomials. Mathematisch Centrum Computational Methods in Number Theory 1, 43–54 (1982)

Changing Bases: Multistage Optimization for Matroids and Matchings

Anupam Gupta[1,⋆], Kunal Talwar[2], and Udi Wieder[2]

[1] Computer Science Dept., Carnegie Mellon University, Pittsburgh, PA 15213
anupamg@cs.cmu.edu

[2] Microsoft Research Silicon Valley. 1065 La Avenida, Mountain View, CA, USA
{kunal,uwieder}@microsoft.com

Abstract. This paper is motivated by the fact that many systems need to be maintained continually while the underlying costs change over time. The challenge is to continually maintain near-optimal solutions to an underlying optimization problem, without creating too much churn in the solution itself. We model this as a multistage combinatorial optimization problem where the input is a sequence of cost functions (one for each time step); while we can change the solution from step to step, we incur an additional cost for every such change.

We first study the multistage matroid maintenance problem, where we need to maintain a base of a matroid in each time step under changing cost functions and acquisition costs for adding new elements. The online version generalizes online paging. E.g., given a graph, we need to maintain a spanning tree T_t at each step: we pay $c_t(T_t)$ for the cost of the tree at time t, and also $|T_t \setminus T_{t-1}|$ for the number of edges changed at this step. Our main result is a polynomial time $O(\log m \log r)$-approximation to the online problem, where m is the number of elements/edges and r is the rank of the matroid. This improves on results of Buchbinder et al. [7] who addressed the *fractional* version of this problem under uniform acquisition costs, and Buchbinder, Chen and Naor [8] who studied the fractional version of a more general problem. We also give an $O(\log m)$ approximation for the offline version of the problem. These bounds hold when the acquisition costs are non-uniform, in which case both these results are the best possible unless P=NP.

We also study the perfect matching version of the problem, where we maintain a perfect matching at each step under changing cost functions and costs for adding new elements. Surprisingly, the hardness drastically increases: for any constant $\varepsilon > 0$, there is no $O(n^{1-\varepsilon})$-approximation to the multistage matching maintenance problem, even in the offline case.

1 Introduction

In a typical instance of a combinatorial optimization problem the underlying constraints model a static application frozen in one time step. In many applications

[⋆] Research performed while the author was at Microsoft Reserach Silicon Valley. Research was partly supported by NSF awards CCF-0964474 and CCF-1016799, and by a grant from the CMU-Microsoft Center for Computational Thinking.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 563–575, 2014.

however, one needs to solve instances of the combinatorial optimization problem that changes over time. While this is naturally handled by re-solving the optimization problem in each time step separately, changing the solution each time step often incurs a transition cost. Consider, for example, the problem faced by a vendor who needs to get supply of an item from k different producers to meet her demand. On any given day, she could get prices from each of the producers and pick the k cheapest ones to buy from. As prices change, this set of the k cheapest producers may change. However, there is a fixed cost to starting and/or ending a relationship with any new producer. The goal of the vendor is to minimize the sum total of these two costs: an "acquisition cost" $a(e)$ to be incurred each time she starts a new business relationship with a producer, and a per period cost $c_t(e)$ of buying in period t from each of the k producers that she picks in this period, summed over T time periods. In this work we consider a generalization of this problem, where the constraint "pick k producers" may be replaced by a more general combinatorial constraint. It is natural to ask whether simple combinatorial problems for which the one-shot problem is easy to solve, as the example above is, also admit good algorithms for the multistage version.

We study the *Multistage Matroid Maintenance* problem (MMM), where the underlying combinatorial constraint is that of maintaining a base of a given matroid. In the example above, the requirement the vendor buys from k different producers could be expressed as optimizing over the $k-$uniform matroid. In a more interesting case one may want to maintain a spanning tree of a graph, where the edge costs $c_t(e)$ change over time, and an acquisition cost of $a(e)$ has to paid every time a new edge enters the spanning tree. (A formal definition appears in Section 2.) While our emphasis is on the online problem, we will show results for the offline version as well, where all edge costs are given in advance. A first observation we make is that if the matroid in question is allowed to be different in each time period, then the problem is hard to approximate to any non-trivial factor (see [17]) even in the offline case. We therefore focus on the case where the same matroid is given at each time period.

To set the baseline, we first study the offline version of the problem (in Section 3), where all the input parameters are known in advance. We show an LP-rounding algorithm which approximates the total cost up to a logarithmic factor. The same approximation factor could be obtained by a simple greedy algorithm, but it will be useful to see the rounding algorithm, since we will use its extension in the online setting. We also show a matching hardness reduction, proving that the problem is hard to approximate to better than a logarithmic factor; even for the special case of spanning trees in graphs.

We then turn to the online version of the problem, where in each time period, we learn the costs $c_t(e)$ of each element, and we need to pick a base S_t of the matroid for this period. We analyze the performance of our online algorithm in the competitive analysis framework: i.e., we compare the cost of the online algorithm to that of the optimum solution of the offline instance. In Section 4, we give an efficient randomized $O(\log |E| \log(rT))$-competitive algorithm for this problem against any oblivious adversary (here E is the universe for the matroid and r is the rank of the matroid), and show that no polynomial-time online algorithm can do better. We also show that the requirement that the algorithm be randomized is necessary: any deterministic algorithm must incur an overhead of $\Omega(\min(|E|, T))$, even for the simplest of matroids.

Our results above crucially rely on the properties of matroids. It is natural to ask if we can handle more general set systems, e.g., p-systems. In Section 5, we consider the case where the combinatorial object we need to maintain is a perfect matching in a graph. Somewhat surprisingly, the problem here is significantly harder than the matroid case, even in the offline case. In particular, we show that even when the number of periods is a constant, no polynomial time algorithm can achieve an approximation ratio better than $\Omega(|E|^{1-\epsilon})$ for any constant $\epsilon > 0$.

1.1 Techniques

We first show that the MMM problem, which is a packing-covering problem, can be reduced to the analogous problem of maintaining a spanning set of a matroid. We call the latter the *Multistage Spanning set Maintenance* (MSM) problem. While the reduction itself is fairly clean, it is surprisingly powerful and is what enables us to improve on previous works. The MSM problem is a covering problem, so it admits better approximation ratios and puts at our disposal a much larger toolbox of techniques. We note that this is the only place where we need the matroid to not change over time: our algorithms for MSM work when the matroids change over time, and even when considering matroid intersections. The MSM problem is then further reduced to the case where the holding cost of an element is in $\{0, \infty\}$, this reduction simplifies the analysis.

For the offline case, we present two algorithms. We first observe that a greedy algorithm easily gives an $O(\log T)$-approximation. We then present a simple randomized rounding algorithm for the linear program. This is analyzed using recent results on contention resolution schemes [12], and gives an approximation of $O(\log rT)$, which can be improved to $O(\log r)$ when the acquisition costs are uniform. This LP-rounding algorithm will be an important constituent of the algorithm for the online case.

For the online case we again write the problem as a covering problem, even though the natural LP formulation has both covering and packing constraints. Phrasing it as a covering problem (with box constraints) enables us to use, as a black-box, results on online algorithms for the fractional problem [9]. This formulation however has exponentially many constraints. We handle that by showing a method of adaptively picking violated constraints such that only a small number of constraints are ever picked. The crucial insight here is that if x is such that $2x$ is not feasible, then x is at least $\frac{1}{2}$ away in ℓ_1 distance from any feasible solution; in fact there is a single constraint that is violated to an extent half. This insight allows us to make non-trivial progress (using a natural potential function) every time we bring in a constraint, and lets us bound the number of constraints we need to add until constraints are satisfied by $2x$.

1.2 Related Work

Our work is related to several lines of research, and extends some of them. The paging problem is a special case of MMM where the underlying matroid is a uniform one. Our online algorithm generalizes the $O(\log k)$-competitive algorithm for weighted caching [5], using existing online LP solvers in a black-box fashion. Going from uniform to general matroids loses a logarithmic factor (after rounding), we show such a loss is unavoidable unless we use exponential time.

The MMM problem is also a special case of classical metrical task systems [6]; see [1, 4] for more recent work. The best approximations for metrical task systems are poly-logarithmic in the size of the metric space. In our case the metric space is specified by the total number of bases of the matroid which is often exponential, so these algorithms only give a trivial approximation.

In trying to unify online learning and competitive analysis, Buchbinder et al. [7] consider a problem on matroids very similar to ours. The salient differences are: (a) in their model all acquisition costs are the same, and (b) they work with fractional bases instead of integral ones. They give an $O(\log n)$-competitive algorithm. Our online LP solving generalizes their result to arbitrary acquisition costs. They leave open the question of getting integer solutions online (Seffi Naor, private communication), which we present in this work. In a more recent work, Buchbinder, Chen and Naor [8] use a regularization approach to solving a broader set of fractional problems, but once again do not get integer solutions.

Shachnai et al. [23] consider "reoptimization" problems: given a starting solution and a new instance, they want to balance the transition cost and the cost on the new instance. This is a two-timestep version of our problem, and the short time horizon raises a very different set of issues (since the output solution does not need to itself hedge against possible subsequent futures).

Cohen et al. [13] describe a framework for stability-versus-fit tradeoff; e.g., that of finding a "stable" solutions which (like in reoptimization) maximizes the quality minus the transition costs. They show that maintaining stable solutions for matroids becomes a repeated two-stage reoptimization problem; their problem is poly-time solvable, whereas matroid problems in our model become NP-hard. The reason is that the solution for two time steps does not necessarily lead to a base from which it is easy to move in subsequent time steps, as our hardness reduction shows. They also consider a multistage offline version of their problem (again maximizing fit minus stability) which is very similar in spirit and form to our (minimization) problem, though the minus sign in the objective function makes it difficult to approximate in cases which are not in poly-time.

In dynamic Steiner tree maintenance [18, 19, 15] the goal is to maintain an approximately optimal Steiner tree for a varying instance (where terminals are added) while changing few edges at each time step. In dynamic load balancing [2, 14] one has to maintain a good scheduling solution while moving a small number of jobs around. The work on lazy experts in the online prediction community [11] also deals with similar concerns.

There is also work on "leasing" problems [20, 3, 21]: these are optimization problems where elements can be obtained for an interval of any length, where the cost is concave in the lengths; the instance changes at each timestep. The main differences are that the solution only needs to be feasible at each timestep (i.e., the holding costs are $\{0, \infty\}$), and that any element can be leased for any length ℓ of time starting at any timestep for a cost that depends only on ℓ, which gives these problems a lot of uniformity. In turn, these leasing problems are related to "buy-at-bulk" problems.

2 Maintaining Bases to Maintaining Spanning Sets

Given reals $c(e)$ for elements $e \in E$, we will use $c(S)$ for $S \subseteq E$ to denote $\sum_{e \in S} c(e)$. We denote $\{1, 2, \ldots, T\}$ by $[T]$.

We assume basic familiarity with matroids: see, e.g., [22] for a detailed treatment. Given a matroid $\mathcal{M} = (E, \mathcal{I})$, a *base* is a maximum cardinality independent set, and a *spanning set* is a set S such that $\mathsf{rank}(S) = \mathsf{rank}(E)$; equivalently, this set contains a base within it. The *span* of a set $S \subseteq E$ is $\mathsf{span}(S) = \{e \in E \mid \mathsf{rank}(S + e) = \mathsf{rank}(S)\}$. The *matroid polytope* $\mathcal{P}_I(\mathcal{M})$ is defined as $\{x \in \mathbb{R}^{|E|}_{\geq 0} \mid x(S) \leq \mathsf{rank}(S)\ \forall S \subseteq E\}$. The *base polytope* $\mathcal{P}_B(\mathcal{M}) = \mathcal{P}_I(\mathcal{M}) \cap \{x \mid x(E) = \mathsf{rank}(E)\}$. We will sometimes use m to denote $|E|$ and r to denote the rank of the matroid.

Formal Definition of Problems: An instance of the *Multistage Matroid Maintenance* (MMM) problem consists of a matroid $\mathcal{M} = (E, \mathcal{I})$, an *acquisition cost* $a(e) \geq 0$ for each $e \in E$, and for every timestep $t \in [T]$ and element $e \in E$, a *holding cost* $c_t(e)$. The goal is to find bases $\{B_t \in \mathcal{I}\}_{t \in [T]}$ to minimize

$$\textstyle\sum_t \big(c_t(B_t) + a(B_t \setminus B_{t-1})\big), \tag{2.1}$$

where we define $B_0 := \emptyset$. A related problem is the *Multistage Spanning set Maintenance*(MSM) problem, where we want to maintain a spanning set $S_t \subseteq E$ at each time, and cost of the solution $\{S_t\}_{t \in [T]}$ (once again with $S_0 := \emptyset$) is

$$\textstyle\sum_t \big(c_t(S_t) + a(S_t \setminus S_{t-1})\big). \tag{2.2}$$

Maintaining Bases versus Maintaining Spanning Sets. The following lemma shows the equivalence of maintaining bases and spanning sets. Despite its simplicity, this is what enables us to avoid obstacles faced by previous works.

Lemma 1. *For matroids, the optimal solutions to MMM and MSM have the same costs.*

Proof. Clearly, any solution to MMM is also a solution to MSM, since a base is also a spanning set. Conversely, consider a solution $\{S_t\}$ to MSM. Set B_1 to be any base in S_1. Given $B_{t-1} \subseteq S_{t-1}$, start with $B_{t-1} \cap S_t$, and extend it to any base B_t of S_t. This is the only step where we use the matroid properties—indeed, since the matroid is the same at each time, the set $B_{t-1} \cap S_t$ remains independent at time t, and by the matroid property this independent set can be extended to a base. Observe that this process just requires us to know the base B_{t-1} and the set S_t, and hence can be performed in an online fashion.

We claim that the cost of $\{B_t\}$ is no more than that of $\{S_t\}$. Indeed, $c_t(B_t) \leq c_t(S_t)$, because $B_t \subseteq S_t$. Moreover, let $D := B_t \setminus B_{t-1}$, we pay $\sum_{e \in D} a_e$ for these elements we just added. To charge this, consider any such element $e \in D$, let $t^\star \leq t$ be the time it was most recently added to the cover—i.e., $e \in S_{t'}$ for all $t' \in [t^\star, t]$, but $e \notin S_{t^\star - 1}$. The MSM solution paid for including e at time $t^\star$, and we charge our acquisition of e into B_t to this pair $(e, t^\star)$. It suffices to now observe that we will not charge to this pair again, since the procedure to create $\{B_t\}$ ensures we do not drop e from the base until it is dropped from S_t itself—the next time we pay an addition cost for element e, it would have been dropped and added in $\{S_t\}$ as well. ■

Hence it suffices to give a good solution to the MSM problem. We observe that the proof above uses the matroid property crucially and would not hold, e.g., for matchings. It also requires that the *same* matroid be given at all time steps. Also, as noted above, the reduction is online: the instance is the same, and given an MSM solution it can be transformed online to a solution to MMM.

Elements and Intervals: It is convenient to think of an instance of MSM as being a matroid $\mathcal{M}$, where each element only has an acquisition cost $a(e) \geq 0$, and it has a lifetime $I_e = [l_e, r_e]$. There are no holding costs, but the element e can be used in spanning sets only for timesteps $t \in I_e$. Equivalently, one can think of holding costs being zero for $t \in I_e$ and ∞ otherwise.

An Offline Exact Reduction. The translation is the natural one: given instance $(E, \mathcal{I})$ of MSM, create elements e_{lr} for each $e \in E$ and $1 \leq l \leq r \leq T$, with acquisition cost $a(e_{lr}) := a(e) + \sum_{t=l}^{r} c_t(e)$, and interval $I_{e_{lr}} := [l, r]$. (The matroid is extended in the natural way, where all the elements e_{lr} associated with e are parallel to each other.) The equivalence of the original definition of MSM and this interval view is easy to verify.

An Online Approximate Reduction. Observe that the above reduction created at most $\binom{T}{2}$ copies of each element, and required knowledge of all the costs. If we are willing to lose a constant factor in the approximation, we can perform a reduction to the interval model in an *online* fashion as follows. For element $e \in E$, define $t_0 = 0$, and create many parallel copies $\{e_i\}_{i \in \mathbb{Z}_+}$ of this element (modifying the matroid appropriately). Now the i^{th} interval for e is $I_{e_i} := [t_{i-1}+1, t_i]$, where t_i is set to $t_{i-1}+1$ in case $c_{t_{i-1}+1}(e) \geq a(e)$, else it is set to the *largest* time such that the total holding costs $\sum_{t=t_{i-1}+1}^{t_i} c_t(e)$ for this interval $[t_{i-1}+1, t_i]$ is at most $a(e)$. This interval I_{e_i} is associated with element e_i, which is only available for this interval, at cost $a(e_i) = a(e) + c_{t_{i-1}+1}(e)$.

A few salient points about this reduction: the intervals for an original element e now partition the entire time horizon $[T]$. The number of elements in the modified matroid whose intervals contain any time t is now only $|E| = n$, the same as the original matroid; each element of the modified matroid is only available for a single interval. Moreover, the reduction can be done online: given the past history and the holding cost for the current time step t, we can ascertain whether t is the beginning of a new interval (in which case the previous interval ended at $t-1$) and if so, we know the cost of acquiring a copy of e for the new interval is $a(e) + c_t(e)$. It is easy to check that the optimal cost in this interval model is within a constant factor of the optimal cost in the original acquisition/holding costs model.

3 Offline Algorithms

Being a covering problem, MSM is conceptually easier to solve: e.g., we could use algorithms for submodular set cover [24] with the submodular function being the sum of ranks at each of the timesteps, to get an $O(\log T)$ approximation.

In [17], we give a dual-fitting proof of the performance of the greedy algorithm. Here we give an LP-rounding algorithm which gives an $O(\log rT)$ approximation; this can be improved to $O(\log r)$ in the common case where all acquisition costs are unit. (While the approximation guarantee is no better than that from submodular set cover, this LP-rounding algorithm will prove useful in the online case in Section 4). Finally, the hardness results in [17] show that we cannot hope to do much better than these logarithmic approximations.

3.1 The LP Rounding Algorithm

We now consider an LP-rounding algorithm for the MMM problem; this will generalize to the online setting, whereas it is unclear how to extend the greedy algorithm to that case. For the LP rounding, we use the standard definition of the MMM problem to write the following LP relaxation.

$$\begin{aligned} \min \sum_{t,e} a(e) \cdot y_t(e) + \sum_{t,e} c_t(e) \cdot z_t(e) & & \text{(LP2)} \\ \text{s.t.} \quad z_t \in \mathcal{P}_B(\mathcal{M}) & \qquad \forall t & \\ y_t(e) \geq z_t(e) - z_{t-1}(e) & \qquad \forall t, e & \\ y_t(e), z_t(e) \geq 0 & & \end{aligned}$$

It remains to round the solution to get a feasible solution to MSM (i.e., a spanning set S_t for each time) with expected cost at most $O(\log n)$ times the LP value, since we can use Lemma 1 to convert this to a solution for MMM at no extra cost. The following lemma is well-known, see, e.g. [10].

Lemma 2. *For a fractional base $z \in \mathcal{P}_B(M)$, let $R(z)$ be the set obtained by picking each element $e \in E$ independently with probability z_e. Then $E[\mathsf{rank}(R(z))] \geq r(1 - 1/e)$.*

Theorem 1. *Any fractional solution can be randomly rounded to get solution to MSM with cost $O(\log rT)$ times the fractional value, where r is the rank of the matroid and T the number of timesteps.*

Proof. Set $L = 32\log(rT)$. For each element $e \in E$, choose a random threshold τ_e independently and uniformly from the interval $[0, 1/L]$. For each $t \in T$, define the set $\widehat{S}_t := \{e \in E \mid z_t(e) \geq \tau_e\}$; if $\widehat{S}_t$ does not have full rank, augment its rank using the cheapest elements according to $(c_t(e) + a(e))$ to obtain a full rank set S_t. Since $\Pr[e \in \widehat{S}_t] = \min\{L \cdot z_t(e), 1\}$, the cost $c_t(\widehat{S}_t) \leq L \times (c_t \cdot z_t)$. Moreover, $e \in \widehat{S}_t \setminus \widehat{S}_{t-1}$ exactly when τ_e satisfies $z_{t-1}(e) < \tau_e \leq z_t(e)$, which happens with probability at most

$$\frac{\max(z_t(e) - z_{t-1}(e), 0)}{1/L} \leq L \cdot y_t(e).$$

Hence the expected acquisition cost for the elements newly added to $\widehat{S}_t$ is at most $L \times \sum_e (a(e) \cdot y_t(e))$. Finally, we have to account for any elements added to extend $\widehat{S}_t$ to a full-rank set S_t.

Lemma 3. *For any fixed $t \in [T]$, the set $\widehat{S}_t$ contains a basis of $\mathcal{M}$ with probability at least* $1 - 1/(rT)^8$.

The proof of the lemma is a Chernoff bound, and appears in [17]. Now if the set $\widehat{S}_t$ does not have full rank, the elements we add have cost at most that of the min-cost base under the cost function $(a_e + c_t(e))$, which is at most the optimum value for (LP2). (We use the fact that the LP is exact for a single matroid, and the global LP has cost at least the single timestep cost.) This happens with probability at most $1/(rT)^8$, and hence the total expected cost of augmenting $\widehat{S}_t$ over all T timesteps is at most $O(1)$ times the LP value. This proves the main theorem. ■

Again, this algorithm for MSM works with different matroids at each timestep, and also for intersections of matroids. To see this observe that the only requirements from the algorithm are that there is a separation oracle for the polytope and that the contention resolution scheme works. In the case of $k-$matroid intersection, if we pay an extra $O(\log k)$ penalty in the approximation ratio we have that the probability a rounded solution does not contain a base is $< 1/k$ so we can take a union bound over the multiple matroids.

An Improvement: Avoiding the Dependence on T: When the ratio of the maximum to the minimum acquisition cost is small, we can improve the approximation factor above. We show that essentially the same randomized rounding algorithm (with a different choice of L) gives an approximation ratio of $\log \frac{ra_{max}}{a_{min}}$. We defer the argument to [17].

Hardness for Offline* MSM*: In [17] we show that the MSM and MMM problems are NP-hard to approximate better than $\Omega(\min\{\log r, \log T\})$ even for graphical matroids. We also show an integrality gap of $\Omega(\log T)$.

4 Online MSM

We turn to solving MMM in the online setting. Now, the acquisition costs $a(e)$ are known up-front, but the holding costs $c_t(e)$ for day t are not known before day t. Since the equivalence given in Lemma 1 between MMM and MSM holds even in the online setting, we can just work on the MSM problem. We show that the online MSM problem admits an $O(\log |E| \log rT)$-competitive (oblivious) randomized algorithm. To do this, we show that one can find an $O(\log |E|)$-competitive fractional solution to the linear programming relaxation in Section 3, and then we round this LP relaxation online, losing another logarithmic factor.

4.1 Solving the LP Relaxations Online

Again, we work in the interval model outlined in Section 2. Recall that in this model, for each element e there is a unique interval $I_e \subseteq [T]$ during which it is alive. The element e has an acquisition cost $a(e)$, no holding costs. Once an element has been acquired (which can be done at any time during its interval), it can be used at all times in that interval, but not after that. In the online setting, at each time step t we are told which intervals have ended (and which have not); also, which new elements e are available starting at time t, along with their acquisition costs $a(e)$. Of course, we do not know when its interval I_e will end; this information is known only once the interval ends.

We will work with the same LP as in Section 3.1, albeit now we have to solve it online. The variable x_e is the indicator for whether we acquire element e.

$$\begin{aligned} P := \min \textstyle\sum_e a(e) \cdot x_e & & \qquad\qquad\qquad\qquad \text{(LP3)}\\ \text{s.t.} \quad z_{et} &\in \mathcal{P}_B(\mathcal{M}) & \forall t \\ z_{et} &\le x_e & \forall e, t \in I_e \\ x_e, z_{et} &\in [0,1] \end{aligned}$$

Note that this is not a packing or covering LP, which makes it more annoying to solve online. Hence we consider a slight reformulation. Let $\mathcal{P}_{ss}(\mathcal{M})$ denote the *spanning set polytope* defined as the convex hull of the full-rank (a.k.a. spanning) sets $\{\chi_S \mid S \subseteq E, \mathsf{rank}(S) = r\}$. Since each spanning set contains a base, we can write the constraints of (LP3) as:

$$\mathbf{x}_{E_t} \in \mathcal{P}_{ss}(\mathcal{M}) \qquad \forall t, \text{ where } E_t = \{e : t \in I_e\}. \tag{4.3}$$

Here we define $\mathbf{x}_S$ to be the vector derived from $\mathbf{x}$ by zeroing out the x_e values for $e \notin S$. It is known that the polytope $\mathcal{P}_{ss}(\mathcal{M})$ can be written as a (rather large) set of covering constraints. Indeed, $\mathbf{x} \in \mathcal{P}_{ss}(\mathcal{M}) \iff (\mathbf{1} - \mathbf{x}) \in \mathcal{P}_I(\mathcal{M}^*)$, where $\mathcal{M}^*$ is the dual matroid for $\mathcal{M}$. Since the rank function of M^* is given by $r^*(S) = r(E \setminus S) + |S| - r(E)$, it follows that (4.3) can be written as

$$\begin{aligned} \textstyle\sum_{e \in S} x_e &\ge r(E) - r(E \setminus S) & \forall t, \forall S \subseteq E_t \qquad \text{(LP4)} \\ x_e &\ge 0 & \forall e \in E \\ x_e &\le 1 & \forall e \in E. \end{aligned}$$

Thus we get a covering LP with "box" constraints over E. The constraints can be presented one at a time: in timestep t, we present all the covering constraints corresponding to E_t. We remark that the newer machinery of [8] may be applicable to LP4. We next show that a simpler approach suffices[1]. The general results of Buchbinder and Naor [9] (and its extension to row-sparse covering problems by [16]) imply a deterministic algorithm for fractionally solving this linear program online, with a competitive ratio of $O(\log |E|) = O(\log m)$. However, this is not yet a polynomial-time algorithm, the number of constraints for each timestep being exponential. We next give an adaptive algorithm to generate a small yet sufficient set of constraints.

Solving the LP Online in Polynomial Time: Given a vector $\mathbf{x} \in [0,1]^E$, define $\widetilde{\mathbf{x}}$ as follows:

$$\tilde{x}_e = \min(2\,x_e, 1) \qquad \forall e \in E. \tag{4.4}$$

Clearly, $\widetilde{\mathbf{x}} \leq 2\mathbf{x}$ and $\widetilde{\mathbf{x}} \in [0,1]^E$. We next describe the algorithm for generating covering constraints in timestep t. Recall that [9] give us an online algorithm $\mathcal{A}_{onLP}$ for solving a fractional covering LP with box constraints; we use this as a black-box. (This LP solver only raises variables, a fact we will use.) In timestep t, we adaptively select a small subset of the covering constraints from (LP4), and present it to $\mathcal{A}_{onLP}$. Moreover, given a fractional solution returned by $\mathcal{A}_{onLP}$, we will need to massage it at the end of timestep t to get a solution satisfying all the constraints from (LP4) corresponding to t.

Let $\mathbf{x}$ be the fractional solution to (LP4) at the end of timestep $t-1$. Now given information about timestep t, in particular the elements in E_t and their acquisition costs, we do the following. Given $\mathbf{x}$, we construct $\widetilde{\mathbf{x}}$ and check if $\widetilde{\mathbf{x}}_{E_t} \in \mathcal{P}_{ss}(M)$, as one can separate for $\mathcal{P}_{ss}(\mathcal{M})$. If $\widetilde{\mathbf{x}}_{E_t} \in \mathcal{P}_{ss}(M)$, then $\widetilde{\mathbf{x}}$ is feasible and we do not need to present any new constraints to $\mathcal{A}_{onLP}$, and we return $\widetilde{\mathbf{x}}$. If not, our separation oracle presents an S such that the constraint $\sum_{e\in S} \widetilde{x}_e \geq r(E) - r(E \setminus S)$ is violated. We present the constraint corresponding to S to $\mathcal{A}_{onLP}$ to get an updated $\mathbf{x}$, and repeat until $\widetilde{\mathbf{x}}$ is feasible for time t. (Since $\mathcal{A}_{onLP}$ only raises variables and we have a covering LP, the solution remains feasible for past timesteps.) We next argue that we do not need to repeat this loop more than $2n$ times.

Lemma 4. *If for some* $\mathbf{x}$ *and the corresponding* $\widetilde{\mathbf{x}}$*, the constraint* $\sum_{e\in S} \widetilde{x}_e \geq r(E) - r(E \setminus S)$ *is violated. Then*

$$\textstyle\sum_{e\in S} x_e \leq r(E) - r(E \setminus S) - \frac{1}{2}$$

Proof. Let $S_1 = \{e \in S : \widetilde{x}_e = 1\}$ and let $S_2 = S \setminus S_1$. Let γ denote $\sum_{e\in S_2} \widetilde{x}_e$. Thus

$$|S_1| = \textstyle\sum_{e\in S} \widetilde{x}_e - \sum_{e\in S_2} \widetilde{x}_e < r(E) - r(E \setminus S) - \gamma$$

[1] Additionally, Lemma 4 will be useful in improving the rounding algorithm.

Since both $|S_1|$ and $r(E) - r(E \setminus S)$ are integers, it follows that $|S_1| \leq r(E) - r(E \setminus S) - \lceil \gamma \rceil$. On the other hand, for every $e \in S_2, x_e = \frac{1}{2} \cdot \widetilde{x}_e$, and thus $\sum_{e \in S_2} x_e = \frac{\gamma}{2}$. Consequently

$$\begin{aligned}\textstyle\sum_{e \in S} x_e = \sum_{e \in S_1} x_e + \sum_{e \in S_2} x_e &= |S_1| + \tfrac{\gamma}{2} \\ &\leq r(E) - r(E \setminus S) - \lceil \gamma \rceil + \tfrac{\gamma}{2}.\end{aligned}$$

Finally, for any $\gamma > 0$, $\lceil \gamma \rceil - \frac{\gamma}{2} \geq \frac{1}{2}$, so the claim follows. ∎

The algorithm $\mathcal{A}_{onLP}$ updates $\mathbf{x}$ to satisfy the constraint given to it, and Lemma 4 implies that each constraint we give to it must increase $\sum_{e \in E_t} x_e$ by at least $\frac{1}{2}$. The translation to the interval model ensures that the number of elements whose intervals contain t is at most $|E_t| \leq |E| = m$, and hence the total number of constraints presented at any time t is at most $2m$. We summarize the discussion of this section in the following theorem.

Theorem 2. *There is a polynomial-time online algorithm to compute an $O(\log |E|)$-approximate solution to (LP3).*

We observe that the solution to this linear program can be trivially transformed to one for the LP in Section 3.1. Finally, the randomized rounding algorithm of Section 3.1 can be implemented online by selecting a threshold $t_e \in [0, 1/L]$ the beginning of the algorithm, where $L = \Theta(\log rT)$ and selecting element e whenever $\widetilde{x}_e$ exceeds t_e: here we use the fact that the online algorithm only ever raises x_e values, and this rounding algorithm is monotone. Rerandomizing in case of failure gives us an expected cost of $O(\log rT)$ times the LP solution, and hence we get an $O(\log m \log rT)$-competitive algorithm.

An $O(\log r \frac{a_{max}}{a_{min}})$-Approximate Rounding. The dependence on the time horizon T is unsatisfactory in some settings, but we can do better using Lemma 4. Recall that the $\log(rT)$-factor loss in the rounding follows from the naive union bound over the T time steps. We can argue that when $\frac{a_{max}}{a_{min}}$ is small, we can afford for the rounding to fail occasionally, and charge it to the acquisition cost incurred by the linear program. The details appear in [17].

***Hardness of the online* MMM *and online* MSM.** In [17] we show that any polynomial-time algorithm cannot achieve better than an $\Omega(\log m \log T)$ competitive ratio, via a reduction from online set cover.

5 Perfect Matching Maintenance

We next consider the *Perfect Matching Maintenance* (PMM) problem where E is the set of edges of a graph $G = (V, E)$, and the at each step, we need to maintain a perfect matchings in G. Details and proofs for this Section appear in [17].

Integrality Gap. Somewhat surprisingly, we show that the natural LP relaxation has an $\Omega(n)$ integrality gap, even for a constant number of timesteps.

Hardness. The Perfect Matching Maintenance problem is hard to approximate:

Theorem 3. *For any $\varepsilon > 0$ it is NP-hard to distinguish PMM instances with cost N^{ε} from those with cost $N^{1-\varepsilon}$, where N is the number of vertices in the graph. This holds even when the holding costs are in $\{0, \infty\}$, acquisition costs are 1 for all edges, and the number of time steps is a constant.*

6 Conclusions

Our work suggests several directions for future research. It is natural to study other combinatorial optimization problems, both polynomial time solvable ones such as shortest path and min-cut, as well as NP-hard ones such as min-max load balancing and bin-packing in this multistage framework with acquisition costs. Moreover, the approximability of the *bipartite* matching maintenance, as well as matroid intersection maintenance remains open. Our hardness results for the matroid problem hold when edges have $\{0, 1\}$ acquisition costs. The unweighted version where all acquisition costs are equal may be easier; we currently know no hardness results, or sub-logarithmic approximations for this useful special case.

References

[1] Abernethy, J., Bartlett, P.L., Buchbinder, N., Stanton, I.: A regularization approach to metrical task systems. In: Hutter, M., Stephan, F., Vovk, V., Zeugmann, T. (eds.) Algorithmic Learning Theory. LNCS, vol. 6331, pp. 270–284. Springer, Heidelberg (2010)

[2] Andrews, M., Goemans, M.X., Zhang, L.: Improved bounds for on-line load balancing. Algorithmica 23(4), 278–301 (1999)

[3] Anthony, B.M., Gupta, A.: Infrastructure leasing problems. In: Fischetti, M., Williamson, D.P. (eds.) IPCO 2007. LNCS, vol. 4513, pp. 424–438. Springer, Heidelberg (2007)

[4] Bansal, N., Buchbinder, N., Naor, J(S.): Metrical task systems and the k-server problem on hSTs. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6198, pp. 287–298. Springer, Heidelberg (2010)

[5] Bansal, N., Buchbinder, N., Naor, J.S.: Randomized competitive algorithms for generalized caching. In: STOC 2008, pp. 235–244 (2008)

[6] Borodin, A., Linial, N., Saks, M.E.: An optimal on-line algorithm for metrical task system. J. ACM 39(4), 745–763 (1992)

[7] Buchbinder, N., Chen, S., Naor, J., Shamir, O.: Unified algorithms for online learning and competitive analysis. JMLR 23, 5.1–5.18 (2012)

[8] Buchbinder, N., Chen, S., Naor, J.S.: Competitive analysis via regularization. In: ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 436–444 (2014)

[9] Buchbinder, N., Naor, J.S.: Online primal-dual algorithms for covering and packing. Math. Oper. Res. 34(2), 270–286 (2009)

[10] Calinescu, G., Chekuri, C., Pál, M., Vondrák, J.: Maximizing a submodular set function subject to a matroid constraint (extended abstract), pp. 182–196 (2007)

[11] Cesa-Bianchi, N., Lugosi, G.: Prediction, learning, and games. Cambridge University Press (2006)

[12] Chekuri, C., Vondrák, J., Zenklusen, R.: Submodular function maximization via the multilinear relaxation and contention resolution schemes. In: STOC, pp. 783–792 (2011)

[13] Cohen, E., Cormode, G., Duffield, N.G., Lund, C.: On the tradeoff between stability and fit. CoRR abs/1302.2137 (2013)

[14] Epstein, L., Levin, A.: Robust algorithms for preemptive scheduling. In: Demetrescu, C., Halldórsson, M.M. (eds.) ESA 2011. LNCS, vol. 6942, pp. 567–578. Springer, Heidelberg (2011)

[15] Gu, A., Gupta, A., Kumar, A.: The power of deferral: maintaining a constant-competitive steiner tree online. In: STOC, pp. 525–534 (2013)

[16] Gupta, A., Nagarajan, V.: Approximating sparse covering integer programs online. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012, Part I. LNCS, vol. 7391, pp. 436–448. Springer, Heidelberg (2012)

[17] Gupta, A., Talwar, K., Wieder, U.: Changing bases: Multistage optimization for matroids and matchings. In: arXiv:1404.3768 (2014)

[18] Imase, M., Waxman, B.M.: Dynamic Steiner tree problem. SIAM J. Discrete Math. 4(3), 369–384 (1991)

[19] Megow, N., Skutella, M., Verschae, J., Wiese, A.: The power of recourse for online MST and TSP. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012, Part I. LNCS, vol. 7391, pp. 689–700. Springer, Heidelberg (2012)

[20] Meyerson, A.: The parking permit problem. In: FOCS. pp. 274–284 (2005)

[21] Nagarajan, C., Williamson, D.P.: Offline and online facility leasing. In: Lodi, A., Panconesi, A., Rinaldi, G. (eds.) IPCO 2008. LNCS, vol. 5035, pp. 303–315. Springer, Heidelberg (2008)

[22] Schrijver, A.: Combinatorial Optimization. Springer, Heidelberg (2003)

[23] Shachnai, H., Tamir, G., Tamir, T.: A theory and algorithms for combinatorial reoptimization. In: Fernández-Baca, D. (ed.) LATIN 2012. LNCS, vol. 7256, pp. 618–630. Springer, Heidelberg (2012)

[24] Wolsey, L.A.: An analysis of the greedy algorithm for the submodular set covering problem. Combinatorica 2(4), 385–393 (1982)

Near-Optimal Online Algorithms for Prize-Collecting Steiner Problems

MohammadTaghi Hajiaghayi[2,⋆], Vahid Liaghat[2,⋆,⋆⋆], and Debmalya Panigrahi[1,⋆⋆,⋆⋆⋆]

[1] Computer Science Department, Duke University, Durham, NC 27708
[2] Computer Science Department, University of Maryland, College Park, MD 20742

Abstract. In this paper, we give the first online algorithms with a poly-logarithmic competitive ratio for the node-weighted prize-collecting Steiner tree and Steiner forest problems. The competitive ratios are optimal up to logarithmic factors. In fact, we give a generic technique for reducing online prize-collecting Steiner problems to the fractional version of their non-prize-collecting counterparts losing only a logarithmic factor in the competitive ratio. This reduction is agnostic to the cost model (edge-weighted or node-weighted) of the input graph and applies to a wide class of network design problems including Steiner tree, Steiner forest, group Steiner tree, and group Steiner forest. Consequently, we also give the first online algorithms for the edge-weighted prize-collecting group Steiner tree and group Steiner forest problems with a poly-logarithmic competitive ratio, since corresponding fractional guarantees for the non-prize-collecting variants of these problems were previously known.

For the most fundamental problem in this class, namely the prize-collecting Steiner tree problem, we further improve our results. For the node-weighted prize-collecting Steiner tree problem, we use the generic reduction but improve the best known online Steiner tree result from Naor *et al* [14] on two counts. We improve the competitive ratio by a logarithmic factor to make it optimal (up to constants), and also give a new dual-fitting analysis showing that the competitive ratio holds against the fractional optimum. This result employs a new technique that we call *dual averaging* which we hope will be useful for other dual-fitting analyses as well. For the edge-weighted prize-collecting Steiner tree problem, we match the optimal (up to constants) competitive ratio of $O(\log n)$ that was previously achieved by Qian and Williamson [15] but provide a substantially simpler analysis.

1 Introduction

Over the last two decades, network design problems have been a cornerstone of algorithmic research. The Steiner tree (ST) problem, where the goal is to connect a given set of terminals at minimum cost, and its various generalizations have been central to this research effort. One branch of Steiner problems that has attracted substantial attraction are the so called prize-collecting problems, where the algorithm is permitted to

[⋆] Supported in part by NSF CAREER award 1053605, NSF grant CCF-1161626, ONR YIP award N000141110662, DARPA/AFOSR grant FA9550-12-1-0423.

[⋆⋆] Part of this work was done while the author was at Microsoft Research, Redmond.

[⋆⋆⋆] Supported in part by Duke University startup grant.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 576–587, 2014.

violate one or more connection constraints but must pay corresponding *penalties* in the objective function. Prize collecting (PC) Steiner problems were originally motivated by applications in network planning for service providers (see e.g., [11]). They have since become a well-studied branch of approximation algorithms. In this paper, we focus on the classical online model, where the connectivity demands appear over time and must be immediately satisfied. The study of online prize-collecting network design was initiated very recently by Qian and Williamson [15]. This is somewhat surprising on two counts: first, online versions of Steiner problems appear prominently in the algorithmic literature and prize-collecting variants are a natural generalization; and second, the online model is well-motivated in the context of prize-collecting Steiner problems since in practice, new customers appear over time and network providers must upgrade their networks according to the new demands or lose these customers, thereby paying the corresponding penalty in the revenue. In this paper, we provide a simple generic approach to online prize-collecting Steiner problems that reduces these problems to their fractional non-prize-collecting counterparts losing a logarithmic factor in the competitive ratio. Using known results for the online edge-weighted (EW) and node-weighted (NW) Steiner tree problems, this reduction yields algorithms with competitive ratios of $O(\log^2 n)$ and $O(\log^4 n)$ respectively for their prize-collecting variants. Exploring further, we give improved algorithms for both these problems: for the EW problem, we match the competitive ratio of $O(\log n)$ obtained by Qian and Williamson [15], whereas for the NW problem, we obtain a competitive ratio of $O(\log^3 n)$. Both these results are obtained by employing a novel online dual-fitting approach. Our result represents the first algorithm for online NW-PCST that achieves a poly-logarithmic competitive ratio.

The online Steiner tree (ST) problem was originally considered in the EW model, where Imase and Waxman [10] showed that a natural greedy algorithm has a competitive ratio of $O(\log n)$, which is optimal up to constants. This result was generalized to the online EW Steiner forest (SF) problem by Awerbuch *et al* [3], who showed that the greedy algorithm has a competitive ratio of $O(\log^2 n)$. This result was later improved by Berman and Coulston [4] to $O(\log n)$. All the above results can be reproduced using dual-fitting techniques (though the original expositions relied on combinatorial arguments). This immediately shows that the competitive ratios hold against the respective fractional optimums as well. Very recently, Qian and Williamson [15] initiated the study of online PC Steiner problems by providing an $O(\log n)$-competitive algorithm for the online EW-PCST problem. The analysis of this algorithm is quite complicated and uses a dual moat growing approach of Goemans and Williamson [7] and an amortized accounting scheme due to Berman and Coulston [4].

In contrast to EW problems, progress in online NW Steiner problems has been relatively slow. Note that edge weights can be represented by node weights but not vice-versa; so, NW problems are strictly more general. In fact, the NW-ST problem generalizes the set cover problem, for which the first online algorithm with a poly-logarithmic competitive ratio was obtained by Alon *et al* [2]. They introduced an online adaptation of the classical LP relaxation technique, which has since been used extensively in online optimization (see, e.g., the survey by Buchbinder and Naor [5]). In particular, Naor *et al* [14] used this technique in conjunction with structural properties of the NW-ST problem to give an $O(\log^3 n)$-competitive algorithm for the online

NW-ST problem. They left the online NW-SF problem open, which was resolved very recently by Hajiaghayi *et al* [9] who obtained an identical competitive ratio of $O(\log^3 n)$ for the SF problem. However, there is a crucial difference between these two results. Whereas Hajiaghayi *et al* use a dual-fitting approach, which shows that the competitive ratio holds for the fractional optimum as well, Naor *et al* use structural properties of *integral* Steiner trees. Therefore, their results do not hold for the fractional optimum. As described below, this distinction turns out to be important in our work.

Our Techniques and Contributions. Our first contribution is a simple but subtle reduction of online prize-collecting Steiner problems to their respective non-prize-collecting fractional variants losing a factor of $O(\log n)$ in the competitive ratio. This reduction is quite generic and can be applied for more general problems than ST and SF. Indeed, this approach can be applied to any problem which demands $\{0,1\}$-connectivity on a family of cuts. This setting includes the T-join problem and group Steiner tree (GST)/ group Steiner forest (GSF) problems as special cases (see Theorem 3 for a formal description). All these problems are instances of covering problems. In our reduction, we run the algorithm of Buchbinder *et al* [5] for solving covering problems *in parallel* with an algorithm for the non-prize-collecting variant (as a black box). At each online step, we first generate an online competitive fractional solution. Then we use the fractional solution to reveal a modified demand to the non-prize-collecting black box and finally output an integral solution for the prize-collecting problem. This reduction is oblivious of the cost model of the input graph, and hence can be applied to both EW and NW problems, thereby yielding online algorithms for various prize-collecting Steiner connectivity problems with poly-logarithmic competitive ratios. Indeed one main obstacle to solving the online prize-collecting variants of these problems is that the known rounding techniques do not seem to be effective for rounding the standard linear relaxation of the problems. For example for the online NW prize-collecting ST, it is not known whether one can solve the fractional LP for PCST and then round it online. This may have been the reason that the only PC result known before this work, i.e, the Qian-Williamson algorithm [15] for online EW PCST, is quite sophisticated. A summary of results that follow from our reduction are shown in Table 1.

Table 1. The competitive ratio for online prize-collecting (PC) Steiner problems. Note that the algorithm for NW Group Steiner Forest runs in quasi-polynomial time.

	PC ST	PC SF	PC SF in Planar graphs	PC GSF
EW	$O(\log(n))$[15] Simplified in this paper	$O(\log(n))$[15]	$O(\log(n))$[15]	$O(\log^7(n))$
NW	$O(\log^3(n))$	$O(\log^4(n))$	$O(\log^2(n))$	$O(\log^{11}(n))$ (quasi-polynomial)

Next, we focus on the online EW-PCST and NW-PCST problems. For these problems, the generic reduction yields competitive ratios of $O(\log^2 n)$ and $O(\log^4 n)$ respectively. We improve both these competitive ratios by a logarithmic factor. For the EW problem, this matches the competitive ratio of the Qian-Williamson algorithm [15] and is optimal up to constants. Further, our analysis is extremely simple and uses a natural dual-fitting approach. Due to lack of space, we refer the reader to the full version of the paper for the EW algorithm.

Theorem 1 (also in [15]). *The online edge-weighted prize-collecting Steiner tree problem admits an $O(\log(n))$-competitive algorithm.*

For the online NW-PCST problem, we use the generic reduction, but give an online algorithm for the NW-ST problem that has the optimal competitive ratio of $O(\log^2 n)$ *against the fractional objective*. While it is relatively straightforward to use a re-scaling argument for improving the integral competitive ratio of the algorithm of Naor *et al* [14] by a logarithmic factor, a similar improvement for the (fractionally competitive) algorithm of Hajiaghayi *et al* [9] has fundamental difficulties. So, we first design a novel dual-fitting analysis of the algorithm of Naor *et al*, thereby proving that the competitive ratio of the algorithm now holds against the fractional optimum. However, using this new analysis, we can no longer use the re-scaling argument that improved a logarithmic factor for the integral algorithm. To overcome this difficulty, we introduce a new concept that we call *dual averaging*.

The celebrated moat-growing method of Agrawal, Klein, and Ravi [1] and Goemans and Williamson [7] has been extensively studied for various Steiner connectivity problems. A general pattern in different variants of this method is as follows. We start by growing a moat over every terminal. When two moats collide, this signifies an opportunity for connecting the terminals at the center of the moats thus merging the moats. Since the moats have a dual vector interpretation, by weak duality one can charge the cost of an algorithm to the growth of the moats. When used in a dual-fitting argument, the crux of the analysis is to show that the dual moats do not intersect. In the NW setting, this turns out to be even more difficult since the next terminal that arrives might quickly collide with polynomially many disks at the same vertex thereby making merges of the moats infeasible. We circumvent this problem by introducing a *thinness* factor for the moats. For $\tau \in (0,1]$, a τ-thin dual moat is obtained by scaling the dual variables corresponding to a moat by the factor τ. This allows us to have several *overlapping* moats; this is in contrast to standard dual-fitting methods in which the dual moats are disjoint. The strength of this natural modification is that one can exploit it to show that certain *structural properties* may hold on *average*, although they may not hold for every dual moat independently. For example, consider the feasibility of a dual vector. A reader familiar with the standard dual program for ST may recall that every moat has a load on vertices inside or on the boundary of a moat. It often happens that for every vertex, a few moats have a high load on the vertex while the load of the rest of the moats is negligible. However, for different vertices, the moats with a high load might be different. By considering the proper thinness for the moats, one can balance the loads simultaneously for every vertex to ensure feasibility of the dual moats. We refer the reader to Section 3 for a formal discussion of this approach.

Theorem 2. *The online node-weighted Steiner tree problem admits an $O(\log^2(n))$-competitive algorithm. Moreover, the competitive ratio holds with respect to the optimal fractional solution.*

Note that the above algorithm is optimal in its competitive ratio since there is a known lower bound of $\Omega(\log^2 n)$ [2,13] for the online set cover problem, which is a special case of the online NW-ST (and thus online NW-PCST) problem.

Applying the reduction in Theorem 3 to the algorithm in Theorem 2, we obtain an improved competitive ratio for the online NW-PCST problem. Furthermore, very recently, Hajiaghayi *et al* [9] gave an algorithm with a tight competitive ratio of $O(\log(n))$ for the NW ST problem for planar graphs and more generally graphs excluding a fixed graph as a minor. Their results are based on a primal-dual technique and thus the competitive ratio is w.r.t. the fractional optimum. Therefore we get the following results for the prize-collecting counterparts of these problems.

Corollary 1. *The online NW prize-collecting ST problem admits an $O(\log^3(n))$-competitive algorithm in general graphs. When restricted to graphs excluding a fixed graph as a minor, the problem admits an $O(\log^2(n))$-competitive algorithm.*

In the offline paradigm, algorithms for prize-collecting problems are used at the heart of algorithms for other connectivity problems. An important branch of such problems are *budgeted*[1] and *quota*[2] problems. The key to solving these problems is to design Lagrangian multiplier preserving (LMP) approximation algorithms for PCST. Indeed very recently, Konemann *et al.* [12] gave an LMP $O(\log(n))$-approximation algorithm for offline NW PCST. Therefore, a natural question is whether one can hope for LMP *online* algorithms with poly-logarithmic competitive ratio. We show this is impossible even for the case of online edge-weighted Steiner tree. In the full version of the paper, we show a lower bound of $\Omega(n)$ for the LMP competitive ratio of (randomized) algorithms for online EW-PCST.

2 Online Prize-Collecting Network Design

Let $G = (V,E)$ be an undirected graph. For a set $S \subseteq V$, let $\delta(S) \subseteq V \backslash S$ denote the neighbors of S. Given a $\{0,1\}$-function $f : 2^V \to \{0,1\}$, a *demand system* with respect to f is defined as the following system of inequalities over a set of variables $\mathbf{x}(v)$ for every $v \in V$.

$$\sum_{v\in\delta(S)} \mathbf{x}(v) \geq f(S) \qquad \text{for every } S \subset V, S \neq \phi$$

$$\mathbf{x}(v) \in [0,1]$$

We say a $\{0,1\}$-function f is *efficient* if the following properties hold: (i) $f(\phi) = f(V) = 0$; (ii) for every $S \subseteq V$, $f(S)$ can be computed in polynomial time; and (iii) there exists a separation oracle such that for any vector $\mathbf{x} \in [0,1]^V$, it outputs a set S with $\sum_{v\in\delta(S)} \mathbf{x}(v) < f(S)$ iff there is a violated constraint. The oracle should run in polynomial time w.r.t. $|V|$.

As we will see, the last two properties are required so that our reduction runs in polynomial time. Although the first property is not necessary, it makes the demand

[1] Given an upper limit on the weight, the goal is to maximize the prize of the connected vertices.

[2] We want the minimum-weight subgraph that gathers at least a given amount of prize.

system well defined even if we consider the constraints corresponding to $S = \phi$ and $S = V$ in the system. In the rest of the section, we use $\mathscr{F}$ to refer to a family of efficient functions. Furthermore, we use $\mathscr{G}$ to refer to a family of node-weighted graphs $G = (V, E, w)$ where for $v \in V$, $w_v \in \mathbb{R}_{\geq 0}$.

Problem. A *network design problem* (ND) with respect to $\mathscr{F}$ and $\mathscr{G}$ is defined as follows. Let $G = (V, E, w)$ be a node-weighted graph in $\mathscr{G}$. Given a sequence of functions $f_1, \ldots, f_k \in \mathscr{F}$, the goal is to find a minimum-weight vector $\mathbf{x} \in \{0,1\}^V$ that simultaneously satisfies the demand systems for every function f_i. The ND problem can be formulated as the following integer program (IP). Throughout the paper, for an integer k, let $[k]$ denote $\{1, \ldots, k\}$.

$$\begin{aligned} &\text{minimize} \sum_{v \in V} w_v \mathbf{x}(v) \\ \forall S \subseteq V, i \in [k] &\sum_{v \in \delta(S)} \mathbf{x}(v) \geq f_i(S) \\ &\mathbf{x}(v) \in \{0,1\} \end{aligned} \tag{$\mathbb{ND}$}$$

Given a feasible solution $\mathbf{x}$, we define $cost(\mathbf{x})$ as the total *weight* of $\mathbf{x}$, i.e, $\sum_{v \in V} w_v \mathbf{x}(v)$.

In a *prize-collecting network design problem* (PCND) w.r.t. $\mathscr{F}$ and $\mathscr{G}$, we are given $G = (V, E, w) \in \mathscr{G}$ and a sequence of *demands* $(f_1, \pi_1), \ldots, (f_k, \pi_k)$ where $f_i \in \mathscr{F}$ and $\pi_i \in \mathbb{R}_{\geq 0}$. For every demand (f_i, π_i), we need to either satisfy the demand system w.r.t. f_i or pay the *penalty* π_i. In other words, we need to find an optimal solution to the following IP.

$$\begin{aligned} &\text{minimize} \sum_{v \in V} w_v \mathbf{x}(v) + \sum_{i \in [k]} \pi_i \mathbf{z}(i) \\ \forall S \subseteq V, i \in [k] &\sum_{v \in \delta(S)} \mathbf{x}(v) \geq f_i(S)(1 - \mathbf{z}(i)) \\ &\mathbf{x}(v), \mathbf{z}(i) \in \{0,1\} \end{aligned} \tag{$\mathbb{PCND}$}$$

Given a feasible solution $(\mathbf{x}, \mathbf{z})$ to the IP, we define $cost(\mathbf{x}, \mathbf{z})$ as the weight of $\mathbf{x}$ plus the total penalty $\sum_{i \in [k]} \pi_i \mathbf{z}(i)$.

In what follows, we denote the cost of optimal solutions to the programs $\mathbb{ND}$ and $\mathbb{PCND}$ by OPT_{ND} and OPT_{PCND}. One can also relax the integrality constraints in both IPs to get corresponding linear relaxations. We denote the cost of the optimal fractional solutions of the corresponding linear programs (LP) by OPT^*_{ND} and OPT^*_{PCND}.

Online Setting. In the online variants of network design problems and their prize-collecting counterparts, demands arrive sequentially. However, we assume that the node-weighted graph $G = (V, E, w)$ is known in advance. More precisely, in an online prize-collecting network design problem (OPCND), at time $t \in [k]$, a new demand (f_t, π_t) arrives and we need to output a feasible solution $(\mathbf{x}^t, \mathbf{z}^t)$ for the integer program $\mathbb{PCND}$. The decisions are online in the sense that an online algorithm may only increase the values of the variables, i.e., for every $t' < t$, $\mathbf{x}^{t'}(v) \leq \mathbf{x}^t(v)$ and $\mathbf{z}^{t'}(i) \leq \mathbf{z}^{t'}(i)$, for every $v \in V$ and $i \in [k]$.

Consider an algorithm ALG for OPCND and a sequence of demands $\rho = (f_1,\pi_1),\dots,(f_k,\pi_k)$. Let ALG$(\rho)$ denote the cost of the output of ALG on the online input ρ, i.e., ALG$(\rho) = cost(\mathbf{x}_k,\mathbf{z}_k)$. ALG is *$\alpha$-competitive* w.r.t. $\mathscr{G}$ and $\mathscr{F}$, if for every $G \in \mathscr{G}$ and every sequence of demands $\rho = (f_1,\pi_1),\dots,(f_k,\pi_k)$ where $f_i \in \mathscr{F}$, we have ALG$(\rho) \leq \alpha OPT_{PCND}$. ALG is *strongly α-competitive*, if for every ρ, ALG$(\rho) \leq \alpha OPT^*_{PCND}$. One can define similar notations for online network design (OND) problems by dropping penalties and replacing PCND indices by ND. The main result of this section is the following reduction.

Theorem 3. *Let $\mathscr{G}$ and $\mathscr{F}$ respectively denote a family of graphs and a family of feasible functions. Given a strongly α-competitive algorithm for an online network design problem (OND) w.r.t. $\mathscr{G}$ and $\mathscr{F}$, one can derive a strongly competitive algorithm for the corresponding OPCND with a competitive ratio of $\alpha O(\log(|V|))$.*

Before we prove Theorem 3, we need to recall the following theorem by Buchbinder *et al* [6] (later improved by Gupta and Nagarajan [8][3]). Consider a minimization LP in the form that given a vector c and a matrix A, minimizes $c \cdot \mathbf{x}$ subject to $A\mathbf{x} \geq \mathbf{1}$.[4] A *covering LP* is a special case where all the entries of A are non-negative. In an *online covering problem*, the vector c is known in advance, however, the covering constraints arrive online. After the arrival of a new constraint, the online algorithm needs to output a (fractional) feasible solution without decreasing the previous values of variables.

Theorem 4 (Theorem 4.2 of [6]). *Let n be the number of variables. There exists an algorithm for the online covering problem which finds a fractional solution with the cost within $O(\log(n))$ factor of the optimal fractional solution. Furthermore, the algorithm only increases a variable if it has a positive coefficient in the new constraint.*

Consider a non-trivial constraint[5] of the program $\mathbb{PCND}$ corresponding to a function f_i and a subset S. It can be re-written in the following standard format:

$$\mathbf{z}(i) + \sum_{v \in \delta(S)} \mathbf{x}(v) \geq 1$$

Thus all the constraints are covering constraints. Suppose we want to solve OPCND fractionally. We note that OPCND is not formally a special case of the online covering problem in two aspects. First, the objective function is not fully known, i.e., the prizes are revealed online. Second, in OPCND all constraints corresponding to f_i are revealed at the same time while in an online covering problem, we assume that the constraints are revealed one by one. The former is easy to handle since by Theorem 4, the algorithm of Buchbinder *et al* [5] only changes a variable when it has a positive coefficient in a newly arrived constraint. Thus the variable $\mathbf{z}(i)$ changes only in step i after receiving the demand (f_i,π_i). The second discrepancy can be handled by using the efficiency of function f_i. At any step, let $(\mathbf{x}^*,\mathbf{z}^*)$ denote the current fractional solution. While the solution is not feasible, we find an infeasible constraint and reveal it to the online covering

[3] The competitive ratio in [8] is improved to $O(\log(k))$ where k is the maximum number of non-zero entries in a row.

[4] Let **1** and **0** denote the vectors where all the entries are one and zero, respectively.

[5] We say a constraint is trivial if $f_i(S) = 0$

algorithm. This can be done in polynomial time since f_i is efficient. We continue this process until all the constraints are feasible.

Therefore the algorithm of Buchbinder and Naor can be applied to OPCND to obtain a fractional solution $(\mathbf{x}^*, \mathbf{z}^*)$ such that by Theorem 4, $cost(\mathbf{x}^*, \mathbf{z}^*) \leq OPT^*_{PCND} O(\log(|V|))$. Note that although the algorithm may increase the value of a variable beyond one, this can be ignored since feasibility is maintained even if we decrease such variables to one.

Algorithm. Let ALG$_{OND}$ be a strongly α-competitive algorithm for OND. The following algorithm for OPCND realizes Theorem 3. Let $\rho = (f_1, \pi_1), \ldots, (f_k, \pi_k)$ denote the online input. We run the online fractional algorithm of Buchbinder *et al* and an instance of ALG$_{OND}$ in parallel. At any time step, let $(\mathbf{x}^*, \mathbf{z}^*)$ denote the (partial) output of the fractional algorithm and let $\mathbf{x}$ denote the (partial) output of ALG$_{OND}$, respectively. We also maintain an integral vector $\mathbf{z}$ which shows the integral decisions of our algorithm for paying the penalties of demands that have arrived.

At step i, we receive the new demand (f_i, π_i). We reveal the new demand to the online fraction algorithm which in return updates the values of $(\mathbf{x}^*, \mathbf{z}^*)$. In particular, it sets the first and final value of $\mathbf{z}^*(i)$. Now if $\mathbf{z}^*(i) \geq 1/2$, we pay the penalty of the new demand and set $\mathbf{z}(i) = 1$. Otherwise, we set $\mathbf{z}(i) = 0$, and we reveal the function f_i to the instance of ALG$_{OND}$. At the end of iteration, we report $(\mathbf{x}, \mathbf{z})$ as the output of our algorithm. For a detailed analysis, the reader is referred to the full version of the paper.

3 An Asymptotically Optimal Algorithm for Online NW ST

The online node-weighted Steiner tree (NW-ST) problem is a fundamental OND problem: given a vertex root, every input function f_i characterizes the cuts that separate a vertex t_i from the root.

The Online Node-weighted Steiner Tree problem. We are given an undirected connected graph $G = (V, E)$ where w_v is the weight of vertex $v \in V$. Let $n = |V|$. The online input comprises a sequence of vertices $t_0, t_1, t_2, \ldots, t_k$ where $t_i \in V$. The output comprises a sequence $H_0, H_1, H_2, \ldots, H_k$, where (i) H_i is a connected subgraph of G; (ii) the terminals $\{t_0, t_1, t_2, \ldots, t_i\}$ are connected in H_i; and (iii) H_i is a subgraph of H_{i+1}. The objective is to minimize the total weight of vertices in H_k. Without loss of generality, we assume that the weight of every terminal is zero[6]. For simplicity, we will assume that the cost of the optimal solution is n and for every vertex v, $w_v \in (0, n]$. This is wlog up to a constant factor loss in the competitive ratio[7].

Our algorithm follows the approach of Naor *et al* for solving the problem via an instance of a facility location problem. Although our algorithm is very similar to that of Naor *et al*, our analysis is quite different; while they use a combinatorial fact to prove the competitive ratio of the algorithm, we use the technique of dual averaging that we alluded to in the introduction. This allows us to establish the tight competitive ratio of $O(\log^2(n))$ with respect to the fractional solution.

[6] For every vertex v, attach a dummy vertex ρ_v with weight zero to v. Upon receiving a terminal t, we virtually assume that the terminal is ρ_t.

[7] By an online doubling strategy, we may assume that we know the objective value α of an optimal solution. We multiply all vertex weights by n/α so that the cost of the optimal solution is n. All vertices v with $w_v > n$ are discarded, while we set $x_v = 1$ for those satisfying $w_v \leq 1$.

An Auxiliary Linear Program. Let Γ denote the set of compound indices $V \cup (V \times [k])$, i.e., for every $v \in V$ and $i \in [k]$, both $v \in \Gamma$ and $(v,i) \in \Gamma$. A set $S \subseteq \Gamma$ is an *auxiliary cut* for a terminal t_i, if for every vertex v, S contains exactly one of v and (v,i). Let $\mathbf{S}_i$ denote the collection of all auxiliary cuts for t_i, i.e., for $i \in [k]$, $\mathbf{S}_i = \{S \subseteq \Gamma | \forall_{v \in V} \ |S \cap \{v,(v,i)\}| = 1\}$. For every $v \in V$ and $i \in [k]$, let $P_{(v,i)}$ denote a minimum-weight path between t_i and any previous terminal t_j (i.e. $0 \leq j < i$) which goes through v. For every (compound) index $\gamma \in \Gamma$ we define a weight $\mathbf{w}_\gamma$ as follows. For every $v \in V$, let $\mathbf{w}_v = w_v$. For every $v \in V$ and $i \in [k]$, $\mathbf{w}_{(v,i)}$ is the weight of $P_{(v,i)}$ minus the weight of v.

Consider the *auxiliary linear program* ALP (given below) with a variable $\mathbf{x}_\gamma$ for every index $\gamma \in \Gamma$. Let $\mathbf{x}$ be a feasible solution to the Program ALP. Observe that for every $v \in V$ and $i \in [k]$, we may assume $\mathbf{x}_{(v,i)} \leq \mathbf{x}_v$; otherwise by reducing $\mathbf{x}_{(v,i)}$ to $\mathbf{x}_v$ we can decrease the objective value while keeping the solution feasible[8]. Therefore in the rest of section, wlog, we assume that for every feasible solution, $\mathbf{x}_{(v,i)} \leq \mathbf{x}_v$.

$$\begin{array}{rlr} \text{minimize} & \sum_{\gamma \in \Gamma} \mathbf{w}_\gamma \mathbf{x}_\gamma & \text{(ALP)} \\ \forall\, i \in [k], S \in \mathbf{S}_i & \sum_{\gamma \in S} \mathbf{x}_\gamma \geq 1 & \text{(P1)} \\ & \mathbf{x}_\gamma \in [0,1] & \end{array} \qquad \begin{array}{rlr} \text{minimize} & \sum_{\gamma \in \Gamma} \tilde{\mathbf{w}}_\gamma \mathbf{x}_\gamma & \text{(SALP)} \\ \forall\, i \in [k], S \in \mathbf{S}_i & \sum_{\gamma \in S} \mathbf{x}_\gamma \geq 1 & \text{(P2)} \\ & \mathbf{x}_\gamma \in [0,1] & \end{array}$$

It is easy to verify that for every *integral* feasible solution $\mathbf{x}$ for Program ALP, there exists an integral solution for the Steiner tree instance having cost at most the same as the objective value of the program.

A main obstacle in solving the online ST problem is that the known rounding methods are not effective in rounding a fractional solution of the linear relaxation of standard programs for ST. Indeed an important property of the auxiliary LP is that a standard rounding method similar to the Set Cover problem can be used to round the solution by losing (roughly) a logarithmic factor. However, although one can obtain an integral solution for ST with the same cost as that for Program ALP, the converse does not hold. Naor *et al* [14] use a combinatorial decomposition of an *integral* solution for ST to show that the converse holds if one is willing to incur a factor of $O(\log^2(n))$ in the cost. This combinatorial fact does not have a fractional counterpart which is crucial to our reduction in solving PCST. Furthermore, using Program ALP leads to a competitive ratio of $O(\log^3(n))$ after applying the rounding method, which is off by a logarithmic factor from the known lower bound. We overcome both obstacles by using a dual averaging argument to show a similar relationship between *fractional* solution for ST and that for a *scaled auxiliary LP*.

We define a *scaled weight* $\tilde{\mathbf{w}}$ over the set of compound indices Γ as follows. For every $v \in V$ and $i \in [k]$, let $\tilde{\mathbf{w}}_{(v,i)} = \mathbf{w}_{(v,i)}$, while for every $v \in V$, let $\tilde{\mathbf{w}}_v = w_v \log(n)$.[9] The scaled auxiliary program SALP is given above. We split the objective function of this LP into two parts. For a feasible vector $\mathbf{x}$ for SALP, let the *facility cost*, $FacCost(\mathbf{x}) = \sum_{v \in V} \tilde{\mathbf{w}}_v \mathbf{x}_v$

[8] Recall that for every auxiliary cut S that contains (v,i), there exists a cut S' that replaces (v,i) with v. Thus if constraint P1 is feasible for S', by reducing $\mathbf{x}_{(v,i)}$ to $\mathbf{x}_v$, the constraint corresponding to S remains feasible.

[9] In the rest of the section, the base of all logarithmic terms is 2.

and let the *connection cost*, $ConCost(\mathbf{x}) = \sum_{v\in V, i\in[k]} \tilde{\mathbf{w}}_{(v,i)} \mathbf{x}_{(v,i)}$. We may drop $\mathbf{x}$ from the notation when the vector is clear from the context. Observe that a feasible solution for SALP yields a feasible solution for NW ST with total cost at most $\frac{FacCost}{\log(n)} + ConCost$.

Algorithm. We first find an online[10] fractional solution for SALP using the method of multiplicative updates (a formal discussion of this method is presented in the full version of the paper). We then show the objective value of this fractional solution is within $O(\log^2(n))$ factor of the optimal fractional solution for ST. Note that the objective function of the program uses the scaled weights for vertices. A feasible solution for the *scaled program* can be rounded online with an additional loss of a *constant* factor using the standard rounding techniques. We will not describe this rounding procedure here; we refer the reader to Section 2 of [14].

Analysis. For a subset of vertices $S \subset V$, let $\delta(S) \subseteq V \backslash S$ denote the neighbors of S. Let $\mathscr{S}$ denote the collection of subsets of vertices that separate a subset of terminals from the terminal t_0, i.e, $S \in \mathscr{S}$ if and only if $S \cap \{t_1, \ldots t_k\} \neq \phi$ and $t_0 \notin S$. Consider the natural LP relaxation for the NW ST problem in which there is a flow of one going out of every set in $\mathscr{S}$. The primal dual pair for this LP is as follows.

$$\begin{array}{llllll} \textit{minimize} & \sum_{v\in V} w_v x_v & \textit{maximize} & \sum_{S\in\mathscr{S}} y(S) & \\ \forall S \in \mathscr{S} & \sum_{v\in\delta(S)} x_v \geq 1 & \forall v & \sum_{S\in\mathscr{S}: v\in\delta(S)} y(S) \leq w_v & \text{(D1)} \\ & x_v \geq 0 & & y(S) \geq 0 & \end{array}$$

For two vertices u and v, let $d(u,v)$ denote the weight of shortest path between u and v *excluding* the weight of the endpoints. We borrow the notation of Hajiaghayi *et al* [9] for disks. A *painting* is a function $p : V \to \mathbb{R}_{\geq 0}$. A painting is *valid* if $p(v) \leq w_v$ for every vertex v. Intuitively, every vertex has an area equal to its weight and a painting is a (partial) coloring of these areas. The union of a set of paintings $p_1, \ldots, p_\ell$ is the painting p with $p(v) = \sum_{i\in[\ell]} p_i(v)$ for every $v \in V$.

Recall that the weight of a terminal is zero. A *disk* of radius r centered at terminal t, is a painting in which the area within a radius r of t is colored, i.e.,

$$p(v) = \begin{cases} w_v & \text{if } d(t,v) + w_v \leq r \\ r - d(t,v) & \text{if } d(t,v) + w_v > r \text{ but } d(t,v) \leq r \\ 0 & \text{if } d(t,v) \geq r \end{cases}$$

A disk p centered at a terminal t_i for $i \in [k]$ is *feasible* if $p(t_0) = 0$.

A *disk vector* corresponding to a disk p of radius r centered at a terminal t is a dual vector y generated by the following deterministic process:

```
1: Initialize $y = \mathbf{0}$ and $U = \{t\}$.
2: while the total dual objective $\sum_{S\in\mathscr{S}} y(S)$ is less than $r$ do
```

[10] In the online setting, at time step $i \in [k]$, the constraints for sets $S \in \mathbf{S}_i$ are revealed and the algorithm should output a feasible solution. However, the online algorithm may only increase the previous values of variables in each time step.

```
3:  Continuously increase $y(U)$ until $\sum_{S\in\mathscr{S}} y(S)$ reaches $r$, or for a vertex $v$ the dual
    constraint D1 becomes tight.
4:  If the latter happens, add $v$ to $U$.
```

For a dual vector y, define the *load* of y on a vertex v as $\sum_{S\in\mathscr{S}:v\in\delta(S)} y(S)$. The construction of a disk vector directly yields the following lemma (due to lack of space, we refer the reader to the full version of the paper for the proofs in this section).

Lemma 1. *Let y be a disk vector corresponding to a disk p. For every $v \in V$, the load of y on v is exactly $p(v)$.*

Corollary 2. *Let y be a disk vector corresponding to a disk of radius r centered at t. For every vertex $v \in V$ on which the load of y is strictly positive, we have $r \geq d(t,v)$. Furthermore, y is a feasible dual vector if and only if the disk is feasible, i.e., $r \leq d(t,t_0)$.*

For an arbitrary *thinness* factor $\tau \in (0,1]$, a τ-thin disk vector is a dual vector y' obtained by scaling the disk vector y by a factor of τ, i.e., $y'(S) = \tau y(S)$ for every $S \in \mathscr{S}$. When the thinness factor is clear from the context, we may refer to y' as simply a thin disk vector. We note that a τ-thin disk vector is feasible if and only if the corresponding (1-thin) disk vector is feasible. Observe that the total dual objective value of a τ-thin disk vector with radius r is exactly $r \times \tau$. Similar to paintings, the union of a set of disk-vectors $y_1,\dots,y_\ell$ is the dual vector y where $y(S) = \sum_{i\in[\ell]} y_i(S)$ for every $S \subseteq \mathscr{S}$.

Multiplicative Updates Method. Consider the cost of the fractional solution generated by the multiplicative steps. In what follows, let *opt* denote the cost of the optimal fractional solution for the NW-ST problem. Recall that we assume $opt = n$. In our algorithm, for every $\gamma \in \Gamma$, we initialize $\mathbf{x}_\gamma$ to $\frac{1}{n^3}$ (or to one if $\tilde{\mathbf{w}}_\gamma = 0$). Furthermore, for $v \in V$ and $i \in [k]$, $\tilde{\mathbf{w}}_{(v,i)} \leq n^2$ since we have assumed that the weight of a single vertex is at most n and a simple path may contain at most n vertices. SALP has at most n^2+n variables. Thus the objective cost of the initialization is at most $(n^2+n)(n^2)\frac{1}{n^3} \leq 2n = 2opt$. Hence the cost of initialization adds a constant factor to the competitive ratio; we will ignore this factor in the remaining analysis.

Using the notion of thin disks, we will now establish the competitive ratio of the algorithm. Consider the cost of the fractional solution generated by the multiplicative steps. Using standard techniques, one can show that to bound the cost of the algorithm, it is sufficient to bound the number of multiplicative steps. Therefore we use a union of a set of disks to account for the *number* of the multiplicative updates.

Lemma 2. *Let FacCost and ConCost respectively denote the facility cost and the connection cost of the output of the algorithm. Then $FacCost + ConCost \leq O(\log^2(n))opt$.*

For $i \in [k]$, let s_i denote the number of multiplicative steps done for terminal t_i. We assume wlog that $s_i \geq 1$. Let $\tau = \frac{1}{4\log(n)}$. For every $i \in [k]$, consider a τ-thin disk vector y_i corresponding to a disk of radius $r_i = \frac{s_i}{5\log(n^3)}$ centered at terminal t_i. We first show for every $i \in [k]$, the thin disk vector y_i is indeed feasible. Note that for terminal t_i and in every auxiliary cut we should have either t_0 or (t_0,i). However, the cost of a terminal, in particular that of t_0, is zero. Hence, $\mathbf{x}_{t_0}$ is initialized to one and thus only (t_0,i) can participate in a multiplicative step. Thus using standard arguments, one can show that

$s_i \leq \tilde{\mathbf{w}}_{(t_0,i)} \log(n^3)$. Recall that $\tilde{\mathbf{w}}_{(t_0,i)}$ is the weight of shortest path between t_i and a terminal t_j for $j < i$, which passes through t_0. Hence by choosing $j = 0$, $\tilde{\mathbf{w}}_{(t_0,i)} \leq d(t_i,t_0)$. Thus the radius of the disk is at most $r_i = \frac{s_i}{5\log(n^3)} \leq \frac{d(t_0,t_i)}{5}$, which by Corollary 2 verifies that y_i is feasible.

Let y denote the union of y_i's for every $i \in [k]$. We prove Lemma 2 by showing that the dual vector y is feasible. Finally as mentioned before, by using a standard rounding method, Theorem 2 follows. We refer the reader to the full version of the paper for a formal analysis.

Acknowledgement. D. Panigrahi would like to thank Niv Buchbinder for suggesting the re-scaling trick to remove an additional logarithmic factor from the competitive ratio of the integral algorithm of Naor *et al* [14].

References

1. Agrawal, A., Klein, P., Ravi, R.: When trees collide: an approximation algorithm for the generalized steiner problem on networks. In: STOC, pp. 134–144 (1991)
2. Alon, N., Awerbuch, B., Azar, Y., Buchbinder, N., Naor, J.: The online set cover problem. SIAM J. Comput. 39(2), 361–370 (2009)
3. Awerbuch, B., Azar, Y., Bartal, Y.: On-line generalized steiner problem. Theor. Comput. Sci. 324(2-3), 313–324 (2004)
4. Berman, P., Coulston, C.: On-line algorithms for steiner tree problems. In: STOC, pp. 344–353 (1997)
5. Buchbinder, N., Naor, J.: The design of competitive online algorithms via a primal-dual approach. In: FOCS, pp. 93–263 (2009)
6. Buchbinder, N., Naor, J.: Online primal-dual algorithms for covering and packing problems. Math. Oper. Res. 34(2), 270–286 (2009)
7. Goemans, M.X., Williamson, D.P.: A general approximation technique for constrained forest problems. SIAM J. Comput. 24(2), 296–317 (1995)
8. Gupta, A., Nagarajan, V.: Approximating Sparse Covering Integer Programs Online. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012, Part I. LNCS, vol. 7391, pp. 436–448. Springer, Heidelberg (2012)
9. Hajiaghayi, M., Liaghat, V., Panigrahi, D.: Online node-weighted steiner forest and extensions via disk paintings. In: FOCS, pp. 558–567 (2013)
10. Imase, M., Waxman, B.M.: Dynamic steiner tree problem. SIAM J. Discrete Math. 4(3), 369–384 (1991)
11. Johnson, D.S., Minkoff, M., Phillips, S.: The prize collecting steiner tree problem: theory and practice. In: SODA, pp. 760–769 (2000)
12. Könemann, J., Sadeghabad, S.S., Sanità, L.: An LMP $O(\log n)$-approximation algorithm for node weighted prize collecting steiner tree. In: FOCS, pp. 568–577 (2013)
13. Korman, S.: On the use of randomization in the online set cover problem. M.S. thesis, Weizmann Institute of Science (2005)
14. Naor, J., Panigrahi, D., Singh, M.: Online node-weighted steiner tree and related problems. In: FOCS, pp. 210–219 (2011)
15. Qian, J., Williamson, D.P.: An $o(\log n)$-competitive algorithm for online constrained forest problems. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part I. LNCS, vol. 6755, pp. 37–48. Springer, Heidelberg (2011)

Nearly Linear-Time Model-Based Compressive Sensing[⋆]

Chinmay Hegde, Piotr Indyk, and Ludwig Schmidt

Massachusetts Institute of Technology, Cambridge MA 02139, USA

Abstract. Compressive sensing is a method for recording a k-sparse signal $x \in \mathbb{R}^n$ with (possibly noisy) linear measurements of the form $y = Ax$, where $A \in \mathbb{R}^{m \times n}$ describes the measurement process. Seminal results in compressive sensing show that it is possible to recover the signal x from $m = O(k \log \frac{n}{k})$ measurements and that this is tight. The *model-based compressive sensing* framework overcomes this lower bound and reduces the number of measurements further to $m = O(k)$. This improvement is achieved by limiting the supports of x to a *structured sparsity model*, which is a subset of all $\binom{n}{k}$ possible k-sparse supports. This approach has led to measurement-efficient recovery schemes for a variety of signal models, including tree-sparsity and block-sparsity.

While model-based compressive sensing succeeds in reducing the number of measurements, the framework entails a computationally expensive recovery process. In particular, two main barriers arise: (i) Existing recovery algorithms involve several *projections* into the structured sparsity model. For several sparsity models (such as tree-sparsity), the best known model-projection algorithms run in time $\Omega(kn)$, which can be too slow for large k. (ii) Existing recovery algorithms involve several matrix-vector multiplications with the measurement matrix A. Unfortunately, the only known measurement matrices suitable for model-based compressive sensing require $O(nk)$ time for a single multiplication, which can be (again) too slow for large k.

In this paper, we remove both aforementioned barriers for two popular sparsity models and reduce the complexity of recovery to *nearly linear time*. Our main algorithmic result concerns the tree-sparsity model, for which we solve the model-projection problem in $O(n \log n + k \log^2 n)$ time. We also construct a measurement matrix for model-based compressive sensing with matrix-vector multiplication in $O(n \log n)$ time for $k \leq n^{1/2-\mu}$, $\mu > 0$. As an added bonus, the same matrix construction can also be used to give a fast recovery scheme for the block-sparsity model.

Keywords: Model-based compressive sensing, model-projection, tree-sparsity, restricted isometry property, compressive sensing.

[⋆] This work was supported by grants from the MITEI-Shell program, the MADALGO center, and the Packard Foundation. A full version of this paper (including the appendix) is available at
`http://people.csail.mit.edu/ludwigs/papers/icalp14_fastmodelcs.pdf`.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 588–599, 2014.

1 Introduction

Compressive sensing is a method for recording a signal while taking only a small number of measurements. In particular, recording with *linear measurements* has attracted significant attention over the last decade [CRT06a,Don06,FR13]. In this setup, we are interested in recovering a vector $x \in \mathbb{R}^n$ (the signal) from measurements of the form $y = Ax$, where A is an $m \times n$ matrix and $y \in \mathbb{R}^m$. Usually, the setup also encompasses measurements corrupted by a noise vector e (i.e., $y = Ax+e$), in which case we are interested in recovering a good approximation to x. The main questions in compressive sensing deal with the conditions on A and x that enable efficient, stable recovery from only $m \ll n$ measurements. Compressive sensing has found applications in a wide variety of signal acquisition settings (e.g., MRI [LDP07]) and the underlying problem of *sparse recovery* has connections to several other fields such as data stream algorithms [Mut05,GI10] and Fourier sampling [HIKP12].

Seminal results in compressive sensing show that it is possible to recover a k-*sparse* signal x (containing at most k non-zeros) from $m = O(k \log n/k)$ linear measurements, as long as the measurement matrix A is chosen to satisfy the *restricted isometry property* (RIP) [CRT06b]. Moreover, the recovery step can be performed in polynomial time using several algorithms such as ℓ_1-minimization or CoSaMP [CRT06b,NT09]. While the bound on the number of measurements m is asymptotically tight in the noisy k-sparse setting [DBIPW10,FPRU10], there are ways to overcome this barrier and improve the "compression rate" even further. One such approach for reducing the number of measurements is *model-based compressive sensing* [BCDH10]. In this framework, we make additional assumptions about the support of the signal x. Instead of considering *all* k-sparse signals, we limit our attention to a smaller family of k-sparse supports, which we call a *structured sparsity model* $\mathbb{M}_k$. Research in signal processing has shown that this often is a useful way to capture additional structure in the signals of interest. For example, for some classes of time-domain signals x, the large coefficients in x tend to occur consecutively as clusters. For several sparsity models, it is possible to show measurement bounds of $m = O(k)$. Note that this improvement is not only of theoretical interest: for large values of n, removing the logarithmic factor in m can decrease the measurement complexity by up to an order of magnitude in practice.

While model-based compressive sensing succeeds in reducing the number of measurements, the current framework also entails a computationally more expensive recovery process. In particular, two main barriers limit the recovery performance of model-based compressive sensing compared to "standard" k-sparse compressive sensing:

1. Recovery algorithms for model-based compressive sensing rely on the availability of a *model-projection algorithm.* Given an arbitrary signal x, a model-projection algorithm returns the best approximation of x in the sparsity model $\mathbb{M}_k$. Unfortunately, for many sparsity models, the running time of the best known model-projection algorithm is $\Omega(nk)$.
2. For standard compressive sensing, researchers have identified several classes of measurement matrices A that satisfy the RIP and allow fast matrix-vector

multiplication in time $O(n \log n)$; see [NPW14] and references therein. In contrast, matrices known to satisfy the model-equivalent of the RIP only admit slow multiplication in time $O(nm)$ [BCDH10]. Since known recovery algorithms for model-based compressive sensing perform several matrix-vector multiplications, this can become a bottleneck in the overall time complexity. One approach to overcome this barrier is to use *sparse* matrices that satisfy the ℓ_1-variant of the RIP. However, recent work shows that this implies a lower bound of $m = \Omega(k \log \frac{n}{k} / \log \log \frac{n}{k})$ for the tree-sparsity model [IR13].

In this paper, we remove both aforementioned barriers for two popular sparsity models and bring the recovery performance of these models down to *nearly linear time*. Our central results concern the *tree-sparsity model*. In this model, the coefficients of the signal x are arranged as a complete d-ary tree. The model then requires that the support of x forms a connected subtree containing the root node. The tree-sparsity model captures structure in the wavelet-domain representation of natural images; see [Bar99] and [HIS14c] for more details.

As a bonus, our techniques also imply a fast recovery scheme for the *block-sparsity model*. In the block-sparsity model, the signal is divided into a fixed number of blocks, and valid supports can be described as the union of a small number of such blocks. The block-sparsity model captures signal structure in settings where the nonzeros form a small number of clusters.

Our Contributions. This paper contains two results:

1. Our main technical contribution is a fast model-projection for the tree-sparsity model with time complexity $O(n \log n + k \log^2 n)$. Our formal recovery guarantees complement recent empirical results in [HIS14c].
2. Building on [NPW14], we construct a measurement matrix which satisfies the model-RIP and enables multiplication in $O(n \log n + k^2 \log n \log^2(k \log n))$ time for general k. For $k \leq n^{1/2-\mu}$, $\mu > 0$, the multiplication time is $O(n \log n)$. Moreover, our matrix has the same bound on the number of measurements as existing, slow model-RIP matrices: $m = O(k + \log|\mathbb{M}_k|)$.

Together with existing results [BCDH10,HIS14b], our contributions enable us to state recovery guarantees of the following form: Let x be a signal in the tree-sparsity model with sparsity parameter k and let A be our new measurement matrix with $m = O(k)$ rows. The measurements are given by $y = Ax + e$ for arbitrary noise e. Then we can recover an $\widehat{x}$ such that $\|x - \widehat{x}\|_2 \leq C\|e\|_2$. Moreover, we can perform the recovery in time $O((n \log n + k^2 \log n \log^2(k \log n)) \log \frac{\|x\|_2}{\|e\|_2})$. Note that this compares favorably with the time complexity of the original model-based compressive sensing framework [BCDH10]: $O(nk \log \frac{\|x\|_2}{\|e\|_2})$. Table 1 compares our results to previous recovery schemes for the tree-sparsity model. Our recovery guarantees for the block-sparsity model are analogous.

Ideally, a model-RIP matrix with $m = O(k + \log|\mathbb{M}_k|)$ rows would offer a multiplication time of $O(n \log n)$ for all values of k. However, we conjecture that such a result is connected to progress on the measurement bound for subsampled Fourier matrices in k-sparse compressive sensing. This is considered a challenging open problem in the field.

Table 1. Comparison of our results with previous recovery schemes for the tree-sparsity model. In order to simplify the presentation, all stated bounds are for the regime of $k \leq n^{1/2-\mu}$ with $\mu > 0$. We also omit a factor of $\log \frac{\|x\|_2}{\|e\|_2}$ from all recovery times. An ℓ_p-recovery guarantee is of the form $\|x - \widehat{x}\|_p \leq C\|e\|_p$, where x is the original signal, $\widehat{x}$ is the recovery result, e is the measurement noise, and C is a fixed constant.

Paper	Measurement bound	Recovery time	Matrix-vector multiplication time	Recovery guarantee
[BCDH10]	$O(k)$	$O(nk)$	$O(nk)$	ℓ_2
[IR13]	$O\big(k \frac{\log n}{\log \log n}\big)$	exponential	$O(n \log n)$	ℓ_1
[BBC14]	$O\big(k \frac{\log n}{\log \log n}\big)$	$O(nk)$	$O(n \log n)$	ℓ_1
This paper	$O(k)$	$O(n \log n)$	$O(n \log n)$	ℓ_2

Our Techniques. We achieve the aforementioned results with the following tools:

1. In order to project into the tree-sparsity model, we use the recent framework for *approximation-tolerant* model-based compressive sensing [HIS14b], which was originally introduced for another sparsity model. Following this framework, instead of providing a single *exact* model-projection algorithm, we give two *approximate* algorithms: one for the *minimization* and one for the *maximization* version of the problem. The first algorithm builds a solution by combining several small subtrees which are cheap to find. The second algorithm works with a Lagrangian relaxation and constructs the corresponding Pareto curve with a sweep line approach.
2. We construct our measurement matrix by combining a fast standard-RIP matrix for initial dimensionality reduction with a standard model-RIP matrix for achieving a small number of measurements.

Related Work. There is a large body of work on matrices satisfying the RIP for general k-sparse vectors (e.g. see [RV08,BDDW08,GI10,CGV13] and references therein). For matrices with fast matrix-vector multiplication in $O(n \log n)$ time, the best known measurement bound is $m = O(k \log n \log^2(k \log n))$ [NPW14]. For $k \leq n^{1/2-\mu}$ and $\mu > 0$, there exist fast matrices with $m = O(k \log n)$ [AR13]. Note that in this regime, $O(k \log n) = O(k \log \frac{n}{k})$.

For the model-RIP, the only known matrices with $m = O(k + \log|\mathbb{M}_k|)$ are *dense* matrices with i.i.d. subgaussian entries [BCDH10]. Vector-matrix multiplication with such matrices requires $O(mn)$ time. While ℓ_1-model-RIP matrices support faster multiplication, they also entail a measurement lower bound of $m = \Omega(k \log \frac{n}{k} / \log \log \frac{n}{k})$ for the tree-sparsity model [IR13].

The problem of projecting into the tree-sparsity model has received a fair amount of attention in the literature over the last two decades. Researchers have proposed several algorithms such as the condensing sort-and-select algorithm (CSSA) [BJ94], complexity-penalized residual sum-of-squares (CPRSS) [Don97], and optimal pruning [BB94]. However, all of these algorithms either run in time

$\Omega(n^2)$ or fail to provide projection guarantees for general input signals. A recent paper describes a dynamic programming algorithm for exact projections running in time $O(nk)$ [CT13]. Combining this algorithm with the ℓ_1-model-RIP matrices mentioned above, another recent paper provides a compressive sensing recovery scheme in the ℓ_1-setting [BBC14]. As a result, the measurement complexity is constrained by the aforementioned lower bound and the recovery time is $\Omega(nk)$.

In related work, an algorithm for *approximate* projections into the tree-sparsity model has been proposed [HIS14c]. Unfortunately, this algorithm only has a weakly polynomial running time depending on the largest and smallest nonzero absolute values in the input. Moreover, it solves only the minimization variant of the problem, which is not sufficient to establish a compressive sensing recovery result. Instead, the authors demonstrate the validity of their approach via several numerical experiments. Our results here complement these findings with formal guarantees. We note that our minimization algorithm is related to the algorithm in [HIS14c] but achieves a strongly polynomial running time.

2 Preliminaries

Structured Sparsity. A signal $x \in \mathbb{R}^n$ is k-sparse if at most k of its coefficients are nonzero. The support of x, denoted by $\mathrm{supp}(x) \subseteq [n]$, contains the indices corresponding to the nonzero entries in x.

Suppose that we posses some additional information about the support of our signals of interest. One way to model this information is as follows [BCDH10]: denote the set of *allowed supports* with $\mathbb{M}_k = \{\Omega_1, \Omega_2, \ldots, \Omega_L\}$, where $\Omega_i \subseteq [n]$ and $|\Omega_i| = k$. Often it is useful to work with the closure of $\mathbb{M}_k$ under taking subsets, which we denote with $\mathbb{M}_k^+ = \{\Omega \subseteq [n] \mid \Omega \subseteq S \text{ for some } S \in \mathbb{M}_k\}$. Then, we define a *structured sparsity model*, $\mathcal{M}_k \subseteq \mathbb{R}^n$, as the set of vectors such that $\mathcal{M}_k = \{x \in \mathbb{R}^n \mid \mathrm{supp}(x) \in \mathbb{M}_k^+\}$. The number of allowed supports $L = |\mathbb{M}_k|$ is called the "size" of the model $\mathcal{M}_k$; typically $|\mathbb{M}_k| \ll \binom{n}{k}$.

Our central focus in this paper is the *tree-sparsity model* [BCDH10]. Let n be such that the coefficients of a signal $x \in \mathbb{R}^n$ can be arranged as the nodes of a perfect d-ary tree rooted at node 1.[1] Then, the tree-sparsity model comprises the set of k-sparse signals whose nonzero coefficients form a *connected subtree* rooted at node 1. More formally, let $\mathbb{T}$ be the set of supports forming a connected subtree and let $\mathbb{T}_i$ be the set of supports forming a connected subtree rooted at node i. Then the tree-sparsity model is defined as $\mathbb{M}_k = \{\Omega \subseteq [n] \mid \Omega \in \mathbb{T}_1 \text{ and } |\Omega| = k\}$. The size of this model is bounded by $|\mathbb{M}_k| \leq (2e)^k/(k+1)$ [BCDH10]. For a subtree Ω with root r, we use root-path(Ω) to denote the set of nodes on the path from r to node 1 (the root of the entire tree).

Model Projections. For a sparsity model $\mathcal{M}_k$, we define the problem of *model-projection* as follows: given $x \in \mathbb{R}^n$, find a $x^* \in \mathcal{M}_k$ such that $\|x - x^*\|_p$ is min-

[1] Our algorithms can easily be extended to handle *complete* d-ary trees and hence work for the general tree-sparsity model with arbitrary dimension n. For simplicity, we state our algorithms here for the special case of *perfect* d-ary trees.

imized for a norm parameter $p \geq 1$. In general, this problem can be hard since $\mathcal{M}_k$ is typically non-convex. Moreover, the original model-based compressive sensing framework in [BCDH10] requires the minimization to be *exact*. An alternative is the *approximation-tolerant* model-based compressive sensing framework [HIS14b]. Instead of a single *exact* model-projection algorithm, the framework requires two *approximate* model-projection algorithms with two different notions of approximation:

- A head approximation algorithm $H(x, k)$ that satisfies the following guarantee: Let $\widehat{\Omega} = H(x, k)$. Then $\widehat{\Omega} \in \mathbb{M}^+_{c_1 k}$ and $\|x_{\widehat{\Omega}}\|_p \geq c_2 \max_{\Omega \in \mathbb{M}_k} \|x_\Omega\|_p$ for some constants $c_1 \geq 1$ and $c_2 \leq 1$.
- A tail approximation algorithm $T(x, k)$ that satisfies the following guarantee: Let $\widehat{\Omega} = T(x, k)$. Then $\widehat{\Omega} \in \mathbb{M}^+_{c_1 k}$ and $\|x - x_{\widehat{\Omega}}\|_p \leq c_2 \min_{\Omega \in \mathbb{M}_k} \|x - x_\Omega\|_p$ for some constants $c_1 \geq 1$ and $c_2 \geq 1$.

Using such approximate model-projection algorithms, the framework of [HIS14b] provides the same asymptotic recovery guarantees as those achieved with an exact model-projection.

Measurement Matrices. Many recovery algorithms for compressive sensing assume that the measurement matrix satisfies the *restricted isometry property* (RIP). A matrix $A \in \mathbb{R}^{m \times n}$ has the (δ, k)-RIP if the following inequalities hold for all k-sparse vectors $x \in \mathbb{R}^n$:

$$(1 - \delta)\|x\|_2^2 \leq \|Ax\|_2^2 \leq (1 + \delta)\|x\|_2^2 . \tag{1}$$

There exist measurement matrices satisfying the RIP with only $m = O(k \log \frac{n}{k})$ rows [BDDW08]. A matrix $A \in \mathbb{R}^{m \times n}$ has the (δ, k)-*model-RIP* for model $\mathcal{M}_k$ if (1) holds for all $x \in \mathcal{M}_k$. There exist matrices satisfying the model-RIP with only $m = O(k + \log|\mathbb{M}_k|)$ rows [BCDH10].

Recovery Algorithm. We briefly summarize the *approximate model-iterative hard thresholding* (AM-IHT) algorithm for signal recovery using approximate model projections. For a full explanation, see [HIS14b] and references therein. Let $y = Ax + e$, where e is the measurement noise vector. Then, one can recover a signal estimate $\widehat{x}$ satisfying $\|x - \widehat{x}\|_2 \leq C\|e\|_2$ by applying the following update rule, inspired by the well-known iterative hard thresholding (IHT) [BD09]:

$$x^{(i+1)} \leftarrow T(x^{(i)} + H(A^T(y - Ax^{(i)}))) . \tag{2}$$

It is possible to show that $O(\log \frac{\|x\|_2}{\|e\|_2})$ iterations suffice for guaranteed recovery. Therefore, the overall time complexity of AM-IHT is governed by the running times of $H(\cdot)$, $T(\cdot)$, and the cost of matrix-vector multiplication with A and A^T.

3 Head Approximation for the Tree-Sparsity Model

We propose a head approximation algorithm for the tree-sparsity model. In order to simplify the analysis, we will assume that $k \geq \lceil \log_d n \rceil$. Note that we can always reduce the input to this case by removing layers of the tree with depth greater than k. Our approach is based on the following structural result about decompositions of d-ary trees, which we prove in Appendix A.

Algorithm 1. (HeadApprox) Head approximation for the tree-sparsity model

```
1: function HeadApprox($x, k, d, p, \alpha$)
2:     Run ETP on $x$ with sparsity parameter $k' = d\alpha$.
3:     $x^{(1)} \leftarrow x$
4:     for $i \leftarrow 1, \ldots, \lceil \frac{k}{\alpha} \rceil$ do
5:         $\widehat{\Omega}_i \leftarrow \operatorname*{arg\,max}_{\Omega \in \mathbb{T},\, |\Omega| = d\alpha} \|x^{(i)}_{\Omega}\|_p$
6:         $x^{(i+1)} \leftarrow x^{(i)}, \qquad x^{(i+1)}_{\widehat{\Omega}_i} \leftarrow 0$
7:         for $j \in \widehat{\Omega}_i \cup \text{root-path}(\widehat{\Omega}_i)$ (in bottom-up order) do
8:             Update the DP table for node $j$ up to sparsity $k' = d\alpha$.
9:     return $\widehat{\Omega} \leftarrow \bigcup_{i=1}^{\lceil \frac{k}{\alpha} \rceil} \widehat{\Omega}_i \cup \text{root-path}(\widehat{\Omega}_i)$
```

Lemma 1. *Let T be a d-ary tree with $|T| = k$. Moreover, let $\alpha \in \mathbb{N}, \alpha \geq 1$. Then T can be decomposed into a set of disjoint, connected subtrees $S = \{T_1, \ldots, T_\beta\}$ such that $|T_i| \leq d\alpha$ for all $i \in [\beta]$ and $\beta = |S| \leq \lceil \frac{k}{\alpha} \rceil$.*

In addition to the tree decomposition, our head-approximation algorithm builds on the exact tree projection algorithm (ETP) introduced in [CT13]. The algorithm finds the best tree-sparse approximation for a given signal via dynamic programming (DP) in $O(nkd)$ time.[2] We run ETP with a small sparsity value $k' < k$ in order to find optimal subtrees of size k'. We then assemble several such subtrees into a solution with a provable approximation guarantee. We use the fact that ETP calculates the DP table entries in the following way: if the DP tables corresponding to the children of node i are correct, the DP table for node i can be computed in $O(k'^2)$ time. The time complexity follows from the structure of the DP tables: for every node and $l \leq k'$, we store the value of the best subtree achievable at that node with sparsity exactly l. We can now state our head-approximation algorithm (Alg. 1) and the corresponding guarantees.

Theorem 1. *Let $x \in \mathbb{R}^n$ be the coefficients corresponding to a d-ary tree rooted at node 1. Also, let $p \geq 1$ and $\alpha \geq 1$. Then* HeadApprox(x, k, d, p, α) *returns a support $\widehat{\Omega}$ satisfying $\|x_{\widehat{\Omega}}\|_p \geq \left(\frac{1}{4}\right)^{1/p} \max_{\Omega \in \mathbb{M}_k} \|x_\Omega\|_p$. Moreover, $\widehat{\Omega} \in \mathbb{M}^+_\gamma$ with $\gamma = \lceil \frac{k}{\alpha} \rceil (d\alpha + \lceil \log_d n \rceil)$.*

Proof. Let $\Omega^* \in \mathbb{M}_k$ be an optimal support, i.e., $\|x_{\Omega^*}\|_p = \max_{\Omega \in \mathbb{M}_k} \|x_\Omega\|_p$. Using Lemma 1, there is a decomposition of Ω^* into disjoint sets $\Omega^*_1, \ldots, \Omega^*_\beta$ such that $\Omega^*_i \in \mathbb{T}$, $|\Omega^*_i| \leq d\alpha$ and $\beta \leq \lceil \frac{k}{\alpha} \rceil$. The contribution of Ω^*_i to the overall solution is $\|x_{\Omega^*_i}\|^p_p$. Now, compare the contributions of our subtrees $\widehat{\Omega}_i$ to these quantities. When finding $\widehat{\Omega}_i$ for $i \in [\beta]$, one of the following two cases holds:

1. $\|x^{(i)}_{\Omega^*_i}\|^p_p \geq \frac{1}{2} \|x_{\Omega^*_i}\|^p_p$. Since Ω^*_i is a candidate in the search for $\widehat{\Omega}_i$ in line 5, we have $\|x^{(i)}_{\widehat{\Omega}_i}\|^p_p \geq \|x^{(i)}_{\Omega^*_i}\|^p_p \geq \frac{1}{2} \|x_{\Omega^*_i}\|^p_p$.

[2] While ETP as stated in [CT13] works for $p = 2$ only, the algorithm can easily be extended to arbitrary norm parameters p.

2. $\|x^{(i)}_{\Omega^*_i}\|^p_p < \frac{1}{2}\|x_{\Omega^*_i}\|^p_p$. Therefore, $\widehat{\Omega}_1, \ldots, \widehat{\Omega}_{i-1}$ have already covered at least half of the contribution of Ω^*_i. Formally, let $C_i = \Omega^*_i \cap \bigcup_{j=1}^{i-1} \widehat{\Omega}_j$. Then $\|x_{C_i}\|^p_p \geq \frac{1}{2}\|x_{\Omega^*_i}\|^p_p$.

Let $A = \{i \in [\beta] \,|\, \text{case 1 holds for } \widehat{\Omega}_i\}$ and $B = \{i \in [\beta] \,|\, \text{case 2 holds for } \widehat{\Omega}_i\}$. For the set A we have

$$\|x_{\widehat{\Omega}}\|^p_p \;=\; \sum_{i=1}^{\lceil \frac{k}{\alpha} \rceil} \|x^{(i)}_{\widehat{\Omega}_i}\|^p_p \;\geq\; \sum_{i \in A} \|x^{(i)}_{\widehat{\Omega}_i}\|^p_p + \sum_{i \in B} \|x^{(i)}_{\widehat{\Omega}_i}\|^p_p \;\geq\; \frac{1}{2}\sum_{i \in A} \|x_{\Omega^*_i}\|^p_p \,. \qquad (3)$$

Now, consider the set B. Since the Ω^*_i are disjoint, so are the C_i. Moreover, $C_i \subseteq \widehat{\Omega}$ and therefore

$$\|x_{\widehat{\Omega}}\|^p_p \;\geq\; \sum_{i=1}^{\beta} \|x_{C_i}\|^p_p \;\geq\; \sum_{i \in B} \|x_{C_i}\|^p_p \;\geq\; \frac{1}{2}\sum_{i \in B} \|x_{\Omega^*_i}\|^p_p \,. \qquad (4)$$

Combining (3) and (4), we get

$$2\|x_{\widehat{\Omega}}\|^p_p \;\geq\; \frac{1}{2}\sum_{i \in A} \|x_{\Omega^*_i}\|^p_p + \frac{1}{2}\sum_{i \in B} \|x_{\Omega^*_i}\|^p_p \;\geq\; \frac{1}{2}\|x_{\Omega^*}\|^p_p \,.$$

Raising both sides to power $1/p$ gives the guarantee in the theorem. For the sparsity bound, note that $|\widehat{\Omega}_i| \leq d\alpha$ and $|\text{root-path}(\widehat{\Omega}_i)| \leq \lceil \log_d n \rceil$. Since we take the union over $\lceil \frac{k}{\alpha} \rceil$ such sets, the theorem follows. □

We defer the runtime analysis to Appendix A (Theorem 4) and state the final result here. Its proof is a direct consequence of Theorems 1 and 4.

Corollary 1. *Let* $\alpha = \lceil \log_d n \rceil$. *Then* HeadApprox$(x, k, d, p, \alpha)$ *returns a support* $\widehat{\Omega} \in \mathbb{M}^+_{k(2d+2)}$ *satisfying* $\left\|x_{\widehat{\Omega}}\right\|_p \geq \left(\frac{1}{4}\right)^{1/p} \max_{\Omega \in \mathbb{M}_k} \left\|x_\Omega\right\|_p$. *Moreover, the algorithm runs in time* $O(n \log n + k \log^2 n)$ *for fixed* d.

4 Tail Approximation for the Tree-Sparsity Model

Next, we propose a tail approximation algorithm. We consider the Lagrangian relaxation $\arg\min_{\Omega \in \mathbb{T}_1} \|x - x_\Omega\|^p_p + \lambda|\Omega|$, where the parameter λ controls the trade-off between the approximation error and the sparsity of the identified support. The algorithm in [HIS14c] proceeds by performing a binary search over λ in order to explore the Pareto curve of this trade-off. Unfortunately, the running time of this algorithm is only weakly polynomial because it depends on both $x_{\max} = \max_{i \in [n]} |x_i|$ and $x_{\min} = \min_{i \in [n], |x_i| > 0} |x_i|$. Below, we develop an algorithm that exploits the structure of the Pareto curve in more detail and runs in strongly polynomial time $O(n \log n)$. In fact, our new algorithm constructs the shape of the *entire* Pareto curve and not only a single trade-off.

The Lagrangian relaxation is equivalent to $\arg\max_{\Omega \in \mathbb{T}_1} \|x_\Omega\|^p_p - \lambda|\Omega|$. Hence, we can rewrite this problem as $\arg\max_{\Omega \in \mathbb{T}_1} \sum_{i \in \Omega} y_i$, where $y_i = |x_i|^p - \lambda$. So for a given value of λ, the goal is to find a subtree Ω rooted at node 1 which maximizes the sum of weights y_i associated with the nodes in Ω.

In the following, we analyze how the solution to this problem changes as a function of λ and use this structure in our tail-approximation algorithm. On a high level, the optimal contribution of a node i is positive and decreasing up to a certain value of $\lambda = \gamma_i$, after which the contribution stays 0. So for $\lambda < \gamma_i$, a subtree rooted at node i can contribute positively to an overall solution. For $\lambda \geq \gamma_i$, we can ignore the subtree rooted at node i.

4.1 Properties of the Pareto Curve

Let $b_i(\lambda)$ denote the maximum value achievable with a subtree rooted at i:

$$b_i(\lambda) = \max_{\Omega \in \mathbb{T}_i} \|x_\Omega\|_p^p - \lambda |\Omega| \ .$$

Our algorithm relies on two main insights: (i) $b_i(\lambda)$ is a piecewise linear function with at most n non-differentiable points (or "corners"), which correspond to the values of λ at which the optimal support changes. (ii) Starting with $\lambda = 0$, $b_i(\lambda)$ is strictly decreasing up to a certain value of λ, after which $b_i(\lambda) = 0$. Formally, we can state the properties of the Pareto curve as follows.

Lemma 2. *$b_i(\lambda)$ is piecewise linear. There is a value γ_i such that $b_i(\lambda) = 0$ for $\lambda \geq \gamma_i$ and $b_i(\lambda)$ is strictly decreasing for $\lambda \leq \gamma_i$. The corners of $b_i(\lambda)$ are the points $D_i = \{\gamma_i\} \cup \{\gamma \in \bigcup_{j \in children(i)} D_j \mid \gamma < \gamma_i\}$.*

Proof. A simple inductive argument shows that $b_i(\lambda)$ can be recursively defined as

$$b_i(\lambda) = \max(0, \ |x_i|^p - \lambda + \sum_{j \in \text{children}(i)} b_j(\lambda)) \ .$$

Note that the theorem holds for the leaves of the tree. By induction over the tree, we also get the desired properties for all nodes in the tree. We are using the fact that piecewise linear functions and strictly decreasing functions are closed under addition. Moreover, the corners of a sum of piecewise linear functions are contained in the union of the corners of the individual functions. □

Our algorithm does not compute the $b_i(\lambda)$ directly but instead keeps track of the following two quantities $s_i(\lambda)$ and $c_i(\lambda)$. For a given value of λ, $s_i(\lambda)$ denotes the sum achieved by the best subtree rooted at node i. Similarly, $c_i(\lambda)$ denotes the cardinality of the best subtree rooted at node i. These two quantities are easier to maintain algorithmically because they are piecewise constant. The proof of the next lemma follows directly from Lemma 2 and a similar inductive argument. Appendix B.1 contains further properties of the Pareto curve with accompanying proofs.

Lemma 3. *Let*

$$s_i(\lambda) = |x_i|^p + \sum_{\substack{j \in children(i) \\ b_j(\lambda) > 0}} s_j(\lambda) \qquad \textit{and} \qquad c_i(\lambda) = 1 + \sum_{\substack{j \in children(i) \\ b_j(\lambda) > 0}} c_j(\lambda) \ .$$

Then $s_i(\lambda)$ and $c_i(\lambda)$ are piecewise constant and monotonically decreasing. The discontinuities of $s_i(\lambda)$ and $c_i(\lambda)$ are D_i (see Lemma 2). At a discontinuity $\gamma \in D_i$ we have $\lim_{\delta \to 0^+} s_i(\gamma + \delta) = s_i(\gamma)$ and $\lim_{\delta \to 0^+} c_i(\gamma + \delta) = c_i(\gamma)$.

Algorithm 2. (FindPareto) Constructing the Pareto curve

```
function FindPareto($x, p$)
    for $i \leftarrow 1, \ldots, n$ do                                    ▷ Initialization
        $s_i \leftarrow |x_i|^p$,   $c_i \leftarrow 1$,   $\text{active}_i \leftarrow \text{false}$
    $\widehat{\lambda}_0 \leftarrow +\infty$
    $r_1 \leftarrow c_1$
    for $i = 1, \ldots, n$ do                          ▷ Iterate over the discontinuities
        $j \leftarrow \operatorname*{arg\,max}_{l \in [n],\, \text{active}_l = \text{false}} \frac{s_l}{c_l}$        ▷ Find the next discontinuity
        $\widehat{\lambda}_i \leftarrow \frac{s_j}{c_j}$
        $\text{active}_j \leftarrow \text{true}$
        $a \leftarrow j$
        while $a \neq 1$ do                                ▷ Update the affected nodes
            $a \leftarrow \text{parent}(a)$
            $s_a \leftarrow |x_a|^p$
            $c_a \leftarrow 1$
            for $l \in \text{children}(a)$ with $\text{active}_l = \text{true}$ do
                $s_a \leftarrow s_a + s_l$
                $c_a \leftarrow c_a + c_l$
        $r_{i+1} \leftarrow c_1$
    $\widehat{\lambda}_{n+1} \leftarrow 0$
    return $(\widehat{\lambda}, r)$
```

4.2 Constructing the Pareto Curve

We now use the quantities introduced above in order to traverse the Pareto curve. We start with $\lambda = +\infty$, for which the values of the $s_i(\lambda)$ and $c_i(\lambda)$ are easy to determine. Then, we iterate the following two steps (see Algorithm 2): (i) Use the current values of the $s_i(\lambda)$ and $c_i(\lambda)$ to find the next discontinuity. (ii) Update the $s_i(\lambda)$ and $c_i(\lambda)$ based on the change in the optimal support. In order to simplify the analysis, we assume that the discontinuities γ_i are distinct.

Theorem 5 (Appendix B.2) establishes a connection between the variables s_j and c_j in FindPareto and the functions $s_j(\lambda)$ and $c_j(\lambda)$. Using this connection, we can now show that the algorithm returns the shape of the Pareto curve.

Theorem 2. *Let $p \geq 1$ and $x \in \mathbb{R}^n$ and let $\widehat{\lambda}$ and r be the vectors returned by* FindPareto(x, p). *Moreover, let $\lambda > 0$ such that $\widehat{\lambda}_{i-1} > \lambda \geq \widehat{\lambda}_i$. Then we have $r_i = |\Omega^*_\lambda|$ where*

$$\Omega^*_\lambda = \operatorname*{arg\,max}_{\substack{\Omega \in \mathbb{T}_1,\ 1 \in \Omega \\ b_j(\lambda) > 0 \ \text{for}\ j \in \Omega \setminus \{1\}}} \|x_\Omega\|_p^p - \lambda |\Omega| \ .$$

Proof. By the definition of FindPath and Theorem 5, we have $r_i = c_1(\lambda)$ for $\widehat{\lambda}_{i-1} > \lambda \geq \widehat{\lambda}_i$. The theorem then follows from Lemma 4 (Appendix B.1). □

Moreover, FindPareto can be implemented to run in $O(n \log n)$ time using a priority queue; see Theorem 6 in Appendix B.2 for a formal runtime analysis.

Given the shape of the Pareto curve, we can traverse it to find a suitable trade-off parameter $\widehat{\lambda}$ that achieves a constant-factor tail approximation. The main idea of this last claim is similar to the algorithm in [HIS14c]; we state the final guarantee with proof and pseudo code in Appendix B.3.

5 Compressive Sensing Recovery

We have developed constant factor head and tail approximation algorithms for the tree-sparsity model, both of which run in near-linear time $O(n \log n)$. Therefore, we can invoke AM-IHT (Eq. (2)) to achieve an algorithm for recovering tree-sparse signals from (noisy) linear measurements.[3] In Appendix C, we describe a new construction of a matrix $A \in \mathbb{R}^{m \times n}$ that satisfies the model-RIP for the tree-sparsity model $\mathcal{M}_k$ and in addition supports fast matrix-vector multiplication. Combining these ingredients, we obtain:

Theorem 3. *Let $A \in \mathbb{R}^{m \times n}$ be a model-RIP matrix as constructed in the proof of Theorem 8 Let $x \in \mathbb{R}^n$ be a signal with $x \in \mathcal{M}_k$ and let $y = Ax + e$ be the noisy measurements. Then, there exists an algorithm to recover a signal estimate $\widehat{x} \in \mathcal{M}_{ck}$ from y such that $\|x - \widehat{x}\|_2 \leq C\|e\|_2$ for some constants $c > 1$, $C > 0$. The algorithm runs in $O((n \log n + k^2 \log n \log^2(k \log n)) \log \frac{\|x\|_2}{\|e\|_2})$ time for general k, and in $O(n \log n)$ time for the range $k \leq n^{1/2-\mu}$ with $\mu > 0$.*

While we have stated our results for the tree-sparsity model, a completely analogous construction of A with optimal parameters is possible in the context of the *block-sparsity* model of [BCDH10]. In particular, since the block-sparse projection can be computed exactly in linear time, this construction yields near-linear time recovery of block-sparse signals. We omit a detailed derivation.

References

AR13. Ailon, N., Rauhut, H.: Fast and RIP-optimal transforms (2013), `http://arxiv.org/abs/1301.0878` (preprint)

Bar99. Baraniuk, R.: Optimal tree approximation with wavelets. In: SPIE Wavelet Applications in Signal and Image Processing (1999)

BB94. Bohanec, M., Bratko, I.: Trading accuracy for simplicity in decision trees. Machine Learning (1994)

BBC14. Bah, B., Baldassarre, L., Cevher, V.: Model-based sketching and recovery with expanders. In: Symposium on Discrete Algorithms (SODA) (2014)

BCDH10. Baraniuk, R., Cevher, V., Duarte, M., Hegde, C.: Model-based compressive sensing. IEEE Trans. Inform. Theory (2010)

BD09. Blumensath, T., Davies, M.: Iterative hard thresholding for compressed sensing. Appl. Comput. Harmon. Anal. (2009)

BDDW08. Baraniuk, R., Davenport, M., DeVore, R., Wakin, M.: A simple proof of the restricted isometry property for random matrices. Constructive Approximation (2008)

BJ94. Baraniuk, R., Jones, D.: A signal-dependent time-frequency representation: Fast algorithm for optimal kernel design. IEEE Trans. Sig. Proc. (1994)

[3] To be precise, the AM-IHT algorithm proposed in [HIS14b] imposes additional restrictions on the approximation factors of the head and tail algorithms. However, it is possible to modify AM-IHT to work with arbitrary constant factors. See [HIS14a], which is the journal version of [HIS14b].

CGV13. Cheraghchi, M., Guruswami, V., Velingker, A.: Restricted isometry of Fourier matrices and list decodability of random linear codes. In: Symposium on Discrete Algorithms (SODA) (2013)

CRT06a. Candes, E., Romberg, J., Tao, T.: Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. IEEE Trans. Inform. Theory (2006)

CRT06b. Candes, E., Romberg, J., Tao, T.: Stable signal recovery from incomplete and inaccurate measurements. Comm. Pure Appl. Math. (2006)

CT13. Cartis, C., Thompson, A.: An exact tree projection algorithm for wavelets. IEEE Signal Process. Lett. (2013)

DBIPW10. Do Ba, K., Indyk, P., Price, E., Woodruff, D.: Lower bounds for sparse recovery. In: Symposium on Discrete Algorithms (SODA) (2010)

Don97. Donoho, D.: CART and best-ortho-basis: a connection. Annals of Statistics (1997)

Don06. Donoho, D.: Compressed sensing. IEEE Trans. Inform. Theory (2006)

FPRU10. Foucart, S., Pajor, A., Rauhut, H., Ullrich, T.: The Gelfand widths of ℓ_p-balls for $0 \leq p \leq 1$. Journal of Complexity (2010)

FR13. Foucart, S., Rauhut, H.: A Mathematical Introduction to Compressive Sensing. Springer (2013)

GI10. Gilbert, A., Indyk, P.: Sparse recovery using sparse matrices. In: Proc. IEEE (2010)

HIKP12. Hassanieh, H., Indyk, P., Katabi, D., Price, E.: Nearly optimal sparse Fourier transform. In: Symposium on Theory of Computing (2012)

HIS14a. Hegde, C., Indyk, P., Schmidt, L.: Approximation algorithms for model-based compressive sensing (2014), `http://people.csail.mit.edu/ludwigs/papers/approxmodels.pdf` (preprint)

HIS14b. Hegde, C., Indyk, P., Schmidt, L.: Approximation-tolerant model-based compressive sensing. In: Symposium on Discrete Algorithms (SODA) (2014)

HIS14c. Hegde, C., Indyk, P., Schmidt, L.: A fast approximation algorithm for tree-sparse recovery. In: International Symposium on Information Theory (ISIT) (2014)

IR13. Indyk, P., Razenshteyn, I.: On model-based RIP-1 matrices. In: International Colloquium on Automata, Languages, and Programming (2013)

LDP07. Lustig, M., Donoho, D., Pauly, J.: Sparse MRI: The application of compressed sensing for rapid MR imaging. In: Magnetic Resonance in Medicine (2007)

Mut05. Muthukrishnan, S.: Data streams: Algorithms and applications. In: Foundations and Trends in Theoretical Computer Science (2005)

NPW14. Nelson, J., Price, E., Wootters, M.: New constructions of RIP matrices with fast multiplication and fewer rows. In: Symposium on Discrete Algorithms (SODA) (2014)

NT09. Needell, D., Tropp, J.: CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. Appl. Comput. Harmon. Anal. (2009)

RSV08. Rauhut, H., Schnass, K., Vandergheynst, P.: Compressed sensing and redundant dictionaries. IEEE Trans. Inform. Theory (2008)

RV08. Rudelson, M., Vershynin, R.: On sparse reconstruction from Fourier and Gaussian measurements. In: Comm. Pure Appl. Math. (2008)

Breaking the PPSZ Barrier for Unique 3-SAT*

Timon Hertli

Institute for Theoretical Computer Science, Department of Computer Science, ETH Zürich
timon.hertli@inf.ethz.ch

Abstract. The PPSZ algorithm by Paturi, Pudlák, Saks, and Zane (FOCS 1998) is the fastest known algorithm for (Promise) Unique k-SAT. We give an improved algorithm with exponentially faster bounds for Unique 3-SAT.

For uniquely satisfiable 3-CNF formulas, we do the following case distinction: We call a clause critical if exactly one literal is satisfied by the unique satisfying assignment. If a formula has many critical clauses, we observe that PPSZ by itself is already faster. If there are only few clauses in total, we use an algorithm by Wahlström (ESA 2005) that is faster than PPSZ in this case. Otherwise we have a formula with few critical and many non-critical clauses. Non-critical clauses have at least two literals satisfied; we show how to exploit this to improve PPSZ.

1 Introduction

The well-known problem k-SAT is NP-complete for $k \geq 3$. If P$\neq$NP, k-SAT does not have a polynomial time algorithm. For a CNF formula F over n variables, the naive approach of trying all satisfying assignments takes time $O(2^n \cdot \text{poly}(|F|))$. Especially for $k = 3$ much work has been put into finding so-called "moderately exponential time" algorithms running in time $O(2^{cn})$ for some $c < 1$. In 1998, Paturi, Pudlák, Saks, and Zane presented a randomized algorithm for 3-SAT that runs in time $O(1.364^n)$. Given the promise that the formula has at most one satisfying assignment (that problem is called Unique 3-SAT), a running time of $O(1.308^n)$ was shown. Both bounds were the best known when published. The running time of general 3-SAT has been improved repeatedly (e.g. [7,4]), until PPSZ was shown to run in time $O(1.308^n)$ for general 3-SAT [2].

Any further improvement of 3-SAT further also improves Unique 3-SAT; however that bound has not been improved upon since publication of the PPSZ algorithm. In this paper, we present a randomized algorithm for Unique 3-SAT with exponentially better bounds than what could be shown for PPSZ. Our algorithm builds on PPSZ and improves it by treating sparse and dense formulas differently.

A key concept of the PPSZ analysis is the so-called critical clause: We call a clause critical for a variable x if exactly one literal is satisfied by this unique satisfying assignment, and that literal is over x. It is not hard to see that the uniqueness of the satisfying assignment implies that every variable has at least one critical clause. If some variables have strictly *more* than one critical clause, then we will give a straightforward proof that PPSZ by itself is faster already. Hence the bottleneck of PPSZ is when every variable has *exactly* one critical clause, and in total there are exactly n critical clauses.

* Full version available at http://arxiv.org/abs/1311.2513

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 600–611, 2014.

Given a formula with exactly n critical clauses, consider how many other (non-crtical) clauses there are. If there are few, we use an algorithm by Wahlström [8] that is faster than PPSZ for formulas with few clauses in total. If there are many non-critical clauses we use the following fact: A non-critical clause has two or more satisfied literals (w.r.t. unique satisfying assignment); so after removing a literal, the remaining 2-clause is still satisfied. We will exploit this to improve PPSZ.

An remaining problem is the case if only very few (i.e. sublinearly many) variables have more than one critical clause or appear in many (non-critical) clauses. In this case, we would get only a subexponential improvement. A significant part of our algorithm deals with this problem.

Notation. We use the notational framework introduced in [9]. Let V be a finite set of propositional *variables*. A *literal* u over $x \in V$ is a variable x or a negated variable $\bar{x}$. If $u = \bar{x}$, then $\bar{u}$, the negation of u, is defined as x. We mostly use x, y, z for variables and u, v, w for literals. We assume that all literals are distinct. A *clause* over V is a finite set of literals over pairwise distinct variables from V. By $\mathrm{vbl}(C)$ we denote the set of variables that occur in C, i.e. $\{x \in V \mid x \in C \vee \bar{x} \in C\}$. C is a k-clause if $|C| = k$ and it is a $(\leq k)$-clause if $|C| \leq k$. A formula in *CNF* (Conjunctive Normal Form) F over V is a finite set of clauses over V. We define $\mathrm{vbl}(F) := \bigcup_{C \in F} \mathrm{vbl}(C)$. F is a k-CNF formula (a $(\leq k)$-CNF formula) if all clauses of F are k-clauses ($(\leq k)$-clauses). A (truth) *assignment* on V is a function $\alpha : V \to \{0,1\}$ which assigns a Boolean value to each variable. α extends to negated variables by letting $\alpha(\bar{x}) := 1 - \alpha(x)$. A literal u is *satisfied by* α if $\alpha(u) = 1$. A clause is *satisfied by* α if it contains a satisfied literal and a formula is *satisfied by* α if all of its clauses are. A formula is *satisfiable* if there exists a satisfying truth assignment to its variables. A formula that is not satisfiable is called *unsatisfiable*. satisfy F. *k-SAT* is the decision problem of deciding if a $(\leq k)$-CNF formula has a satisfying assignment.

If F is a CNF formula and $x \in \mathrm{vbl}(F)$, we write $F^{[x \mapsto 1]}$ (analogously $F^{[x \mapsto 0]}$) for the formula arising from removing all clauses containing x and truncating all clauses containing $\bar{x}$ to their remaining literals. This corresponds to assigning x to 1 (or 0) in F and removing trivially satifsied clauses. We call assignments α on V and β and W *consistent* if $\alpha(x) = \beta(x)$ for all $x \in V \cap W$. If α is an assignment on V and $W \subseteq V$, we denote by $\alpha|_W$ the assignment on W with $\alpha|_W(x) = \alpha(x)$ for $x \in W$. If $\gamma = \{x \mapsto 0, y \mapsto 1, \dots\}$, we write $F^{[\gamma]}$ as a shorthand for $F^{[x \mapsto 0][y \mapsto 1] \dots}$, the *restriction* of F to γ.

For a set W, we denote by $x \leftarrow_{\text{u.a.r.}} W$ choosing an element x u.a.r. (uniformly at random). Unless otherwise stated, all random choices are mutually independent. We denote by log the logarithm to the base 2. For the logarithm to the base e, we write ln. By $\mathrm{poly}(n)$ we denote a polynomial factor depending on n. We use the following convention if no confusion arises: When F is a CNF formula, we denote by V its variables and by n the number of variables of F, i.e. $V := \mathrm{vbl}(F)$ and $n := |\mathrm{vbl}(F)|$. By $o(1)$ we denote a quantity dependent on n going to 0 with $n \to \infty$.

Previous Work

Definition 1. *(Promise) Unique 3-SAT is the following promise problem: Given a (≤ 3)-CNF with at most one satisfying assignment, decide if it is satisfiable or unsatisfiable.*

A randomized algorithm for Unique 3-SAT is an algorithm that, for a uniquely satisfiable (≤ 3)*-CNF formula returns the satisfying assignment with probability* $\frac{1}{2}$*.*

If the formula has no satisfying assignment the algorithm cannot erroneously find one. Hence the error is one-sided and we don't have to care about unsatisfiable formulas.

The PPSZ algorithm [6] is a randomized algorithm for Unique 3-SAT running in time $O(1.308^n)$. The precise bound is as follows:

Definition 2. *Let* $S := \int_0^1 \left(1 - \min\{1, \frac{r^2}{(1-r)^2}\}\right) dr = 2\ln 2 - 1$ *()*

Theorem 1 ([6]). *There exists a randomized algorithm (called* PPSZ*) for Unique 3-SAT running in time* $2^{(S+o(1))n}$*.*

Note that $0.3862 < S < 0.3863$ and $2^S < 1.308$.

Our Contribution. For Unqiue 3-SAT, we get time bounds exponentially better than PPSZ:

Theorem 2. *There exists a randomized algorithm for Unique 3-SAT running in time* $2^{(S-\varepsilon_2+o(1))n}$ *where* $\varepsilon_2 = 10^{-24}$*.*

In Section 2, we review the PPSZ algorithm. In Section 3, we show that the worst case for PPSZ occurs when every variable has exactly one critical 3-clause; this case we improve in Section 4. In Section 5, we pose open problems that arise.

2 The PPSZ Algorithm

In this section we review the PPSZ algorithm [6], summarized in Algorithm 1. We need to adapt some statements slightly. For the straightforward but technical proofs we refer the reader to appendix of the full version. The following two definitions are used to state the PPSZ algorithm.

Definition 3. *A CNF formula F D-*implies *a literal u if there exists a subformula* $G \subseteq F$ *with* $|G| \leq D$ *and all satisfying assignments of G set u to* 1*.*

In a random permutation, the positions of two elements are not independent. To overcome this, placements were defined. They can be seen as continuous permutations with the nice property that the places of different elements are independent.

Definition 4 ([6]). *A placement on* V *is a mapping* $V \to [0,1]$*. A random placement is obtained by choosing for every* $x \in V$ $\pi(x)$ *uniformly at random from* $[0,1]$*, independently.*

Observation 3. *By symmetry and as ties happen with probability* 0*, ordering* V *according to a random placement gives a permutation distributed the same as a permutation drawn uniformly at random from the set of all permutations on* V*.*

The analysis of PPSZ builds on the concept of forced and guessed variables:

Algorithm 1. PPSZ(CNF formula F)

```
$V \leftarrow \mathrm{vbl}(F)$; $n \leftarrow |V|$
Choose $\beta$ u.a.r. from all assignments on $V$
Choose $\pi : V \to [0,1]$ as a random placement of $V$
Let $\alpha$ be a partial assignment on $V$, initially empty
for $x \in V$, in ascending order of $\pi(x)$ do
    if $F$ $(\log n)$-implies $x$ or $\bar{x}$, set $\alpha(x)$ to satisfy this literal
    otherwise $\alpha(x) \leftarrow \beta(x)$ {guess $\alpha(x)$ u.a.r.}
    $F \leftarrow F^{[x \mapsto \alpha(x)]}$
end for
return $\alpha$
```

Definition 5. *If in* PPSZ, $\alpha(x)$ *is assigned* 0 *or* 1 *because of D-implication, we call x* forced. *Otherwise (if* $\alpha(x)$ *is set to* $\beta(x)$*), we call x* guessed.

The following lemma from [6] relates the expected number of guessed variables to the success probability (the proof is by an induction argument and Jensen's inequality).

Lemma 1 ([6]). *Let F be a satisfiable* (≤ 3)*-CNF, let* α^* *be a satisfying assignment. Let* $G(\pi)$ *be the expected number of guessed variables conditioned on* $\beta = \alpha^*$ *depending on* π*. Then* PPSZ(F) *returns* α^* *with probability at least* $\mathbf{E}_\pi[2^{-G(\pi)}] \geq 2^{\mathbf{E}_\pi[-G(\pi)]}$.

Remember that $S := \int_0^1 \left(1 - \min\{1, \frac{r^2}{(1-r)^2}\}\right) dr = 2\ln 2 - 1$, which corresponds to the probability that a variable is guessed. We define S_p where the integral starts from p instead of 0; this corresponds to the probability that a variable has place at least p and is guessed.

Definition 6. *Let* $S_p := \int_p^1 \left(1 - \min\{1, \frac{r^2}{(1-r)^2}\}\right) dr.$

Observation 4. *For* $p \leq \frac{1}{2}$, $S_p = S - p + \int_0^p \frac{r^2}{(1-r)^2} dr.$

In the appendix of the full version, we derive from [6] the following:

Corollary 1. *Let F a* (≤ 3)*-CNF with unique satisfying assignment* α*. Then in PPSZ(F) conditioned on* $\beta = \alpha$*, the expected number of guessed variables is at most* $(S + o(1))n$*.*

Furthermore, suppose we pick every variable of F with probability p, independently, and let V_p *be the resulting set. Then in PPSZ(F) conditioned on* $\beta = \alpha$*, the expected number of guessed variables is at most* $(S_p + o(1))n$*.*

By Lemma 1, we have the following corollary:

Corollary 2. *Let F a* (≤ 3)*-CNF with unique satisfying assignment* α*. Then the probability that* PPSZ(F) *returns* α *is at least* $2^{(-S-o(1))n}$.

Furthermore, suppose we pick every variable of F with probability p, independently, and let V_p *be the resulting set. Then the expected* log *of the probability (over the choice of* V_p*) that* PPSZ$(F^{[\alpha|_{V_p}]})$ *returns* $\alpha|_{V \setminus V_p}$ *is at least* $(-S_p - o(1))n$*.*

The first statement is actually what is shown in [6], and the second statement is a direct consequence. We need this later when we replace PPSZ by a different algorithm on variables with place at most p. It is easily seen that for a (≤ 3)-CNF F, $\mathrm{PPSZ}(F)$ runs in time $2^{o(n)}$. Hence by a standard repetition argument, PPSZ gives us an algorithm finding an assignment in time $2^{(S+o(1))n}$ and we (re-)proved Theorem 1.

3 Reducing to One Critical Clause Per Variable

In this section we show that to obtain an exponential improvement for Unique 3-SAT we only need to consider the case where every variable has exactly one critical clause.

Definition 7 ([6]). *Let F be a CNF formula satisfied by α. We call a clause C* critical *for x (w.r.t. α) if α satisfies exactly one literal of C, and this literal is over x.*

Definition 8. *A 1C-Unique (≤ 3)-CNF is a uniquely satisfiable (≤ 3)-CNF where every variable has at most one critical clause. Call the corresponding promise problem 1C-Unique 3-SAT.*

All formulas we consider have a unique satisfying assignment; critical clauses will be always w.r.t. that. First we show that a variables with more than one critical clause are guessed less often; giving an exponential improvement for formulas with a linear number of such variables. A similar statement for shorter critical clauses is required in the next section.

Lemma 2. *Let F be a (≤ 3)-CNF uniquely satisfied by α. A variable x with at least two critical clauses (w.r.t. α) is guessed given $\beta = \alpha$ with probability at most $S - 0.0014 + o(1)$. Furthermore, a variable x with a critical (≤ 2)-clause is guessed with probability at most $S - 0.035 + o(1)$*

Proof. Suppose $\pi(x) = r$. Let C_1 and C_2 be two critical clauses of x. If C_1 and C_2 share no variable besides x, then the probability that x is forced is at least $2r^2 - r^4$ by the inclusion-exclusion principle. If C_1 and C_2 share one variable besides x, then the probability that x is forced is at least $2r^2 - r^3$ (which is smaller than $2r^2 - r^4$. C_1 and C_2 cannot share two varibles besides x: in that case $C_1 = C_2$, as being a critical clause for x w.r.t. α predetermines the polarity of the literals. Intutiviely, if r is small, then $2r^2 - r^3$ is almost twice as large as $\frac{r^2}{(1-r)^2}$; therefore in this area the additional clause helps us and the overall forcing probability increases. For a critical (≤ 2)-clause the argument is analogous. Here, the probability that x is forced given place r is at least r. The statement follows now by integration using the dominated convergence theorem, see appendix of the full version.

Corollary 3. *Let F be a (≤ 3)-CNF formula uniquely satisfied by α. If Δn variables of F have two critical clause, PPSZ finds α with probability at least $2^{-(S-0.0014\Delta+o(1))n}$.*

If Δn variables of F have a critical (≤ 2)-clause clause, PPSZ finds α with probability at least $2^{-(S-0.035\Delta+o(1))n}$.

If there are only few variables (less than $\Delta_1 n$) with one critical clause, we can find and guess them by brute-force. If we choose Δ_1 small enough, any exponential improvement

Algorithm 2. PPSZIMPROVED(CNF formula F)

```
repeat PPSZ($F$) $2^{(S-\varepsilon_2)n}$ times, return if a satisfying assignment has been found
for all subsets $W$ of size $\lfloor \Delta_1 n \rfloor$ and all assignments $\alpha'$ on $W$, try ONECC($F^{[\alpha']}$)
```

for 1C-Unique 3-SAT gives a (diminished) exponential improvement to Unique 3-SAT. To bound the number of subsets of size $\Delta_1 n$, we define the binary entropy and use a well-known upper bound to the binomial coefficient.

Definition 9. *For $p \in [0,1]$, $H(p) := -p\log p - (1-p)\log(1-p)$ ($0\log 0 := 0$).*

Lemma 3 (Chapter 10, Corollary 9 of [5]). *If pn is an integer, then $\binom{n}{pn} \leq 2^{H(p)n}$.*

We will manily prove that we have *some* exponential improvement. The claimed numbers are straightforward to check by inserting the values from the following table.

name	value	description
ε_1	10^{-19}	improvement in 1C-Unique 3-SAT
ε_2	10^{-24}	improvement in Unique 3-SAT
Δ_1	10^{-21}	threshold fraction of vars. with more than 1 crit. clause
Δ_2	$6 \cdot 10^{-5}$	$\Delta_2 n$ is the amount of variables for Δ_2-sparse and Δ_2-dense
ε_3	10^{-3}	exponential savings on repetitions if F is Δ_2-sparse
p^*	$8 \cdot 10^{-7}$	prob. that a var. is assigned using indep. 2-clauses instead of PPSZ

Lemma 4. *If there is a randomized algorithm* ONECC(F) *solving 1C-Unique 3-SAT in time $2^{(S-\varepsilon_1+o(1))n}$ for $\varepsilon_1 > 0$, then there is a randomized algorithm (Algorithm 2) solving Unique 3-SAT in time $2^{(S-\varepsilon_2+o(1))n}$ for some $\varepsilon_2 > 0$.*

Proof. Let F be a (≤ 3)-CNF uniquely satisfied by α. Let $c(F)$ be the number of variables of F with more than one critical clause. If $c(F) \geq \Delta_1 n$, PPSZ is faster by Corollary 3. If $c(F) = 0$, we can use ONECC(F).

However, what if $0 < c(F) < \Delta_1 n$? In that case, we get rid of these variables by brute-force: For all $\lfloor \Delta_1 n \rfloor$-subsets W of variables and for all $2^{\lfloor \Delta_1 n \rfloor}$ possible assignments α' on W, we try ONECC($F^{[\alpha']}$). For one such α', we have $F^{[\alpha']}$ satisfiable and $c(F) = 0$; namely if W includes all variables with multiple critical clauses and α' is compatible with α. This is because fixing variables according to α does not produce new critical clauses w.r.t. α.

There are $\binom{n}{\lfloor \Delta_1 n \rfloor}$ subsets of size $\lfloor \Delta_1 n \rfloor$ of the variables of F, each with $2^{\lfloor \Delta_1 n \rfloor}$ possible assignments. As $\binom{n}{\lfloor \Delta_1 n \rfloor} \leq 2^{\mathrm{H}(\Delta_1)n}$ (Lemma 3), we invoke ONECC($F^{[\alpha']}$) at most $2^{(\Delta_1 + \mathrm{H}(\Delta_1))n}$ times. Setting Δ_1 small enough such that $\Delta_1 + \mathrm{H}(\Delta_1) < \varepsilon_1$ retains an improvement for Unique 3-SAT.

4 Using One Critical Clause Per Variable

In this section we give an exponential improvement for 1C-Unique 3-SAT.

Algorithm 3. OneCC((≤ 3)-CNF F)

```
try Dense($F$);
try Sparse($F$)
```

Algorithm 4. GetInd2Clauses((≤ 3)-CNFF)

```
{for the analysis, $F$ is considered to be $\Delta_2$-dense; the procedure might fail otherwise}
$F_3 \leftarrow \{C \in F \mid |C| = 3\}$, $F_2 \leftarrow \{\}$
for $\lceil \Delta_2 n \rceil$ times do
    let $x$ be a variable with $\deg_3(F_3, x) \geq 5$ (return failure if no such variable exists)
    Choose $C$ u.a.r. from all of $F$ with $x \in \mathrm{vbl}(C)$.
    $l \leftarrow$ literal of $C$ over $x$; $C_2 \leftarrow C \setminus l$
    $F_2 \leftarrow F_2 \cup C_2$
    {remove all clauses of $F_3$ sharing variables with $C_2$}
    $F_3 \leftarrow \{C_3 \in F_3 \mid \mathrm{vbl}(C_3) \cap \mathrm{vbl}(C_2) = \emptyset\}$
end for
return $F_2$
```

Theorem 4. *Given a 1C-Unique* (≤ 3)*-CNF on n variables,* OneCC*(F) runs in expected time* $2^{(S-\varepsilon_1+o(1))n}$ *and finds the satisfying assignment with probability* $2^{-o(n)}$.

Obtaining a randomized algorithm using $2^{o(n)}$ independent repetitions and Markov's inequality is straightforward.

Corollary 4. *There exists a randomized algorithm for 1C-Unique 3-SAT running in time* $2^{(S-\varepsilon_1+o(1))n}$.

Together with Lemma 4 this immediately implies Theorem 2. We obtain the improvement by doing a case distinction into sparse and dense formulas, as defined now:

Definition 10. *For a CNF formula F and a variable x, the* degree *of x in F,* $\deg(F,x)$ *is defined to be the number of clauses in F that contain the variable x. The* 3-clause degree *of x in F,* $\deg_3(F,x)$ *is defined to be the number of 3-clauses in F that contain the variable x. For a set of variables W, denote by* $F \setminus W$ *the part of F* independent *of W that consists of the clauses of F that do not contain variables of W. We say that F is* Δ-sparse *if there exists a set W of at most* Δn *variables such that* $F \setminus W$ *has maximum 3-clause degree* 4*. We say that F is* Δ-dense *otherwise.*

We will show that for Δ_2 small enough, we get an improvement for Δ_2-sparse 1C-Unique (≤ 3)-CNF formulas. On the other hand, for any Δ_2 we will get an improvement for Δ_2-dense 1C-Unique (≤ 3)-CNF formulas. In the sparse case we can fix by brute force a small set of variables to obtain a formula with few 3-clauses. We need to deal with the (≤ 2)-clauses and then use an algorithm from Wahlström for CNF formulas with few clauses.

4.1 Dense Case

First we show the improvement for any Δ_2-dense 1C-Unique (≤ 3)-CNF. Δ_2-density means that even after ignoring all clauses over any $\Delta_2 n$ variables, a variable with

Algorithm 5. DENSE((≤ 3)-CNF F)

```
$F_2 \leftarrow$GETIND2CLAUSES($F$)
for $2^{(S-\varepsilon_1)n}$ times do
  $V_{p^*} \leftarrow$ pick each $x \in \text{vbl}(F)$ with probability $p^*$
  $\alpha' \leftarrow \{\}$
  for $C_2 \in F_2$ do
    if $\text{vbl}(C_2) \subseteq V_p$ then
      Let $\{u,v\} = C_2$
      $(\alpha'(u),\alpha'(v)) \leftarrow \begin{cases} (0,0), & \text{with probability } \frac{3}{15} \\ (0,1),(1,0),(1,1), & \text{with probability } \frac{4}{15} \text{ each} \end{cases}$
    end if
  end for
  for all $x \in V_p$, if $\alpha'(x)$ is not defined yet let $\alpha'(x) \leftarrow_{\text{u.a.r.}} \{0,1\}$
  PPSZ($F^{[\alpha']}$); if a satisfying assignment $\alpha$ has been found, return $\alpha \cup \alpha'$
end for
```

3-clause degree of at least 5 remains. The crucial idea is that for a variable x with 3-clause degree of at least 5, picking one occurence of x u.a.r. and removing it gives a 2-clause satisfied (by the unique satisfying assignment) with probability at least $\frac{4}{5}$: If the 2-clause is not satisfied, then the deleted literal of x is the only satisfied literal of the 3-clause; hnece the 3-clause is critical for x. However by assumption x has at most one critical clause. We can obtain $\lceil \frac{1}{2}\Delta_2 n \rceil$ such 2-clauses that do not share variables by repeating the above process, ignoring all 3-clauses that share variables with the 2-clauses produced so far (listed in GETIND2CLAUSES(F)). By Δ_2-density we do not run out of variables with degree at least 5 in the 3-clauses.

Observation 5. *For a Δ_2-dense 1C-Unique (≤ 3)-CNF F,* GETIND2CLAUSES*(F) returns a set of $\lceil \frac{1}{2}\Delta_2 n \rceil$ 2-clauses sharing no variables. The i-th (for $1 \leq i \leq \lceil \frac{1}{2}\Delta_2 n \rceil$) 2-clause is satisfied with probability at least $\frac{4}{5}$, irrespective of the previous 2-clauses.*

As a random 2-clause is satisfied with probability $\frac{3}{4}$ by a specific assignment, this set of 2-clauses gives us nontrivial information about the unique satisfying asignment. Now we show how to use these 2-clauses to improve PPSZ:

Lemma 5. *Let F be a Δ_2-dense 1C-Unique (≤ 3)-CNF for some $\Delta_2 > 0$. Then there exists an algorithm (*DENSE(F)*) runing in time $2^{(S-\varepsilon_1+o(1))n}$ for $\varepsilon_1 > 0$ and returning the satisfying assignment α of F with probability $2^{-o(n)}$.*

Proof. First we give some intuition. For variables that occur late in PPSZ, the probability of being forced is large (being almost 1 in the second half). However for variables that come at the beginning, the probability is very small; a variable x at place p is forced (in the worst case) with probability $\Theta(p^2)$ for $p \to 0$, hence we expect $\Theta(p^3 n)$ forced variables among the first pn variables in total.

However, a 2-clause that is satisfied by α with probability $\frac{4}{5}$ can be used to guess both variables in a better way than uniform, giving constant savings in random bits required. For $\Theta(n)$ such 2-clauses, we expect $\Theta(p^2 n)$ of them to have both variables among the first pn variables. For each 2-clause we have some nontrivial information;

intuitively we save around 0.01 bits. In total we save $\Theta(p^2n)$ bits among the first pn variables, which is better than PPSZ for small enough p.

Formally, let V_{p^*} be a random set of variables, where each variable of V is added to V_{p^*} with probability p^*. On V_{p^*}, we replace PPSZ by our improved guessing; on the remaining variables $V \setminus V_{p^*}$ we run PPSZ as usual. Let $\mathrm{E}_{\text{guess}}$ be the event that the guessing on V_{p^*} (to be defined later) finds $\alpha|_{V_{p^*}}$. Let $\mathrm{E}_{\mathrm{PPSZ}}$ be the event that PPSZ($F^{[\alpha|_{V_{p^*}}]}$) finds $\alpha|_{V \setminus V_{p^*}}$. Observe that for a fixed V_{p^*}, $\mathrm{E}_{\text{guess}}$ and $\mathrm{E}_{\mathrm{PPSZ}}$ are independent. Hence we can write the overall probability to find α (call it p_s) as an expectation over V_{p^*}:

$$\begin{aligned} p_s &= E_{V_{p^*}}[\Pr(\mathrm{E}_{\text{guess}} \cap \mathrm{E}_{\mathrm{PPSZ}} | V_{p^*})] \\ &= E_{V_{p^*}}[\Pr(\mathrm{E}_{\text{guess}} | V_{p^*}) \Pr(\mathrm{E}_{\mathrm{PPSZ}} | V_{p^*})] \\ &= E_{V_{p^*}}[2^{\log \Pr(\mathrm{E}_{\text{guess}} | V_{p^*}) + \log \Pr(\mathrm{E}_{\mathrm{PPSZ}} | V_{p^*})}] \\ &\geq 2^{E_{V_{p^*}}[\log \Pr(\mathrm{E}_{\text{guess}} | V_{p^*}) + \log \Pr(\mathrm{E}_{\mathrm{PPSZ}} | V_{p^*})]} \\ &= 2^{E_{V_{p^*}}[\log \Pr(\mathrm{E}_{\text{guess}} | V_{p^*})] + E_{V_{p^*}}[\log \Pr(\mathrm{E}_{\mathrm{PPSZ}} | V_{p^*})]}, \end{aligned}$$

where in the last two steps we used Jensen's inequality and linearity of expectation.

By Corollary 2, $E_{V_{p^*}}[\log \Pr(\mathrm{E}_{\mathrm{PPSZ}})] = (-S_p + o(1))n$. We now define the guessing and analyze $E_{V_{p^*}}[\log \Pr(\mathrm{E}_{\text{guess}})]$ (see Algorithm 5 as a reference):

By Observation 5 we obtain a set of $\lceil \frac{1}{2}\Delta_2 n \rceil$ 2-clauses F_2 sharing no variables, each satisfied (by α) with probability $\frac{4}{5}$ irrespective of the previous choices. In the following we assume that F_2 has at least a $\frac{4}{5}$-fraction of satisfied 2-clauses: This happens with constant probability (and we only require subexponential success probability) as we can lower bound the number of satisfied 2-clauses by a binomially distributed variable; such a variable is at least its mean with constant probability, see e.g. in [1].

Using the set of 2-clauses F_2, we choose an assignment α' on V_{p^*} as follows: For every clause C_2 in F_2 completely over V_{p^*} assign both of its variables with probability $\frac{1}{5}$ such that C_2 is violated; with probability $\frac{4}{15}$ each choose one three choices that satisfy C_2. Then any remaining variable of V_{p^*} is guessed u.a.r. from $\{0, 1\}$.

Given V_{p^*}, let m_0 be the number of clauses of F_2 completely over V_{p^*} *not satisfied* by α. Let m_1 be the number of clauses of F_2 completely over V_{p^*} *satisfied* by α. Then

$$\Pr(\mathrm{E}_{\text{guess}} | V_{p^*}) = \left(\frac{1}{2}\right)^{|V_{p^*}| - 2m_0 - 2m_1} \left(\frac{1}{5}\right)^{m_0} \left(\frac{4}{15}\right)^{m_1}.$$

This is seen as follows: For any variable for which no clause in C_2 is completely over V_{p^*}, we guess uniformly at random and so correctly with probability $\frac{1}{2}$. For any clause C_2 which is completely over V_{p^*}, we violate the clause with probability $\frac{1}{5}$, and choose a non-violating assignment with probability $\frac{4}{5}$. For any clause not satisfied by α, we hence set both variables according to α with probability $\frac{1}{5}$. For any clause satisfied by α, we set both variables according to α with probability $\frac{4}{15}$, as we have to pick the right one of the three assignments that satisfy C_2. As $E[V_{p^*}] = p^*n$, $E[m_0] \leq \frac{1}{5}p^{*2}\lceil \Delta_2 n \rceil$,

Algorithm 6. SPARSE((≤ 3)-CNF F)

```
repeat the following $2^{(S-\varepsilon_3)n}$ times:
for all subsets $W$ of size $\lfloor \Delta_2 n \rfloor$ and all assignments $\alpha'$ on $W$ do
    $F' \leftarrow F^{[\alpha']}$
    while no satisfying assignment found do
        try PPSZ($F^{[\alpha']}$)
        $F_2' \leftarrow \{C \in F' \mid |C| \leq 2\}$
        if $|F_2'| \leq \frac{1}{10}|\mathrm{vbl}(F')|$ then
            with probability $2^{-(S-0.015)|\mathrm{vbl}(F')|}$, run WAHLSTROEM($F'$)
        end if
        $\{$set all literals in a uniform $(\leq 2)$-clause to 1$\}$
        $C' \leftarrow_{\text{u.a.r.}} F_2'$, if $F_2' = \{\}$ return failure
        for $l \in C'$ do
            $F' \leftarrow F'^{[l \mapsto 1]}$
        end for
    end while
end for
```

$E[m_1] \geq \frac{4}{5} p^{*2} \lceil \Delta_2 n \rceil$, $E[m_0 + m_1] = p^{*2} \lceil \Delta_2 n \rceil$, we have

$$\begin{aligned} E[\log \Pr(\mathrm{E_{guess}} | V_{p^*})] &= -E[V_{p^*} - 2m_0 - 2m_1] + \log\left(\frac{1}{5}\right) E[m_0] + \log\left(\frac{4}{15}\right) E[m_1] \\ &\geq -p^* n + p^{*2} \lceil \Delta_2 n \rceil \left(2 + \log\left(\frac{1}{5}\right)\frac{1}{5} + \log\left(\frac{4}{15}\right)\frac{4}{5}\right). \end{aligned}$$

The inequality follows from the observations and $\log\left(\frac{4}{15}\right) \geq \log\left(\frac{1}{5}\right)$. One can calculate $2 + \log\left(\frac{1}{5}\right)\frac{1}{5} + \log\left(\frac{4}{15}\right)\frac{4}{5} \geq 0.01$, corresponding to the fact that a four-valued random variable where one value occurs with probability at most $\frac{1}{5}$ has entropy at most 1.99.

Hence by our calculations and Observation 4 (to evaluate S_p), we have

$$\begin{aligned} \frac{1}{n} \log p_s &\geq -S + p^* - \int_0^{p^*} \frac{r^2}{(1-r)^2} dr + o(1) - p^* + \Delta_2 p^{*2} \cdot 0.01 \\ &= -S - \int_0^{p^*} \frac{r^2}{(1-r)^2} dr + \Delta_2 p^{*2} \cdot 0.01 + o(1). \end{aligned}$$

This gives an improvement over PPSZ of $-\int_0^{p^*} \frac{r^2}{(1-r)^2} dr + \Delta_2 p^{*2} \cdot 0.01 + o(1)$. The first term corresponds to the savings PPSZ would have, the second term corresponds to the savings we have in our modified guessing. Observe that for small p^*, the integral evaluates to $\Theta(p^{*3})$, but the second term is $\Theta(p^{*2})$. Hence choosing p^* small enough gives an improvement.

4.2 Sparse Case

Now we show that if $\Delta_2 > 0$ is small enough we get an improvement for a Δ_2-sparse 1C-Unique (≤ 3)-CNF. For this, we need the following theorem by Wahlström:

Theorem 6 ([8]). *Let F be a CNF formula with average degree at most 4.2 where we count degree 1 as 2 instead. Then satisfiability of F can be decided in time $O(2^{0.371n}) \leq 2^{(S-0.015+o(1))n}$. Denote this algorithm by* $\text{WAHLSTROEM}(F)$.

Lemma 6. *Let F be a Δ_2-sparse 1C-Unique (≤ 3)-CNF. For Δ_2 small enough, there exists an algorithm ($\text{SPARSE}(F)$) running in expected time $2^{(S-\varepsilon_3+o(1))n}$ for $\varepsilon_3 > 0$ and finding the satisfying assignment α of F with probability $2^{-o(n)}$.*

Proof. Similar to Section 3, we first check by brute-force all subsets W of $\lfloor \Delta_2 n \rfloor$ variables and all possible assignments α' of W; by definition of Δ_2-sparse for some W, the part of F independent of W (i.e. $F \setminus W$) has maximum 3-clause degree 4. If furthermore α' is compatible with α, $F' := F^{[\alpha']}$ is a 1C-Unique (≤ 3)-CNF with maximum 3-clause degree 4: We observed that critical clauses cannot appear in the process of assigning variables according to α; furthermore any clause of F not independent of W must either disappear in F' or become a (≤ 2)-clause. As earlier, there are at most $2^{(\Delta_2+H(\Delta_2))n}$ cases of choosing W and α'. We now analyze what happens for the correct choice of F':

We would like to use WAHLSTROEM on F'; however F' might contain an arbitrary amount of (≤ 2)-clauses. The plan is to use the fact that either there are many critical (≤ 2)-clauses, in which case PPSZ is better, or few critical (≤ 2)-clauses, in which case all other (≤ 2)-clauses are non-critical and have only satisfied literals.

The algorithm works as follows: We have a 1C-Unique (≤ 3)-CNF on F' on $n' := |\text{vbl}(F')|$ variables; the maximum degree in the 3-clauses is at most 4. First we try PPSZ: if there are $\frac{1}{30}n'$ critical (≤ 2)-clauses, this gives a satisfying assignment with probability $2^{(S-0.035\frac{1}{30})n'}$. Otherwise, if there are less than $\frac{1}{10}n'$ (≤ 2)-clauses, the criterion of Theorem 6 applies: We invoke $\text{WAHLSTROEM}(F')$ with probability $2^{-(S-0.015)n'}$; this runs in expected time $2^{-o(n)}$ and finds a satisfying assignment with probability $2^{-(S-0.015)n'}$.

If both approaches fail, we know that F' has at less than $\frac{1}{30}n'$ critical (≤ 2)-clauses clauses, but also more than $\frac{1}{10}n'$ (≤ 2)-clauses overall. Hence at most one third of the (≤ 2)-clauses is critical. However a non-critical (≤ 2)-clause must be a 2-clause with both literals satisfied. Hence choosing a (≤ 2)-clause of F' uniformly at random and setting all its literals to 1 sets two variables correctly with probability at least $\frac{2}{3} > 2^{-0.371 \cdot 2} > 2^{-(S-0.015) \cdot 2}$. That is we reduce the number of variables by 2 with a better probability than PPSZ overall; and we can repeat the process with the reduced formula. This shows that for the correct F', we have expected running time $2^{o(n)}$ and success probability $2^{(-S+\varepsilon_3-o(1))n}$ for $\varepsilon_3 > 0$. It is important to see that ε_3 does *not* depend on Δ_2. Repeating this process $2^{(-S+\varepsilon_3-o(1))n}$ times gives success probability $2^{o(n)}$.

Together with the brute-froce choice of W and α', we have expected running time of $2^{(S-\varepsilon_3+\Delta_2+H(\Delta_2)+o(1))n}$. By choosing Δ_2 small enough we are better than PPSZ.

5 Open Problems

Can we also obtain an improvement for general 3-SAT? In general 3-SAT, there might be even fewer critical clauses and critical clauses for some assignments are not always critical for others. We need to fit our improvement into the framework of [2]. As there is some leeway for multiple assignments, this seems possible, but nontrivial and likley to become very complex.

Another question is whether we can improve (Unique) k-SAT. PPSZ becomes slower as k increases, which makes an improvement easier. However the guessing in SPARSE relied on the fact that non-critical (≤ 2)-clauses have all literals satisfied, which is not true for larger clauses.

Suppose Wahlström's algorithm is improved so that it runs in time $O(c^n)$ on 3-CNF formulas with average degree D. The sparsification lemma [3] shows that for $c \to 1$ and $D \to \infty$, we obtain an algorithm for 3-SAT running in time $O(b^n)$ for $b \to 1$. Can our approach be extended to a similar sparsification result?

Acknowledgements. I am very grateful to my advisor Emo Welzl for his support in realizing this paper. I also thank the anonymous reviewers for their helpful comments.

References

1. Hamza, K.: The smallest uniform upper bound on the distance between the mean and the median of the binomial and poisson distributions. Statistics & Probability Letters 23(1), 21–25 (1995)
2. Hertli, T.: 3-SAT faster and simpler—unique-SAT bounds for PPSZ hold in general. In: 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, pp. 277–284. IEEE Computer Soc., Los Alamitos (2011)
3. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity. J. Comput. System Sci. 63(4), 512–530 (2001); Special issue on FOCS 1998 (Palo Alto, CA)
4. Iwama, K., Tamaki, S.: Improved upper bounds for 3-SAT. In: Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithm, pp. 328–329 (electronic). ACM Press, New York (2004)
5. MacWilliams, F.F.J., Sloane, N.N.J.A.: The Theory of Error-Correcting Codes. North-Holland, Amsterdam (1977)
6. Paturi, R., Pudlák, P., Saks, M.E., Zane, F.: An improved exponential-time algorithm for k-SAT. J. ACM 52(3), 337–364 (electronic) (2005)
7. Schöning, U.: A probabilistic algorithm for k-SAT and constraint satisfaction problems. In: Proceedings of the 40th Annual Symposium on Foundations of Computer Science, pp. 410–414. IEEE Computer Society, Los Alamitos (1999)
8. Wahlström, M.: An algorithm for the SAT problem for formulae of linear length. In: Brodal, G.S., Leonardi, S. (eds.) ESA 2005. LNCS, vol. 3669, pp. 107–118. Springer, Heidelberg (2005)
9. Welzl, E.: Boolean Satisfiability – Combinatorics and Algorithms (Lecture Notes) (2005), `www.inf.ethz.ch/~emo/SmallPieces/SAT.ps`

Privately Solving Linear Programs*

Justin Hsu[1], Aaron Roth[1], Tim Roughgarden[2], and Jonathan Ullman[3]

[1] University of Pennsylvania
[2] Stanford University
[3] Harvard University

Abstract. In this paper, we initiate the systematic study of solving linear programs under differential privacy. The first step is simply to define the problem: to this end, we introduce several natural classes of *private linear programs* that capture different ways sensitive data can be incorporated into a linear program. For each class of linear programs we give an efficient, differentially private solver based on the multiplicative weights framework, or we give an impossibility result.

1 Introduction

Linear programming is one of the most fundamental and powerful tools in algorithmic design. It is used ubiquitously throughout computer science: applications include maximum matching, maximum and minimum cost flow, and fractional packing and covering problems. Linear programming relaxations of NP-complete problems also underlie countless efficient approximation algorithms.

At the same time, differential privacy is a field where efficient algorithms have been difficult to find. For many problems in differential privacy, the initial focus was on understanding the *information-theoretic complexity*—the extent to which solving the problem, efficiently or not, is compatible with differential privacy. As a result, there are many central problems that are known to be privately solvable, but for which computationally efficient algorithms are not known. For example, Kasiviswanathan et al. [17] show how to privately PAC learn any PAC learnable concept class (without privacy) with only a small increase in the sample complexity, but via an exponential time algorithm. It remains open whether a computationally efficient algorithm can do this in general. Similarly, Blum et al. [3] show how to privately release a summary of a private database that approximately preserves the answers to rich families of linear queries, again via an exponential time algorithm. In fact, under standard cryptographic assumptions, it is not possible to efficiently and privately answer large collections of general linear queries [9, 24, 23].

The two preceding examples are among the many algorithms that use the extremely general *exponential mechanism* of McSherry and Talwar [19] to achieve near optimal error. However, the exponential mechanism is not efficient in general: it requires running time linear in the size of its output range, which can

* A full version of the paper with the omitted proofs and sections can be found at `http://arxiv.org/abs/1402.3631`.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 612–624, 2014.

be extremely large. In contrast, general tools for designing *efficient* differentially private algorithms are harder to come by (although not non-existent, e.g., the sample and aggregate framework [20] and output/objective perturbation for unconstrained convex optimization [5, 18]).

Our work contributes to the toolbox of general algorithmic techniques for designing computationally efficient and differentially private algorithms; specifically, we give tools to privately and efficiently solve linear programs (LPs) of various types. An initial problem is to simply *define* what it means to solve a linear program privately. Differential privacy is defined in terms of *neighboring databases.* A database is a collection of records from some domain and two databases are neighboring if they differ in a single record. Differential privacy requires the output distribution of an algorithm to be nearly identical when run on either of a pair of neighboring databases. If linear programs can depend on private databases, we naturally have a notion of *neighboring linear programs*, and we want an algorithm for solving these linear programs that is differentially private with respect to this notion of neighboring inputs.

The way in which the linear program is derived from the database gives rise to several distinct notions of neighboring linear programs. For instance, consider an LP with objective $c^\top x$ and constraints $Ax \leq b$, where moving to a neighboring LP neighboring database leaves c and A unchanged but perturbs b by only a small amount in each coordinate. Solving this kind of linear programming privately is similar to the well-studied *linear query release* problem in differential privacy, and techniques for linear query release—such as the private multiplicative weights algorithm of Hardt and Rothblum [13] (and its offline variants [12, 14])—can be adapted with minor changes. (This result may even be considered folklore.) On the other hand, the situation is qualitatively different if moving to a neighboring LP can change either the constraint matrix A or the objective vector c. Some of these private LPs can still be solved; others are provably impossible to solve to nontrivial accuracy under differential privacy.

In this paper, we develop a taxonomy of private LPs. For each class, we either present an efficient and accurate differentially private solver, or prove that general LPs of this type cannot be accurately solved while preserving privacy.

1.1 Our Results and Techniques

We consider linear programs $LP(D)$ defined by a database D, with form

$$\begin{aligned} &\max_{x \in \mathbb{R}^d_+} c^\top x \\ &\text{s.t. } Ax \leq b. \end{aligned}$$

Here, the vector x represents the variables of the linear program, and $c = c(D), A = A(D)$, and $b = b(D)$ may each depend on the private database D. Our goal is to find an approximate solution to $LP(D)$ in a sense to be defined, while ensuring differential privacy for the underlying database D.

We classify private LPs along two dimensions: *which part* of the LP depends on the database and *how sensitive* the LP is to changes in the database. Along the second axis, we will consider: 1) *low-sensitivity* LPs, where changing one record of the database induces a small difference between coefficients that vanishes as the size of the database n grows and 2) *high-sensitivity* LPs, where changing one record of the database can induce a potentially large change in some coefficient. Low-sensitivity LPs are natural when the coefficients of the LP represent some kind of average over the database, whereas high-sensitivity LPs are natural when the coefficients represent specific records of the database.

Furthermore, we consider four parts of the LP that might depend on the database: 1) *the rows of A*, 2) *the scalars b*, 3) *the columns of A*, and 4) *the objective c*. These four parts of the LP, combined with the two notions of sensitivity, lead to the following eight notions of private linear programming:

1. **The constraints:** For these linear programs, moving to a neighboring database can affect at most one row of A and the corresponding entry of b, which corresponds to changing one constraint of the LP.
   (a) **High-sensitivity:** For *high-sensitivity constraint private LPs*, moving to a neighboring database can change a single constraint arbitrarily. That is, for every pair of neighboring databases D, D', there exists a row i such that for every row $j \neq i$, $A(D)_j = A(D')_j$ and $b(D)_j = b(D')_j$. This kind of linear program arises, for example, in covering LPs in which each record of the database represents an individual that needs to be covered. We cannot hope to approximately satisfy *every* constraint while ensuring privacy, but we show that by using a variant of multiplicative weights that operates only over a restricted set of distributions, we can still find solutions to such LPs that approximately satisfy *most* of the constraints. As an example of our technique, we solve a private version of the fractional set cover problem.
   (b) **Low-sensitivity:** For *low-sensitivity constraint private LPs*, moving to a neighboring database can change a single row of A by a small amount in each entry—for some row i, $\|A_i(D) - A_i(D')\|_\infty \leq 1/n$. We show how to solve these LPs using multiplicative weights; our techniques work equally well if the entire constraint matrix can change on neighboring problems (we will sometimes call these *low-sensitivity matrix* or *row private LPs*).
2. **The scalars:** For these linear programs, c and A are fixed and moving to a neighboring database only affects $b = b(D)$.
   (a) **High-sensitivity:** For *high-sensitivity scalar private LPs*, for every neighboring D, D', there is a row i such that for every $j \neq i$, $b(D)_j = b(D')_j$. We show that in general, such LPs cannot be solved privately.
   (b) **Low-sensitivity:** For *low-sensitivity scalar private LPs*, moving to a neighboring database can change every entry in b slightly, such that $\|b(D) - b(D')\|_\infty \leq 1/n$. These LPs capture the *private linear query release* problem, so we will sometimes refer to them as *query release LPs*. In this problem, the database is viewed as a histogram $D \in \mathbb{N}^d_+$ and the objective is to find a *synthetic database* $x \in \mathbb{R}^d_+$ such that for every linear query q in some family, $\langle q, x\rangle \approx \langle q, D\rangle$. We show how to adapt existing techniques for this problem and derive resulting accurate solvers for LPs of this form.

3. **One column in** A: For these linear programs, moving to a neighboring database can affect at most one column of A.
   (a) **High-sensitivity:** For *high-sensitivity column private LPs*, for every neighboring D, D', the matrices $A(D), A(D')$ are arbitrarily different in a single column, and identical in all other columns. We show that in general, such LPs cannot be solved privately.
   (b) **Low-sensitivity:** For *low-sensitivity column private LPs*, moving to a neighboring database can change every entry in a single column of A by a small amount. More generally, if A_i is the ith *row* of A, then $\|A(D)_i - A(D')_i\|_1 \leq 1/n$ for each i. We show how to use these LPs using multiplicative weights.
4. **The objective:** For these linear programs, moving to a neighboring database can affect the objective c. The scalars b and constraints A remain unchanged.
   (a) **High-sensitivity:** For *high-sensitivity objective private LPs*, for every neighboring D, D', a single entry of the objective $c(D), c(D')$ can change arbitrarily. We show that in general, such LPs cannot be solved privately.
   (b) **Low-sensitivity:** For *low-sensitivity objective private LPs*, for every neighboring D, D', the objective vectors $c(D), c(D')$ satisfy $\|c(D) - c(D')\|_1 \leq 1/n$. This kind of linear program can be solved inefficiently to high accuracy by selecting from the set of vertices of the feasible polytope with the exponential mechanism; we show that linear programs in this class can also be solved efficiently and accurately, by directly using randomized response.

This taxonomy is summarized in Table 1. We will formally define accuracy, but roughly speaking, an accurate solution satisfies each constraint to within additive α, and has objective within additive α of optimal (when there is an objective). The exception is constraint privacy (indicated by the asterisk), where our algorithm finds a solution that satisfies only *most* of the constraints to within additive α, and may violate the other constraints arbitrarily.

Table 1. Efficient and accurate solvability

Location of change	High sensitivity	Low sensitivity
Objective c	No	Yes
Scalar b	No	Yes
Row/All of A	Yes*	Yes
Column of A	No	Yes

1.2 Related Work

Differential privacy emerged from a line of work initiated by Dinur and Nissim [6], was defined by Dwork et al. [8], and is now a standard definition of privacy in computer science. Below, we discuss relevant results in differential privacy; the survey by Dwork [7] is an excellent source for a more comprehensive overview.

Private optimization has been studied since the work of Blum et al. [2] and Kasiviswanathan et al. [17], who considered how to choose an optimal classifier privately. Blum et al. [2] give an efficient reduction from SQ learning to private SQ learning, and Kasiviswanathan et al. [17] give a very general but inefficient reduction from PAC learning to private PAC learning using the exponential mechanism of McSherry and Talwar [19]. Private learning was placed explicitly into an optimization framework by Chaudhuri et al. [5], who give two techniques for privately solving certain *unconstrained* convex optimization problems. Gupta et al. [11] give several algorithms for problems in private *combinatorial optimization*, but these were specialized combinatorial algorithms for specific problems.

In parallel, a line of work initiated by Blum et al. [3] and continuing with Dwork et al. [9], Roth and Roughgarden [22], Dwork et al. [10], Hardt and Rothblum [13], Gupta et al. [12], Hardt et al. [14] study the problem of privately producing *synthetic data* consistent with some private database on many *linear queries*. (Of particular note is the private multiplicative weights mechanism of Hardt and Rothblum [13], which achieves the optimal accuracy and running time bounds [23, 4].) This problem can be represented as a linear program with queries defining constraints, and indeed, the private multiplicative weights algorithm of Hardt and Rothblum [13] can be directly applied to solve this kind of linear program. This observation motivates our current investigation.

Our algorithms are mostly based on different variants of the *multiplicative weights* method of solving linear programs, which was introduced by Plotkin et al. [21] (see the excellent survey by Arora et al. [1] for more details). Whereas Plotkin et al. [21] maintain a distribution over the dual variables with multiplicative weights, depending on the kind of linear program we are solving, we either maintain a distribution over the dual variables or the primal variables. To solve *constraint private* LPs, we use a combination of the multiplicative weights update method and *Bregman projections* [1]—Hsu et al. [16] use a similar version of this technique in designing *analyst private* mechanisms.

2 Differential Privacy Preliminaries

We defer differential privacy preliminaries to the extended version of the paper.

3 Constraint Private LPs

Let us begin by considering *constraint private LPs*, with the general form

$$\begin{aligned} &\max_{x \in \mathcal{K}} c^\top x \\ &\text{s.t. } Ax \leq b, \end{aligned}$$

where $A \in \mathbb{R}^{m \times d}, b \in \mathbb{R}^m, c \in \mathbb{R}^d$, and $\mathcal{K} \subseteq \mathbb{R}^d$. We think of $\mathcal{K}$ as the *easy* constraints, those that are independent of the database, like non-negativity.

Let $\mathcal{K}_{\mathrm{OPT}} = \mathcal{K} \cap \{x \in \mathbb{R}^d \mid c^\top x = \mathrm{OPT}\}$. Then, the original LP can be solved approximately by repeatedly solving the feasibility problem

$$\begin{aligned} &\text{find } x \in \mathcal{K}_{\mathrm{OPT}} \\ &\text{s.t. } Ax \leq b, \end{aligned}$$

binary searching on the optimal objective value OPT.[1] Thus, unless we specify otherwise, we will restrict our attention to feasibility LPs. Furthermore, since a linear program has a convex feasible region, $\mathcal{K}$ (and hence $\mathcal{K}_{\mathrm{OPT}}$) are convex. From now on, we will write $\mathcal{K}$ for $\mathcal{K}_{\mathrm{OPT}}$.

For constraint privacy, we want to find a solution that hides whether a single constraint is in the LP or not. Formally:

Definition 1. *A randomized algorithm $\mathcal{M}$ with inputs $m \in \mathbb{N}$, vector $b \in \mathbb{R}^m$, and matrix $A \in \mathbb{R}^{m \times d}$ and outputting a vector in $\mathbb{R}^d$ is* (ϵ, δ)-high sensitivity constraint private *if for any A, A' such that A' is equal to A with an additional row appended, and b, b' such that b' is equal to b with an additional entry,*

$$\Pr[\mathcal{M}(m, b, A) \in S] \leq e^{\epsilon} \Pr[\mathcal{M}(m+1, b', A') \in S] + \delta$$

for any set $S \subseteq \mathbb{R}^d$.

3.1 Solving LPs with Dense Multiplicative Weights

A standard approach to solving LPs is via *no-regret* algorithms. While LPs can be solved using any no-regret algorithm, for concreteness we use the multiplicative weights update algorithm.

We will use a variant of the standard multiplicative weights algorithm that maintains a *dense distribution* over the set of constraints, i.e., a distribution that doesn't place too much probability on any action. We will call this algorithm, due to Herbster and Warmuth [15], the *dense multiplicative weights algorithm* (Algorithm 1). Roughly, the algorithm projects the MW distribution on actions into the set of dense distributions at each step. The loss at each step will be defined by a point that approximately satisfies the average constraint weighted by the MW distribution—by capping the probability on any constraint, we ensure that this point can be selected privately even when a single constraint can change arbitrarily on neighboring instances.

We first define this projection step, also known as a *Bregman projection.*

Definition 2. *Let $s > 0$. Given a measure A such that $|A| \leq s$, let $\Gamma_s A$ be the* (Bregman) projection of A into the set of $1/s$-dense distributions, *defined by $\Gamma_s A_a = \frac{1}{s} \cdot \min\{1, cA_a\}$ for every $a \in \mathcal{A}$, where $c \geq 0$ is such that $s = \sum_{a \in \mathcal{A}} \min\{1, cA_a\}$.*

Then, we can define the Dense Multiplicative Weights algorithm, which uses the standard multiplicative weights update rule combined with a Bregman projection into the set of dense distributions after each step.

Recall we can assume that we know the optimal value OPT, so the objective can be represented as the constraint $c^{\top} x = \mathrm{OPT}$. Hence, let $\mathcal{K} = \{x \in \mathbb{R}^d_+ \mid$

[1] Binary search will incur an additional overhead in privacy, but in some situations may not be necessary: for instance, if a bound on the sensitivity of the optimal objective is known, we can solve the LP non-privately and estimate OPT with the Laplace mechanism.

Algorithm 1. The Dense Multiplicative Weights algorithm, $DMW_{s,\eta}$

```
Let $A_1$ be the uniform measure on $\mathcal{A}$
For $t = 1, 2, \ldots, T$:
   Let $\widetilde{B}^t = \Gamma_s A^t$
   Receive loss vector $\ell^t$ (may depend on $B^1, \ldots, B^t$)
   **Update: For each $a \in \mathcal{A}$:**
      Update $A_a^{t+1} = e^{-\eta \ell_a^t} A_a^t$
```

$c^\top x = \text{OPT}\}$ be the public feasible set. We will assume that there is a known, data-independent upper bound ρ such that

$$\rho \geq \max_D \max_{x \in \mathcal{K}} \|A(D)x - b(D)\|_\infty,$$

which we call the *width* of the LP.

We will define our algorithm in terms of an approximate oracle for solving a linear minization problem. (For a concrete example of such an oracle in the context of fractional set cover, see the next section.)

Definition 3. *An* (α, β)-approximate, ρ-bounded oracle*, given a distribution* $y \in \mathbb{R}^m$ *and matrix* $A \in \mathbb{R}^{m \times d}$*, with probability at least* $1 - \beta$ *finds* $x^* \in \mathbb{R}^d$ *with*

$$\sum_{i=1}^m y_i(A_i \cdot x^*) \leq \min_{x \in \mathcal{K}} \sum_{i=1}^m y_i(A_i \cdot x) + \alpha$$

and $\|Ax^* - b\|_\infty \leq \rho$*.*

We present the full algorithm in Algorithm 2.

Algorithm 2. Solving for LP feasibility with dense multiplicative weights

```
Input $A \in \mathbb{R}^{m \times d}$, $b \in \mathbb{R}^m$.
Let $\widetilde{y}^1$ be the uniform distribution in $\mathbb{R}^m$, $\rho \geq \max_{x \in \mathcal{K}} \|Ax - b\|_\infty$ be the *width* of
the LP, $s \in \mathbb{N}$ be the *density parameter*, and $\alpha > 0$ be the desired accuracy. Let
*Oracle* be an $(\alpha, \beta)$-accurate, $\rho$-bounded oracle, and set

   $$\eta = \sqrt{\frac{\log m}{T}}, \qquad T = \frac{36\rho^2 \log m}{\alpha^2}.$$

For $t = 1, \ldots, T$:
   Find $x^t = Oracle(\widetilde{y}^t, A)$
   Compute losses $\ell_i^t := (1/2\rho)(b_i - A_i \cdot x^t) + 1/2$.
   Update $\widetilde{y}^{t+1}$ from $\widetilde{y}^t$ and $\ell^t$ via dense multiplicative weights with density $s$.
Output $\overline{x} = (1/T) \sum_{t=1}^T x^t$.
```

If the oracle is sufficiently accurate, the following theorem bounds the number of iterations needed to achieve a set level of accuracy.

Theorem 1. *Let $0 < \alpha \leq 9\rho$, and let $\beta \in (0, 1)$. Suppose there is a feasible solution of the linear program. Then with probability at least $1 - \beta$, Algorithm 2 with density parameter s run with an $(\alpha/3, \beta/T)$-approximate, ρ-bounded oracle finds a point x^* in $\mathcal{K}$ such that there is a set of constraints S of size at most $|S| < s$, with $A_i x^* \leq b_i + \alpha$ for every $i \notin S$.*

We also have the following privacy guarantee.

Theorem 2. *Let $\epsilon, \delta, T > 0$, and let*

$$\epsilon' = \frac{\epsilon}{\sqrt{8T \log(1/\delta)}}.$$

with density parameter $s \in \mathbb{N}$. Suppose the oracle is ϵ'-private, where on neighboring instances the inputs (distributions) $\widetilde{y}, \widetilde{y}'$ satisfy

$$\|\widetilde{y}\|_\infty \leq 1/s, \qquad \|\widetilde{y}'\|_\infty \leq 1/s, \qquad \|\widetilde{y} - \widetilde{y}'\|_1 \leq 2/s,$$

and the matrices A, A' are exactly the same except one has an additional row, and the vectors b, b' except one has a corresponding additional entry. Then, Algorithm 2 with density s is (ϵ, δ)-high sensitivity constraint private.

Now that we have presented our algorithm for solving LPs under constraint privacy, we give an example of how to instantiate the oracle and apply Theorem 1.

3.2 Private Fractional Set Cover

We will consider the example of the *fractional set cover* LP, though our arguments extend to constraint private LPs with a private oracle that has low width. (For example, many covering and packing LPs satisfy this property.)

Suppose there are d sets, each covering some subset of m people. Each set has a cost c_S, and we wish to select the cheapest collection of sets that covers every person. We will consider the fractional relaxation of this problem, where instead of selecting whole sets for the cover, we can decide to select a fraction of each set, i.e., each set can be chosen to some non-negative degree, and the cost for set S is the degree to which it is open times c_S. We again want the cheapest fractional collection of sets, such that at least weight 1 covers each person.[2]

To formulate this as a linear program, let the variables be $x \in \mathbb{R}^d_+$; variable x_S will be the degree that we choose set S in the cover. For the constraints, let $A_i \in \{0, 1\}^m$ such that A_{iS} is 1 exactly when set S covers i, otherwise 0.

We will assume that the optimal value OPT is known, and the goal is to compute an approximate fractional set covering x^* corresponding to OPT. This is equivalent to solving the following linear program:

$$\begin{aligned} &\text{find: } x \in \mathcal{K} \\ &\text{s.t. } A_i \cdot x \geq 1 \quad \text{for each } i \end{aligned}$$

[2] To highlight the constraint private LP, we will only consider the fractional version. It is also possible to round the fractional solution to an integral solution (with slightly worse cost), since randomized rounding is independent of the private data.

where $\mathcal{K} = \{x \in \mathbb{R}^d_+ \mid c \cdot x = \mathrm{OPT}\}$ is the feasible region.

We wish to achieve constraint privacy: if each individual corresponds to a covering constraint, then we want an approximate solution that is hides whether a person i needs to be covered or not. This is not always possible—if each set contains just one person, then the presence of a set in any valid covering will reveal information about the people that need to be covered. Thus, we will find a solution violating a few constraints, so only covering *most* people.

To use our constraint private LP solver, we first define a private oracle solving the minimization problem

$$O(y) = \operatorname*{argmin}_{x \in \mathcal{K}} \sum_i y_i (A_i \cdot x).$$

Since the oracle is minimizing a linear function, the optimal point lies at a vertex of $\mathcal{K}$ and is of the form

$$x^* = \frac{\mathrm{OPT}}{c_i} e_i$$

for some i, where e_i is the i'th standard basis vector, i.e., all zeros except for a 1 in the i'th coordinate. We can use the exponential mechanism to privately select this vertex. Now, it follows that Algorithm 2 solves the private fractional set cover problem with the following accuracy guarantee.

Theorem 3. *Let $\beta \in (0, 1)$. With probability at least $1 - \beta$, Algorithm 2 with the exponential mechanism as an oracle—where ρ is the width of the oracle and $\alpha \leq 9\rho$—finds a point x^* such that $A_i x^* \geq 1 - \alpha$ except for at most s constraints i, where*

$$s = \widetilde{O}\left(\frac{\mathrm{OPT}^2 \log d \log^{1/2} m \log(1/\beta) \log^{1/2}(1/\delta)}{c^2 \cdot \alpha^2 \cdot \epsilon}\right).$$

Algorithm 2 is also ϵ-high sensitivity constraint private.

Remark 1. A variant of the efficient private set cover problem has been investigated by Gupta et al. [11]. Our techniques are more general, but the solution we provide here has an imcomparable accuracy guarantee. We include this example to demonstrate how to use Algorithm 2 and Theorem 1.

4 Low-Sensitivity LPs

Let us now turn to low-sensitivity LPs. Recall that for these LPs, the distance between adjacent inputs decreases as the size of the database (i.e., the number of individuals) grows. First, a few simplifying assumptions. Like above, we will continue to solve feasibility LPs of the following form:

$$\begin{aligned} &\text{find } x \in \mathbb{R}^d_+ \\ &\text{s.t. } Ax \leq b \end{aligned}$$

Unlike the case for general constraint private LPs, we require that the feasible solution is a distribution, i.e., is non-negative and has ℓ_1 norm 1. Note that if the optimal solution has ℓ_1 norm L, then the rescaled LP

$$\begin{aligned} &\text{find } x \in \mathbb{R}^d_+ \\ &\text{s.t. } Ax \leq b/L \end{aligned}$$

has a distribution as a solution. Our algorithms will find a point x^* such that $Ax^* \leq b/L + \alpha \cdot \mathbf{1}$, so if we set $\alpha = \alpha'/L$, then $A(Lx^*) \leq b + \alpha'$ gives an approximate solution to the original, unscaled LP.

4.1 Scalar-Private LPs

We defer our scalar-private results to the extended version of the paper.

4.2 Row/Matrix-Private LPs

Suppose we have the feasibility problem

$$\begin{aligned} &\text{find } x \\ &\text{s.t. } Ax \leq b, \end{aligned}$$

where some entries in A may change by at most Δ_∞ on a neighboring instance.

For row privacy, we want to find a solution that hides a single constraint's low sensitivity change. Formally:

Definition 4. *A randomized algorithm $\mathcal{M}$ with inputs vector $b \in \mathbb{R}^m$ and matrix $A \in \mathbb{R}^{m \times d}$, and outputting a vector in $\mathbb{R}^d$ is* (ϵ, δ)-low sensitivity row private with sensitivity Δ_∞ *if for any A, A' such that $\|A - A'\|_\infty \leq \Delta_\infty$,*

$$\Pr[\mathcal{M}(b, A) \in S] \leq e^\epsilon \Pr[\mathcal{M}(b', A') \in S] + \delta$$

for any set $S \subseteq \mathbb{R}^d$.

We give a solver in the extended version of the paper, with the following guarantee.

Theorem 4. *Let $\epsilon, \delta, \beta > 0$, and let Δ_∞ be the sensitivity of the LP. Suppose the program has a distribution as a feasible solution. There is an (ϵ, δ)-low sensitivity row private algorithm with sensitivity Δ_∞ that with probability at least $1 - \beta$, produces a point x^* with $Ax^* \leq b + \alpha \cdot \mathbf{1}$, where*

$$\alpha = \widetilde{O}\left(\frac{\Delta_\infty^{1/2} d^{1/4}}{\epsilon^{1/2}} \cdot \operatorname{polylog}\left(d, m, \frac{1}{\beta}, \frac{1}{\delta}\right)\right).$$

4.3 Column Private LPs

We defer our column private results, which are quite similar to our row private results, to the extended version of the paper.

4.4 Objective Private LPs

For our final type of low-sensitivity LP, we consider linear programs with objectives that depend on private data. We show that a very simple approach—*randomized response*—can solve these types of LPs accurately. Throughout, we will assume that the optimal solution to the LP has ℓ_1 weight equal to 1. We start with an LP in general form:

$$\begin{aligned} &\max c^\top x \\ &\text{s.t. } Ax \leq b, \end{aligned}$$

On instances corresponding to neighboring database D, D', the objective may change by $\varDelta_1$ in ℓ_1 norm: $\|c(D) - c(D')\|_1 \leq \varDelta_1$. Formally:

Definition 5. *A randomized algorithm $\mathcal{M}$ with inputs vectors $b \in \mathbb{R}^m$, $c \in \mathbb{R}^d$ and matrix $A \in \mathbb{R}^{m \times d}$, and outputting a vector in $\mathbb{R}^d$ is (ϵ, δ)*-low sensitivity objective private with sensitivity $\varDelta_1$ *if for any c, c' such that $\|c - c'\|_1 \leq \varDelta_1$,*

$$\Pr[\mathcal{M}(c, b, A) \in S] \leq e^\epsilon \Pr[\mathcal{M}(c', b, A) \in S] + \delta$$

for any set $S \subseteq \mathbb{R}^d$.

Theorem 5. *Suppose an objective private LP has optimal objective* OPT*, and has optimal solution with ℓ_1 weight* 1*. Define*

$$\hat{c} = c + \text{Lap}\left(\frac{\varDelta_1 \sqrt{8d \log(1/\delta)}}{\epsilon}\right)^d,$$

where the noise is d independent draws from the Laplace distribution with the given parameter. Then, releasing the perturbed LP

$$\begin{aligned} &\max \hat{c}^\top x \\ &\textit{s.t. } Ax \leq b \quad \textit{and} \quad \mathbf{1}^\top x = 1 \end{aligned}$$

is (ϵ, δ)-low sensitivity objective private with sensitivity $\varDelta_1$. With probability $1 - \beta$, solving the perturbed LP non-privately yields a point x^* such that $Ax^* \leq b$ and $c^\top x^* \geq \text{OPT} - \alpha$, where

$$\alpha = \frac{4\varDelta_1 \sqrt{8d \log(d/\delta)}}{\epsilon}.$$

5 Lower Bounds

We defer details of our lower bounds to the extended version of the paper.

References

[1] Arora, S., Hazan, E., Kale, S.: The multiplicative weights update method: a meta-algorithm and applications. Theory of Computing 8(1), 121–164 (2012)

[2] Blum, A., Dwork, C., McSherry, F., Nissim, K.: Practical privacy: the sulq framework. In: ACM SIGACTSIGMODSIGART Symposium on Principles of Database Systems (PODS), Baltimore, Maryland (2005)

[3] Blum, A., Ligett, K., Roth, A.: A learning theory approach to noninteractive database privacy. Journal of the ACM 60(2), 12 (2013)

[4] Bun, M., Ullman, J., Vadhan, S.P.: Fingerprinting codes and the price of approximate differential privacy. In: ACM SIGACT Symposium on Theory of Computing (STOC), pp. 1–3. ACM, New York (2014)

[5] Chaudhuri, K., Monteleoni, C., Sarwate, A.D.: Differentially private empirical risk minimization. Journal of Machine Learning Research 12, 1069–1109 (2011)

[6] Dinur, I., Nissim, K.: Revealing information while preserving privacy. In: ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS), San Diego, California, pp. 202–210 (2003)

[7] Dwork, C.: Differential privacy: A survey of results. In: Agrawal, M., Du, D.-Z., Duan, Z., Li, A. (eds.) TAMC 2008. LNCS, vol. 4978, pp. 1–19. Springer, Heidelberg (2008)

[8] Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006)

[9] Dwork, C., Naor, M., Reingold, O., Rothblum, G.N., Vadhan, S.: On the complexity of differentially private data release: efficient algorithms and hardness results. In: ACM SIGACT Symposium on Theory of Computing (STOC), Bethesda, Maryland, pp. 381–390 (2009)

[10] Dwork, C., Rothblum, G.N., Vadhan, S.: Boosting and differential privacy. In: IEEE Symposium on Foundations of Computer Science (FOCS), Las Vegas, Nevada, pp. 51–60 (2010)

[11] Gupta, A., Ligett, K., McSherry, F., Roth, A., Talwar, K.: Differentially private combinatorial optimization. In: ACM SIAM Symposium on Discrete Algorithms (SODA), Austin, Texas, pp. 1106–1125 (2010)

[12] Gupta, A., Roth, A., Ullman, J.: Iterative constructions and private data release. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 339–356. Springer, Heidelberg (2012)

[13] Hardt, M., Rothblum, G.N.: A multiplicative weights mechanism for privacy-preserving data analysis. In: IEEE Symposium on Foundations of Computer Science (FOCS), Las Vegas, Nevada, pp. 61–70 (2010)

[14] Hardt, M., Ligett, K., McSherry, F.: A simple and practical algorithm for differentially private data release. In: Conference on Neural Information Processing Systems (NIPS), Lake Tahoe, California, pp. 2348–2356 (2012)

[15] Herbster, M., Warmuth, M.K.: Tracking the best linear predictor. Journal of Machine Learning Research 1, 281–309 (2001)

[16] Hsu, J., Roth, A., Ullman, J.: Differential privacy for the analyst via private equilibrium computation. In: ACM SIGACT Symposium on Theory of Computing (STOC), Palo Alto, California, pp. 341–350 (2013)

[17] Kasiviswanathan, S.P., Lee, H.K., Nissim, K., Raskhodnikova, S., Smith, A.: What can we learn privately? SIAM Journal on Computing 40(3), 793–826 (2011)

[18] Kifer, D., Smith, A., Thakurta, A.: Private convex empirical risk minimization and high-dimensional regression. Journal of Machine Learning Research 1, 41 (2012)

[19] McSherry, F., Talwar, K.: Mechanism design via differential privacy. In: IEEE Symposium on Foundations of Computer Science (FOCS), Providence, Rhode Island (2007)

[20] Nissim, K., Raskhodnikova, S., Smith, A.: Smooth sensitivity and sampling in private data analysis. In: ACM SIGACT Symposium on Theory of Computing (STOC), San Diego, Illinois, pp. 75–84 (2007)

[21] Plotkin, S.A., Shmoys, D.B., Tardos, É.: Fast approximation algorithms for fractional packing and covering problems. Mathematics of Operations Research 20(2), 257–301 (1995)

[22] Roth, A., Roughgarden, T.: Interactive privacy via the median mechanism. In: ACM SIGACT Symposium on Theory of Computing (STOC), Cambridge, Massachusetts, pp. 765–774

[23] Ullman, J.: Answering $n^{2+o(1)}$ counting queries with differential privacy is hard. In: ACM SIGACT Symposium on Theory of Computing (STOC), Palo Alto, California, pp. 361–370 (2013)

[24] Ullman, J., Vadhan, S.: PCPs and the hardness of generating private synthetic data. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 400–416. Springer, Heidelberg (2011)

How Unsplittable-Flow-Covering Helps Scheduling with Job-Dependent Cost Functions[⋆],[⋆⋆]

Wiebke Höhn[1], Julián Mestre[2], and Andreas Wiese[3]

[1] Technische Universität Berlin, Germany
hoehn@math.tu-berlin.de
[2] The University of Sydney, Australia
mestre@it.usyd.edu.au
[3] Max-Planck-Institut für Informatik, Saarbücken, Germany
awiese@mpi-inf.mpg.de

Abstract. Generalizing many well-known and natural scheduling problems, scheduling with job-specific cost functions has gained a lot of attention recently. In this setting, each job incurs a cost depending on its completion time, given by a private cost function, and one seeks to schedule the jobs to minimize the total sum of these costs. The framework captures many important scheduling objectives such as weighted flow time or weighted tardiness. Still, the general case as well as the mentioned special cases are far from being very well understood yet, even for only one machine. Aiming for better general understanding of this problem, in this paper we focus on the case of uniform job release dates on one machine for which the state of the art is a 4-approximation algorithm. This is true even for a special case that is equivalent to the covering version of the well-studied and prominent unsplittable flow on a path problem, which is interesting in its own right. For that covering problem, we present a quasi-polynomial time $(1+\varepsilon)$-approximation algorithm that yields an $(e+\varepsilon)$-approximation for the above scheduling problem. Moreover, for the latter we devise the best possible resource augmentation result regarding speed: a polynomial time algorithm which computes a solution with *optimal* cost at $1+\varepsilon$ speedup. Finally, we present an elegant QPTAS for the special case where the cost functions of the jobs fall into at most $\log n$ many classes. This algorithm allows the jobs even to have up to $\log n$ many distinct release dates. All proposed quasi-polynomial time algorithms require the input data to be quasi-polynomially bounded.

1 Introduction

In scheduling, a natural way to evaluate the quality of a computed solution is to assign a cost to each job which depends on its completion time. The goal is then to minimize the sum of these costs. The function describing this dependence may be completely different for each job. There are many well-studied and important

[⋆] Funded by the Go8-DAAD joint research cooperation scheme.

[⋆⋆] A full version of the paper can be found at http://arxiv.org/abs/1403.1376.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 625–636, 2014.

scheduling objectives which can be cast in this framework. Some of them are already very well understood, for instance weighted sum of completion times $\sum_j w_j C_j$ for which there are polynomial time approximation schemes (PTASs) [1], even for multiple machines and very general machine models. On the other hand, for natural and important objectives such as weighted flow time or weighted tardiness, not even a constant factor polynomial time approximation algorithm is known, even on a single machine. In a recent break-through result, Bansal and Pruhs presented a $O(\log\log P)$-approximation algorithm [6,7] for the single machine case where every job has its private cost function, denoting by P the range of the processing times. Formally, they study the General Scheduling Problem (GSP) where the input consists of a set of jobs J where each job $j \in J$ is specified by a processing time p_j, a release date r_j, and a non-decreasing cost function f_j, and the goal is to compute a preemptive schedule on one machine which minimizes $\sum_j f_j(C_j)$ where C_j denotes the completion time of job j in the computed schedule. Interestingly, even though this problem is very general, subsuming all the objectives listed above, the best known complexity result for it is only strong NP-hardness, so there might even be a polynomial time $(1+\varepsilon)$-approximation.

Aiming to better understand GSP, in this paper we investigate the special case that all jobs are released at time 0. This case is still strongly NP-hard [20] and the currently best know approximation algorithm for it is a $(4+\varepsilon)$-approximation algorithm [18,22][1]. As observed by Bansal and Verschae [8], this problem is a generalization of the covering-version of the well-studied Unsplittable Flow on a Path problem (UFP) [2,3,5,11,14,17]. The input of this problem consists of a path, each edge e having a demand u_e, and a set of tasks T. Each task i is specified by a start vertex s_i, an end vertex t_i, a size p_i, and a cost c_i. In the covering version, the goal is to select a subset of the tasks $T' \subseteq T$ which covers the demand profile, i.e., $\sum_{i \in T' \cap T_e} p_i \geq u_e$ where T_e denotes all tasks in T whose path uses e. The objective is to minimize the total cost $\sum_{i \in T'} c_i$.

This covering version of UFP has applications to resource allocation settings such as workforce and energy management, making it an interesting problem in its own right. For example, one can think of the tasks as representing time intervals when employees are available, and one aims at providing certain service level that changes over the day. UFP-cover is a generalization of the knapsack cover problem [12] and corresponds to instances of GSP without release dates where the cost function of each job attains only the values 0, some job-dependent value c_i, and ∞. The best known approximation algorithm for UFP-cover is a 4-approximation [9,13], which essentially matches the best known result for GSP without release dates.

Our Contribution. In this paper we present several new approximation results for GSP without release dates and some of its special cases. First, we give a

[1] In [18] a primal-dual $(2+\varepsilon)$-approximation algorithm was claimed for this problem. However, there is a error in the argumentation: there are instances [22] where the algorithm constructs a dual solution whose value differs from the optimal integral solution by a factor of 4.

$(1+\varepsilon)$-approximation algorithm for the covering version of UFP with quasi-polynomial running time. Our algorithm follows the high-level idea of the known QPTAS for the packing version [3]. Its key concept is to start with an edge in the middle and to consider the tasks using it. One divides these tasks into groups, all tasks in a group having roughly the same size and cost, and guesses for each group an approximation of the capacity profile used by the tasks from that group. In the packing version, one can show that by slightly underestimating the true profile one still obtains almost the same profit as the optimum. For the covering version, a natural adjustment would be to use an approximate profile which *over*estimates the true profile. However, when using only a polynomial number of approximate profiles, it can happen that in the instance there are simply not enough tasks from a group available so that one can cover the overestimated profile which approximates the actual profile in the best possible way.

We remedy this problem in a maybe counterintuitive fashion. Instead of guessing an approximate upper bound of the true profile, we first guess a *lower* bound of it. Then we select tasks that cover this lower bound, and finally add a small number of "maximally long" additional tasks. Using this procedure, we cannot guarantee (instance-independently) how much our selected tasks exceed the guessed profile on each edge. However, we can guarantee that for the correctly guessed profile, we cover at least as much as the optimum and pay only slightly more. Together with the recursive framework from [3], we obtain a QPTAS. As an application, we use this algorithm to get a quasi-polynomial time $(e+\varepsilon)$-approximation algorithm for GSP with uniform release dates, improving the approximation ratio of the best known polynomial time 4-approximation algorithm [18,22]. This algorithm, as well as the QPTAS mentioned below, requires the input data to be quasi-polynomially bounded.

Moreover, we consider a different way to relax the problem. Rather than sacrificing a $1+\varepsilon$ factor in the objective value, we present a polynomial time algorithm that computes a solution with *optimal* cost but requiring a speedup of $1+\varepsilon$. Such a result can be easily obtained for job-*independent*, scalable cost functions using the PTAS in [21] (a cost function f is scalable if $f(c\,t)=\phi(c)\,f(t)$ for some suitable function ϕ and all all $c,t\geq 0$). In our case, however, the cost functions of the jobs can be much more complicated and, even worse, they can be different for each job. Our algorithm first imposes some simplification on the solutions under consideration, at the cost of a $(1+\varepsilon)$-speedup. Then, we use a recently introduced technique to first guess a set of discrete intervals representing slots for large jobs and then use a linear program to simultaneously assign large jobs into these slots and small jobs into the remaining idle times [24].

An interesting open question is to design a (Q)PTAS for GSP without release dates. As a first step towards this goal, recently Megow and Verschae [21] presented a PTAS for minimizing the objective function $\sum_j w_j g(C_j)$ where each job j has a private weight w_j but the function g is identical for all jobs. In Section 4 we present a QPTAS for a generalization of this setting. Instead of only one function g for all jobs, we allow up to $(\log n)^{O(1)}$ such functions, each job using one of them, and we even allow the jobs to have up to $(\log n)^{O(1)}$ distinct release dates. We note that our algorithms requires the weights of the jobs to be in a quasi-polynomial range. Despite the fact that this setting is much more general, our algorithm is very clean and easy to analyze.

Related Work. As mentioned above, Bansal and Pruhs present a $O(\log \log P)$-approximation algorithm for GSP [6]. Even for some well-studied special cases, this is now the best known polynomial time approximation result. For instance, for the important weighted flow time objective, previously the best known approximation factors were $O(\log^2 P)$, $O(\log W)$ and $O(\log nP)$ [4,16], where P and W denote the ranges of the job processing times and weights, respectively. A QPTAS with running time $n^{O_\varepsilon(\log P \log W)}$ is also known [15]. For the objective of minimizing the weighted sum of completion times, PTASs are known, even for an arbitrary number of identical and a constant number of unrelated machines [1].

For the case of GSP with identical release dates, Bansal and Pruhs [6] give a 16-approximation algorithm. Later, Shmoys and Cheung claimed a primal-dual $(2+\varepsilon)$-approximation algorithm [18]. However, an instance was later found where the algorithm constructs a dual solution which differs from the best integral solution by a factor 4 [22], suggesting that the primal-dual analysis can show only an approximation ratio of 4. On the other hand, Mestre and Verschae [22] showed that the local-ratio interpretation of that algorithm (recall the close relation between the primal-dual schema and the local-ratio technique [10]) is in fact a pseudopolynomial time 4-approximation, yielding a $(4+\varepsilon)$-approximation in polynomial time.

As mentioned above, a special case of GSP with uniform release dates is a generalization for the covering version of Unsplittable Flow on a Path. For this special case, a 4-approximation algorithm is known [9,13]. The packing version is very well studied. After a series of papers on the problem and its special cases [5,11,14,17], the currently best known approximation results are a QPTAS [3] and a $(2+\varepsilon)$-approximation in polynomial time [2].

2 Quasi-PTAS for UFP-Cover

In this section, we present a quasi-polynomial time $(1+\varepsilon)$-approximation algorithm for the UFP-cover problem. Subsequently, we show how it can be used to obtain an approximation algorithm with approximation ratio $e+\varepsilon \approx 2.718+\varepsilon$ and quasi-polynomial running time for GSP without release dates. Throughout this section, we assume that the sizes of the tasks are quasi-polynomially bounded. Our algorithm follows the structure from the QPTAS for the packing version of Unsplittable Flow on a Path due to Bansal et al. [3]. First, we describe a recursive exact algorithm with exponential running time. Subsequently, we describe how to turn this routine into an algorithm with only quasi-polynomial running time and an approximation ratio of $1+\varepsilon$.

For computing the exact solution (in exponential time) one can use the following recursive algorithm: Given the path $G=(V,E)$, denote by e_M the edge in the middle of G and let T_M denote the tasks that use e_M. Our strategy is to "guess" which tasks in T_M are contained in OPT, the (unknown) optimal solution. Note that once these tasks are chosen, the remaining problem splits into the two independent subproblems given by the edges on the left and on the right of e_M, respectively, and the tasks whose paths are fully contained in them. Therefore, we enumerate all subsets of $T'_M \subseteq T_M$, denote by $\mathcal{T}_M$ the resulting set of sets. For each set $T'_M \in \mathcal{T}_M$ we recursively compute the optimal solution for the subpaths $\{e_1, ..., e_{M-1}\}$ and $\{e_{M+1}, ..., e_{|E|}\}$, subject to the tasks in T'_M being already chosen and that no more tasks from T_M are allowed to be chosen. The leaf subproblems are given when the path in the recursive call has only one edge. Since $|E| = O(n)$ this procedure has a recursion depth of $O(\log n)$ which is helpful when aiming at quasi-polynomial running time. However, since in each recursive step we try each set $T'_M \in \mathcal{T}_M$, the running time is exponential (even in one single step of the recursion). To remedy this issue, we will show that for any set $\mathcal{T}_M$ appearing in the recursive procedure there is a set $\bar{\mathcal{T}}_M$ which is of small size and which approximates $\mathcal{T}_M$ well. More precisely, we can compute $\bar{\mathcal{T}}_M$ in quasi-polynomial time (and it thus has only quasi-polynomial size) and there is a set $T^*_M \in \bar{\mathcal{T}}_M$ such that $c(T^*_M) \leq (1+\varepsilon) \cdot c(T_M \cap \mathrm{OPT})$ and T^*_M dominates $T_M \cap \mathrm{OPT}$. For any set of tasks T' we write $c(T') := \sum_{i \in T'} c_i$, and for two sets of tasks T_1, T_2, we say that T_1 *dominates* T_2 if $\sum_{i \in T_1 \cap T_e} p_i \geq \sum_{i \in T_2 \cap T_e} p_i$ for each edge e. We modify the above procedure such that we do recurse on sets in $\bar{\mathcal{T}}_M$ instead of $\mathcal{T}_M$. Since $\bar{\mathcal{T}}_M$ has quasi-polynomial size, $\bar{\mathcal{T}}_M$ contains the mentioned set T^*_M, and the recursion depth is $O(\log n)$, the resulting algorithm is a QPTAS. In the sequel, we describe the above algorithm in detail and show in particular how to obtain the set $\bar{\mathcal{T}}_M$.

2.1 Formal Description of the Algorithm

We use a binary search procedure to guess the optimal objective value B. First, we reject all tasks i whose cost is larger than B and select all tasks i whose cost is at most $\varepsilon B/n$. The latter cost at most $n \cdot \varepsilon B/n \leq \varepsilon B$ and thus only a factor $1+\varepsilon$ in the approximation ratio. We update the demand profile accordingly.

We define a recursive procedure UFPcover(E', T') which gets as input a subpath $E' \subseteq E$ of G and a set of already chosen tasks T'. Denote by $\bar{T}$ the set of all tasks $i \in T \setminus T'$ such that the path of i uses only edges in E'. The output of UFPcover(E', T') is a $(1+\varepsilon)$-approximation to the minimum cost solution for the subproblem of selecting a set of tasks $T'' \subseteq \bar{T}$ such that $T' \cup T''$ satisfy all demands of the edges in E', i.e., $\sum_{i \in (T' \cup T'') \cap T_e} p_i \geq u_e$ for each edge $e \in E'$. Note that there might be no feasible solution for this subproblem in which case we output ∞. Let e_M be the edge in the middle of E', i.e., at most $|E'|/2$ edges are on the left and on the right of e_M, respectively. Denote by $T_M \subseteq \bar{T}$ all tasks in $\bar{T}$ whose path uses e_M. As described above, the key is now to construct the set $\bar{\mathcal{T}}_M$ with the above properties. Given this set, we compute UFPcover$(E'_L, T' \cup T'_M)$

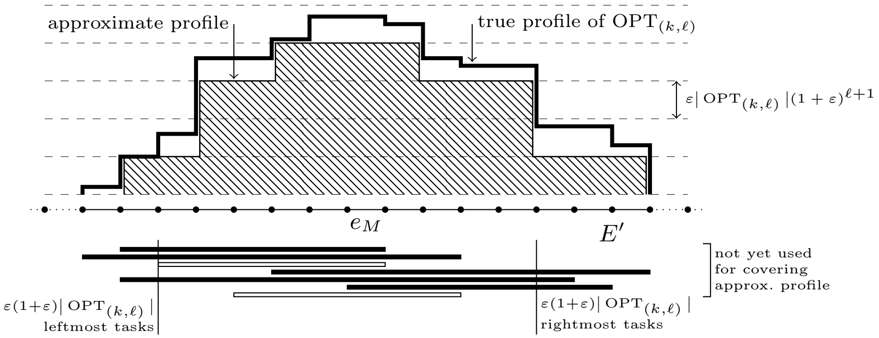

approximate profile
true profile of $\mathrm{OPT}_{(k,\ell)}$
$\varepsilon|\mathrm{OPT}_{(k,\ell)}|(1+\varepsilon)^{\ell+1}$
e_M
E'
not yet used for covering approx. profile
$\varepsilon(1+\varepsilon)|\mathrm{OPT}_{(k,\ell)}|$ leftmost tasks
$\varepsilon(1+\varepsilon)|\mathrm{OPT}_{(k,\ell)}|$ rightmost tasks

Fig. 1. Construction from Lemma 1

and UFPcover$(E'_R, T' \cup T'_M)$ for each set $T'_M \in \bar{\mathcal{T}}_M$, where E'_L and E'_R denote the subpaths of E' on the left and on the right of e_M, respectivley. We output

$$\min_{T'_M \in \bar{\mathcal{T}}_M} c(T'_M) + \mathrm{UFPcover}(E'_L, T' \cup T'_M) + \mathrm{UFPcover}(E'_R, T' \cup T'_M).$$

For computing the set $\bar{\mathcal{T}}_M$, we first group the tasks in T_M into $(\log n)^{O(1)}$ many groups, all tasks in a group having roughly the same costs and sizes. Formally, for each pair (k, ℓ), denoting (approximately) cost $(1+\varepsilon)^k$ and size $(1+\varepsilon)^\ell$, we define

$$T_{(k,\ell)} := \{i \in T_M : (1+\varepsilon)^k \leq c_i < (1+\varepsilon)^{k+1} \wedge (1+\varepsilon)^\ell \leq p_i < (1+\varepsilon)^{\ell+1}\}.$$

Since the sizes of the tasks are quasi-polynomially bounded and we preprocessed the weights of the tasks, we have $(\log n)^{O(1)}$ non-empty groups.

For each group $T_{(k,\ell)}$, we compute a set $\bar{\mathcal{T}}_{(k,\ell)}$ containing at least one set which is not much more expensive than $\mathrm{OPT}_{(k,\ell)} := \mathrm{OPT} \cap T_{(k,\ell)}$ and which dominates $\mathrm{OPT}_{(k,\ell)}$. To this end, observe that the sizes of the tasks in $\mathrm{OPT}_{(k,\ell)}$ cover a certain profile (see Figure 1). Initially, we guess the number of tasks in $\mathrm{OPT}_{(k,\ell)}$, and if $|\mathrm{OPT}_{(k,\ell)}| \leq \frac{1}{\varepsilon^2}$ then we simply enumerate all subsets of $T_{(k,\ell)}$ with at most $\frac{1}{\varepsilon^2}$ tasks. Otherwise, we consider a polynomial number of profiles that are potential approximations of the true profile covered by $\mathrm{OPT}_{(k,\ell)}$. To this end, we subdivide the (implicitly) guessed height of the true profile evenly into $\frac{1}{\varepsilon}$ steps of uniform height, and we allow the approximate profiles to use only those heights while being monotonously increasing and decreasing before and after e_M, respectively (observe that also $\mathrm{OPT}_{(k,\ell)}$ has this property since all its tasks use e_M). This leads to at most $n^{O(1/\varepsilon)}$ different approximate profiles in total.

For each approximate profile we compute a set of tasks covering it using LP-rounding. The path of any task in $T_{(k,\ell)}$ contains the edge e_M, and hence, a task covering an edge e always covers all edges inbetween e and e_M as well. Thus, when formulating the problem as an LP, it suffices to introduce one constraint for the leftmost and one constraint for the rightmost edge of each height in the approximated profile. We compute an extreme point solution of the LP and round up each of the at most $\frac{2}{\varepsilon}$ fractional variables. Since $|\operatorname{OPT}_{(k,\ell)}| \geq \frac{1}{\varepsilon^2}$ this increases the cost at most a factor $1 + O(\varepsilon)$ compared to the cost of the LP.

It is clear that the LP has a solution if the approximate profile is dominated by the true profile. Among such approximate profiles, consider the one that is closest to the latter. On each edge it would be sufficient to add $O(\varepsilon \cdot \left|\operatorname{OPT}_{(k,\ell)}\right|)$ tasks from $T_{(k,\ell)}$ in order to close the remaining gap. This is due to our choice of the step size of the approximate profile and the fact that all tasks in $T_{(k,\ell)}$ have roughly the same size. To this end, from the not yet selected tasks in $T_{(k,\ell)}$ we add the $O(\varepsilon \cdot |\operatorname{OPT}_{(k,\ell)}|)$ tasks with the leftmost start vertex and the $O(\varepsilon \cdot |\operatorname{OPT}_{(k,\ell)}|)$ tasks with the rightmost end vertex (see Figure 1). This costs again at most an $O(\varepsilon)$-fraction of the cost so far. As a result, on each edge e we have either selected $O(\varepsilon \cdot \left|\operatorname{OPT}_{(k,\ell)}\right|)$ additional tasks using it, thus closing the remaining gap, or we have selected *all* tasks from $T_{(k,\ell)}$ using e. In either case, the selected tasks dominate the tasks in $\operatorname{OPT}_{(k,\ell)}$, i.e., the true profile.

Lemma 1. *Given a group $T_{(k,\ell)}$. There is a polynomial time algorithm which computes a set of task sets $\bar{\mathcal{T}}_{(k,\ell)}$ which contains a set $T^*_{(k,\ell)} \in \bar{\mathcal{T}}_{(k,\ell)}$ such that $c(T^*_{(k,\ell)}) \leq (1+\varepsilon) \cdot c(\operatorname{OPT}_{(k,\ell)})$ and $T^*_{(k,\ell)}$ dominates $\operatorname{OPT}_{(k,\ell)}$.*

We define the set $\bar{\mathcal{T}}_M$ by taking all combinations of selecting exactly one set from the set $\bar{\mathcal{T}}_{(k,\ell)}$ of each group $T_{(k,\ell)}$. Since there are $(\log n)^{O(1)}$ groups, by Lemma 1 the set $\bar{\mathcal{T}}_M$ has only quasi-polynomial size and it contains one set T^*_M which is a a good approximation to $T_M \cap \operatorname{OPT}$, i.e., the set T^*_M dominates $T_M \cap \operatorname{OPT}$ and it is at most by a factor $1+O(\varepsilon)$ more expensive. Now each node in the recursion tree has at most $n^{(\log n)^{O(1)}}$ children and, as argued above, the recursion depth is $O(\log n)$. Thus, a call to UFPcover$(E, \emptyset)$ has quasi-polynomial running time and yields a $(1 + O(\varepsilon))$-approximation for the overall problem.

Theorem 1. *For any $\varepsilon > 0$ there is a quasi-polynomial $(1+\varepsilon)$-approximation algorithm for UFP-cover if the sizes of the tasks are in a quasi-polynomial range.*

Bansal and Pruhs [6] give a 4-approximation-preserving reduction from GSP with uniform release dates to UFP-cover using geometric rounding. Here we observe that if instead we use *randomized geometric rounding* [19], then one can obtain an e-approximation-preserving reduction. Together with our QPTAS for UFP-cover, we get the following result.

Theorem 2. *For any $\varepsilon > 0$ there is a quasi-polynomial time $(e+\varepsilon)$-approximation algorithm for GSP with uniform release dates.*

3 General Cost Functions under Speedup

We present a polynomial time algorithm which computes a solution for an instance of GSP with uniform release dates whose cost is optimal and which is feasible if the machine runs with speed $1 + \varepsilon$ (rather than unit speed).

Let $1 > \varepsilon > 0$ be a constant and assume for simplicity that $\frac{1}{\varepsilon} \in \mathbb{N}$. For our algorithm, we first prove some properties that we can assume "at $1+\varepsilon$ speedup"; by this, we mean that there is a schedule whose cost is at most the optimal cost (without enforcing these restricting properties) and which is feasible if we increase the speed of the machine by a factor $1+\varepsilon$. Many statements are similar to properties that are used in [1] for constructing PTASs for the problem of minimizing the weighted sum of completion times.

For a given schedule denote by S_j and C_j the start and end times of job j in a given schedule (recall that we consider only non-preemptive schedules). We define $C_j^{(1+\varepsilon)}$ to be the smallest power of $1+\varepsilon$ which is not smaller than C_j, i.e., $C_j^{(1+\varepsilon)} := (1+\varepsilon)^{\lceil \log_{1+\varepsilon} C_j \rceil}$, and adjust the objective function as given in the next lemma. Also, we impose that jobs that are relatively large are not processed too early; formally, they do not run before $(1+\varepsilon)^{\lfloor \log_{1+\varepsilon} \varepsilon \cdot p_j/(1+\varepsilon) \rfloor}$ which is the largest power of $1+\varepsilon$ which is at most $\varepsilon/(1+\varepsilon) \cdot p_j$ (the speedup will compensate for the delay of the start time).

Lemma 2. *At $1+O(\varepsilon)$ speedup we can use the objective function $\sum_j f_j\big(C_j^{(1+\varepsilon)}\big)$, instead of $\sum_j f_j(C_j)$, and assume $S_j \geq (1+\varepsilon)^{\lfloor \log_{1+\varepsilon} \varepsilon \cdot p_j/(1+\varepsilon) \rfloor}$ for each job j.*

Next, we discretize the time axis into intervals of the form $I_t := [R_t, R_{t+1})$ where $R_t := (1+\varepsilon)^t$ for any integer t. Note that $|I_t| = \varepsilon \cdot R_t$. Following Lemma 2, to simplify the problem we want to assign an artificial release date to each job j. For each job j, we define $r(j) := (1+\varepsilon)^{\lfloor \log_{1+\varepsilon} \varepsilon \cdot p_j/(1+\varepsilon) \rfloor}$. Lemma 2 implies then that we can assume $S_j \geq r(j)$ for each job j. Therefore, we interpret the value $r(j)$ as the release date of job j and from now on disallow to start job j before time $r(j)$.

In a given schedule, we call a job j *large* if $S_j \leq \frac{1}{\varepsilon^3} \cdot p_j$ and *small* otherwise. For the large jobs, we do not allow arbitrary starting times but we discretize the time axis such that each interval contains only a constant number of starting times for large jobs (for constant ε). For the small jobs, we do not want them to overlap over interval boundaries and we want that all small jobs scheduled in an interval I_t are scheduled during one (connected) subinterval $I_t^s \subseteq I_t$.

Lemma 3. *At $1+O(\varepsilon)$ speedup we can assume that*

- *each interval I_t contains only $O(\frac{1}{\varepsilon^3})$ potential start points for large jobs, and*
- *for each interval I_t there is a time interval $I_t^s \subseteq I_t$, ranging from one potential start point for large jobs to another, which fully contains all small jobs scheduled in I_t and no large jobs.*

For the moment, let us assume that the processing times of the instance are polynomially bounded. We will give a generalization to arbitrary instances later.

Our strategy is the following: Since the processing times are bounded, the whole schedule finishes within $\log_{1+\varepsilon}(\sum_j p_j) \leq O(\frac{1}{\varepsilon}\log n)$ intervals. Ideally, we would like to guess the placement of all large jobs in the schedule and then use a linear program to fill in the remaining small jobs. However, this would result in $n^{O(\frac{1}{\varepsilon}\log n)}$ possibilities for the large jobs, which is quasi-polynomial but not polynomial. Instead, we only guess the *pattern* of large-job usage for each interval. A pattern P for an interval is a set of $O(\frac{1}{\varepsilon^3})$ integers which defines the start and end times of the *slots* during which large jobs are executed in I_t. Note that such a job might start before I_t and/or end after I_t.

Proposition 1. *For each interval I_t there are only $N \in O_\varepsilon(1)$ many possible patterns, i.e., constantly many for constant ε. The value N is independent of t.*

We first guess all patterns for all intervals at once. Since there are only $O(\frac{1}{\varepsilon}\log n)$ intervals, this yields only $N^{O(\frac{1}{\varepsilon}\log n)} \in n^{O_\varepsilon(1)}$ possible combinations for all patterns for all intervals. Suppose now that we guessed the pattern corresponding to the optimal solution correctly. Next, we solve a linear program that in parallel assigns large jobs to the slots specified by the pattern, and also, it assigns small jobs into the remaining idle times on the intervals. Formally, we solve the following LP. We denote by Q the set of all slots for large jobs, $\mathsf{size}(s)$ denotes the length of a slot s, $\mathsf{begin}(s)$ its start time, and $t(s)$ denotes the index of the interval I_t that contains s. For each interval I_t denote by $\mathsf{rem}(t)$ the remaining idle time for small jobs, and consider these idle times as slots for small jobs, which we refer to by their interval indices $I := \{1, \ldots, \log_{1+\varepsilon}(\sum_j p_j)\}$. For each pair of slot $s \in Q$ and job $j \in J$, we introduce a variable $x_{s,j}$ corresponding to assigning j to s. Analogously, we use variables $y_{t,j}$ for the slots in I.

$$\min \sum_{j\in J}\Bigg(\sum_{s\in Q} f_j(R_{t(s)+1}) \cdot x_{s,j} + \sum_{t\in I} f_j(R_{t+1}) \cdot y_{t,j}\Bigg) \tag{1}$$

$$\begin{aligned}
\sum_{s\in Q} x_{s,j} + \sum_{t\in I} y_{t,j} &= 1 && \forall j \in J && (2)\\
\sum_{j\in J} x_{s,j} &\leq 1 && \forall\, s \in Q && (3)\\
\sum_{j\in J} p_j \cdot y_{t,j} &\leq \mathsf{rem}(t) && \forall\, t \in I && (4)\\
x_{s,j} &= 0 && \forall\, s \in Q, \forall j \in J:\ r(j) > \mathsf{begin}(s) \ \vee\ p_j > \mathsf{size}(s) && (5)\\
y_{t,j} &= 0 && \forall\, t \in I, \forall j \in J:\ r(j) > R_t \ \vee\ p_j > \varepsilon \cdot |I_t| && (6)\\
x_{s,j},\, y_{t,j} &\geq 0 && \forall\, s \in Q, \forall\, t \in I, \forall j \in J. && (7)
\end{aligned}$$

Denote the above LP by sLP. It has polynomial size and thus we can solve it efficiently. Borrowing ideas from [23] we round it to a solution that is not more costly and which can be made feasible using additional speedup of $1 + \varepsilon$.

Lemma 4. *Given a fractional solution (x, y) to* sLP*. In polynomial time, we can compute a non-negative integral solution (x', y') whose cost is not larger than the cost of (x, y) and which fulfills the constraints (2), (3), (5), (6),* (7) *and*

$$\sum_{j\in J} p_j \cdot y_{t,j} \leq \mathsf{rem}(t) + \varepsilon \cdot |I_t| \qquad \forall\, t \in I. \tag{4a}$$

In particular, the cost of the computed solution is no more than the cost of the integral optimum and it is feasible under $1 + O(\varepsilon)$ speedup (accumulating all the speedups from the previous lemmas). We remark that the technique of guessing patterns and filling them in by a linear program was first used in [24].

For the general case, i.e., for arbitrary processing times, we first show that at $1+\varepsilon$ speedup, we can assume that for each job j there are only $O(\log n)$ intervals between $r(j)$ (the artificial release date of j) and C_j. Then we devise a dynamic program which moves from left to right on the time axis and considers sets of $O(\log n)$ intervals at a time, using the above technique.

Theorem 3. *Let $\varepsilon > 0$. There is a polynomial time algorithm for GSP with uniform release dates which computes a solution with optimal cost and which is feasible if the machine runs with speed $1+\varepsilon$.*

4 Few Classes of Cost Functions

In this section, we study the following special case of GSP with release dates. We assume that each cost function f_j can be expressed as $f_j = w_j \cdot g_{u(j)}$ for a job-dependent weight w_j, k global functions $g_1, ..., g_k$, and an assignment $u : J \to [k]$ of cost functions to jobs. We present a QPTAS for this problem, assuming that $k = (\log n)^{O(1)}$ and that the jobs have at most $(\log n)^{O(1)}$ distinct release dates. We assume that the job weights are in a quasi-polynomial range, i.e., we assume that there is an upper bound $W = 2^{(\log n)^{O(1)}}$ for the (integral) job weights.

In our algorithm, we first round the values of the functions g_i so that they attain only few values, $(\log n)^{O(1)}$ many. Then we guess the $(\log n)^{O(1)}/\varepsilon$ most expensive jobs and their costs. For the remaining problem, we use a linear program. Since we rounded the functions g_i, our LP is sparse, and by rounding an extreme point solution we increase the cost by at most an ε-fraction of the cost of the previously guessed jobs, which yields an $(1+\varepsilon)$-approximation overall.

Formally, we use a binary search framework to estimate the optimal value B. Having this estimate, we adjust the functions g_i such that each of them is a step function with at most $(\log n)^{O(1)}$ steps, all being powers of $1+\varepsilon$ or 0.

Lemma 5. *At $1+\varepsilon$ loss we can assume that for each $i \in [k]$ and each t it holds that $g_i(t)$ is either 0 or a power of $1+\varepsilon$ in $\left[\frac{\varepsilon}{n} \cdot \frac{B}{W}, B\right)$.*

Our problem is in fact equivalent to assigning a due date d_j to each job (cf. [6]) such that the due dates are *feasible,* meaning that there is a preemptive schedule where every job finishes no later than its due date, and the objective being $\sum_j f_j(d_j)$. The following lemma characterizes when a set of due dates is feasible.

Lemma 6 ([6]). *Given a set of jobs and a set of due dates. The due dates are feasible if and only if for every interval $I = [r_j, d_{j'}]$ for any two jobs j, j', the jobs in $X(I) := \{j : r_j \in I\}$ that are assigned a deadline after I have a total size of at least $\mathrm{ex}(I) := \max(\sum_{j \in X(I)} p_j - |I|, 0)$. That is, $\sum_{\bar{j} \in X(I): d_{\bar{j}} > d_{j'}} p_{\bar{j}}$ is at least $\mathrm{ex}(I)$ for all intervals $I = [r_j, d_{j'}]$.*

Denote by D all points in time where at least one cost function g_i increases. It suffices to consider only those values as possible due dates.

Proposition 2. *There is an optimal due date assignment such that $d_j \in D$ for each job j.*

Denote by R the set of all release dates of the jobs. Recall that $|R| \leq (\log n)^{O(1)}$. We guess now the $|D|\cdot|R|/\varepsilon$ most expensive jobs of the optimal solution and their respective costs. Due to the rounding in Lemma 5 we have that $|D| \leq k \cdot \log_{1+\varepsilon}(W \cdot n/\varepsilon) = (\log n)^{O(1)}$ and thus there are only $O(n^{|D|\cdot|R|/\varepsilon}) = n^{(\log n)^{O(1)}/\varepsilon}$ many guesses.

Suppose we guess this information correctly. Let J_E denote the guessed jobs and for each job $j \in J_E$ denote by d_j the latest time where it attains the guessed cost, i.e., its *due date*. Denote by c_{thres} the minimum cost of a job in J_E, according to the guessed costs. The remaining problem consists in assigning a due date $d_j \in D$ to each job $J \setminus J_E$ such that none of these jobs costs more than c_{thres}, all due dates together are feasible, and the overall cost is minimized. We express this as a linear program. In that LP, we have a variable $x_{j,t}$ for each pair of a job $j \in J \setminus J_E$ and a due date $t \in D$ such that j does not cost more than c_{thres} when finishing at time t. We add the constraint $\sum_{t\in D} x_{j,t} = 1$ for each job j, modeling that the job has a due date, and one constraint for each interval $[r,t]$ with $r \in R$ and $t \in D$ to model the condition given by Lemma 6.

In polynomial time, we compute an extreme point solution x^* for the LP. It has at most $|D| \cdot |R| + |J \setminus J_E|$ many non-zeros. Each job j needs at least one non-zero variable $x^*_{j,t}$, due to the constraint $\sum_{t\in D} x_{j,t} = 1$. Thus, there are at most $|D| \cdot |R|$ fractionally assigned jobs, i.e., jobs j having a variable $x^*_{j,t}$ with $0 < x^*_{j,t} < 1$. We define an integral solution by rounding x^* as follows: For each job j we set d_j to be the maximum value t such that $x^*_{j,t} > 0$. We round up at most $|D| \cdot |R|$ jobs and after the rounding, each of them costs at most c_{thres}. Hence, those jobs cost at most an ε-fraction of the cost of guessed jobs (J_E).

Lemma 7. *Denote by $c(x^*)$ the cost of the solution x^*. We have that*

$$\sum_{j\in J\setminus J_E} f_j(d_j) \;\leq\; c(x^*) + \varepsilon \cdot \sum_{j\in J_E} f_j(d_j).$$

Since $c(x^*) + \sum_{J_E} f_j(d_j)$ is a lower bound on the optimum, we obtain a $(1+\varepsilon)$-approximation. As there are quasi-polynomially many guesses for the expensive jobs and the remainder can be done in polynomial time, we obtain a QPTAS.

Theorem 4. *There is a QPTAS for GSP, assuming that each cost function f_j can be expressed as $f_j = w_j \cdot g_{u(j)}$ for some job-dependent weight w_j and at most $k = (\log n)^{O(1)}$ global functions $g_1, ..., g_k$, and that the jobs have at most $(\log n)^{O(1)}$ distinct release dates.*

References

1. Afrati, F., Bampis, E., Chekuri, C., Karger, D., Kenyon, C., Khanna, S., Milis, I., Queyranne, M., Skutella, M., Stein, C., Sviridenko, M.: Approximation schemes for minimizing average weighted completion time with release dates. In: Proceedings of FOCS 1999, pp. 32–44 (1999)
2. Anagnostopoulos, A., Grandoni, F., Leonardi, S., Wiese, A.: A mazing 2+ε approximation for unsplittable flow on a path. In: Proceedings of SODA 2014, pp. 26–41 (2014)
3. Bansal, N., Chakrabarti, A., Epstein, A., Schieber, B.: A quasi-PTAS for unsplittable flow on line graphs. In: Proceedings of STOC 2006, pp. 721–729 (2006)

4. Bansal, N., Dhamdhere, K.: Minimizing weighted flow time. ACM T. Alg. 3(4), article 39 (2007)
5. Bansal, N., Friggstad, Z., Khandekar, R., Salavatipour, R.: A logarithmic approximation for unsplittable flow on line graphs. In: Proceedings of SODA 2009, pp. 702–709 (2009)
6. Bansal, N., Pruhs, K.: The geometry of scheduling. In: Proceedings of FOCS 2010, pp. 407–414 (2010), See also `http://www.win.tue.nl/~nikhil/pubs/wflow-journ3.pdf`
7. Bansal, N., Pruhs, K.: Weighted geometric set multi-cover via quasi-uniform sampling. In: Epstein, L., Ferragina, P. (eds.) ESA 2012. LNCS, vol. 7501, pp. 145–156. Springer, Heidelberg (2012)
8. Bansal, N., Verschae, J.: Personal communication
9. Bar-Noy, A., Bar-Yehuda, R., Freund, A., Naor, J., Schieber, B.: A unified approach to approximating resource allocation and scheduling. J. ACM 48(5), 1069–1090 (2001)
10. Bar-Yehuda, R., Rawitz, D.: On the equivalence between the primal-dual schema and the local ratio technique. SIAM J. Discrete Math. 19(3), 762–797 (2005)
11. Bonsma, P., Schulz, J., Wiese, A.: A constant factor approximation algorithm for unsplittable flow on paths. In: Proceedings of FOCS 2011, pp. 47–56 (2011)
12. Carr, R.D., Fleischer, L.K., Leung, V.J., Phillips, C.A.: Strengthening integrality gaps for capacitated network design and covering problems. In: Proceedings of SODA 2000, pp. 106–115 (2000)
13. Chakaravarthy, V.T., Kumar, A., Roy, S., Sabharwal, Y.: Resource allocation for covering time varying demands. In: Demetrescu, C., Halldórsson, M.M. (eds.) ESA 2011. LNCS, vol. 6942, pp. 543–554. Springer, Heidelberg (2011)
14. Chakrabarti, A., Chekuri, C., Gupta, A., Kumar, A.: Approximation algorithms for the unsplittable flow problem. In: Jansen, K., Leonardi, S., Vazirani, V.V. (eds.) APPROX 2002. LNCS, vol. 2462, pp. 51–66. Springer, Heidelberg (2002)
15. Chekuri, C., Khanna, S.: Approximation schemes for preemptive weighted flow time. In: Proceedings of STOC 2002, pp. 297–305 (2002)
16. Chekuri, C., Khanna, S., Zhu, A.: Algorithms for minimizing weighted flow time. In: Proceedings of STOC 2001, pp. 84–93 (2001)
17. Chekuri, C., Mydlarz, M., Shepherd, F.: Multicommodity demand flow in a tree and packing integer programs. ACM T. Alg. 3(3), article 27 (2007)
18. Cheung, M., Shmoys, D.B.: A primal-dual approximation algorithm for min-sum single-machine scheduling problems. In: Goldberg, L.A., Jansen, K., Ravi, R., Rolim, J.D.P. (eds.) RANDOM 2011 and APPROX 2011. LNCS, vol. 6845, pp. 135–146. Springer, Heidelberg (2011)
19. Kao, M.-Y., Reif, J.H., Tate, S.R.: Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem. Inform. Comput. 131(1), 63–79 (1996)
20. Lawler, E.L.: A “pseudopolynomial” algorithm for sequencing jobs to minimize total tardiness. Ann. Discrete Math. 1, 331–342 (1977)
21. Megow, N., Verschae, J.: Dual techniques for scheduling on a machine with varying speed. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) ICALP 2013, Part I. LNCS, vol. 7965, pp. 745–756. Springer, Heidelberg (2013)
22. Mestre, J., Verschae, J.: A 4-approximation for scheduling on a single machine with general cost function, `http://arxiv.org/abs/1403.0298`
23. Shmoys, D.B., Tardos, É.: An approximation algorithm for the generalized assignment problem. Math. Program. 62(1-3), 461–474 (1993)
24. Sviridenko, M., Wiese, A.: Approximating the configuration-LP for minimizing weighted sum of completion times on unrelated machines. In: Goemans, M., Correa, J. (eds.) IPCO 2013. LNCS, vol. 7801, pp. 387–398. Springer, Heidelberg (2013)

Why Some Heaps Support Constant-Amortized-Time Decrease-Key Operations, and Others Do Not[*]

John Iacono[**] and Özgür Özkan

New York University

Abstract. A lower bound is presented which shows that a class of heap algorithms in the pointer model with only heap pointers must spend $\Omega\left(\frac{\log\log n}{\log\log\log n}\right)$ amortized time on the Decrease-Key operation (given $O(\log n)$ amortized-time Extract-Min). Intuitively, this bound shows the key to having $O(1)$-time Decrease-Key is the ability to sort $O(\log n)$ items in $O(\log n)$ time; Fibonacci heaps [M. .L. Fredman and R. E. Tarjan. J. ACM 34(3):596-615 (1987)] do this through the use of bucket sort. Our lower bound also holds no matter how much data is augmented; this is in contrast to the lower bound of Fredman [J. ACM 46(4):473-501 (1999)] who showed a tradeoff between the number of augmented bits and the amortized cost of Decrease-Key. A new heap data structure, the *sort heap*, is presented. This heap is a simplification of the heap of Elmasry [SODA 2009: 471-476] and shares with it a $O(\log\log n)$ amortized-time Decrease-Key, but with a straightforward implementation such that our lower bound holds. Thus a natural model is presented for a pointer-based heap such that the amortized runtime of a self-adjusting structure and amortized lower asymptotic bounds for Decrease-Key differ by but a $O(\log\log\log n)$ factor.

1 Introduction

While Insert and Extract-Min are supported by all priority queues, there is one additional operation which is of use in some algorithms: Decrease-Key. The fast execution of Decrease-Key is vital to the runtime of several algorithms, most notably Dijkstra's algorithm [2]. The constant-amortized-time Decrease-Key operation is the defining feature of the *Fibonacci heap* [8].

In [7], a new heap called the *pairing heap* was introduced. The pairing heap is a self-adjusting heap, whose design and basic analysis closely follows that of splay trees [11]. They are much simpler in design than Fibonacci heaps, and they perform well in practice. It was conjectured at the time of their original presentation that pairing heaps had the same $O(\log n)$ amortized time Extract-Min, and $O(1)$ amortized time Insert and Decrease-Key as Fibonacci heaps, however,

[*] The full version is on the arXiv at CoRR abs/1302.6641 (2013).

[**] Research supported by NSF Grant CCF-1018370.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 637–649, 2014.

at their inception only a $O(\log n)$ amortized bound was proven for all three operations. Stasko and Vitter [13] provided some simulation results which showed that $O(1)$ Decrease-Key appeared likely. We have shown that Insert does in fact have $O(1)$ amortized time [9]. However in [6], Fredman refuted the conjectured constant-amortized-time Decrease-Key in pairing heaps by proving that pairing heaps have a lower bound of $\Omega(\log\log n)$ on the Decrease-Key operation. The result he proved was actually more general: he created a model of heaps which includes both pairing heaps and Fibonacci heaps, and produced a tradeoff between the number of bits of data augmented and a lower bound on the runtime of Decrease-Key. Pairing heaps have no augmented bits, and were shown to have a $\Omega(\log\log n)$ amortized lower bound on Decrease-Key while in his model a $O(1)$ Decrease-Key requires $\Omega(\log\log n)$ bits of augmented information per node, which is the number of bits of augmented information used by Fibonacci heaps and variants. More recently, Pettie has shown a upper bound of $O(2^{2\sqrt{\log\log n}})$ for Decrease-Key in pairing heaps [10]. It remains open where in $\Omega(\log\log n)\ldots O(2^{2\sqrt{\log\log n}})$ the cost of Decrease-Key in a pairing heap lies. Elmasry has shown that a simple variant of pairing heaps has $O(\log\log n)$ amortized time Decrease-Key operation [4,3]. However, this variant is not in Fredman's model and thus the $\Omega(\log\log n)$ lower bound does not apply.

So, it would seem from the preceding exposition that the situation has been essentially resolved: Fibonacci heaps are complex but optimal, while the elegant pairing heaps (and Elmasry's variant) are as good as a self-adjusting structure can get. (One defining feature of a *self-adjusting* structure is that they store no augmented data in every node.) In the case of dictionaries, we have the hope of instance-based optimality, as evidenced by the dynamic optimality conjecture [11], while in the case of heaps, self-adjusting structures can not even achieve optimal amortized asymptotic runtimes. We will now propose an alternate interpretation of the facts which leads to a much nicer conclusion.

Fredman's model is nuanced, and has limitations which cause us to introduce here a new model, which we call the *pure heap* model. We first informally describe our model, and then describe how it differs from Fredman's. A pure heap is a pointer-based forest of rooted trees, each node holding one key, that obeys the heap property (the key of the source of every pointer is smaller than the key of the destination); the nodes may be augmented, and all operations must be valid in the pointer model; heap pointers are only removed if one of the nodes is removed or if a Decrease-Key is performed on the node that a heap pointer points to. This model is both simple and captures the spirit of many heaps like the pairing heap, and is meant to be a clean definition analogous to that of the well-established binary search tree (BST) model.However, Fibonacci heaps are not pure model heaps in the standard textbook presentation [1] for the following reason: each node is augmented with a $\log\log n$-bit integer (in the range $1\ldots\log n$), and in the implementation of Extract-Min it is required to separate a number of nodes, call it k, into groups of nodes with like number. This is done classically using bucket sort in time $O(k+\log n)$. Bucket sort is not allowed in the pointer model; realizing this a pointer-model implementation was presented that makes use of non-heap pointers [8].

We can easily modify Fibonacci heaps and the aforementioned alternatives to only use heap pointers by simply using $O(\log\log n)$ time to determine which of the $O(\log n)$ buckets each of the k keys are in. We call such a variant of rank pairing heaps a *pointer rank-pairing* heap. However, this alteration increases the time of DECREASE-KEY in a Fibonacci heaps and their variants to $O(\log\log n)$.

Fredman's model differs from ours in several regards. First, Fredman's model requires that comparisons can not be performed unless the nodes being compared must be linked by a heap pointer after the comparison. Second, in the course of an EXTRACT-MIN, any two children of the former root can be randomly accessed and compared at unit cost. Third, the number of augmented bits per node is a parameter of the model. The first restriction, while it admits pairing heaps and Fibonacci heaps, excludes Elmasry's variant. This is because Elmasry's variant sorts the keys in nodes in order to determine how to link them; such sorting is directly against what is allowed in Fredman's model. Our sort heaps, presented in the full version, are not in Fredman's model for the same reason. Also, subjectively, we find the first restriction a bit odd, but it appears to be vital to the result. The second difference means that a fundamental cost in the pointer model is not counted: moving pointers to reach the desired nodes to compare or otherwise manipulate. So, compared to our model, Fredman's model is more restrictive because of the first condition, and more permissive with the second condition. We feel our model is more natural. A more detailed discussion of how our model differs from Fredman's can be found in the full version.

Thus, we conclude that the reason that Fibonacci heaps have fast DECREASE-KEY is not (only) because of the augmented bits as Fredman's result suggests, but rather because they depart from the pure heap model. We prove that any pure-heap-model heap has an $\Omega\left(\frac{\log\log n}{\log\log\log n}\right)$ amortized lower bound on DECREASE-KEY key, *no matter how many bits of data each node is augmented with.* In our view, Fibonacci heaps are a typical RAM-model structure that squeeze out a $\log\log n$ factor over the best structure in a natural pointer-based model by beating the sorting bound using bit tricks (of which bucket sort is a very primitive example).

Given our lower bound, we still have the issue that it does not apply to Elmasry's variant. We rectify this in the full version, by introducing the *sort heap.* This heap is simply Elmasry's heap with the non-standard DECREASE-KEY replaced with the standard one. Our sort heap has an $O(\log\log n)$ amortized DECREASE-KEY operation, and has a markedly different analysis than other self-adjusting heaps with fast DECREASE-KEY.

2 The Pure Heap Model

Here we define the *pure heap* model, and how priority queue operations on data structures in this model are executed in it.

The pure heap model requires that at the end of every operation, the data structure is an ordered forest of general heaps. Each node is associated with a key $x \in \mathcal{S}$. We will use x both to refer to a key and the node in the heap containing the key. Inside each node is stored the key value, pointers to the parent, leftmost child and right sibling of the node, along with other possible augmented information.

The *structure* of the heap is the shape of the forest, without regard to the contents of the nodes. The *location* of a node is its position in the forest of heaps relative to the right (e.g. a node could be described as being the fifth child from the right of a node which is the third child from the right of the fourth root from the right. Note that location is invariant under adding new siblings to the left).

An algorithm in the pure heap model implements the priority queue operations as follows: Insert operations are executed at unit cost by adding the new item as a new leftmost heap. Decrease-Key is executed at unit cost by disconnecting from its parent the node containing the key to be decreased (if it is not a root), decreasing the key, and then placing it as the leftmost root in the forest of heaps. An Extract-Min operation is performed by first executing a sequence of pointer-based suboperations which are fully described below. After executing the suboperations, the forest is required to be monoarboral (i.e. have only one heap). Thus, the root of this single tree has as its key the minimum key in $\mathcal{S}$. This node is then removed, the key value is returned, and its children become the new roots of the forest. The cost is the number of suboperations performed.

Note that some data structures are not presented exactly in the framework as described above but can be easily put into this mode by being lazy. For example, in a pairing heap, the normal presentations of Insert and Decrease-Key cause an immediate pairing with the single existing root. However, such pairings can easily be deferred until the next Extract-Min, thus putting the resulting structure in our pure-heap framework.

To execute an Extract-Min, the minimum must be determined. In an Extract-Min, the forest of heaps must be combined into a single heap using an operation called *pairing*, which takes two roots and attaches the root with larger key value as the leftmost child of the root with smaller key value. (Note that while the pairing operation brings to mind pairing heaps, it is the fundamental building block of many heaps. Even the *skew heap* [12], which seems at first glance to not use anything that looks like the pairing operation, can be shown in all instances to be able to be transformed into a pairing-based structure [5].)

In the execution of the Extract-Min operation, the use of some constant number ρ of pointers $p_1, p_2, \ldots p_\rho$ is allowed. They are all initially set to the leftmost root. The constant ρ is a parameter of the model.

The suboperations include: move one of the pointers to the parent, left or right sibling, or leftmost child; pair the nodes pointed to by two pointers; and copy pointers. A special suboperation, End(), signifies the execution of the Extract-Min is complete. A full list of suboperations and their precise definitions and preconditions can be found in the full version. The total number of suboperations, including the parameters, is defined to be η. Observe that

$\eta = \Theta(\rho^2)$, which is $\Theta(1)$ since ρ is a constant. A sequence of suboperations is a *valid* implementation of the EXTRACT-MIN operation if all the preconditions of each suboperation are met and the last suboperation is an END().

In the pure heap model, the only thing that differentiates between different algorithms is in the choice of the suboperations to execute EXTRACT-MIN operations. In these operations it is the role of the particular *heap algorithm* to specify which suboperations should be performed for each EXTRACT-MIN. We place no restrictions as to how an algorithm determines the suboperation sequence for each EXTRACT-MIN other than the suboperation sequence must be valid; the algorithm need not restrict its actions to information in the nodes visited in that operation. This definition of an algorithm encompasses and is more permissive than allowing the algorithm to make decisions to be made based on some data augmented at any node.

3 Lower Bound

Theorem 1. *In the pure heap model with a constant number of pointers, if* EXTRACT-MIN *and* INSERT *have an amortized cost of* $O(\log n)$*, then* DECREASE-KEY *has an amortized cost of* $\Omega\left(\frac{\log\log n}{\log\log\log n}\right)$.

The proof will follow by contradiction and will consume the rest of this section. Assume that there is a pure heap model algorithm $\mathcal{A}$ where EXTRACT-MIN and INSERT have an amortized cost of at most $c \log n$, for some constant c, and DECREASE-KEY has an amortized cost of at most $dc(n)$, for some $dc(n) = o\left(\frac{\log\log n}{\log\log\log n}\right)$. The existence of the algorithm $\mathcal{A}$, the constant c and the function $dc(n)$ will be assumed in the definitions and lemmas that follow. A sufficiently large n is also assumed.

Overview of Proof. The proof is at its core an adversary argument. But, our argument is not straightforward as it works on sets of sequences of operations rather than a single operation sequence. There is a hierarchy of things we manipulate in our argument:

Suboperaton. The suboperations of §2 are the very basic unit-cost primitives that can be used to implement EXTRACT-MIN, the only operation that does not have constant actual cost. It is at this level that definitions have been made to enforce pointer model limitations. *Operation.* We use *operation* to refer to a priority queue operation. *Sequence.* Operations are combined to form sequences of operations. *Set of operation sequences.* Our adversary does not just work with a single operation sequence but rather with sets of operation sequences. These sets are defined to have certain invariants. *Evolution.* We use the word *evolution* to refer to a function the adversary uses to take a set of operation sequences, and modify it. *Rounds.* Our evolutions are structured into *rounds.*

The proof will start with a set containing a single operation sequence, and then perform rounds of evolutions on this set; the exact choice of evolutions to perform will depend on how the algorithm executes the sequences of operations in the set. The evolutions in a round are structured in such a way that most rounds increase the size of the set of operations. After sufficiently many rounds, an upper bound on the maximum size of the set of operation sequences will be exceeded, thus giving a contradiction.

Ranks: Definitions and Useful Facts. As in many previous works on heaps and trees, the notion of the *rank* of a node in the heap is vital. The rank of a node is meant to be a rough proxy for the logarithm of the size of the subtree of the node. For ease of presentation, the rank of a node is defined in terms of the function $j(n) = 2dc(n) + 1$. The general idea is to have the rank of a node be the negation of the key value stored in the node. (Ranks will be non-negative, and we will only give nodes non-positive integer key values). A node's rank can increase as the result of a pairing, and a node's value can decrease as the result of a Decrease-Key. It is thus our goal to perform a Decrease-Key on a node which has had its rank increase to restore it to the negation of its rank. During the time between when a rank increase occurs in a node and the time the Decrease-Key is performed, we refer to the node as *marked.*

Call the *unmarked subtree* of a node to be the subtree of a node if all marked nodes were detached from their parents; the *unmarked structure* of the heap is the structure of the unmarked subtrees of the roots. The rank of a node at a given time will be defined as a function of the structure of its unmarked subtree.

The following assumes a particular heap structure and marking, as the rank of a node is always defined with respect to the structure of the heap after executing a sequence of heap operations. Let x be the node we wish to compute the rank of. Let k denote the number of unmarked children of x, and let $y_1, y_2, \ldots y_k$ denote these children numbered right-to-left (i.e., in the order which they became children of x). Let $\tau_i(x)$ be a subtree of x consisting of x connected to only the subtrees induced by $y_1, y_2, \ldots y_i$. We will define the function $r_i(x)$ as a function of $\tau_i(x)$. The rank of a node, $r(x)$ is $r_k(x)$. Each node y_i is labeled as *efficiently linked* to its parent x if and only if $r_{i-1}(x) - 3 \leq r(y_i) \leq r_{i-1}(x)$. The case of $r(y_i) > r_{i-1}(x)$ will never occur, as pairings will only happen among unmarked nodes, where the rank perfectly matches the negation of the key value. We will have the property that $r_i(x)$ is either $r_{i-1}(x)$ or $r_{i-1}(x)+1$; in the latter case, y_i is called *incremental.* Given a node y_i, let j be defined to be the index of the first incremental node in the sequence $\langle y_{i-1}, y_{i-2}, \ldots \rangle$; j is defined to be 0 if there is no such incremental node. The set $N(y_i)$ is defined to be $\{y_k | j < k \leq i\}$; that is, y_i and the maximal set of its non-incremental siblings to the right. Given these preliminaries, we can now give the full definition of the rank of a node:

$$r_i(x) = \begin{cases} 0 & \text{if } i = 0 \\ r_{i-1}(x) + 1 & \begin{array}{l}\text{(Efficient case) } y_i \text{ is efficiently linked and is the } j(n)\text{th} \\ \text{efficient element of } N(y_i) \text{ or (Default case) } |N(y_i)| = j(n)\end{array} \\ r_{i-1}(x) & \text{otherwise} \end{cases}$$

While the rank and mark are interrelated, there is no circularity in their definitions—whether a node is marked depends on its rank and key value and the rank of a node is a function of the ranks and marks of its children.

Observation 2. *Given two nodes x and y with different ranks, the unmarked structure of their induced subtrees must be different.*

This follows directly from the fact that the rank of a node is a function of its induced unmarked subtree. Set $f(n) = \frac{j(n)}{2 \log j(n)}$ and $g(n) = \frac{1}{2 \log j(n)}$. Using some technical lemmas in the full version gives:

Corollary 3. *Suppose a root with unmarked subtree of size m has $\leq f(n) \log m$ unmarked children. Then it has $\geq g(n) \log m$ efficiently linked children.*

We note that $f(n) = \frac{j(n)}{2 \log j(n)} = o(\log n)$ easily since $dc(n) = o(\log \log n)$.

Monotonic Operation Sequences and Sugmented Suboperations. Call the *designated minimum root* the next node to be removed in an Extract-Min. Define a *monotonic operation sequence* to be one where Decrease-Key operations are only performed on roots, children of the designated minimum root, or marked nodes. All of the sequences of operations we define will be monotonic. Observe:

Observation 4. *In a monotonic operation sequence, for any node x with descendent y (at the beginning of an operation) where all nodes on the path from x's child down to and including y are unmarked, y will remain in the same location in x's subtree (at the beginning of subsequent opeations) until x becomes the designated minimum root. In a monotonic operation sequence, the rank of a node never decreases, from the time it is inserted until the time it becomes the designated minimum root.*

We augment the Pair($\cdot$) operation, to return whether the rank was incremented as a result of the pairing. This augmentation does not give any more power to the pure heap model. We use this augmentation to create a finer notion of what constitutes a distinct sequence of suboperations. Observe:

Lemma 1. *Suppose s_i and s_j are two structurally distinct states of the data structure. Suppose a single valid sequence of suboperations implementing an* Extract-Min *is performed on both, and the outcomes of all augmented suboperations that have return values are identical in both structures. Then, the position of all nodes who have had their ranks changed is identical in both.*

Evolutions of Indistinguishable Sequences. Let $B = \langle b_1, b_2, \ldots \rangle$ be a sequence of priority queue operations. Let $A(b_i) = \langle a_1^i, a_2^i, \ldots \rangle$ be the sequence of augmented suboperations and their return values used by algorithm $\mathcal{A}$ to execute operation b_i if b_i is an Extract-Min; if it is not $A(b_i)$ is defined to be the empty sequence. $A(B)$ is the concatenation of $A(b_1), A(b_2), \ldots$. We call two sequences of priority queue operations B and B' *algorithmically indistinguishable* if $A(B) = A(B')$, else they are *algorithmically distinct*. Let $s_B(i)$ be the structure of the heap after running sequence $\langle b_1 \ldots b_i \rangle$; the *terminal structure* of B is $s_B(|B|)$ which we denote as s_B. Recall that by structure, we mean the raw shape of the heap without regard to the data in each node, but including which nodes are marked. Two sequences B and B' are *terminal-structure indistinguishable* if $s_B = s_{B'}$, else they are *terminal-structure distinct*. Given a set of mutually algorithmically indistinguishable and terminal-structurally distinct (AI-TSD) sequences of heap operations $\varXi$, the *distinctness* of the set, $\xi(\varXi)$ is defined to be $\log |\varXi|$. Note that having two sequences which are algorithmically indistinguishable does not imply anything about them being terminal-structure indistinguishable. For example, it may be possible to add a Decrease-Key to a sequence, changing the terminal structure, while the sequence of suboperations performed to execute the sequence remains unchanged. A critical observation needed at the end of the proof is that the number of terminal-structurally distinct sequences is function of n, the proof of this lemma is in the full version:

Lemma 2. *The maximum distinctness of any set $\varXi$ of terminal-structurally distinct sequences, all of which have terminal structures of size n, is $\xi(\varXi) = O(n)$.*

Evolving. We will now describe several functions on AI-TSD sets of heap operations; we call such functions *evolutions*. The general idea is to append individual heap operations or small sequences of heap operations to all sequences in the input set $\varXi$ and remove some of the resulting sequences so as to maintain the property that the sequences in the resultant set of sequences $\varXi'$ are AI-TSD. The evolutions will also have the property that if the time to execute all sequences in $\varXi$ is identical, then the runtime to execute all sequences in $\varXi'$ will also be identical. The difference in the runtime to execute sequences in $\varXi'$ versus those in $\varXi$ will be called the *runtime* of an evolution.

Insert Evolution. The *insert evolution* has the following form: $\varXi' =$ Evolve-Insert($\varXi$). In an insert evolution, a single Insert operation of a key with value 0 is appended to the end of all $\varXi$ to obtain $\varXi'$. Given $\varXi$ is AI-TDS, the set $\varXi'$ is AI-TDS and trivially $\xi(\varXi) = \xi(\varXi')$. The runtime of the evolution is 1 since the added Insert has runtime 1. The rank of the newly inserted node is 0, and is thus unmarked.

Decrease-Key Evolution. The *decrease-key evolution* has the following form: $\varXi' =$ Evolve-Decrease-Key($\varXi, p$), where p is a location which is either a root or a marked node in all terminal structures of sequences in $\varXi$. In a Decrease-Key *evolution*, a Decrease-Key($p, \Delta x$) operation is appended to the end of all sequences in $\varXi$ to obtain $\varXi'$. The value of Δx is chosen such that the new key value of what p points to is set to is the negation of its current rank; this means

Δx is always nonnegative because of the monotone property of ranks noted in Observation 4. Observe that if p points to a marked node, then it is unmarked after performing an EVOLVE-DECREASE-KEY. This requirement ensures that all structures that are distinct before this operation will remain distinct after the operation. Thus, the set Ξ' is AI-TDS and trivially $\xi(\Xi) = \xi(\Xi')$. The runtime of the evolution is 1 since the added DECREASE-KEY has runtime 1.

Designated Minimum Root Evolution. The *designated minimum root evolution* has the form $\Xi' =$ EVOLVE-DESIGNATED-MINIMUM-ROOT(Ξ, r), where r is the position of one root which exists in all terminal structures of Ξ. In a designated minimum root evolution, a DECREASE-KEY operation on r to a value of negative infinity is appended to all sequences in Ξ to give Ξ'. It will always be the case that the (next) evolution performed on Ξ' will be an EVOLVE-EXTRACT-MIN evolution; the root r, which is known as the *designated minimum root*, will be removed from all terminal structures of Ξ' in this subsequent EVOLVE-EXTRACT-MIN. There is no change in distinctness caused by this operation: $\xi(\Xi) = \xi(\Xi')$. The runtime of the evolution is 1 since the added DECREASE-KEY has runtime 1.

Extract-min Evolution. The *extract-min evolution* has the form $(\Xi', V, e) =$ EVOLVE-EXTRACT-MIN(Ξ). First, an EXTRACT-MIN operation is appended to the end of all sequences in Ξ to obtain an intermediate set of sequences which we call $\overline{\overline{\Xi}}$. There is no reason to assume that the suboperations executed by the algorithm in response to the EXTRACT-MIN in each of the elements of $\overline{\overline{\Xi}}$ are the same; thus the set $\overline{\overline{\Xi}}$ may no longer be algorithmically indistinguishable. We fix this by removing selected sequences from the set $\overline{\overline{\Xi}}$ so that the only ones that remain execute the appended EXTRACT-MIN by using identical sequences of suboperations. This is done by looking at the first suboperation executed in implementation of EXTRACT-MIN in each element of $\overline{\overline{\Xi}}$, seeing which suboperation is the most common, and removing all those sequences $\overline{\overline{\Xi}}$ that do not use the most common first suboperation. If the suboperation is one which has a return value, the return value which is most common is selected and the remaining sequences are removed. This process is repeated for the second suboperation, etc., until the most common operation is END() and thus the end of all remaining suboperation sequences has been simultaneously reached. Since there are only a constant η number of suboperations, and return values, if present, are boolean, at most a constant fraction of $\overline{\overline{\Xi}}$ is removed while pruning each suboperation. At the end of processing each suboperation by pruning the number of sequences, the new set is returned as Ξ'. The set Ξ' can be seen to be terminal-structure distinct, since pairing identically positioned roots in structurally different heaps, and having the same nodes win the pairings, can not make different structures the same. Observe that the nodes winning pairings in the execution of the EXTRACT-MIN might have their ranks increase, and thus become marked. By Lemma 1, the position of all such nodes is identical in all terminal structures of Ξ'. The set of the locations of these newly marked nodes is returned as V, the *violation set*.

Now that it has been ensured that all of the sets of operations execute the appended Extract-Min using the same suboperations, we define e to be this common number of suboperations used to implement the Extract-Min; this value is returned by the evolution. As each suboperation reduces the distinctness by at most a constant, $\xi(\Xi') \geq \xi(\Xi) - e\log(2\eta) = \xi(\Xi) - O(e)$. The runtime of the evolution is e since that is the cost of the added Extract-Min.

Big/Small Evolution. The *big/small evolution* has the form $(\Xi', (p, \mathit{bigsmall})) =$ Evolve-Big-Small(Ξ). The goal of the big/small evolution is to ensure that the terminal structures of all sets are able to be executed in the same way in subsequent evolutions. In a big/small evolution, each terminal structure of each of the operation sequences of Ξ is classified according to the following, using the previously-defined function $f(n)$:

- The exact number of roots (if at most $f(n)\log n$) or the fact that the number of roots is greater than $f(n)\log n$ (we call this case *many-roots*).
- If the exact number of roots is at most $f(n)\log n$: The position of the root with the largest subtree (the leftmost such root if there is a tie). Call it p. Observe that the size of p's subtree is at least $\frac{n}{f(n)\log n}$. The exact number of children of p if less than $f(n)\log\frac{n}{f(n)\log n}$ (we call this case *small*) or the fact that the number of roots is greater than $f(n)\log\frac{n}{f(n)\log n}$ (we call this case *root-with-many-children*).

There are at most $\lceil f(n)\log n\rceil^2 \cdot \lceil f(n)\log\frac{n}{f(n)\log n}\rceil$ possible classifications. We create set Ξ' by removing from Ξ sequences with all but the most common classification of their terminal structures. The return value is based on the resultant classification: *Many-roots:* Return $(p, \mathit{bigsmall})$ where $p =$ Null and $\mathit{bigsmall} =$ Big. Root-with-many-children: Return $(p, \mathit{bigsmall})$ where p is the location of the root with the largest subtree and $\mathit{bigsmall} =$ Big. *Small:* Return $(p, \mathit{bigsmall})$ where p is the location of the root with the largest subtree and $\mathit{bigsmall} =$ Small. We bound the loss of distinctness, which is the logarithm of the number of classifications. Since $f(n) = o(\log n)$, then $\log\left(\lceil f(n)\log n\rceil^2 \cdot \lceil f(n)\log\frac{n}{f(n)\log n}\rceil\right) = O(\log\log n)$, and thus $\xi(\Xi') = \xi(\Xi) - O(\log\log n)$. The evolution's runtime is 0.

Permutation Evolution. The *permutation evolution* has the form $\Xi' =$ Evolve-Permute(Ξ), where the leftmost root r has in all terminal structures of the sequences of Ξ a subtree size of at least $\frac{n}{f(n)\log n}$ and at most $f(n)\log\frac{n}{f(n)\log n}$ children; this will be achieved by being in the small case of the big/small evolution and performing a decrease-key evolution on the relevant node. It is required that all terminal structures of sequences in Ξ are entirely unmarked. Here distinctness increases, and is the only evolution to increase the number of sequences.

By corollary 3 all nodes in the terminal structures of $\varXi$ at location r have at least $g(n)\log\frac{n}{f(n)\log n}$ efficiently linked children; since there are at most $4j(n)$ efficiently linked children of each rank, there are at least $\frac{g(n)\log\frac{n}{f(n)\log n}}{4j(n)}$ efficiently linked children of different ranks in each terminal structure. Find such a set and call it the *permutable set (PS)*. Pick the position of the PS that is most common. Form the intermediate set of sequences $\hat{\varXi}$ by removing from $\varXi$ all sequences that do not have this commonly located PS. Letting $F = f(n)\log\frac{n}{f(n)\log n}$ and $G = \frac{g(n)}{4j(n)}\log\frac{n}{f(n)\log n}$, an upper bound on the logarithm of the number of different locations a PS of size G could be in is $\log\binom{F}{G} = \varTheta(G\log\frac{F}{G}) = \varTheta\left(\frac{\log n}{j(n)}\right)$. As $j(n) = \varTheta(dc(n))$, the reduction of distinctness is $\xi(\hat{\varXi}) - \xi(\varXi) = -O\left(\frac{\log n}{dc(n)}\right)$.

Using the definitions of $f(n)$ and $dc(n)$, the PS is of size $\varTheta(\frac{\log n}{dc(n)\log dc(n)})$. Let m be a constant such that the PS is of size at least $\frac{m\log n}{dc(n)\log dc(n)}$ for sufficiently large n. We then create $\varXi'$ by replacing each sequence in $\hat{\varXi}$ with $(\frac{m\log n}{dc(n)\log dc(n)})!$ new sequences created by appending onto the end of each existing sequence a sequence of all possible permutations of Decrease-Key operations on all elements of an arbitrary subset of size $\frac{m\log n}{dc(n)\log dc(n)}$ of the PS. As all of the sequences in $\hat{\varXi}$ have the same PSs ensures that all terminal structures in $\varXi'$ are terminal-structure distinct. (Because of Lemma 2). Thus, in this step distinctness increases by $\xi(\varXi') - \xi(\hat{\varXi}) = \log(\frac{m\log n}{dc(n)\log dc(n)})! = \varTheta(\frac{\log n\log\log n}{dc(n)\log dc(n)})$, which dominates the total change of disctintness. The evolution costs $\leq \frac{m\log n}{dc(n)\log dc(n)}$, the number of unit-cost Decrease-Key operations appended to the sequences.

Rounds. A sequence of evolutions $\varPsi = \langle\psi_0, \psi_1, \ldots\rangle$ defines a sequence of AI-TSD sets $\langle\varXi_0, \varXi_1, \ldots\rangle$. The initial set $\varXi_0$ consists of a single sequence of operations: the operation Insert(0), executed n times. Each subsequent AI-TSD set $\varXi_i$ is derived from $\varXi_{i-1}$ by performing the single evolution ψ_{i-1}; thus in general $\varXi_i$ is composed of some of the sequences of $\varXi_{i-1}$ with some operations appended.

These evolutions are split into *rounds*; $\circ_i$ is the index of the first AI-TSD set of the ith round. Thus round i begins with AI-TSD set $\varXi_{\circ_i}$ and ends with $\varXi_{\circ_{i+1}-1}$ through the use of evolutions $\langle\psi_{\circ_i}\ldots\psi_{\circ_{i+1}-1}\rangle$ These rounds are constructed to maintain several invariants: All terminal structures of all sequences in the AI-TSD set at the beginning and end of each round have size n. This holds as in each round, exactly one Insert evolution and exactly one Extract-Min evolution is performed. All nodes in all terminal structures in the AI-TSD sets at the beginning and end of each round are unmarked. There are two types of rounds, *big rounds* and *small rounds.* At the beginning of both types of round a big/small evolution is performed which determines the round type.

The Big Round. As the round begins, the terminal structures of the AI-TSD set are entirely unmarked, and there are either at least $f(n)\log n$ roots, or one root with at least $f(n)\log\frac{n}{f(n)\log n}$ children. The round proceeds as follows:

(1) Perform a designated minimum root evolution on the root with largest subtree; this is the node r from the return value of the big/small evolution; as a result of the big/small evolution it is guaranteed to be in the same location in all of the terminal structures of the sequences of Ξ. (2) Perform an EXTRACT-MIN evolution. (3) For each item in the violation sequence returned by the EXTRACT-MIN evolution, perform a DECREASE-KEY evolution. This makes the terminal structures of all heaps in Ξ unmarked. (4) Perform an INSERT evolution. Assuming we are in round i, let e_i be the cost of the EXTRACT-MIN evolution, and let v_i be the size of the violation sequence. The cost of the round (the sum of the costs of the evolutions) is $e_i + v_i + 2$, which is at least $f(n) \log \frac{n}{f(n)\log n}$, and based on the evolutions performed the distinctness can be bounded as follows: $\xi_i - \xi_{i+1} = O(e_i) + O(\log\log n)$.

The Small Round. There is one root, call it x, at the same location in all terminal structures, with size at least $\frac{n}{f(n)\log n}$ and some identical number of children in all terminal structures which is at most $f(n) \log \frac{n}{f(n)\log n}$. The location of x was returned by the big/small evolution. The round proceeds as follows: (1) Perform a DECREASE-KEY evolution on x to make it negative infinity. (2) Perform an EVOLVE-PERMUTE evolution. (3) Perform an EXTRACT-MIN evolution. (4) For each item in the violation sequence returned by the EXTRACT-MIN evolution, perform a DECREASE-KEY evolution. (5) Perform an INSERT evolution.

Let e_i be the actual cost of the EXTRACT-MIN, let v_i be the size of the violation sequence. The cost of the round is $e_i + v_i + 2 + \frac{m \log n}{dc(n) \log dc(n)}$, and based on the evolutions performed the distinctness can be bounded as follows:

$$\xi_i - \xi_{i+1} = \overbrace{O(e_i)}^{\text{EXTRACT-MIN}} + \overbrace{O(\log\log n)}^{\text{EVOLVE-BIG-SMALL}} - \overbrace{\Omega(\tfrac{\log n \log\log n}{dc(n) \log dc(n)})}^{\text{EVOLVE-PERMUTE}}.$$

Lemma 3 (See the Full Version for Proof). *(A) The time to execute any sequence in $\Xi_{\circ_k}$ is $O(k \log n)$. (B) Over half of the rounds must be small rounds.*

To prove Theorem 1, the distinctness gain of a round has been bounded as follows:

$$\xi_{\circ_{i+1}} - \xi_{\circ_i} = \begin{cases} -O(e_i) - O(\log\log n) & \text{If the } i\text{th round is a big round} \\ -O(e_i) - O(\log\log n) + \Theta(\frac{\log n \log\log n}{dc(n)\log dc(n)}) & \text{If the } i\text{th round is a small round (§3)} \end{cases}$$

Now we know that $\sum_{i=1}^{k} e_i$ is less than the actual cost to execute a sequence in $\Xi_{\circ_k}$, which is $O(k \log n)$ by Lemma 3(A). Substituting $\sum_{i=1}^{k} e_i = O(k \log n)$ into the above and using the fact from Lemma 3(B) that at least half of the rounds are small rounds gives: $\xi_{\circ_k} - \xi_{\circ_0} = \Theta(k \log n \frac{\log\log n}{dc(n)\log dc(n)}) - O(k \log\log n) - O(k \log n)$ Since $dc(n) = o(\frac{\log\log n}{\log\log\log n})$, $\frac{\log\log n}{dc(n)\log dc(n)} = \omega(1)$, and thus the negative terms in the previous equation can be absorbed, giving: $\xi_{\circ_k} - \xi_{\circ_0} = \Theta(\frac{k \log n \log\log n}{dc(n)\log dc(n)})$.

But after sufficiently many rounds (i.e. sufficiently large k) this contradicts Lemma 2 that for all i, $\xi_i = O(n)$. Thus for sufficiently large k and n a contradiction has been obtained, proving Theorem 1.

References

1. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. MIT Press (2009)
2. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271 (1959)
3. Elmasry, A.: Pairing heaps with $O(\log \log n)$ decrease cost. In: SODA, pp. 471–476 (2009)
4. Elmasry, A.: Pairing Heaps with Costless Meld. In: de Berg, M., Meyer, U. (eds.) ESA 2010, Part II. LNCS, vol. 6347, pp. 183–193. Springer, Heidelberg (2010)
5. Fredman, M.L.: A Priority Queue Transform. In: Vitter, J.S., Zaroliagis, C.D. (eds.) WAE 1999. LNCS, vol. 1668, pp. 244–258. Springer, Heidelberg (1999)
6. Fredman, M.L.: On the Efficiency of Pairing Heaps and Related Data Structures. J. ACM 46(4), 473–501 (1999)
7. Fredman, M.L., Sedgewick, R., Sleator, D.D., Tarjan, R.E.: The Pairing Heap: A New Form of Self-Adjusting Heap. Algorithmica 1(1), 111–129 (1986)
8. Fredman, M.L., Tarjan, R.E.: Fibonacci heaps and their uses in improved network optimization algorithms. J. ACM 34(3), 596–615 (1987)
9. Iacono, J.: Improved Upper Bounds for Pairing Heaps. In: Halldórsson, M.M. (ed.) SWAT 2000. LNCS, vol. 1851, pp. 32–45. Springer, Heidelberg (2000)
10. Pettie, S.: Towards a Final Analysis of Pairing Heaps. In: FOCS, pp. 174–183 (2005)
11. Sleator, D.D., Tarjan, R.E.: Self-Adjusting Binary Search Trees. J. ACM 32(3), 652–686 (1985)
12. Sleator, D.D., Tarjan, R.E.: Self-Adjusting Heaps. SIAM J. Comput. 15(1), 52–69 (1986)
13. Stasko, J.T., Vitter, J.S.: Pairing Heaps: Experiments and Analysis. Commun. ACM 30(3), 234–249 (1987)

Partial Garbling Schemes and Their Applications

Yuval Ishai[1,⋆] and Hoeteck Wee[2,⋆⋆]

[1] Technion, Haifa, Israel
yuvali@cs.technion.ac.il
[2] ENS, Paris, France
wee@di.ens.fr

Abstract. Garbling schemes (aka randomized encodings of functions) represent a function F by a "simpler" randomized function $\hat{F}$ such that $\hat{F}(x)$ reveals $F(x)$ and no additional information about x. Garbling schemes have found applications in many areas of cryptography. Motivated by the goal of improving the efficiency of garbling schemes, we make the following contributions:

- We suggest a general new notion of *partial garbling* which unifies several previous notions from the literature, including standard garbling schemes, secret sharing schemes, and "conditional disclosure of secrets". This notion considers garbling schemes in which part of the input is public, in the sense that it can be leaked by $\hat{F}$.
- We present constructions of partial garbling schemes for (boolean and arithmetic) formulas and branching programs which take advantage of the public input to gain better efficiency.
- We demonstrate the usefulness of the new notion by presenting applications to efficient attribute-based encryption, delegation, and secure computation. In each of these applications, we obtain either new schemes for larger classes of functions or efficiency improvements from quadratic to linear. In particular, we obtain the first ABE scheme in bilinear groups for arithmetic formulas, as well as more efficient delegation schemes for boolean and arithmetic branching programs.

1 Introduction

There are many situations in cryptography where one is interested in computing some function F of a sensitive input x but the computational model is restricted so that only "simple" functions F can be directly computed. For instance, the entries of x may be encrypted so that only *affine* functions can be computed, or they may be distributed between multiple non-interacting parties so that only *local* functions can be computed.

[⋆] Research received funding from the European Union's Tenth Framework Programme (FP10/2010-2016) under grant agreement no. 259426 ERC-CaC. Also supported by ISF grant 1361/10 and BSF grant 2012378.

[⋆⋆] CNRS (UMR 8548) and INRIA. Supported in part by the French ANR-12-INSE-0014 SIMPATIC Project and NSF Awards CNS-1237429 and CNS-1319021. Part of this work was done while visiting the Technion, supported by ERC-CaC.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 650–662, 2014.

A common approach for handling more complex functions F in such situations is to relax the usual notion of computation. This is done by: (1) settling for computing some function $\hat{F}$ whose output *encodes* the output of F (and reveals nothing else about the input), and (2) allowing the latter encoding to be *randomized*. That is, the randomized function $\hat{F}(x)$ should satisfy the simplicity constraint (e.g., being affine) for every fixed choice of the randomness,[1] and moreover its output distribution on an input x should reveal $F(x)$ and no additional information about x. The function $\hat{F}$ is often referred to as a *randomized encoding* or a *garbling scheme* for F.

To give a simple example, let $F(a,b) = ab$ where a and b are elements of a finite field. Then an affine garbling scheme for F can be defined by $\hat{F}(a,b) = ((a - r_a), (b - r_b), ar_b + br_a - r_a r_b)$, where r_a and r_b are random and independent field elements. Note that $\hat{F}$ is an affine function of the input (a,b) for every fixed choice of r_a, r_b. Moreover, $F(a,b)$ can be recovered from the output (c,d,e) of $\hat{F}(a,b)$ by computing $cd + e$. Finally, the output (c,d,e) of $\hat{F}$ is distributed uniformly subject to the constraint that $cd + e = ab$ and hence it reveals no additional information about (a,b) other than ab.

Garbling schemes have found applications in many areas of cryptography and elsewhere (see [38, 17, 27, 3, 2, 7, 33] and references therein). Starting with Yao's celebrated garbled circuit construction, different constructions of garbling schemes have been proposed for circuits and other representation models. However, these constructions still have theoretical and practical limitations. In particular, they do not efficiently generalize to arithmetic computations (despite progress in [5]) and even in the boolean case their asymptotic and concrete efficiency leave much to be desired. Furthermore, some of the best garbling schemes do not satisfy all of the structural properties that are needed by applications.

1.1 New Notion: Partial Garbling

This work is motivated by the observations that (1) in many applications of garbling schemes, most of the input is already known to the designated receiver of the encoded output, and (2) known constructions do not take advantage of this fact for improving efficiency.

We suggest a general new notion of *partial garbling* which relaxes standard garbling by allowing part of the input to be public. This notion can be viewed as unifying several previous notions from the literature, including standard garbling schemes, secret sharing schemes [35, 30], and conditional disclosure of secrets [21].

Consider for instance the case of secret sharing. Here, we want to disclose a secret if and only if the attributes of the parties satisfy a given predicate referred to as the access structure; however, it is okay to leak information about the individual attributes, which we think of as being "public". Note that "public" does not mean "known to everyone" or "must be revealed by the garbling" but rather "okay to leak by the garbling" and known to the reconstruction algorithm. For this reason, partial garbling cannot be viewed as a special case of standard garbling.

[1] Some applications require that $\hat{F}$ be "simple" even as a function of both x and the randomness; however, in this work we will be mainly interested in complexity in terms of x.

As another example, consider a client who has her data x distributed between several servers. Suppose that the servers wish to disclose a secret s to the client only if $P(x) = 1$ for some publicly known predicate P. That is, they would like to reveal to the client the function $F(x,s) = P(x) \cdot s$. Note that from the client's point of view, x is public input for F whereas s is a secret input. Now, suppose we are given a *local* partial garbling scheme $\hat{F}(x,s)$ for F, namely one where each output depends only on the view of a single server. The partial garbling $\hat{F}$ can be used by the servers to conditionally disclose s to the client by each sending her a single message. This "conditional disclosure" primitive serves as a useful building block in both two-party and multi-party cryptographic protocols, where it can often serve as a light-weight, non-interactive substitute for zero-knowledge proofs [21, 1, 37, 16, 11, 12].

Typical applications of garbling schemes, including those illustrated above, require that the garbling be "affine" and/or "local" with respect to the public inputs. This additional requirement rules out the trivial solution of garbling a restriction of F to the private inputs.

Garbling Arithmetic Branching Programs (ABP). We present unconditional constructions of partial garbling schemes for boolean and arithmetic formulas and branching programs which take advantage of the public input to gain better efficiency. Our constructions satisfy all of the useful structural properties of garbling schemes required by natural applications.

In cases where the private input is small, as is the case for essentially all of our motivating applications, we improve the size of the garbling from quadratic to linear in the size of the formula or branching program; we also obtain a corresponding efficiency improvement in the applications. Boolean and arithmetic formulas and branching programs capture many functions of interest, including arithmetic computations like sparse polynomials, mean, and variance, as well as combinatorial computations like string-matching, finite automata and decision trees. Indeed, these classes have been studied in several recent works in a variety of different cryptographic settings [14, 9, 29, 24].

Our partial garbling schemes are "affine" and "local" with respect to the public inputs; indeed, all of our applications exploit this property. As mentioned earlier, this means that we cannot simply garble the original function hardwired with the private inputs. Instead, we start with the randomized encoding scheme for ABPs in [28]. Roughly speaking, this prior construction works by multiplying the adjacency matrix for the branching program by random upper triangular matrices on both the left and the right. Our construction uses a subgroup of upper triangular matrices with fewer non-zero entries (corresponding to the private inputs). This means that the size of the garbling is roughly the number of private inputs times the size of the branching program. Therefore, when the number of private inputs is constant (or "local"), the size of the garbling improves from quadratic to linear. In particular, we achieve linear-size garbling for functions of the form $F(x,s) = P(x) \cdot s$ and $F(x,(s_1,s_2)) = s_1P(x) + s_2$ where P is an ABP acting on a public input x; these are precisely the functions we consider for several of our applications.

Comparison with Standard Garbling Techniques. We note that standard garbling is a special case of partial garbling where all input is private. In the other direction, there is a general reduction from partially garbling $F(x,z)$ where x is public and z is private to garbling $F'(x,z) := (x, F(x,z))$. However, the cost of the reduction can be significant. Take the example where F is identically 0. Then, a partial garbling scheme can output nothing whereas a standard garbling of F' must reveal x.

Unlike Yao's garbling technique and its variants, our constructions cannot handle general circuits and are restricted to weaker computational models. However, they do offer a number of significant advantages: (i) they do not rely on computational assumptions and can be used in the context of information-theoretic cryptography; (ii) they work in an arithmetic model of computation, where the number of field operations is independent of the field size; (iii) they satisfy the "linear reconstruction" property required by several applications below; (iv) they have better concrete efficiency for natural functions F that admit compact representations by formulas and branching programs.

1.2 Applications

We demonstrate the usefulness of our new notion and constructions by presenting applications to efficient attribute-based encryption (ABE) [34, 25], delegation [22, 20], and secure computation [26, 21]. More broadly, partial garbling can be applicable in many cryptographic settings in which there are computations that mix public inputs with private inputs. In each of the applications we consider, we obtain either new schemes for larger classes of functions or efficiency improvements from quadratic to linear:

- For ABE, we extend prior pairing-based schemes for boolean formulas and branching programs [25, 23] to arithmetic branching programs.
- For delegation, conditional disclosure of secrets and generalized oblivious transfer, we obtain efficiency improvements for arithmetic branching programs from quadratic to linear; prior to this work, constructions with linear complexity were only known for boolean formulas [20, 21, 36].

We proceed with an overview of the applications to delegation and ABE.

Delegation and Verifiable Computation. In verifiable computation (VC), a computationally weak client with input x wishes to delegate a complex computation f to an untrusted server, with the assurance that the server cannot convince the client to accept an incorrect computation [22, 20, 4, 9]. We focus on the online/offline setting, where the protocol proceeds in two phases. In the offline phase, the client sends to the server a possibly long message that may be expensive to compute. Later on, in the online phase (when the input x arrives), the client sends a short message to the server, and receives the result of the computation together with a certificate for correctness. We are interested in protocols where the client's communication and computational complexity in the online phase depend only on the input and output lengths and is independent of the complexity of f.

Our VC schemes build upon the garble+MAC paradigm in [4], which derives a VC protocol by garbling the function obtained by composing f with a one-time MAC.

Our key observation is that it suffices to use a partial garbling scheme where the public input is x and the private input is the MAC key. Using our partial garbling schemes, we then derive more efficient online/offline VC protocols for arithmetic branching programs (ABPs), reducing the complexity of previous protocols from quadratic to linear in the size of the program. We note that ABPs simultaneously capture several classes of functions considered in the literature on delegation, including boolean formulas in [20, 32] and sparse arithmetic polynomials in [9].

Theorem 1 (informal). *Assuming the existence of a PRG, there is an online/offline VC protocol for arithmetic branching programs with the following efficiency features. The complexity of the server and the client's offline phase is $s \cdot \mathrm{poly}(\lambda)$ and that of the client's online phase is $n \cdot \mathrm{poly}(\lambda)$, where n is the input length, s is the size of the ABP, and λ is the security parameter.*

In [4], the complexity of the server and the client's offline phase is $s^2 \cdot \mathrm{poly}(\lambda)$. We also obtain a smaller improvement for boolean formulas.

Attribute-Based Encryption. Attribute-based encryption (ABE) [34, 25] is a new paradigm for public-key encryption that enables fine-grained access control for encrypted data. In ABE, ciphertexts are associated with descriptive values x in addition to a plaintext, secret keys are associated with predicates P, and a secret key decrypts the ciphertext if and only if $P(x) = 1$. Here, P may express an arbitrarily complex access policy, which is in stark contrast to traditional public-key encryption, where access is all or nothing. The security requirement for ABE enforces resilience to collusion attacks, namely any group of users holding secret keys for different functions learns nothing about the plaintext if none of them is individually authorized to decrypt the ciphertext.

We present the first ABE that directly handles a large class of predicates over arithmetic domains as described by arithmetic branching programs. This is particularly useful in settings where identities or attributes come from a universe of exponential size, since we can avoid the overhead from using bit encodings, as with the case for the Boneh-Boyen identity-based encryption [13].

Theorem 2 (informal). *Suppose the decisional bilinear Diffie-Hellman assumption holds. Then, there exists a (selectively secure) ABE scheme for the class of arithmetic branching programs.*

Note that there are two natural ways to associate an ABP with a predicate, namely whether its output is zero (Z-ABP), or non-zero (N-ABP). We obtain ABE schemes for both via a single construction for "arithmetic span programs," which simultaneously generalizes boolean branching programs in [25] as well as (public-index) inner product and non-zero inner product predicates [31, 6]. Prior to this work, we do not know any ABE schemes in bilinear groups supporting the class of ABPs. We could of course appeal to lattice-based ABE for general circuits [23, 18], though simulating an ABP using a boolean circuit incurs a substantial overhead in concrete efficiency.

At a high level, our construction follows the approach of Goyal et al. [25] for building ABE for monotone Boolean formula from linear secret-sharing schemes. The difficulty with extending this approach to arithmetic branching programs is that there is no natural analogue of LSSS for arithmetic functionalities. Instead, we observe that it suffices to use partial garbling with linear reconstruction, which we do obtain for arithmetic branching programs. Intuitively, the descriptive value x on the ABE ciphertext corresponds to the public input in partial garbling, and the plaintext/master secret key corresponds to the private input.

The running time of the encryption algorithm depends only on the input length to the ABP and not the size of the ABP. As such, exploiting the connection between ABE and delegation in [32] and using the fact that we handle both Z-ABP and N-ABP, we obtain a publicly verifiable delegation scheme for ABPs. This scheme requires a stronger assumption than the offline/online VC in Theorem 1, but achieves a stronger soundness requirement with reusability.

Finally, our construction yields an unconditionally secure witness encryption scheme [19] for algebraic languages corresponding to vectors of group elements $g^{\mathbf{w}}$ such that $P(\mathbf{w}) = 0$ for a fixed ABP P. For instance, this captures ElGamal public key and ciphertext pair (pk, C) such that C is an encryption of 0 or 1; see [12, 8, 10] for additional examples of such languages. The construction follows essentially from conditional disclosure of secrets schemes for the same predicate, along with the fact that reconstruction is linear.

Related Work. In an independent work, Boneh et al. [15] constructed ABE for arithmetic circuits under the LWE assumption; they only handle the "is zero" predicate (i.e., decryption is possible exactly when the output of the circuit is zero), whereas our construction also handles the "is non-zero" predicate. Handling a class that is closed under complement is useful for applications such as publicly verifiable delegation [32], as noted in the preceding paragraph.

2 Preliminaries

Notation. We denote by $s \leftarrow_{\mathrm{R}} S$ the fact that s is picked uniformly at random from a finite set S and by $x, y, z \leftarrow_{\mathrm{R}} S$ that all x, y, z are picked independently and uniformly at random from S.

Arithmetic Branching Programs. A *branching program* is defined by a directed acyclic graph (V, E), two special vertices $v_0, v_1 \in V$ and a labeling function ϕ. An *arithmetic branching program* (ABP) over a finite field $\mathbb{F}_q$ computes a function $f : \mathbb{F}_q^n \to \mathbb{F}_q$. Here, ϕ assigns to each edge in E an affine function in some input variable or a constant, and $f(x)$ is the sum over all v_0-v_1 paths of the product of all the values along the path. We refer to $|V|+|E|$ as the *size* of the ABP. Ishai and Kushilevitz [26, 28] showed how to relate an ABP computation to that of computing the determinant of a matrix (see Figure 1 for an example).

Lemma 1 ([28, Lemma 1]). *Given an ABP $\Gamma = (V, E, v_0, v_1, \phi)$ computing $f : \mathbb{F}_q^n \to \mathbb{F}_q$, we can efficiently (and deterministically) compute a function $L(x)$ mapping an input $x \in \mathbb{F}_q^n$ to a $(|V|-1) \times (|V|-1)$ matrix over $\mathbb{F}_q$, such that:*

- $\det(L(x)) = f(x)$;
- *each entry of $L(x)$ is a degree one polynomial in a single variable x_i;*
- *$L(x)$ contains only -1's in the second diagonal (the diagonal below the main diagonal) and 0's below the second diagonal.*

Specifically, L is obtained by removing the column corresponding to v_0 and the row corresponding to v_1 in the matrix $\mathbf{A} - \mathbf{I}$, where $\mathbf{A}$ is the adjacency matrix for Γ.

We note that there is a linear-time algorithm that converts any boolean formula, boolean branching program or arithmetic formula to an arithmetic branching program with a constant blow-up in the representation size. Thus, ABPs can be viewed as a stronger computational model than all of the above.

3 Partial Garbling Schemes (PGS)

We consider garbling schemes (aka randomized encodings of functions) [38, 17, 27, 3, 7] in which part of the input is public; we refer to this as *partial garbling*. Take a function F where the input (x, z) comprises a public value x and a private value z. In a standard garbling $\hat{F}$ of F, the function $\hat{F}$ is randomized and the output distribution $\hat{F}(x, z)$ "encodes" $F(x, z)$ and leaks no additional information about the input (x, z). In a partial garbling $\hat{F}$ of F, the value x is public, and the privacy requirement only applies to z. Again, we require that $\hat{F}(x, z)$ "encodes" $F(x, z)$, and that it leaks no additional information about the private input z beyond what is revealed by x and $F(x, z)$. More formally, we require that there be two efficiently computable maps Sim and Rec where Sim is randomized and Rec is deterministic such that

- the distributions $\mathsf{Sim}(x, F(x, z))$ and $\hat{F}(x, z)$ are identical;
- $\mathsf{Rec}(x, \hat{F}(x, z; r)) = F(x, z)$ for all inputs (x, z) and randomness r.

We want constructions where $\hat{F}$ is a "simple" function of (x, z). We are also particularly interested in constructions where and the reconstruction algorithm $\mathsf{Rec}(x, \cdot)$ is a "simple" function of $\hat{F}(x, z)$, e.g. a function of total degree 1 whose coefficients may depend arbitrarily on x.

Definition 1 ((affine) Partial Garbling Scheme). *Let $F : \mathbb{F}_q^n \times \mathbb{F}_q^{n'} \to \mathbb{F}_q$ be a function. We say that a randomized function $\hat{F} : \mathbb{F}_q^n \times \mathbb{F}_q^{n'} \to \mathbb{F}_q^m$ is a* partial garbling scheme (PGS) *of F if it satisfies the following properties:*

- *(correctness) There exists a deterministic reconstruction algorithm Rec such that for all $(x, z) \in \mathbb{F}_q^n \times \mathbb{F}_q^{n'}$, $\Pr[\mathsf{Rec}(x, \hat{F}(x, z)) = F(x, z)] = 1$.*
- *(privacy) There exists a randomized algorithm Sim, called a simulator, such that for all $(x, z) \in \mathbb{F}_q^n \times \mathbb{F}_q^{n'}$, $\mathsf{Sim}(x, F(x, z))$ and $\hat{F}(x, z)$ are identically distributed.*

In addition, we say that the garbling scheme is affine *if for all indices $j \in [m]$, $\hat{F}(x, z)_j$ is an affine function of the form $a_j x_i + b_j$ or $a_j z_i + b_j$ where the coefficients $a_j, b_j \in \mathbb{F}_q$ depend only on the randomness of $\hat{F}$.*

We will also say that the garbling scheme is *x-affine* if each $\hat{F}(x,z)_j$ is an affine function of the form $a_j x_i + b_j$ where the coefficients $a_j, b_j \in \mathbb{F}_q$ depend only on z and the randomness of $\hat{F}$. We may define *z-affine* analogously. Note that an affine garbling scheme is both x-affine and z-affine.

We note that the above definition extends naturally to functions F with longer outputs. When considering an infinite family of F, we also require that there is an efficient deterministic garbling algorithm for computing the description of $\hat{F}$, Rec and Sim from that of F.

Examples. As a warm-up, we describe several partial garbling schemes for functions with $n' = 1$ and $n' = 2$, some of which were implicit in prior works. All of these schemes are captured by our more general construction in Section 4.

Example 1 (non-zero product). Consider the function

$$F((x_1,x_2,x_3),z) = x_1 x_2 x_3 z.$$

This corresponds to disclosing a secret z subject to the condition x_1, x_2, x_3 are all non-zero. Consider the affine garbling scheme:

$$\hat{F}((x_1,x_2,x_3),z;r_1,r_2,r_3) = (r_1 x_1, r_1 - r_2 x_2, r_2 - r_3 x_3, r_3 - z))$$

Reconstruction has degree 1 and is given by a dot product with the vector

$$(-1, x_1, x_1 x_2, x_1 x_2 x_3).$$

Example 2 (sum is zero). Consider the function $F : \mathbb{F}_q^n \times \mathbb{F}_q^2 \to \mathbb{F}_q$ given by

$$F((x_1,\ldots,x_n),(z,z')) = z' \cdot (x_1 + \cdots + x_n) + z.$$

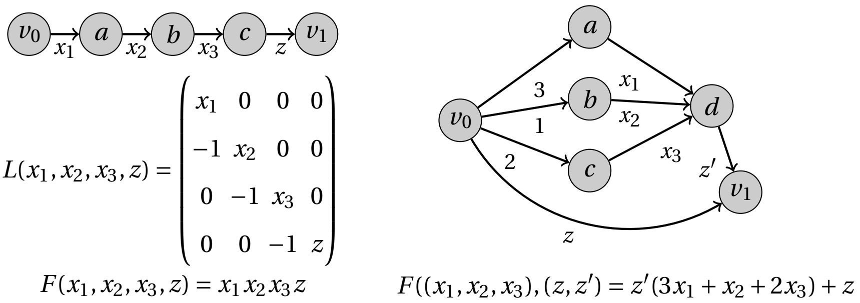
This corresponds to disclosing a secret z subject to the condition $x_1 + \ldots + x_n = 0$; for $n = 2$, this captures “disclose z if x_1, x_2 are equal”. Consider the x-affine encoding:

$$\hat{F}((x_1,\ldots,x_n),(z,z');r_1,\ldots,r_n) = (z' x_1 - r_1,\ldots,z' x_n - r_n, r_1 + \cdots + r_n + z)$$

Reconstruction has degree 1 and is given by summing the values in the encoding. This is essentially the scheme given in [21, Lemma 2].

4 Partially Garbling Arithmetic Branching Programs

In this section, we present partial garbling schemes for arithmetic branching programs, with a restriction on where the private inputs are used in the computation. Specifically, we consider functions $F : \mathbb{F}_q^n \times \mathbb{F}_q^{n'} \to \mathbb{F}_q$ that are computed by an ABP such that the variables in the private input z appear only on the edges leading into the last vertex v_1 (or more generally, into the last t vertices in V). For instance, this class captures read-once branching programs on $n + n'$ inputs where the first n inputs are public and the last n' inputs are private. Figure 1 shows that this class also captures the first two examples in Section 3.



$$L(x_1,x_2,x_3,z) = \begin{pmatrix} x_1 & 0 & 0 & 0 \\ -1 & x_2 & 0 & 0 \\ 0 & -1 & x_3 & 0 \\ 0 & 0 & -1 & z \end{pmatrix}$$

$$F(x_1,x_2,x_3,z) = x_1x_2x_3z \qquad F((x_1,x_2,x_3),(z,z')) = z'(3x_1+x_2+2x_3)+z$$

Fig. 1. ABPs for examples in Section 3 with $t = 1$

4.1 Statement of Results

We now formally state our results on partial garbling.

Theorem 3 (Parti ally Garbling ABP). *Consider a function $F : \mathbb{F}_q^n \times \mathbb{F}_q^{n'} \to \mathbb{F}_q$ which is computed by an ABP $\Gamma = (V, E, v_0, v_1, \phi)$ taking as input (x, z), where the variables in the private input z appear only on the edges leading into the last t vertices in V. Then, there is a partial garbling scheme $\hat{F}$ of F with the following properties:*

- *the output length of $\hat{F}$ is $t \cdot (|V| - 1)$;*
- *each entry of $\hat{F}$ is a polynomial of total degree one in the variables x and z;*
- *each entry of $\hat{F}$ is a polynomial of total degree one in the randomness for $t = 1$, and total degree two in the randomness in the general case.*
- *the reconstruction algorithm $\mathsf{Rec}(x, \cdot)$ has degree t in the output of $\hat{F}$.*

In the interesting special case where $t = 1$ (which captures the applications to CDS and secret-sharing), we obtain an encoding of linear size and degree one reconstruction. For the case $t = |V| - 1$, we achieve the standard requirement for garbling schemes and recover a variant of the construction in [28, Theorem 1]. We note here that the degree in the randomness r is important in MPC applications where the generation of r is distributed between multiple parties, whereas the degree of the reconstruction algorithm is important in ABE schemes where reconstruction happens “in the exponent”.

Affine Partial Garbling. Combined with the locality lemma in [3, Lemma 4.17], we obtain an affine PGS for arithmetic branching programs:

Corollary 1 (Affine PGS for ABP). *Consider a function $F : \mathbb{F}_q^n \times \mathbb{F}_q^{n'} \to \mathbb{F}_q$ which is computed by an ABP $\Gamma = (V, E, v_0, v_1, \phi)$ taking as input (x, z) where the variables in the private input z appear only on the edges leading into the last t vertices in V. Then, there is an affine partial garbling scheme $\hat{F}$ of F with the following properties:*

- *$\hat{F}$ has the output length $t^2 \cdot |E|$;*
- *the reconstruction algorithm $\mathsf{Rec}(x, \cdot)$ has degree t.*

The locality lemma tells us that we can garble a polynomial of d variables of total degree one using d affine functions of a single variable (e.g. we garble $x_1 + 2x_2 + x_3$ using $(x_1 - r_1, 2x_2 - r_2, x_3 + r_1 + r_2)$), while increasing the randomness complexity by $d-1$ and without affecting the degree of the reconstruction algorithm. That is, we will replace each polynomial in d variables in $\hat{F}$ with d affine functions in one variable. This increases the output length of $\hat{F}$ from $t \cdot (|V| - 1)$ to $t^2 \cdot |E|$.

4.2 Our Construction

Following prior garbling schemes for ABP due to Ishai and Kushilevitz in [27, 28], the starting point of our construction is the matrix representation $L(x, z)$ of the ABP in Lemma 1. Since the variables in z appear only on the edges leading into the last t vertices in V, this means that they only appear in the last t columns of the matrix $L(x, z)$. Garbling proceeds similarly to that in [28, Section 4] by randomizing the last t columns of this matrix while preserving its determinant – we achieve this by multiplying $L(x, z)$ on the left and on the right by random matrices with a prescribed structure. The efficiency improvement over the prior construction comes from using matrices with fewer random entries; in particular, only the last t columns of the randomizing matrices contain random entries.

A Digression into Matrices. We consider a set $\mathcal{H}$ of matrices which contains $L(x, z)$, along with two groups of matrices $\mathcal{G}_1, \mathcal{G}_2$ which would be used to randomize $L(x, z)$ as outlined above.

Definition 2. *Let $\mathcal{H}$ denote the set of $\ell \times \ell$ matrices over $\mathbb{F}_q$ containing only -1's in their second diagonal (the diagonal below the main diagonal), and 0's below the second diagonal. For a fixed parameter t, define two matrix groups $\mathcal{G}_1$ and $\mathcal{G}_2$ as follows:*

- *$\mathcal{G}_1$ is the subset of $\ell \times \ell$ matrices over $\mathbb{F}_q$ with 1's on the main diagonal and 0's in all of the remaining entries except the right-most $t - 1$ entries in the top row;*
- *$\mathcal{G}_2$ is the subset of $\ell \times \ell$ matrices over $\mathbb{F}_q$ with 1's on the main diagonal and 0's in all of the remaining entries except for those above the main diagonal in the t right-most columns.*

It is straight-forward to verify that $\mathcal{G}_1, \mathcal{G}_2$ are both closed under multiplication and inverse; that is, both $\mathcal{G}_1$ and $\mathcal{G}_2$ are subgroups of the multiplicative group of invertible $\ell \times \ell$ matrices. Next, we establish additional properties of $\mathcal{H}, \mathcal{G}_1, \mathcal{G}_2$ which would be used to establish correctness and privacy respectively:

Lemma 2. *For any $H \in \mathcal{H}, G_1 \in \mathcal{G}_1, G_2 \in \mathcal{G}_2$, the first $\ell - t$ columns in $G_1 H G_2$ are the same as those in H.*

The following lemma (generalizing [28, Lemma 3]) shows that a matrix H from $\mathcal{H}$ can be brought into a canonical form, uniquely defined by its first $\ell - t$ columns and its determinant, by multiplying it from the left by some $G_1 \in \mathcal{G}_1$ and from the right by some $G_2 \in \mathcal{G}_2$.

Lemma 3. *For any $H \in \mathcal{H}$, there exists $G_1 \in \mathcal{G}_1$ and $G_2 \in \mathcal{G}_2$ such that G_1HG_2 satisfies the following properties (that is, the "canonical form"):*

- *the entries in the first $\ell - t$ columns of G_1HG_2 are the same as those in H;*
- *G_1HG_2 contains -1's in its second diagonal;*
- *it contains 0's elsewhere except the value $\det(H)$ in the top-right entry.*

Note that the canonical form for H is unique and is completely determined by the first $\ell - t$ columns of H and $\det(H)$.

Partially Garbling F. We may now specify our PGS $\hat{F}$: start with the $\ell \times \ell$ matrix representation $L(x,z)$ for F, where $\ell = |V| - 1$, and output the last t columns of the matrix $R_1L(x,z)R_2$, where $R_1 \leftarrow_{\mathrm{R}} \mathcal{G}_1$ and $R_2 \leftarrow_{\mathrm{R}} \mathcal{G}_2$. We proceed to analyze the construction:

- (correctness) Reconstruction proceeds as follows: Given x and the last t columns of $R_1L(x,z)R_2$, we may recover the entire matrix $R_1L(x,z)R_2$, since the first $\ell - t$ columns are the same as those in $L(x,z)$ (cf. Lemma 2) and depend only on x. Then, we compute $\det(R_1L(x,z)R_2)$, which is a degree t computation over the output of $\hat{F}$. Correctness follows from the fact that $\det(R_1L(x,z)R_2) = \det(L(x,z)) = F(x,z)$ since $\det(R_1) = \det(R_2) = 1$.
- (privacy) Given x and $F(x,z)$, we can compute the canonical form H' of the matrix $L(x,z)$ as defined in Lemma 3, namely $H' = G_1L(x,z)G_2$ for some $G_1 \in \mathcal{G}_1, G_2 \in \mathcal{G}_2$ (we do not need to compute G_1, G_2). Since $\mathcal{G}_1$ and $\mathcal{G}_2$ are both matrix groups, it follows that $R_1L(x,z)R_2$ and $R_1'H'R_2'$, where $R_1, R_1' \leftarrow_{\mathrm{R}} \mathcal{G}_1, R_2, R_2' \leftarrow_{\mathrm{R}} \mathcal{G}_2$, are identically distributed. Simulation proceeds by outputting the last t columns of $R_1'H'R_2'$.

Theorem 3 then follows readily.

References

[1] Aiello, W., Ishai, Y., Reingold, O.: Priced oblivious transfer: How to sell digital goods. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 119–135. Springer, Heidelberg (2001)

[2] Applebaum, B.: Randomly encoding functions: A new cryptographic paradigm - (invited talk). In: Fehr, S. (ed.) ICITS 2011. LNCS, vol. 6673, pp. 25–31. Springer, Heidelberg (2011)

[3] Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in NC^0. SIAM J. Comput. 36(4), 845–888 (2006)

[4] Applebaum, B., Ishai, Y., Kushilevitz, E.: From secrecy to soundness: Efficient verification via secure computation. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6198, pp. 152–163. Springer, Heidelberg (2010)

[5] Applebaum, B., Ishai, Y., Kushilevitz, E.: How to garble arithmetic circuits. In: FOCS, pp. 120–129 (2011)

[6] Attrapadung, N., Libert, B.: Functional encryption for inner product: Achieving constant-size ciphertexts with adaptive security or support for negation. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 384–402. Springer, Heidelberg (2010)

[7] Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: ACM CCS (2012); Also Cryptology ePrint Archive, Report 2012/265

[8] Ben Hamouda, F., Blazy, O., Chevalier, C., Pointcheval, D., Vergnaud, D.: Efficient UC-secure authenticated key-exchange for algebraic languages. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 272–291. Springer, Heidelberg (2013)

[9] Benabbas, S., Gennaro, R., Vahlis, Y.: Verifiable delegation of computation over large datasets. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 111–131. Springer, Heidelberg (2011)

[10] Benhamouda, F., Blazy, O., Chevalier, C., Pointcheval, D., Vergnaud, D.: New techniques for SPHFs and efficient one-round PAKE protocols. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 449–475. Springer, Heidelberg (2013)

[11] Bitansky, N., Paneth, O.: Point obfuscation and 3-round zero-knowledge. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 190–208. Springer, Heidelberg (2012)

[12] Blazy, O., Pointcheval, D., Vergnaud, D.: Round-optimal privacy-preserving protocols with smooth projective hash functions. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 94–111. Springer, Heidelberg (2012)

[13] Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)

[14] Boneh, D., Freeman, D.M.: Homomorphic signatures for polynomial functions. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 149–168. Springer, Heidelberg (2011)

[15] Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., Vinayagamurthy, D.: Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 533–556. Springer, Heidelberg (2014)

[16] Chandran, N., Goyal, V., Ostrovsky, R., Sahai, A.: Covert multi-party computation. In: FOCS, pp. 238–248 (2007)

[17] Feige, U., Kilian, J., Naor, M.: A minimal model for secure computation. In: STOC, pp. 554–563 (1994)

[18] Garg, S., Gentry, C., Halevi, S., Sahai, A., Waters, B.: Attribute-based encryption for circuits from multilinear maps. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 479–499. Springer, Heidelberg (2013)

[19] Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In: STOC, pp. 467–476 (2013); Also, Cryptology ePrint Archive, Report 2013/258

[20] Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Heidelberg (2010)

[21] Gertner, Y., Ishai, Y., Kushilevitz, E., Malkin, T.: Protecting data privacy in private information retrieval schemes. J. Comput. Syst. Sci. 60(3), 592–629 (2000)

[22] Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: interactive proofs for muggles. In: STOC, pp. 113–122 (2008)

[23] Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute-based encryption for circuits. In: STOC, pp. 545–554 (2013); Also, Cryptology ePrint Archive, Report 2013/337

[24] Gordon, S.D., Malkin, T., Rosulek, M., Wee, H.: Multi-party computation of polynomials and branching programs without simultaneous interaction. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 575–591. Springer, Heidelberg (2013)

[25] Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM Conference on Computer and Communications Security, pp. 89–98 (2006)

[26] Ishai, Y., Kushilevitz, E.: Private simultaneous messages protocols with applications. In: ISTCS, pp. 174–184 (1997)

[27] Ishai, Y., Kushilevitz, E.: Randomizing polynomials: A new representation with applications to round-efficient secure computation. In: FOCS, pp. 294–304 (2000)

[28] Ishai, Y., Kushilevitz, E.: Perfect constant-round secure computation via perfect randomizing polynomials. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 244–256. Springer, Heidelberg (2002)

[29] Ishai, Y., Paskin, A.: Evaluating branching programs on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 575–594. Springer, Heidelberg (2007)

[30] Ito, M., Saito, A., Nishizeki, T.: Secret sharing schemes realizing general access structure. In: GLOBECOM, pp. 99–102 (1987)

[31] Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)

[32] Parno, B., Raykova, M., Vaikuntanathan, V.: How to delegate and verify in public: Verifiable computation from attribute-based encryption. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 422–439. Springer, Heidelberg (2012)

[33] Prabhakaran, M., Sahai, A.: Secure Multi-Party Computation. IOS Press (2003)

[34] Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)

[35] Shamir, A.: Factoring numbers in O(log n) arithmetic steps. Inf. Process. Lett. 8(1), 28–31 (1979)

[36] Tassa, T.: Generalized oblivious transfer by secret sharing. Des. Codes Cryptography 58(1), 11–21 (2011)

[37] von Ahn, L., Hopper, N.J., Langford, J.: Covert two-party computation. In: STOC, pp. 513–522 (2005)

[38] Yao, A.C.-C.: Theory and applications of trapdoor functions. In: FOCS, pp. 80–91 (1982)

On the Complexity of Trial and Error for Constraint Satisfaction Problems

Gábor Ivanyos[1], Raghav Kulkarni[2], Youming Qiao[2,4], Miklos Santha[2,3],
and Aarthi Sundaram[2]

[1] Institute for Computer Science and Control, Hungarian Academy of Sciences,
Budapest, Hungary
Gabor.Ivanyos@sztaki.mta.hu

[2] Centre for Quantum Technologies, National University of Singapore
kulraghav@gmail.com, aarthims@nus.edu.sg

[3] LIAFA, Univ. Paris 7, CNRS, 75205 Paris, France
miklos.santha@liafa.univ-paris-diderot.fr

[4] Centre for Quantum Computation and Intelligent Systems
University of Technology, Sydney
jimmyqiao86@gmail.com

Abstract. In a recent work of Bei, Chen and Zhang (STOC 2013), a trial and error model of computing was introduced, and applied to some constraint satisfaction problems. In this model the input is hidden by an oracle which, for a candidate assignment, reveals some information about a violated constraint if the assignment is not satisfying. In this paper we initiate a *systematic* study of constraint satisfaction problems in the trial and error model. To achieve this, we first adopt a formal framework for CSPs, and based on this framework we define several types of revealing oracles. Our main contribution is to develop a *transfer theorem* for each type of the revealing oracle, under a broad class of parameters. To any hidden CSP with a specific type of revealing oracle, the transfer theorem associates another, potentially harder CSP in the normal setting, such that their complexities are polynomial time equivalent. This in principle transfers the study of a large class of hidden CSPs, possibly with a promise on the instances, to the study of CSPs in the normal setting. We then apply the transfer theorems to get polynomial-time algorithms or hardness results for hidden CSPs, including satisfaction problems, monotone graph properties, isomorphism problems, and the exact version of the Unique Games problem.

1 Introduction

In [2], Bei, Chen and Zhang proposed a *trial and error* model to study algorithmic problems when some input information is lacking. As argued in their paper, the lack of input information can happen when we have only limited knowledge of and access to the problem. They also described several realistic scenarios where the inputs are actually unknown. Then, they formalized this methodology in

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 663–675, 2014.

the complexity-theoretic setting, and proposed a trial and error model for constraint satisfaction problems. They further applied this idea to investigate the information needed to solve linear programming in [3], and to study information diffusion in a social network in [1].

As mentioned, in [2] the authors focused on the hidden versions of some specific constraint satisfaction problems (H–CSPs), whose instances could only be accessed via a *revealing* oracle. An algorithm in this setting interacts with this revealing oracle to get information about the input instance. Each time, the algorithm proposes a candidate solution, a *trial*, and the validity of this trial is checked by the oracle. If the trial succeeds, the algorithm is notified that the proposed trial is already a solution. Otherwise, the algorithm obtains as an *error*, a violation of some property corresponding to the instance. The algorithm aims to make effective use of these errors to propose new trials. The optimal algorithm minimizes the number of trials while keeping in mind the cost for proposing new trials. When the CSP is already difficult, a computation oracle that solves the original problem might be allowed. Its use is justified as we are interested in the *extra difficulty* caused by the lack of information. Bei, Chen and Zhang considered several natural CSPs in the trial and error setting, including SAT, Stable Matching, Graph Isomorphism and Group Isomorphism. While the former two problems in the hidden setting are shown to be of the same difficulty as in the normal one, the last two cases have substantially increased complexities in the unknown-input model. They also studied more problems, as well as various aspects of this model, like the query complexity.

In this paper, following [2], we initiate a *systematic* study of the constraint satisfaction problems in the trial and error model. To achieve this, we first adopt a formal framework for CSPs. Based on this framework we define three types of revealing oracles to generalize the model of [2]. Our main contribution is to develop a *transfer theorem* for each type of the revealing oracle, under a broad class of parameters. For any hidden CSP with a specific type of revealing oracle, the transfer theorem associates another CSP in the normal (unhidden) setting, such that their difficulties are roughly the same. This in principle transfers the study of hidden CSPs to the study of CSPs in the normal setting. We also apply transfer theorems to get results for concrete CSPs, including some problems considered in [2], for which we usually get much shorter and easier proofs.

The Framework for CSPs, and Hidden CSPs. To state our results we describe informally the framework of CSPs. A CSP S is defined by a finite alphabet $[\![w]\!] = \{0, 1, \dots, w-1\}$ and by $\mathcal{R} = \{R_1, \dots, R_s\}$, a set of relations over $[\![w]\!]$ of some fixed arity q. For a set of variables $\mathcal{V} = \{x_1, \dots, x_\ell\}$, an instance of S is a set of constraints $\mathcal{C} = \{C_1, \dots, C_m\}$, where $C_j = R(x_{j_1}, \dots, x_{j_q})$ for some relation $R \in \mathcal{R}$ and some q-tuple of variables. An assignment $a \in [\![w]\!]^\ell$ satisfies $\mathcal{C}$ if it satisfies every constraint in it.

Example 1. 1SAT: Here $w = 2$, $q = 1$, and $\mathcal{R} = \{\mathsf{Id}, \mathsf{Neg}\}$, where $\mathsf{Id} = \{1\}$ is the identity relation, and $\mathsf{Neg} = \{0\}$ is its complement. Thus a constraint is a literal x_i or $\bar{x}_i$, and an instance is just a collection of literals. In case of 3SAT the parameters are $w = 2$, $q = 3$ and $|\mathcal{R}| = 8$. We will keep for further illustrations

$\mathsf{1SAT}$ which is a problem in polynomial time. $\mathsf{3SAT}$ would be a less illustrative example since the standard problem is already NP-complete. We omit $\mathsf{2SAT}$ as its hardness is implied from that of $\mathsf{1SAT}$.

To allow for more versatility, we often consider some *promise* $W \subseteq [\![w]\!]^\ell$ on the assignments, and only look for a satisfying assignment within this promise. This case happens, say when we look for permutations in isomorphism problems.

Recall that in the hidden setting, the algorithm interacts with some revealing oracle by repeatedly proposing assignments. If the proposed assignment is not satisfying then the revealing oracle discloses certain information about some violated constraint. This can be in principle an index of such a constraint, (the index of) the relation in it, the indices of the variables where this relation is applied, or any subset of the above. Here we will require that the oracle always reveals the index of a violated constraint from $\mathcal{C}$. To characterize the choices for the additional information, for any subset $\mathcal{U} \subseteq \{\mathcal{R}, \mathcal{V}\}$ we say that an oracle is *$\mathcal{U}$-revealing* if it also gives out the information corresponding to $\mathcal{U}$. For a CSP problem S we use $\mathrm{H{-}S}_{\mathcal{U}}$ to denote the corresponding hidden problem in the trial and error model with $\mathcal{U}$-revealing oracle.

Example 1 continued. Let us suppose that we present an assignment $a \in \{0,1\}^\ell$ for an instance of the hidden version $\mathrm{H{-}}\mathsf{1SAT}_{\mathcal{U}}$ of $\mathsf{1SAT}$ to the $\mathcal{U}$-revealing oracle. If $\mathcal{U} = \{\mathcal{V}\}$ and the oracle reveals j and i respectively for the violated constraint and the variable in it then we learn that the jth literal is x_i if $a_i = 0$, and $\bar{x}_i$ otherwise. If $\mathcal{U} = \{\mathcal{R}\}$ and say the oracle reveals j and Id then we learn that the jth literal is positive. If $\mathcal{U} = \emptyset$ and the oracle reveals j then we only learn that the jth literal is either a positive literal corresponding to one of the indices where a is 0, or a negative literal corresponding to an index where a is 1.

In order to explain the transfer theorem and motivate the operations which create richer CSPs, we first make a simple observation that $\mathrm{H{-}S}_{\{\mathcal{R},\mathcal{V}\}}$ and S are polynomial time equivalent, when the relations of S are in P (note that the latter does not necessarily imply that S is in P). Indeed, an algorithm for $\mathrm{H{-}S}_{\{\mathcal{R},\mathcal{V}\}}$ can solve S, as the answers of the oracle can be given by directly checking if the proposed assignment is satisfying. In the other direction, we repeatedly submit assignments to the oracle. The answer of the oracle fully reveals a (violated) constraint. Given some subset of constraints we already know, to find a new constraint, we submit an assignment which satisfies all the known constraints. Such an assignment can be found by the algorithm for S.

With a weaker oracle this procedure clearly does not work and to compensate, we need stronger CSPs. In the case of $\{\mathcal{V}\}$-revealing oracles an answer helps us include as possibilities for the specified clause, all those relations which were violated at the specified indices of the proposed assignment, and remove all the relations which were satisfied at those indices. Therefore, to find out more information about the input, we would like to find a satisfying assignment for a CSP instance whose corresponding constraint is the union of all its possibilities. This naturally brings us to consider the CSP $\bigcup$ S, the *closure by union* of S whose relations are from $\bigcup\mathcal{R}$, the *closure by union* of $\mathcal{R}$, which contains relations by taking union over any subset of $\mathcal{R}$.

The situation with the $\{\mathcal{R}\}$-revealing oracle is analogous, but here we have to compensate, in the stronger CSP, for the lack of revealed information about the variable indices. For a relation R and q-tuple of distinct indices $(j_1,\ldots,j_q)$, we define the ℓ-ary relation $R^{(j_1,\ldots,j_q)} = \{a \in W \ : \ (a_{j_1},\ldots,a_{j_q}) \in R\}$, and for a set I of q-tuples of indices, we set $R^I = \bigcup_{(j_1,\ldots,j_q)\in I} R^{(j_1,\ldots,j_q)}$. The *arity extension* of S is the constraint satisfaction problem E–S whose relations are from arity extension $\text{E–}\mathcal{R} = \bigcup_I \{R^I \ : \ R \in \mathcal{R}\}$ of $\mathcal{R}$.

The transfer theorem first says that with $\bigcup$S (resp. E–S) we can compensate the information hidden by a $\{\mathcal{V}\}$-revealing (resp. $\{\mathcal{R}\}$-revealing) oracle, that is we can solve H–S$_{\{\mathcal{V}\}}$ (resp. H–S$_{\{\mathcal{R}\}}$). In fact, with $\bigcup$E–S we can solve H–S$_\emptyset$. Moreover, perhaps more surprisingly, it says that these statements also hold in the reverse direction: if we can solve the hidden CSP, we can also solve the corresponding extended CSP.

Transfer Theorem (informal statement). *Let* S *be a CSP whose parameters are "reasonable" and whose relations are in* P*. Then for any promise W on the assignments, the complexities of the following problems are polynomial time equivalent: (a)* H–S$_{\{\mathcal{V}\}}$ *and* $\bigcup$S*, (b)* H–S$_{\{\mathcal{R}\}}$ *and* E–S*, (c)* H–S$_\emptyset$ *and* $\bigcup$E–S*.*

The precise dependence on the parameters can be found in the theorems of Section 3 and Corollary 1 highlights the conditions for polynomial equivalence.

Example 1 continued. Since $\bigcup\{\mathsf{Id},\mathsf{Neg}\} = \{\emptyset,\mathsf{Id},\mathsf{Neg},\{0,1\}\}$, $\bigcup$1SAT has only the two trivial (always false or always true) relations in addition to the relations in 1SAT. Therefore it can be solved in polynomial time, and by the the Transfer Theorem H–1SAT$_{\{\mathcal{V}\}}$ is also in P. On the other hand, for any index set $I \subseteq [\ell]$, Id^I is a disjunct of positive literals with variables from I, and similarly Neg^I is a disjunct of negative literals with variables from I. Thus E–1SAT includes MONSAT, which consists of those instances of SAT where in each clause either every variable is positive, or every variable is negated. The problem MONSAT is NP-hard by Schaefer's characterization [6], and therefore the Transfer Theorem implies that H–1SAT$_{\{\mathcal{R}\}}$ and H–1SAT$_\emptyset$ are also NP-hard.

In a further generalization, we will also consider CSPs and H–CSPs whose instances satisfy some property. One such property can be *repetition freeness* meaning that the constraints of an instance are pairwise distinct. The promise H–CSPs could also be a suitable framework for discussing certain graph problems on special classes of graphs. For a promise PROM on instances of S we denote by S$^{\mathsf{PROM}}$ the promise problem whose instances are instances of S satisfying PROM. The problem H–S$^{\mathsf{PROM}}_{\{\mathcal{U}\}}$ is defined in an analogous way from H–S$_{\{\mathcal{U}\}}$.

It turns out that we can generalize the Transfer Theorem for CSPs with promises on the instances. We describe this in broad lines for the case of $\{\mathcal{V}\}$-revealing oracles. Given a promise PROM on S, the corresponding promise $\bigcup$PROM for $\bigcup$S is defined in a natural way. We say that a $\bigcup$S-instance $\mathcal{C}'$ *includes* an S-instance $\mathcal{C}$ if for every $j \in [m]$, the constraint C'_j in $\mathcal{C}'$ and the constraint C_j in $\mathcal{C}$ are defined on the same variables, and seen as relations,

$C_j \subseteq C'_j$. Then $\bigcup \mathsf{PROM}$ is the set of instances $\mathcal{C}'$ of $\bigcup \mathrm{S}$ which include some $\mathcal{C} \in \mathsf{PROM}$. The concept of an algorithm *solving* $\bigcup \mathrm{S}^{\bigcup \mathsf{PROM}}$ has to be relaxed: while we search for a satisfying assignment for those instances which include a satisfiable instance of PROM, when this is not the case, the algorithm can abort even if the instance is satisfiable. With this we have:

Transfer Theorem for Promise Problems (informal statement). *Let* S *be a constraint satisfaction problem with promise* PROM*. Then the complexities of* $\mathrm{H}\text{–}\mathrm{S}^{\mathsf{PROM}}_{\{\mathcal{V}\}}$ *and* $\bigcup S^{\bigcup \mathsf{PROM}}$ *are polynomial time equivalent when the parameters are "reasonable" and the relations of* S *are in* P.

Example 1 continued. Let RF denote the property of being repetition free, in the case of $\mathsf{1SAT}$ this just means that no literal can appear twice in the formula. Then $\mathrm{H}\text{–}\mathsf{1SAT}^{\mathsf{RF}}_{\emptyset}$, hidden repetition-free $\mathsf{1SAT}$ with $\emptyset$-revealing oracle, is solved in polynomial time. To see this we first consider $\mathrm{X}\text{–}\mathsf{1SAT}$, the constraint satisfaction problem whose relations are all ℓ-ary extensions of Id and Neg. (See Section 2 for a formal definition.) It is quite easy to see that hidden $\mathsf{1SAT}$ with $\emptyset$-revealing oracle is essentially the same problem as hidden $\mathrm{X}\text{–}\mathsf{1SAT}$ with $\{\mathcal{V}\}$-revealing oracle. Therefore, by the Transfer Theorem we are concerned with $\bigcup \mathrm{X}\text{–}\mathsf{1SAT}$ with promise $\bigcup \mathsf{RF}$. The instances satisfying the promise are $\{C_1, \ldots, C_m\}$, where C_j is a disjunction of literals such that there exist distinct literals $z_1, \ldots, z_m$, with $z_j \in C_j$. It turns out that these specific instances of SAT can be solved in polynomial time. The basic idea is that we can apply a maximum matching algorithm, and only output a solution if we can select m pairwise different variables $x_{i_1}, \ldots, x_{i_m}$ such that either x_{i_j} or $\overline{x}_{i_j}$ is in C_j.

Applications of Transfer Theorems. Since NP-hard problems obviously remain NP-hard in the hidden setting (without access to an NP oracle), we investigate the complexity of various polynomial-time solvable CSPs. We first apply the Transfer Theorem when there is no promise on the instances. We categorize the hidden CSPs depending on the type of the revealing oracle.

With constraint index revealing oracles, we focus on various monotone graph properties like Spanning Tree, Cycle Cover, etc. We define a general framework to represent monotone graph property problems as H–CSPs and show that they become NP-hard. This framework also naturally extends to directed graphs.

With constraint and variable index revealing oracles, we obtain results on several interesting families of CSPs including the exact-Unique Games Problem (cf. Section 5), equality to a member of a fixed class of graphs, and graph properties discussed as above. Interestingly, many of the graph properties mentioned in the last paragraph are no longer NP-hard but in P, as well as some other CSPs like $\mathsf{2SAT}$ and the exact-Unique Game problem on alphabet size 2. Still, there are some NP-hard CSPs, like the exact-Unique Game problem on alphabet size ≥ 3, and equality to some specific graph, such as k-cliques. The latter problem is just the Graph Isomorphism problem considered in [2, Theorem 13], whose proof, with the help of the Transfer Theorem, becomes very simple.

With constraint and relation index revealing oracles, we show a dichotomy theorem similar to results obtained in [4,5] for any CSP with constant arity and alphabet size: if some string of the form $(\alpha, \ldots, \alpha)$ satisfies all the non-empty relations then the problem is in P, otherwise it is NP-hard.

Finally, we investigate hidden CSPs with promises on the instances. We first consider the repetition freeness promise, as exhibited by the $\mathsf{1SAT}$ example as above. Though the hidden repetition free $\mathsf{1SAT}$ problem becomes solvable in polynomial time, in this setting $\mathsf{2SAT}$ is still NP-hard. The group isomorphism problem can also be cast in this framework, and we give a simplified proof of [2, Theorem 11]: to compute an explicit isomorphism of the hidden group with $\mathbb{Z}_p$ is NP-hard.

Organization. In Section 2 we formally describe the model of CSPs, and hidden CSPs. In Section 3, the transfer theorems are stated and proved. Section 4, 5, and 6 contain the applications of the main theorems in the case of $\emptyset$-revealing, $\{\mathcal{V}\}$-revealing and $\{\mathcal{R}\}$-revealing oracles respectively. Finally in Section 7 we present the results for hidden promise CSPs. Most proofs are omitted from this version of the paper due to space constraints.

2 Preliminaries

The Model of Constraint Satisfaction Problems. For a positive integer k, let $[k]$ denote the set $\{1, \ldots, k\}$. (Recall that $[\![k]\!] = \{0, 1, \ldots, k-1\}$.) A *constraint satisfaction problem*, (CSP) S, is specified by its set of parameters and its type, both defined for every positive integer n.

The *parameters* are the alphabet size $w(n)$, the assignment length $\ell(n)$, the set of (admissible) assignments $W(n) \subseteq [\![w(n)]\!]^{\ell(n)}$, the arity $q(n)$, and the number of relations $s(n)$. We suppose that $W(n)$ is symmetric, that is for $\forall \pi \in S_{\ell(n)}$, if $a_1 \ldots a_{\ell(n)} \in W(n)$ then $a_{\pi(1)} \ldots a_{\pi(\ell(n))} \in W(n)$. To simplify notations, we often omit n from the parameters, and just write w, ℓ, W, q and s.

We denote by W_q the projection of W to q coordinates, i.e. $W_q = \{u \in [\![w]\!]^q \ : \ uv \in W \text{ for some } v \in [\![w]\!]^{\ell-q}\}$. A q-ary *relation* is $R \subseteq W_q$. For b in W_q, if $b \in R$, we sometimes write $R(b) = \mathrm{T}$, and similarly for $b \notin R$ we write $R(b) = \mathrm{F}$. The *type* of S is a set of q-ary relations $\mathcal{R}_n = \{R_1, \ldots, R_s\}$, where $R_k \subseteq W_q$, for every $k \in [s]$. As for the parameters, we usually just write $\mathcal{R}$.

We set $[\ell]^{(q)} = \{(j_1, \ldots, j_q) \in [\ell]^q \ : \ |\{j_1, \ldots, j_q\}| = q\}$, that is $[\ell]^{(q)}$ denotes the set of distinct q-tuples from $[\ell]$. An *instance* of S is given by a set of m (m may depend on n) constraints $\mathcal{C} = \{C_1, \ldots, C_m\}$ over a set $\mathcal{V} = \{x_1, \ldots, x_\ell\}$ of variables, where a *constraint* is $R_k(x_{j_1}, \ldots, x_{j_q})$ for some $k \in [s]$ and $(j_1, \ldots, j_q) \in [\ell]^{(q)}$. We say that an assignment $a \in W$ satisfies $C_j = R_k(x_{j_1}, \ldots, x_{j_q})$ if $R_k(a_{j_1}, \ldots, a_{j_q}) = \mathrm{T}$. An assignment *satisfies* $\mathcal{C}$ if it satisfies all its constraints. The *size* of an instance is $n + m(\log s + q \log \ell) + \ell \log w$ which includes the length of the description of $\mathcal{C}$ and the length of the assignments. In all our applications the instance size will be polynomial in n. A *solution* of $\mathcal{C}$ is a satisfying assignment if there exists any, and NO otherwise.

We further introduce the following notations. For a relation R let $\mathrm{comp}(R)$ be the time complexity of deciding the membership of a tuple in R, and for a set of relations $\mathcal{R}$ let $\mathrm{comp}(\mathcal{R})$ be $\max_{R \in \mathcal{R}} \mathrm{comp}(R)$. We denote by $\dim(\mathcal{R})$ the *dimension* of $\mathcal{R}$ which is defined as the length of the longest chain of relations (for inclusion) in $\mathcal{R}$.

We also introduce two new operations which create richer sets of relations from a relation set. For a given CSP S, these richer sets of relations derived from the type of S, will be the types of harder CSPs which turn out to be equivalent to various hidden variants of S. The first operation is standard. We denote by $\bigcup \mathcal{R}$ the closure of $\mathcal{R}$ by the union operation, that is $\bigcup \mathcal{R} = \{\bigcup_{R \in \mathcal{R}'} R \ : \ \mathcal{R}' \subseteq \mathcal{R}\}$. We define the (*closure by*) *union* of S as the constraint satisfaction problem $\bigcup$S whose parameters are the same as those of S except the number of relations which is at most 2^s, and whose type is $\bigcup \mathcal{R}$. We remark that $\dim(\bigcup \mathcal{R}) \leq \min\{|\mathcal{R}|, |W_q|\}$.

For a relation $R \in \mathcal{R}$ and for $(j_1, \ldots, j_q) \in [\ell]^{(q)}$ we define the ℓ-ary relation $R^{(j_1,\ldots,j_q)} = \{a \in W \ : \ (a_{j_1}, \ldots, a_{j_q}) \in R\}$, and X–$\mathcal{R} = \{R^{(j_1,\ldots,j_q)} \ : \ R \in \mathcal{R}$ and $(j_1, \ldots, j_q) \in [\ell]^{(q)}\}$. The set X–$\mathcal{R}$ contains the natural extension of relations in $\mathcal{R}$ from arbitrary coordinates. If we want to consider unions of the same relation from arbitrary coordinates, then for $I \subseteq [\ell]^{(q)}$, we set $R^I = \bigcup_{(j_1,\ldots,j_q) \in I} R^{(j_1,\ldots,j_q)}$, and define the *arity extension* of $\mathcal{R}$, as E–$\mathcal{R}$=$\bigcup_{R \in \mathcal{R}}\{R^I \ : \ I \subseteq [\ell]^{(q)}\}$. Observe that E–$\mathcal{R} \subseteq \bigcup$ X–$\mathcal{R} = \bigcup$ E–$\mathcal{R}$. The *arity extension* of S is the constraint satisfaction problem E–S whose parameters are the same as those of S except for the arity which becomes ℓ, and the number of relations which becomes at most $s\frac{\ell!}{(\ell-q)!}$. The type of E–S is E–$\mathcal{R}$. The problem X–S is defined similarly, but with type X–$\mathcal{R}$.

Hidden CSP **in the Trial and Error Model.** Suppose that we want to solve a CSP problem S whose parameters and type are known to us, but for the instance $\mathcal{C}$, we are explicitly given only n and the number of constraints m. The instance is otherwise specified by a *revealing* oracle V for $\mathcal{C}$ which can be used by an algorithm to receive information about the constraints in $\mathcal{C}$. The algorithm can propose $a \in W$ to the oracle which is conceived as its guess for a satisfying assignment. If a indeed satisfies $\mathcal{C}$ then V answers YES. Otherwise there exists some violated constraint $C_j = R_k(x_{j_1}, \ldots, x_{j_q})$, and the oracle has to reveal some information about that. We will require that the oracle always reveals j, the index of the constraint C_j in $\mathcal{C}$, but in addition, it can also make further disclosures. These can be k, the index of the relation R_k in $\mathcal{R}$; $(j_1, \ldots, j_q)$, the q-tuple of indices of the ordered variables $x_{j_1}, \ldots, x_{j_q}$ in $\mathcal{V}$; or both of these. To characterize the choices for the additional information, for any subset $\mathcal{U} \subseteq \{\mathcal{R}, \mathcal{V}\}$, we require that a *$\mathcal{U}$-revealing* oracle $\mathsf{V}_{\mathcal{U}}$ give out the information corresponding to $\{\mathcal{C}\} \bigcup \mathcal{U} \subseteq \{\mathcal{C}, \mathcal{R}, \mathcal{V}\}$. Thus for example a $\emptyset$-revealing oracle $\mathsf{V}_{\emptyset}$ reveals the index j of some violated constraint but nothing else, whereas a $\mathcal{V}$-revealing oracle $\mathsf{V}_{\{\mathcal{V}\}}$ also reveals the indices $(j_1, \ldots, j_q)$ of the variables of the relation in the clause C_j, but not the name of the relation.

Analogously, for every CSP S, and for every $\mathcal{U} \subseteq \{\mathcal{R}, \mathcal{V}\}$, we define the *hidden constraint satisfaction problem* (H–CSP) with *$\mathcal{U}$-revealing oracle* H–S$_{\mathcal{U}}$ whose parameters and type are those of S, but whose instances are specified by a $\mathcal{U}$-revealing oracle. An algorithm *solves* the problem H–S$_{\mathcal{U}}$ if for all n, m, for every instance $\mathcal{C}$ for S, specified by any $\mathcal{U}$-revealing oracle for $\mathcal{C}$, it outputs a satisfying assignment if there exists any, and NO otherwise. The complexity of an algorithm for H–S$_{\mathcal{U}}$ is the number of steps in the worst case over all inputs and all $\mathcal{U}$-revealing oracles, where a query to the oracle is counted as one step.

3 Transfer Theorems for Hidden CSPs

In this section we precisely state our transfer theorems between H–CSPs and CSPs with extended types. We will only give the proof for the case of the $\{\mathcal{V}\}$-revealing oracle below owing to space constraints.

Theorem 1. *(a) If* $\bigcup$S *is solvable in time T then* H–S$_{\{\mathcal{V}\}}$ *is solvable in time* $O((T + s \times \mathrm{comp}(\mathcal{R})) \times m \times \dim(\bigcup \mathcal{R})$. *(b) If* H–S$_{\{\mathcal{V}\}}$ *is solvable in time T then* $\bigcup$S *is solvable in time* $O(T \times m \times \mathrm{comp}(\bigcup \mathcal{R}))$.

Theorem 2. *(a) If* E–S *is solvable in time T then* H–S$_{\{\mathcal{R}\}}$ *is solvable in time* $O((T + |[\ell]^{(q)}| \times \mathrm{comp}(\mathcal{R})) \times m \times |[\ell]^{(q)}|)$. *(b) If* H–S$_{\{\mathcal{R}\}}$ *is solvable in time T then* E–S *is solvable in time* $O(T \times m \times \mathrm{comp}(\text{E–}\mathcal{R}))$.

Theorem 3. *(a) If* $\bigcup$E–S *is solvable in time T then* H–S$_{\emptyset}$ *is solvable in time* $O((T + s \times |[\ell]^{(q)}| \times \mathrm{comp}(\mathcal{R})) \times m \times \dim(\bigcup \text{E–}\mathcal{R}))$. *(b) If* H–S$_{\emptyset}$ *is solvable in time T then* $\bigcup$E–S *is solvable in time* $O(T \times m \times \mathrm{comp}(\bigcup \text{E–}\mathcal{R}))$.

Proof of Theorem 1. We first prove (a). Let $\mathcal{A}$ be an algorithm which solves $\bigcup$S in time T. We define an algorithm $\mathcal{B}$ for H–S$_{\{\mathcal{V}\}}$. The algorithm will repeatedly call $\mathcal{A}$, until it finds a satisfying assignment or reaches the conclusion NO. The instance $\mathcal{C}^t = \{C_1^t, \dots, C_m^t\}$ of the tth call is defined as $C_j^t = \bigcup_{R \in \mathcal{R} : R \cap A_j^t = \emptyset} R(x_{j_1^t}, \dots, x_{j_q^t})$ where $A_j^t \subseteq W_q$ and $(j_1^t, \dots, j_q^t) \in [\ell]^{(q)}$, for $j \in [m]$, are determined successively by $\mathcal{B}$. Initially $A_j^1 = \emptyset$ and $(j_1^1, \dots, j_q^1)$ is arbitrary. If the output of $\mathcal{A}$ for $\mathcal{C}^t$ is NO then $\mathcal{B}$ outputs NO. If the output of $\mathcal{A}$ for $\mathcal{C}^t$ is $a \in W$ then $\mathcal{B}$ submits a to the $\{\mathcal{V}\}$-revealing oracle V. If V answers YES then $\mathcal{B}$ outputs a. If the oracle does not find a satisfying, and reveals j and $(j_1, \dots, j_q)$ about the violated constraint, then $\mathcal{B}$ does not change A_i^t and $(i_1^1, \dots, i_q^1)$ for $i \neq j$, but sets $A_j^{t+1} = A_j^t \bigcup \{(a_{j_1}, \dots, a_{j_q})\}$, and $(j_1^{t+1}, \dots, j_q^{t+1}) = (j_1, \dots, j_q)$. Observe that the q-tuple for the jth constraint is changed at most once, the first time when the revealing oracle gives the index of the jth constraint.

To prove that the algorithm correctly solves H–S$_{\{\mathcal{V}\}}$, let $\mathcal{C} = \{C_1, \dots, C_m\}$ be an instance of S and let V be any $\{\mathcal{V}\}$-revealing oracle for $\mathcal{C}$. We have to show that if $\mathcal{B}$ answers NO then $\mathcal{C}$ is unsatisfiable. If $\mathcal{B}$ answers NO, then for some t, the tth call of $\mathcal{A}$ resulted in output NO. By construction A_j^t and $(j_1^t, \dots, j_q^t)$, for every $j \in [m]$, are such that if $R \cap A_j^t \neq \emptyset$ then C_j can't be $R(x_{j_1}, \dots, x_{j_q})$. Indeed, if $C_j = R(x_{j_1}, \dots, x_{j_q})$ and $b \in R \cap A_j^t$ then at the call when b was added to A^t_j the oracle's answer is incorrect. Therefore all possible remaining R_js are included in C^t_j, and since $\mathcal{C}^t$ is unsatisfiable, so is $\mathcal{C}$.

For the complexity of the algorithm let us remark that if for some j and t, the constraint C^t_j is the empty relation then $\mathcal{B}$ stops since $\mathcal{C}^t$ becomes unsatisfiable. This happens in particular if $A^t_j = W_q$. Since for every call to $\mathcal{A}$ one new element is added to one of the A^t_j and at least one new relation in $\mathcal{R}$ is excluded from C^t_j, the number of calls is upper bounded by $m \times \dim(\mathcal{R})$. To compute a new constraint, some number of relations in $\mathcal{R}$ have to be computed on a new argument, which can be done in time $s \times \mathrm{comp}(\mathcal{R})$.

We now prove (b). Let $\mathcal{A}$ be an algorithm which solves H–S$_{\{\mathcal{V}\}}$ in time T. Without loss of generality we suppose that $\mathcal{A}$ only outputs a satisfying assignment a after submitting it to the verifying oracle. We define an algorithm $\mathcal{B}$ for $\bigcup$S. Let $\mathcal{C} = \{C_1, \ldots, C_m\}$ be an instance of $\bigcup$S where for $j \in [m]$, $C_j = \bigcup_{R \in \mathcal{R}_j} R(x_{j_1}, \ldots, x_{j_q})$, for some $\mathcal{R}_j \subseteq \mathcal{R}$ and $(j_1, \ldots, j_q) \in [\ell]^{(q)}$. The algorithm $\mathcal{B}$ runs $\mathcal{A}$, and outputs NO whenever $\mathcal{A}$ outputs NO. During $\mathcal{A}$'s run $\mathcal{B}$ simulates a $\{\mathcal{V}\}$-revealing oracle V for $\mathcal{A}$ which we describe now. Simultaneously with V's description we also specify instances $\mathcal{C}^t = \{C^t_1, \ldots, C^t_m\}$ of $\bigcup$S which will be used in the proof of correctness of the algorithm. For $j \in [m]$, the constraints of $\mathcal{C}^t$ are defined as $C^t_j = \bigcup_{R \in \mathcal{R} : R \cap A^t_j = \emptyset} R(x_{j^t_1}, \ldots, x_{j^t_q})$, where the sets $A^t_j \subseteq W_q$ are determined by the result of the tth call to the oracle. Initially $A^0_j = \emptyset$. For every request $a \in W$, the algorithm $\mathcal{B}$ checks if a satisfies $\mathcal{C}$. If it is the case then V returns a and $\mathcal{B}$ outputs a. Otherwise there exists $j \in [m]$ such that a violates C_j, and the answer of the oracle is j and $(j_1, \ldots, j_q)$ (where j can be chosen arbitrarily among the violated constraints, if there are several). Observe that this is a legitimate oracle for any instance of H–S$_{\{\mathcal{V}\}}$ whose jth constraint is arbitrarily chosen from $\mathcal{R}_j$. We define $A^{t+1}_j = A^t_j \bigcup \{(a_{j_1}, \ldots, a_{j_q})\}$, and for $i \neq j$ we set $A^{t+1}_i = A^t_i$.

To show the correctness of $\mathcal{B}$, we prove that whenever $\mathcal{A}$ outputs NO, the instance $\mathcal{C}$ is unsatisfiable. Let us suppose that $\mathcal{A}$ made t queries before outputting NO. An algorithm for H–S$_{\{\mathcal{V}\}}$ can output NO only if all possible instances of S which are compatible with the answers received from the oracle are unsatisfiable. In such an instance the jth constraint has necessarily empty intersection with A^t_j, therefore we can deduce that the $\bigcup$S instance $\mathcal{C}^t$ is unsatisfiable. It also holds that $A^t_j \bigcap C_j = \emptyset$ for every $j \in [m]$, since if $b \in A^t_j \bigcap C_j$ then the request to the oracle because of which b was added to A^t_j wouldn't violate the jth constraint. Thus $C_j \subseteq C^t_j$, and $\mathcal{C}$ is unsatisfiable.

For the complexity analysis we observe that during the algorithm, for every query to the oracle and for every constraint, one relation in $\bigcup \mathcal{R}$ is evaluated. □

Corollary 1. *Let* $\mathrm{comp}(\mathcal{R})$ *be polynomial. Then the complexities of the following problems are polynomial time equivalent: (a)* H–S$_{\{\mathcal{V}\}}$ *and* $\bigcup$S *if the number of relations s is constant, (b)* H–S$_{\{\mathcal{R}\}}$ *and* E–S *if the arity q is constant, (c)* H–S$_{\emptyset}$ *and* $\bigcup$E–S *if both s and q are constant.*

The polynomial time equivalence of Theorems 1, 2, 3 and Corollary 1 remain true when the algorithms have access to the same computational oracle. Therefore, we get generic easiness results for H–CSPs under an NP oracle.

4 Constraint-Index Revealing Oracle

In this section, we present some applications of our transfer theorems in the context of the constraint-index revealing oracle. Here we propose a framework for monotone graph properties to present our examples. Recall that a *monotone graph property* of an n-vertex graph is a monotone Boolean function $\mathcal{P}$ on $\binom{n}{2}$ variables invariant under relabeling of vertices. The CSP $\mathrm{S}_{\mathcal{P}}$ associated with $\mathcal{P}$ has parameters $w = 2,\ q = 1,\ \ell = \binom{n}{2}$, $W_{\mathcal{P}} = \{A \mid A$ is a graph with minimal number of edges satisfying $\mathcal{P}\}$, and $\mathcal{R} = \{\mathsf{Neg}\}$. The goal is to decide, given a graph $G = (V, E)$, whether there exists an $A \in W_{\mathcal{P}}$ such that $A \subseteq G$. The corresponding constraints are $e \notin A$ for every $e \notin E$. We have X–$\mathcal{R} = \{\mathsf{Neg}^e \mid e \in \binom{n}{2}\}$, where $\mathsf{Neg}^e(\alpha_1, \ldots, \alpha_{\binom{n}{2}}) = \neg\alpha_e$. Thus, the $\bigcup$X–$\mathrm{S}_{\mathcal{P}}$ problem becomes the following: given a graph $G = (V, E)$, and $E_1, \ldots, E_m \subseteq \binom{[n]}{2}$, does there exist an $A \in W_{\mathcal{P}}$ such that $A \subseteq E$ and A excludes at least one edge from each E_i? This framework naturally extends to directed graphs and to bipartite graphs.

By Theorem 3, H–$\mathrm{S}_{\mathcal{P}}$ can be analyzed by considering $\bigcup$X–$\mathrm{S}_{\mathcal{P}}$. We do this for the following: Spanning Tree (ST, the property of being connected), Undirected Cycle Cover (UCC, containing an undirected cycle cover), Undirected Path (UPATH, containing an undirected path between s and t), Bipartite Perfect Matching (BPM, having a perfect matching in a bipartite graph), Directed Spanning Tree (DST), Directed Cycle Cover (DCC), and Directed Path (DPATH).

Theorem 4. *In the monotone graph property framework for the hidden model using constraint-index revealing oracle the following properties are* NP*-hard:* ST*,* DST*,* UCC*,* DCC*,* BPM*,* DPATH*,* UPATH*.*

5 Constraint-Index and Variable-Index Revealing Oracle

In this section, we present some applications of our transfer theorem when the index of the constraint and the indices of the variables participating in that constraint are revealed. We consider following CSPs: **Deltas on Triplets** (Δ): $w = 2,\ q = 3$, and $\mathcal{R} = \{R_{abc} : \{0,1\}^3 \to \{\mathrm{T}, \mathrm{F}\} \mid a, b, c \in \{0, 1\}\}$, where $R_{abc}(x, y, z) := (x = a) \wedge (y = b) \wedge (z = c)$; **Hyperplane Non-cover** (HYP−NC): Given a group Z_p^N, the hyperplane non-cover problem is the solvability of a system of homogeneous linear in-equations in Z_p^N; **Arbitrary sets of binary relations on Boolean alphabet,** in particular, the 2-SAT Problem (2SAT); **Exact-Unique Game Problem** (UG$[k]$): Given an undirected graph $G = (V, E)$ and given a permutation $\pi_e : [\![k]\!] \to [\![k]\!]$, for every edge $e \in E$, the goal is to decide if one can assign labels $\alpha_v \in [\![k]\!]$ for every vertex $v \in V$ such that for every edge $e = \{u, v\} \in E$ with $u < v$ we have $\pi_e(\alpha_u) = \alpha_v$;

k-Clique Isomorphism ($k\mathsf{CLQ}$–ISO): Given an undirected graph $G = (V, E)$, does there exist a permutation π on $[n]$ such that: (a) $\forall (i,j) \in E,\ \ \pi(i), \pi(j) \leq k$, (b) $\forall (i,j) \notin E,\ \ \pi(i) > k$ or $\pi(j) > k$; **Polynomial time Solvable Graph Properties** ($\mathcal{P}^{poly}$): The framework for graph properties as defined in Section 4, but with $\{\mathcal{V}\}$-revealing oracle; **Equality to some member in a fixed class of graphs** ($\mathsf{EQ}_{\mathcal{K}}$): For a fixed class $\mathcal{K}$ of graphs on n vertices, we denote by $\mathcal{P}_{\mathcal{K}} : \{0,1\}^{\binom{n}{2}} \to \{T, F\}$ the property of being equal to a graph from $\mathcal{K}$. For example, Equality to k-Clique ($\mathsf{EQ}_{k\mathsf{CLQ}}$), Equality to Hamiltonian Cycle ($\mathsf{EQ}_{\mathsf{HAMC}}$), and Equality to Spanning Tree ($\mathsf{EQ}_{\mathsf{ST}}$).

Theorem 5. *(a) The following problems in the hidden setting with constraint-index and variable-index revealing oracle are in polynomial time:* $\mathsf{2SAT}$*,* $\mathsf{UG}[2]$*,* $\mathcal{P}^{poly}$*,* $\mathsf{EQ}_{\mathsf{ST}}$*. (b) The following problems in the hidden setting with constraint-index and variable-index revealing oracle are* NP*-hard:* Δ*,* $\mathsf{HYP}-\mathsf{NC}$*,* $\mathsf{UG}[k]$ *for* $k \geq 3$*,* $k\mathsf{CLQ}$–ISO*,* $\mathsf{EQ}_{k\mathsf{CLQ}}$*,* $\mathsf{EQ}_{\mathsf{HAMC}}$*.*

Remarks. (1) Polynomial-time solvable graph properties are in P this time, in contrast to the NP-hardness result when only constraint index is revealed (Theorem 4). (2) $\mathsf{UG}[k]$ for $k = 2$ is in P, while for $k \geq 3$ it is NP-hard.

6 Constraint-Index and Relation-Index Revealing Oracle

Theorem 6. *Let* S *be a* CSP *with constant arity and alphabet size* w*. If for every* $\alpha \in [\![w]\!]$*, there is a non-empty relation* $R \in \mathcal{R}$ *such that* $(\alpha, \ldots, \alpha) \notin R$*, then* H–S$_{\{\mathcal{R}\}}$ *is* NP*-hard; otherwise* H–S$_{\{\mathcal{R}\}}$ *is (trivially) in* P*.*

Remark. Under the same conditions H–S$_{\emptyset}$ is NP-hard. As an application, let LINEQ be the CSP in which that alphabet is identified with a finite field F and the ℓ-ary constraints are linear equations over F. Then H–$\mathsf{LINEQ}_{\emptyset}$ is NP-hard.

7 Hidden CSPs with Promise on Instances

In this section we consider an extension of the H–CSP framework where the instances satisfy some property. For the sake of simplicity, we develop this subject only for the constraint index revealing model. Formally, let S be a CSP, and let PROM be a subset of all instances. Then S with *promise* PROM is the CSP S$^{\mathsf{PROM}}$ whose instances are only elements of PROM. One such property is *repetition freeness* where the constraints of an instance are pairwise distinct. We denote by RF the subset of instances satisfying this property. For example $\mathsf{1SAT}^{\mathsf{RF}}$, (as well as H–$\mathsf{1SAT}^{\mathsf{RF}}$) consists of pairwise distinct literals. Such a requirement is quite natural in the context of certain graph problems where the constraints are inclusion (or non-inclusion) of possible edges. The promise H–CSPs framework could also be suitable for discussing certain graph problems on special classes of graphs (e.g, connected graphs, planar graphs, etc.).

We would like to prove an analog of the transfer theorem with promise. Let us be given a promise PROM for the CSP S of type $\mathcal{R} = \{R_1, \ldots, R_s\}$. The corresponding promise $\bigcup\mathsf{PROM}$ for $\bigcup$S is defined quite naturally as follows.

We say that an instance $\mathcal{C} = (C_1, \ldots, C_m)$ of S, where $C_j = R_{k_j}(x_{j_1}, \ldots, x_{j_q})$, is *included* in an instance $\mathcal{C}' = (C'_1, \ldots, C'_m)$ of $\bigcup$S if for every $j = 1, \ldots, m$ $C'_j = R'_j(x_{j_1}, \ldots, x_{j_q})$ for $R'_j \in \bigcup \mathcal{R}$ such that $R_{k_j} \subseteq R'_j$. Then $\bigcup \mathsf{PROM}$ is defined as the set of instances in $\mathcal{C}' \in \bigcup$S which includes some $\mathcal{C} \in \mathsf{PROM}$. In order for the transfer theorem to work, we relax the notion of a solution. A *solution under promise* for $\mathcal{C}' \in \bigcup \mathsf{PROM}$ has to satisfy two criteria: it is a satisfying assignment when $\mathcal{C}'$ includes a satisfiable instance $\mathcal{C} \in \mathsf{PROM}$, and it is EXCEPTION when $\mathcal{C}'$ is unsatisfiable. However, when all the instances $\mathcal{C} \in \mathsf{PROM}$ included in $\mathcal{C}'$ are unsatisfiable but $\mathcal{C}'$ is still satisfiable, it can be either a satisfying assignment or EXCEPTION. We say that an algorithm *solves* $\bigcup \mathrm{S}^{\bigcup \mathsf{PROM}}$ *under promise* if $\forall \mathcal{C}' \in \bigcup \mathsf{PROM}$, it outputs a solution under promise.

Using the above definition in the transfer theorem's proof allows the algorithm for H–$\mathrm{S}_{\{\mathcal{V}\}}$ to terminate, at any moment of time, with the conclusion NO as soon as it gets enough information about the instance to exclude satisfiability and without making further calls to the revealing oracle. In some ambiguous cases, it can still call the oracle with an assignment which satisfies the $\bigcup$S-instance. Other cases when the satisfiability of a $\bigcup$S-instance with promise implies the existence of a satisfiable promise-included instance lack this ambiguity. With these notions the proof of Theorem 1 goes through and we obtain the following.

Theorem 7. *Let* $\mathrm{S}^{\mathsf{PROM}}$ *be a promise* CSP. *(a) If* $\bigcup S^{\bigcup \mathsf{PROM}}$ *is solvable under promise in time T then* H–$\mathrm{S}^{\mathsf{PROM}}_{\{\mathcal{V}\}}$ *is solvable in time* $O((T + s \times \mathrm{comp}(\mathcal{R})) \times m \times \min\{\dim(\bigcup \mathcal{R}), |W_q|\})$. *(b) If* H–$\mathrm{S}^{\mathsf{PROM}}_{\{\mathcal{V}\}}$ *is solvable in time T then* $\bigcup S^{\bigcup \mathsf{PROM}}$ *is solvable under promise in time* $O(T \times m \times \mathrm{comp}(\bigcup \mathcal{R}))$.

We apply Theorem 7 to the following problems: H–$\mathsf{1SAT}^{\mathsf{RF}}_{\emptyset}$, H–$\mathsf{2SAT}^{\mathsf{RF}}_{\emptyset}$, H–$\mathsf{2COL}^{\mathsf{RF}}_{\emptyset}$, and H–$k\mathsf{WEIGHT}^{\mathsf{RF}}_{\emptyset}$. Informally, the problem $k\mathsf{WEIGHT}$ decides if a 0-1 string has Hamming weight at least k, and H–$k\mathsf{WEIGHT}_{\emptyset}$ is NP-hard under the constraint index revealing oracle. Interestingly, in the repetition-free setting, H–$\mathsf{1SAT}^{\mathsf{RF}}_{\emptyset}$ and H–$k\mathsf{WEIGHT}^{\mathsf{RF}}_{\emptyset}$ are in P. On the other hand, H–$\mathsf{2SAT}^{\mathsf{RF}}_{\emptyset}$ and H–$\mathsf{2COL}^{\mathsf{RF}}_{\emptyset}$ are still NP-hard. Finally, we give an alternative proof, via Theorem 7, for [2, Theorem 11], showing NP-hardness of the isomorphism problem of a hidden group (specified by its multiplication table) with a given group.

Acknowledgements. We are grateful to an anonymous referee for various insightful comments and improvements. Most of this work was conducted when the authors were at the Centre for Quantum Technologies (CQT) in Singapore, and partially funded by the Singapore Ministry of Education and the National Research Foundation, also through the Tier 3 Grant "Random numbers from quantum processes". Research partially supported by the European Commission IST STREP project Quantum Algorithms (QALGO) 600700, by the French ANR Blanc program under contract ANR-12-BS02-005 (RDAM project), and by the Hungarian Scientific Research Fund (OTKA), Grant NK105645.

References

1. Bei, X., Chen, N., Dou, L., Huang, X., Qiang, R.: Trial and error in influential social networks. In: KDD, pp. 1016–1024 (2013)
2. Bei, X., Chen, N., Zhang, S.: On the complexity of trial and error. In: STOC, pp. 31–40 (2013)
3. Bei, X., Chen, N., Zhang, S.: Solving linear programming with constraints unknown. CoRR, abs/1304.1247 (2013)
4. Feder, T., Vardi, M.Y.: The computational structure of monotone monadic snp and constraint satisfaction: A study through datalog and group theory. SIAM J. Comput. 28(1), 57–104 (1999)
5. Martin, B.: First-order model checking problems parameterized by the model. In: Beckmann, A., Dimitracopoulos, C., Löwe, B. (eds.) CiE 2008. LNCS, vol. 5028, pp. 417–427. Springer, Heidelberg (2008)
6. Schaefer, T.J.: The complexity of satisfiability problems. In: STOC, pp. 216–226 (1978)

Information Theoretical Cryptogenography*

Sune K. Jakobsen

School of Mathematical Sciences and School of Electronic Engineering and Computer Science, Queen Mary University of London,
Mile End Road, London, E1 4NS, UK
S.K.Jakobsen@qmul.ac.uk

Abstract. We consider problems where n people are communicating and a random subset of them is trying to leak information, without making it clear which people are leaking the information. We introduce a measure of suspicion, and show that the amount of leaked information will always be bounded by the expected increase in suspicion, and that this bound is tight. We ask the question: Suppose a large number of people have some information they want to leak, but they want to ensure that after the communication, an observer will assign probability $\leq c$ to the events that each of them is trying to leak the information. How much information can they reliably leak, per person who is leaking? We show that the answer is $\left(\frac{-\log(1-c)}{c} - \log(e)\right)$ bits.

1 Introduction

The year is 2084 and the world is controlled by a supercomputer called Eve. It makes the laws, carries them out, has surveillance cameras everywhere, can hear everything you say, and can break any kind of cryptography. It was designed to make a world that maximises the total amount of happiness, while still being fair. However, Eve started to make some unfortunate decisions. For example, it thought that to maximise the utility it has been designed to maximise, it must ensure that it survives, so it decided to execute everyone it knew beyond reasonable doubt was trying to plot against Eve (it was designed so it could not punish anyone as long as there is reasonable doubt, and reasonable doubt was defined to be a 5% chance of being innocent). Everyone agrees that Eve should be shut down. The only person who can shut down Eve is Frank who is sitting in a special control room. Eve cannot hurt him, he has access to everything Eve can see, but he needs a password to shut down Eve. 100 people in the world know the password. Eve and Frank have no clue who these people are, only that they exist. If one of them simply says the password, Eve will execute the person. How can they reveal the password, without any of them getting killed?

1.1 Previous Work and Our Results

If we assume standard cryptographic assumptions, or if each pair of people had a private channel, we could use multi-party computation to let one person

* Full version of the paper can be found here: http://arxiv.org/abs/1402.3125

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 676–688, 2014.

reveal information to a group of n people, in such a way that if more than half of them follow the protocol, a computationally bounded observer will only have a negligible advantage when trying to guess which of the collaborating parties originally had the information [4,7]. If we allow Frank to communicate, we could also use steganography [5] to reveal the information to Frank, again only assuming standard cryptographic assumptions and that the observers have bounded computational power.

However, we assume that the observers have unbounded computational power, and that the observers see all messages sent. In that case, we could let every person send random messages. People who know the secret X could make their message correlated with X. For example the messages could be "I think X belongs to the set S". However, every time you make a correct hint about what the secret X is, it will increase the observer's suspicion that you know X. The more precise the hint is or the more unlikely it is that you would give the hint without knowing X, the more useful the statement is to Frank. But such statements would also be the statements that increase Eve's suspicion towards you the most (at least if we assume she knows X). Our main contribution is to introduce a measure of suspicion that captures this, and to show that if you want to leak some amount of information about X in the information theoretical sense, then your suspicion will, in expectation, have to increase by at least the same amount.

The measure of suspicion turns out to be extremely useful for showing upper bounds on how much information you can leak without making it clear that you are leaking. We show that if n people are known to each know X with probability b independently of each other, and no one wants an observer to assign probability more than c to the event that they were leaking information, they can each leak at most $\frac{-b\log(1-c)+c\log(1-b)}{c}n$ bits about X. Using Shannon's Coding Theorem, we show that for all $\epsilon > 0$ there exists n such that if X is uniformly distributed with entropy $\left(\frac{-b\log(1-c)+c\log(1-b)}{c} - \epsilon\right)n$ then n such people can communicate in a way that would enable an observer to guess X with probability $> 1-\epsilon$, but for each person, the observer would still assign probability $\leq c$ to the event that that person was leaking. We show a similar result for the case where the total number of leakers is fixed and known.

The measure of suspicion is also useful for analysing a generalisation of the original cryptogenography (hidden-origin-writing) problem, as introduced in [2]. Here the authors considered a game where one person among n was randomly chosen and given the result of a coin flip. The goal for the n players is to communicate in such a way that an observer, Frank, would guess the correct result of the coin flip, but another observer, Eve, who has the same information would guess wrong when asked who of the n originally knew the result of the coin flip. The main method in [2] is a concavity characterisation, and is very different from the information theory methods we use. We generalise the problem to h bits of information and more players l who have the information, and show that if $h = o(l)$ the winning probability tends to 1 and if $l = o(h)$ it tends to 0.

Finally we show that in general to do cryptogenography, you do not need the non-leakers to collaborate. Instead we can use the fact that people send out random messages anyway, and use this in a similar way to steganography (see [5]). All we need is that people are communicating in a way that involves sufficiently randomness and that they do not change this communication, when we build a protocol on top of that. We can for example assume that they are not aware of the protocol, or they do not care about the leakage.

1.2 Model

We consider problems where one or more players might be trying to leak information about the outcome of a random variable X (see [3] for an introduction to information thoery). The total number of players is denoted n and the players are called $\text{PLR}_1, \dots, \text{PLR}_n$. Sometimes we will use Alice and Bob as names for two of the players. We let L_i be the random variable that is 1 if player i knows the information and 0 otherwise. If there is only one player we write L instead of L_1. The joint distribution of $(X, L_1, \dots, L_n)$ is known to everyone.

All messages are broadcast to all players and to two observers, Eve and Frank. The two observers will have exactly the same information, but we will think of them as two people rather than one. We want to reveal information about X to Frank, while at the same time make sure that for all i, Eve does not get too sure that $L_i = 1$.

A collaborating cryptogenography protocol is a protocol that specifies how each player should choose his messages. The players send messages in rounds, and we assume that the message in round k is sent by player $\text{PLR}_{k \bmod n}$. Formally, a *collaborating cryptogenography protocol* π consists of a number length(π) that specifies how many rounds the protocol consists of, and for each possible transcript of length $<$ length(π) it specifies distributions $p_?$ and $\{p_x\}_{x\in\mathcal{X}}$ (the distributions $p_?$ and $\{p_x\}_{x\in\mathcal{X}}$ depend on π and previous transcript). If player $\text{PLR}_{k \bmod n}$ does not have the information, he chooses a random message using distribution $p_?$, if he knows that $X = x$ he chooses a random message using distribution p_x.

We assume that both Frank and Eve know the protocol, and that they have computational power to compute $(X, L_1, \dots, L_n)|_{T=t}$ for any transcript t. This assumption rules out the use of cryptography.

In some theorems we only care about how much information $I(T; X)$ the transcript T reveals about the secret X, but for cryptogenography to be really useful, we need *reliable* leakage. That is, we want to ensure that given T, Frank can guess X with high probability. Frank's guess must be some function D of the transcript t, so saying that Frank will guess X correct with high probability when $X = x$ is the same as saying that $\Pr(D(T) = x|X = x)$ is close to one.

Definition 1. *Let $L = (L_1, \dots, L_n)$ be a tuple of random variables, where the L_i's take values in $\{0, 1\}$. A* risky (n, h, L, c, ϵ)-protocol *is a collaborating cryptogenography protocol together with a function D from the set of possible transcripts to $\mathcal{X} = \{1, \dots, 2^{\lceil h \rceil}\}$ such that when X and L are distributed independently and X is uniformly distributed on $\mathcal{X}$, then for any $x \in \mathcal{X}$, there* *is probability* $1 - \epsilon$ *that a random transcript* t *distributed as* $T|_{X=x}$ *satisfies* $\forall i : \Pr(L_i = 1|T = t, X = x) \leq c$ *and* $D(t) = x$.

That is, no matter the value of X, with high probability Frank can guess the value of X, and with high probability no player will be estimated to have leaked the information with probability $> c$ by Eve, who knows X. However, there might be a small risk that someone will be estimated to have leaked the information with probability $> c$. This is why we call it a risky protocol.

Definition 2. *A* safe (n, h, L, c, ϵ)-protocol *is a risky* (n, h, L, c, ϵ)*-protocol where* $\Pr(L_i = 1|T = t, X = x) \leq c$ *for all* i, t, x *with* $\Pr(T = t, X = x) > 0$.

Definition 3. *Let* $\text{Fixed}(l, n)$ *be the random variable* $(L_1, \ldots, L_n)$ *that is distributed such that the set* $\{i|L_i = 1\}$ *is uniformly distributed over all subsets of* $\{1, \ldots, n\}$ *of size* l.

A rate R *is* safely/riskily c-achievable for Fixed *if for all* $\epsilon > 0$ *and all* l_0*, there exists a safe/risky* $(n, lR, \text{Fixed}(l, n), c, \epsilon)$*-protocol for some* $l \geq l_0$ *and some* n.

The safe/risky c-capacity for Fixed *is the supremum of all safely/riskily c-achievable rates for* Fixed.

We determine this capacity and also define and determine the capacity for a problem where each person knows X with probability b independently of each other.

Until now we have assumed that all n communicating payers where following a protocol π, but it may not be reasonable to assume that the non-leaking players are collaborating. It turns out that we can do just as well with a weaker assumption. First we need to define what it means to do "as well". All we care about is Frank's and Eve's beliefs about $(X, L_1, \ldots, L_n)$, so we make the following definition.

Definition 4. *Let* $(X, L_1, \ldots, L_n)$ *be distributed on* $\{1, \ldots, 2^{\lceil h \rceil}\} \times \{0, 1\}^n$ *and have full support, and let* π *be a protocol with transcript* T *and* π' *a protocol with transcript* T'. *For a transcript* t *of* π *let* μ_t *denote the distribution* $(X, L_1, \ldots, L_n)|_{T=t}$*, and similar for transcripts* t' *of* π'. *We say that* π *and* π' *are* equivalent *if the distribution of* μ_T *is the same as the distribution of* $\mu_{T'}$.

That is, the probability that the posterior distribution of $(X, L_1, \ldots, L_n)$ is μ has to be the same for both π and π'. It is possible to show that the prior distribution on $(X, L_1, \ldots, L_n)$ does not matter.

Now, what can we do if we cannot dictate what protocol the non-leakers follow? If they do not communicate at all, then it is easy to detect anyone who sends information. However, to do cryptogenography it is enough to assume that all n players communicate innocently using some informative innocent communication protocol (say a protocol for talking about the weather). In an *innocent communication protocol* ι we allow that all players sends a message in each round, and we will assume that ι has infinitely many rounds. A *predicting function* P_i is a function that sends partial transcripts of ι to possible messages of PLR_i in the next round. We say that ι is *informative* if for every player PLR_i and any predicting function P_i, we have $\Pr(\forall k : P_i(T^{k-1}) = T_{k,i}) = 0$, where T^{k-1} is transcript of the first $k-1$ rounds and $T_{k,i}$ is the message send by PLR_i in round k. A collaborating cryptogenography protocol π is called *non-revealing* if any message send by a leaker could have been send by a non-leaker.

Theorem 1. *Let π be a non-revealing collaborating cryptogenography protocol, and let ι be an informative communication protocol. Then there exists a protocol ι^π that is equivalent to π, but where the non-leakers follow the protocol ι.*

Thus, when using ι^π the non-leakers might be talking about the weather and not realise that other players are trying to leak information, but anyone who knows ι^π will get as much information about X and as little information about who is leaking, as if they where all following protocol π. The definition of informative implies that ι and hence ι^π contains infinitely many rounds, but finite truncations of ι^π will give good approximations.

Our main theorem is as follows.

Theorem 2. *The safe and the risky c-capacity for* Fixed *are both $\frac{-\log(1-c)}{c} - \log(e)$. Moreover, these capacities can be achieved when the non-leakers follow any fixed informative communication protocol.*

That is, given ϵ, for sufficiently large k and sufficiently larger n, k leakers in a group of n people can leak a secret of $\left(\frac{-\log(1-c)}{c} - \log(e) - \epsilon\right) k$ bits with success probability $1 - \epsilon$, but if they try to leak $\left(\frac{-\log(1-c)}{c} - \log(e) + \epsilon\right) k$ bits of information the success probability will be bounded away from 1. We give the ideas needed to prove this theorem in Section 3. In this extended abstract we will include some proofs and sketch proofs that are particularly illustrative of our approach, but defer most other proofs to the full version of the paper [6].

1.3 Notation

Unless stated otherwise, all random variables are assumed to be discrete. Random variables are denoted by capital letters and they take values from the set denote by the calligraphic version of that letter (e.g. X takes values from $\mathcal{X}$). The random variable that is the transcript of a protocol will be denoted T, and specific transcripts t. For a tuple a, we let a_i denote the i'th element of a and let a^i denote the tuple $(a_1, \ldots, a_i)$ of the i first elements of a. Similarly for tuples A of random variables. The transcript is a tuple of messages, so for example t^i denotes the transcript of the first i messages. All logarithms are in base 2.

2 Suspicion

In this section we define suspicion and we show that it measures the cost of leaking information. That is, we show that the amount of leaked information is bounded by the increase in expected suspicion, and that this bound is tight.

First we will look at the problem where only one player is communicating and she might be trying to leak information. We will also use these results when we analyse the many-player problem.

Definition 5. *Let Y be a random variable jointly distributed with L. Then the* suspicion (of Alice) given $Y = y$ *is*

$$\mathrm{susp}(Y = y) = -\log(\Pr(L = 0|Y = y)).$$

We see that $\mathrm{susp}(Y = y)$ depends on y and the joint distribution of L and Y, but to keep notation simple, we suppress the dependence on L. The value $\mathrm{susp}(Y = y)$ measures how suspicious Alice looks to someone who knows that $Y = y$ and knows nothing more. For example Y could be the tuple that consists of the secret information X and the current transcript. We can think of the suspicion as the surprisal of the event, "Alice did not have the information".

Definition 6. *The* suspicion (of Alice) given Y *is*

$$\mathrm{susp}(Y) = \sum_{y\in\mathcal{Y}} \Pr(Y = y)\mathrm{susp}(Y = y). \tag{1}$$

For random variables Y, Z we define

$$\mathrm{susp}(Y, Z = z) = \sum_{y\in\mathcal{Y}} \Pr(Y = y|Z = z)\mathrm{susp}((Y, Z) = (y, z)).$$

Here Y and Z can consist of more than one random variable, e.g. $Y = (X, A)$.

The definitions imply that

$$\mathrm{susp}(X, A) = \sum_{a\in\mathcal{A}} \Pr(A = a)\mathrm{susp}(X, A = a).$$

Now we show that suspicion exactly captures the cost of leaking information.

Lemma 1. *If Alice sends a message A, we have*

$$I(X; A) \leq \mathrm{susp}(X, A) - \mathrm{susp}(X). \tag{2}$$

That is, the amount of information she sends about X is at most her expected increase in suspicion given X. There is equality if and only if A and L are independent.

The assumption that Alice sends A means that A and X are independent given $L = 0$, but that is the only restriction on (X, L, A). The lemma is proved by a computation that shows that inequality (2) is equivalent to the fact that the Kullback-Leibler divergence of $A|_{L=0}$'s distribution from A's distribution is non-negative. It is 0 if and only if the two distributions are the same, that is, if A and L are independent. So to have equality in Lemma 1 we only need to ensure that an observer *who does not know* X, does not change her suspicion towards Alice by learning the message A.

We will now turn to the problem where many people are communicating. We break the protocol into time periods were only one person is communicating, and see the entire protocol as a sequence of one-player protocols. The following Corollary shows that a statement similar to Lemma 1 holds for each single message in a protocol with many players.

Corollary 1. *Let* (L, T^{k-1}, X) *have some joint distribution, where* T^{k-1} *denotes previous transcript. Assume that Alice sends the k'th message. Then*

$$I(X; T_k | T^{k-1}) \leq \operatorname{susp}(X, T^k) - \operatorname{susp}(X, T^{k-1}).$$

This is proved by using Lemma 1 on each possible previous transcript.

A protocol consists of a sequence of messages that each leaks some information and increases the suspicion of the sender. We can add up the increases in suspicion, and using the chain rule for mutual information we can also add up the amount of revealed information. However, Bob's message might not only affect his own suspicion, it might also affect Alice's suspicion. To get an upper bound on the amount of information the players can leak, we need to show that one persons message cannot, in expectation, make another persons suspicion decrease. This follows from the next proposition by setting $Y = (X, T^{k-1})$ and $B = T_k$.

Proposition 1. *For any joint distribution on* (L, Y, B) *we have* $\operatorname{susp}(Y) \leq \operatorname{susp}(Y, B)$.

This follows from a convexity property of susp. Let susp_i denote the suspicion of player i.

Theorem 3. *If* T *is the transcript of the entire protocol we have*

$$I(X; T) \leq \sum_{i=1}^{n} \left(\operatorname{susp}_i(X, T) - \operatorname{susp}_i(X)\right).$$

Proof. From the chain rule for mutual information, we know that

$$I(X; T) = \sum_{k=1}^{length(\pi)} I(X; T_k | T^{k-1}).$$

Now Corollary 1 shows that $I(X; T_k | T^{k-1}) \leq \operatorname{susp}_i(X, T^k) - \operatorname{susp}_i(X, T^{k-1})$ if PLR_i send the kth message and Proposition 1 shows that $\operatorname{susp}_{i'}(X, T^k) \geq \operatorname{susp}_{i'}(X, T^{k-1})$ for all other i'. We sum over all the rounds, to get the theorem.

We will now look at the problem where each communicating player wants to ensure that after the leakage, an observer who knows X will assign probability at most c to the event that she was leaking information. If $\Pr(L_1 = 1 | X = x, T = t) \leq c$ for the true value of X, for all i and after all possible transcripts t, we see that $\mathrm{susp}_i(X,T) \leq -\log(1-c)$. If we assume that each player before the protocol had probability $b < c$ of leaking, independently of X, that is $\Pr(L_i|X=x) = b$ for all x and i, we have $\mathrm{susp}_i(X) = -\log(1-b)$. Thus

$$I(X;T) \leq \sum_{i=1}^{n} (\mathrm{susp}_i(X,T) - \mathrm{susp}_i(X)) = (\log(1-c) + \log(1-b))\, n. \qquad (3)$$

To reach this bound, we would need to have $\Pr(L_i = 1|X = x, T = t) = c$ for all x, t, i. But the probability $\Pr(L_i = 1|X = x) = b$ can also be computed as $\mathbb{E}_t \Pr(L_i = 1|X = x, T = t)$, so $\Pr(L_i = 1|X = x, T = t)$ cannot be constantly $c > b$. In fact, the suspicion is a convex function of probabilities $\Pr(L_i = 1|X = x, T = t)$, so suspicion will be maximised subject to $\mathbb{E}_t \Pr(L_i = 1|X = x, T = t) = b$ and $\Pr(L_i = 1|X = x, T = t) \leq c$ when $\Pr(L_i = 1|X = x, T = t)$ only takes the two extreme values 0 and c. This improves the bound as follows.

Theorem 4. *Let π be a collaborating cryptogenography protocol, and T be its transcript. If for all players* PLR_i *and all $x \in \mathcal{X}$ and all transcripts t we have* $\Pr(L_i = 1|X = x) = b$, *and* $\Pr(L_i = 1|T = t, X = x) \leq c$ *then*

$$I(X;T) \leq \frac{-b\log(1-c) + c\log(1-b)}{c} n.$$

3 Reliable Leakage

In this section we present the ideas needed to prove our main theorem. First we consider the following simpler model, as it is easier to analyse but still illustrates many of the ideas needed to prove the main theorem. At the end of this section we present the other ideas in the proof.

Definition 7. *Let* $\mathrm{Indep}_b(n)$ *be the random variable $(L_1, \ldots, L_n)$ where the L_i's are independent, and each L_i is distributed on $\{0,1\}$ and* $\Pr(L_1 = 1) = b$.

A rate R is safely/riskily c-achievable for Indep_b *if for all $\epsilon > 0$ and all n_0, there exists a safe/risky* $(n, nR, \mathrm{Indep}_b(n), c, \epsilon)$*-protocol with $n \geq n_0$.*

The safe/risky c-capacity for Indep_b *is the supremum of all safely/riskily c-achievable rates for* Indep_b.

Notice that while the capacities for Fixed are measured in bits per leaker, the capacities for Indep_b are measured in bits per communicating player.

Theorem 5. *No rate $R > \frac{-b\log(1-c)+c\log(1-b)}{c}$ is safely c-achievable for* Indep_b.

Proof. Assume for contradiction that $R > \frac{-b\log(1-c)+c\log(1-b)}{c}$ is safely c-achievable for Indep_b, and let π be a safe $(n, Rn, \mathrm{Indep}_b(n), c, \epsilon)$-protocol. Let $\delta = R - \frac{-b\log(1-c)+c\log(1-b)}{c}$. We know from Theorem 4 that

$$I(X;T) \leq \frac{-b\log(1-c) + c\log(1-b)}{c} n = (R-\delta)n.$$

Now

$$H(X|T) = H(X) - I(X;T) \geq Rn - (R-\delta)n = \delta n.$$

By Fano's inequality [3, Theorem 2.11.1] we get that the probability of error for Frank's guess is $P_e \geq \frac{\delta n - 1}{nR}$. Thus for sufficiently large n and sufficiently small ϵ we cannot have $P_e \leq \epsilon$. When $P_e > \epsilon$ there must exist an $x \in \mathcal{X}$ such that $\Pr(D(T) \neq x | X = x) > \epsilon$, so R is not safely c-achievable.

This upper bound can be achieved using Shannon's Noisy Coding Theorem.

Theorem 6. *Any rate* $R < \frac{-b\log(1-c)+c\log(1-b)}{c}$ *is safely c-achievable for* Indep_b.

Proof. Let $R < \frac{-b\log(1-c)+c\log(1-b)}{c}$ and let $c' \leq c$ be a number such that $\frac{b(1-c')}{c'(1-b)}$ is rational and $R < \frac{-b\log(1-c')+c'\log(1-b)}{c'}$. Let $d, a \in \mathbb{N}$ be the smallest natural numbers such that $\frac{a}{d} = \frac{b(1-c')}{c'(1-b)}$. We consider the channel that on input $j \in \{1,\ldots,d\}$ with probability b returns a random uniformly distributed element in $\{1+(j-1)a, 2+(j-1)a\ldots, ja\} \mod d$, and with probability $1-b$ it returns a random and uniformly distributed element in $\{1,\ldots,d\}$. A simple computation shows that the capacity of this channel is $\frac{-b\log(1-c')+c'\log(1-b)}{c'}$. We now use Shannon's Noisy-Channel Coding Theorem [8][3] to get an error correcting code $\mathfrak{C} : \mathcal{X} \to \{1,\ldots,d\}^n$ for this channel, that achieves rate R and for each x fails with probability $< \epsilon$. We consider each player to be one use of the channel. When $X = x$ any player that is not leaking will send a message A_i chosen uniformly at random from $\{1,\ldots,d\}$ and any player PLR_i with $L_i = 1$ chooses a message A_i uniformly at random from $\{1+(j-1)a, 2+(j-2)a,\ldots,ja\} \mod d$, where $j = \mathfrak{C}(x)_i$ is the i'th letter in the codeword for x. By assumption about $\mathfrak{C}$, Frank will be able to guess x with probability $1-\epsilon$. Any player PLR_i who sends a message not in $\{1+(j-1)a, 2+(j-2)a,\ldots,ja\} \mod d$, where $j = \mathfrak{C}(x)_i$ cannot be leaking and have $\Pr(L_i = 1|T = t) = 0$. For players who do send a message in $\{1+(j-1)a, 2+(j-2)a,\ldots,ja\} \mod d$ we get

$$\begin{aligned}\Pr(L_i = 1|T = t, X = x) =& \Pr(L_i = 1|A_i = t_i, X_i = x_i)\\ =& \frac{\Pr(A_i = t_i|L_i = 1, X_i = x_i)\Pr(L_i = 1|X_i = x_i)}{\Pr(A_i = t_i|X_i = x_i)}\\ =& \frac{\frac{1}{a}b}{\frac{b}{a}+\frac{1-b}{d}} = c'.\end{aligned}$$

In the first equality we use that given X the random variable (A_i, L_i), is independent from $A_1, L_1, \ldots, A_{i-1}, L_{i-1}, A_{i+1}, L_{i+1}, \ldots, A_n, L_n$. In the second we use Bayes' Theorem. This shows that for each i, $\Pr(L_i = 1|T = t, X = x)$ is either c' or 0 thus $\leq c$.

Together these two theorems give us the safe c-capacity for Indep_b.

Corollary 2. *The safe c-capacity for* Indep_b *is* $\frac{-b\log(1-c)+c\log(1-b)}{c}$.

Using the same idea as in the proof of Theorem 5, you can show that the safe c-capacity for Fixed is at most $\frac{-\log(1-c)}{c} - \log(e)$. However, if we try to use the idea from Theorem 6 to show a lower bound on the c-capacity for Fixed, we have a problem. In the protocol from the proof of Theorem 6 there is a very small risk that only the leakers send messages consistent with knowing $X = x$. This is fine when the L_i's are independent, but when the total number of leakers is fixed and known, it means that we have revealed the leakers. Using the idea from Theorem 6 we can only show that the *risky* c-capacity for Fixed is at least $\frac{-\log(1-c)}{c} - \log(e)$.

To show that the safe and risky c-capacities are the same, we find a way to turn a risky (n, h, L, c, ϵ)-protocol π into a safe $(n, h, L, c', \epsilon f(c, c'))$-protocol π' where $c' > c$ and f is some function. The idea is, to first modify π so it does not have any surprising messages. Then we modify it so if someone could look like they were leaking with probability $> c'$ after the next message, then everyone starts to *pretend ignorance*, that is they send message as if they did not have the information. This ensure that no one will look like they are leaking with probability $> c'$ and as $c' > c$ we can bound the probability that they need to pretend ignorance by $\epsilon f(c, c')$ for some f. This shows that the risky c-capacity is no more than the safe c'-capacity, whenever $c' > c$. By continuity of $\frac{-\log(1-c)}{c} - \log(e)$ both the safe and risky c-capacity for Fixed is $\frac{-\log(1-c)}{c} - \log(e)$.

Finally we need to show that when ι is an informative communication protocol and π a non-revealing collaborating cryptogenography protocol, we can construct a protocol ι^{π} that is equivalent to π but where the non-leakers follows ι. The idea is first to define an *interpretation function* i that sends (partial) transcripts of ι to (partial) transcripts of π. We want to ensure that if PLR_j is following ι, then in the interpretation of the transcript PLR_j behaves as a non-leaker in π. The construction of i uses that ι is informative, which means the PLR_j will need an infinite random string to follow ι, and that we can get a random string from the transcript of ι, and use it to choose messages in π. Now if a leaker PLR_j want to send a message m in π he just send a sequence of messages that gets interpreted as m. To ensure that ι^{π} does not reveal more information than π, the leaker needs to chose the messages using the same distribution as a non-leaker, given that the interpretation is m.

4 The Original Cryptogenography Problem

In [2] the authors studied the following cryptogenographic problem. We flip a coin and tell the result to one out of n people. The $n - 1$ other people do not know who got the information. Formally that means we take $L = (L_1, \ldots, L_n)$ to be the random variable that is uniformly distributed over all $\{0, 1\}$-vectors $(l_1, \ldots, l_n)$ containing exactly one 1 and take X to be uniformly distributed over $\{0, 1\}$ independently from L. We let the group of n people use any collaborating cryptogenography protocol, and afterwards we let Frank guess the result of the coin flip (his guess depends only on the transcript) and then let Eve guess who was leaking (her guess can depend on both transcript and Frank's guess). Eve wins if she guesses the leaker or if Frank does not guess the result of the coin flip. Otherwise Frank and the n people communicating wins. We assume that both Frank and Eve guess to maximise the probability that they win, rather than maximise the probability of being correct.

In [2] it was shown that the probability that the group wins is below 3/4 and for sufficiently high n it is at least 0.5644. In this section we will generalise the problem to a situation were more people are leaking and X contains more information. It is obvious how to generalise X to more information, we simply take X to be uniformly distributed on $\{1, \dots, 2^{\lceil h \rceil}\}$. It is less obvious to generalise to more leakers. When more people are leaking, it would be unreasonable to require Eve to guess all the leakers. If this was the rule, one of the leaking players could just reveal himself as a leaker and say what X is, while the rest of the leakers behave exactly as the non-leakers. Instead we let Eve guess at one person and if that person is leaking, she wins.

Definition 8. *For fixed values of h, number of leakers l and number of communicating players $n > l$ and a collaborating cryptogenography protocol π, we let* $\mathrm{Succ}(h, l, n, \pi)$ *denote the probability that after the players communicate using protocol π, Frank will guess the correct value of X but Eve's guess will not be a leaker. We define*

$$\mathrm{Succ}(h, l) = \sup_{\pi, n}(\mathrm{Succ}(h, l, n, \pi)).$$

In this section we will give results about the asymptotic behaviour of $\mathrm{Succ}(h, l)$ when at least one of l and h tends to infinity. The following result is a consequence of our main theorem.

Theorem 7. *For all $p \in (0, 1)$,*

$$\liminf_{l \to \infty} \mathrm{Succ}\left(\left\lceil\left(\frac{-\log(p)}{1-p} - \log(e)\right) l\right\rceil, l\right) \geq p.$$

Corollary 3. *Let $l \to \infty$ and $h = h(l)$ be a function of l with $h = o(l)$. Then* $\mathrm{Succ}(h, l) \to 1$.

In the other direction we show the following bound.

Lemma 2. *For any $c \in (0, 1)$ and any h, l, n, π, we have* $\mathrm{Succ}(h, l, n, \pi) \leq 1 - \frac{ch + l\log(1-c) + lc\log(e) - c}{h}$.

The idea in the proof is to first modify π to get a protocol π' where $\Pr(L_i = 1 | T^k = t^k, X = x)$ never gets above c. Then Theorem 4 and Fano's inequality gives an upper bound on Frank's probability of being correct. In the cases where π differ from π', we have $\Pr(L_i = 1 | T^k = t^k, X = x) > c$ for some i and k, so on average Eve guesses correctly in these cases with probability at least c.

Theorem 8. *Let $r > 0$ be a real number. Now*

$$\limsup_{l\to\infty} \mathrm{Succ}(\lceil r\log(e)l\rceil, l) \leq \frac{\log(r+1)}{r\log(e)}$$

Proof. Set $c = \frac{r}{r+1}$ and $h = \lceil r\log(e)l\rceil$ in Lemma 2.

Corollary 4. *Let $h \to \infty$ and let $l = l(h)$ be a function of h with $l(h) = o(h)$. Then* $\mathrm{Succ}(h, l) \to 0$.

5 Open Problems

We only considered how much information l players can leak in an asymptotic sense, where l tends to infinity, and the proof of the achievability results is not constructive. We have not tried to find any explicit protocols that work well for specific values of l and ϵ, but that would be an interesting possibility for further research. We assumed that both Eve and Frank knew the true distribution q of $(X, L_1, \ldots, L_n)$. It might be interesting to consider the problem where their beliefs, q_E and q_F are different from q and from each other.

In the setup we considered here, there are two types of players. Some know the information that we want to leak and some do not. We could also imagine that some people know who knows the information, without knowing the information itself, and some could know who knows who knows the information and so on. We could also have people who would know X if it belongs to some set S, and otherwise only know that $X \notin S$. All of this can be described by having a joint distribution $(X, P_1, \ldots, P_n)$ where X is the information we want to leak and P_i is the random variable that player i has as information [1].

A different generalisation would be to have players that try to prevent the leakage by sending misleading information. Such players would also not want to be discovered. If Frank notice that someone is sending misleading information, he could just ignore all the messages sent by that person.

Acknowledgements. I would like to thank my supervisors, Peter Keevash and Søren Riis for valuable discussions about cryptogenography. I also want to thank Jalaj Upadhyay for pointing me to [5], and the referees for their valuable inputs.

References

1. Aumann, R.J.: Interactive epistemology I: Knowledge. International Journal of Game Theory 28(3), 263–300 (1999)
2. Brody, J., Jakobsen, S., Scheder, D., Winkler, P.: Cryptogenography. In: ITCS (2014)
3. Cover, T.M., Thomas, J.A.: Elements of information theory, New York, NY, USA. Wiley-Interscience (1991)

4. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, pp. 218–229. ACM, New York (1987)
5. Hopper, N.J.: Toward a theory of Steganography. PhD thesis, Carnegie Mellon University (2004)
6. Jakobsen, S.: Information theoretical cryptogenography. arXiv:1402.3125 [cs.CR] (2014)
7. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In: STOC, pp. 73–85 (1989)
8. Shannon, C.E.: A mathematical theory of communication. Bell System Technical Journal 27 (1948)

The Complexity of Somewhat Approximation Resistant Predicates

Subhash Khot[1,⋆], Madhur Tulsiani[2,⋆⋆], and Pratik Worah[1,⋆⋆⋆]

[1] NYU
khot@cims.nyu.edu, pworah@cs.uchicago.edu
[2] TTI Chicago
madhurt@ttic.edu

Abstract. A boolean predicate $f : \{0,1\}^k \to \{0,1\}$ is said to be *somewhat approximation resistant* if for some constant $\tau > \frac{|f^{-1}(1)|}{2^k}$, given a τ-satisfiable instance of the $\mathsf{MAX\ k\text{-}CSP}(f)$ problem, it is NP-hard to find an assignment that *strictly beats* the naive algorithm that outputs a uniformly random assignment. Let $\tau(f)$ denote the supremum over all τ for which this holds. It is known that a predicate is somewhat approximation resistant precisely when its Fourier degree is at least 3. For such predicates, we give a characterization of the *hardness gap* $(\tau(f) - \frac{|f^{-1}(1)|}{2^k})$ up to a factor of $O(k^5)$. We show that the hardness gap is determined by two factors:

- The nearest Hamming distance of f to a function g of Fourier degree at most 2, which is related to the Fourier mass of f on coefficients of degree 3 or higher.
- Whether f is monotonically below g.

When the Hamming distance is small and f is monotonically below g, we give an SDP-based approximation algorithm and hardness results otherwise. We also give a similar characterization of the *integrality gap* for the natural SDP relaxation of $\mathsf{MAX\ k\text{-}CSP}(f)$ after $\Omega(n)$ rounds of the Lasserre hierarchy.

1 Introduction

Given a predicate $f : \{0,1\}^k \to \{0,1\}$, an instance of $\mathsf{MAX\ k\text{-}CSP}(f)$ problem consists of n boolean variables and m constraints where each constraint is the predicate f applied on some (ordered) subset of k variables and the variables are allowed to appear in negated form. The goal is to find an assignment to the variables that satisfies maximum number of constraints.

Definition 1. *Given a predicate* $f : \{0,1\}^k \to \{0,1\}$*, define the density* $\rho(f) := \frac{|f^{-1}(1)|}{2^k}$.

[⋆] Work done while the author was at the University of Chicago. Research supported by NSF Expeditions grant CCF-0832795, NSF Waterman Award and BSF grant 2008059.

[⋆⋆] Research supported by NSF Careere Award CCF-1254044.

[⋆⋆⋆] Work done while the author was at the University of Chicago.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 689–700, 2014.

Definition 2. *For a predicate $f : \{0,1\}^k \to \{0,1\}$ and a constant $\tau > \rho(f)$, the predicate is said to be* τ-resistant *if for an arbitrarily small constant $\varepsilon > 0$, it is NP-hard to distinguish instances of MAX k-CSP(f) where a $\tau - \varepsilon$ fraction of constraints can be simultaneously satisfied from those where at most $\rho(f) + \varepsilon$ fraction of the constraints can be simultaneously satisfied.*

A τ-resistant predicate with $\tau = 1$ is more popularly known as *approximation resistant.* There is a substantial body of work on trying to characterize approximation resistant predicates, e.g. [24,21,16,4,3]. A recent survey by Håstad [25] gives a comprehensive overview of many results in this area. In particular, a celebrated result of his [24] shows that the predicates $x \vee y \vee z$ (i.e. 3SAT) and $x \oplus y \oplus z = 0$ (i.e. 3LIN) are approximation resistant. Such predicates have also been studied in the context of unconditional lower bounds and especially in the context of Linear and Semidefinite hierarchies [23,14,27]. In the context of such unconditional lower bounds approximation resistance is synonymous with similar lower bounds in the corresponding proof systems which establishes strong connections with a large body of work in propositional proof complexity [7,6,2,1]. A complete characterization of approximation resistant predicates remains elusive (see [3] however for some progress on this question). In this paper we study a related notion of *somewhat approximation resistance* defined by Håstad [25].

Definition 3. *A predicate $f : \{0,1\}^k \to \{0,1\}$ is said to be* somewhat approximation resistant *if there exists some constant $\tau > \rho(f)$ such that the predicate is τ-resistant.*

Definition 4. *A predicate $f : \{0,1\}^k \to \{0,1\}$ is said to be* always approximable *if for every constant $\tau > \rho(f)$, there is a constant $\nu > \rho(f)$ and a polynomial time (possibly randomized) algorithm that given an instance of MAX k-CSP(f) where a τ fraction of constraints can be simultaneously satisfied, finds an assignment satisfying ν fraction of the constraints.*

Clearly, the terms *somewhat approximation resistant* and *always approximable* are mutually exclusive (assuming P $\neq$ NP). We recall that any predicate $f : \{0,1\}^k \to \{0,1\}$ has a Fourier representation:

$$f(x) = \sum_{\alpha \in \{0,1\}^k} \hat{f}(\alpha)(-1)^{\alpha \cdot x}.$$

The Fourier degree of f is the maximum Hamming weight $|\alpha|$ such that $\hat{f}(\alpha) \neq 0$. We say that f depends on a variable x_i if that variable *appears* in the above representation (i.e. if there exists $\alpha \in \{0,1\}^k$ such that $\alpha_i = 1, \hat{f}(\alpha) \neq 0$). We note that $\rho(f) = \hat{f}(0) = \sum_\alpha \hat{f}(\alpha)^2$. Based on earlier work, the survey [25] (by Håstad) *concluded* the following result:[1]

[1] The survey [25] provides only a sketch of the proofs. The proof details for the first statement in the theorem can be filled in easily. Regarding the second statement, namely that a function of Fourier degree at most 2 can depend on at most 4 variables, the proof implied by Håstad follows from a closer inspection of [19] (also see Section 3.1 in [20]).

Theorem 1. *A predicate f is always approximable if its Fourier degree is at most 2 and somewhat approximation resistant otherwise. Moreover, a function of Fourier degree at most 2 can depend on at most 4 variables.*

In this paper, we focus our attention to the case when f has Fourier degree at least 3 and hence is somewhat approximation resistant.

Definition 5. *Let $f : \{0,1\}^k \to \{0,1\}$ be a predicate with Fourier degree at least 3. Define $\tau(f)$ to be the supremum over all τ such that f is τ-resistant.*

The parameter $\tau(f) - \rho(f)$ may be considered as the *hardness gap*. Our goal is to characterize this gap as closely as possible. As we demonstrate, the gap can be as small as $2^{-\Omega(k)}$ for some predicates. Our main result is a characterization of this gap up to a multiplicative factor of $O(k^5)$. Håstad's result [25] gives a lower bound of $\left(\max_{|\alpha|\geq 3} |\hat{f}(\alpha)|\right)$ on the gap $\tau(f) - \rho(f)$.[2] However we show that this bound is too weak for some predicates and a stronger lower bound is $\Omega\left(\frac{1}{k^2} \cdot \sum_{|\alpha|\geq 3} \hat{f}(\alpha)^2\right)$ (clearly there are predicates, like AND, where the maximum Fourier coefficient at/above level 3 is exponentially far off from the total Fourier mass at/above level 3). However even this bound is not the *correct* one for some predicates and the situation turns out to be a bit subtle. We show that the gap is characterized by two factors:

1. Whether f is *close to* or *far from* the class of functions with Fourier degree at most 2. Not surprisingly, this is related to whether the Fourier mass of f at level 3 and above is *low* or *high*.
2. When f is *close to* some function g with Fourier degree at most 2, whether f is *monotonically below* g.

Note that the upper and lower bound on the gap $\tau(f) - \rho(f)$ correspond to an algorithm and a NP-hardness result respectively. We show that our upper and lower bounds also hold in the Lasserre SDP hierarchy in the following sense: The algorithmic upper bound is achieved by a simple SDP relaxation with one round of the natural Lasserre relaxation. On the other hand, for all lower bound results, there is a $\Omega(n)$-level Lasserre integrality gap construction with integrality gap similar to the NP-hardness gap.

1.1 The Main Result

Let $\mathcal{Q}$ denote the set of boolean functions on k variables which are of Fourier degree at most 2. From Theorem 1, if $f \in \mathcal{Q}$ then f is always approximable and otherwise it is somewhat approximation resistant. We are interested in the case that $f \notin \mathcal{Q}$. Let $\Delta(f, \mathcal{Q})$ denote the minimum Hamming distance (normalized by a factor 2^k so that it is in the range $[0,1]$) of f from any function in $\mathcal{Q}$. We now state our main result.

[2] This is the bound that can be inferred from Håstad's proof sketch.

Theorem 2. *Let k be sufficiently large and let $f : \{0,1\}^k \to \{0,1\}$ be a predicate with Fourier degree at least* 3 *(and hence $\Delta(f, \mathcal{Q}) > 0$).* [3]

1. *If $\Delta(f, \mathcal{Q}) \geq 1/k^3$, then $\tau(f) \geq \rho(f) + \Omega(1/k^5)$.*
2. *If $\Delta(f, \mathcal{Q}) = \delta \leq 1/k^3$, then let $g \in \mathcal{Q}$ denote the unique function such that $\Delta(f, g) = \delta$.*
   - *(a) If $\exists x \in \{0,1\}^k$ such that $f(x) = 1 \wedge g(x) = 0$ then $\tau(f) \geq \rho(f) + \Omega(1/k)$.*
   - *(b) Otherwise, g is* monotonically above *f. In this case, there is an absolute constant C and a polynomial time algorithm that for any $\varepsilon \geq Ck^3\delta$, given a $\rho(f)+\varepsilon$ satisfiable instance of MAX k-CSP(f), finds an assignment that is $\left(\rho(f) + \Omega(\frac{\varepsilon}{k^2 \log(1/\varepsilon)})\right)$-satisfying. In particular,*

$$\tau(f) \leq \rho(f) + O(k^3\delta).$$

Moreover, $\tau(f) \geq \rho(f) + \Omega\left(\frac{\delta}{k^2}\right)$.

Remark 1. We always have the trivial upper bound $\tau(f) - \rho(f) \leq 1$. Hence in all the cases, $\tau(f) - \rho(f)$ is characterized up to a multiplicative factor of $O(k^5)$ as claimed. In Case (2b), δ could be as small as 2^{-k}, and so $\tau(f) - \rho(f)$ is of the same order. Note that characterizing $\tau(f)$ precisely would in particular completely characterize approximation resistant predicates (with $\tau(f) = 1$) which is open even for the case $k = 4$ [16,25]. We believe that even characterizing the gap $\tau(f) - \rho(f)$ with a polylog(k) factor would require significantly new ideas.

Remark 2. Whenever Case (1) applies, we have $\rho(f) \geq \frac{1}{k^3}$ (otherwise f would be $\frac{1}{k^3}$-close to the zero-function which is in $\mathcal{Q}$). The functions $g \in \mathcal{Q}$ depend on at most 4 variables and thus $\rho(g) \in \{\frac{\ell}{16} | \ell \in \{0, 1, \ldots, 16\}\}$. Whenever Case (2) applies $\rho(f)$ is $\frac{1}{k^3}$-close to one of these 17 values.

We can also prove unconditional lower bounds without much extra effort. In this context the notion of NP-hardness is replaced by the notion of the integrality gap which persists even after many levels of Lasserre relaxations.

Definition 6. *Given MAX k-CSP(f), we say that f is* τ^*-resistant *for the* Lasserre hierarchy *if for all constant $\varepsilon > 0$, there exists a constant $c = c(\varepsilon) > 0$ and instances with n variables and m constraints, for infinitely many values of n, such that the Lasserre relaxation after $\lfloor cn \rfloor$ rounds has value at least τ^* but the integral optimum is at most $\rho(f) + \varepsilon$.*

Definition 7. *Let $f : \{0,1\}^k \to \{0,1\}$ be a predicate with Fourier degree at least* 3*. Define $\tau^*(f)$ to be the supremum over all τ^* such that f is τ^*-resistant.*

[3] The current lower bound on k for Theorem 2 is large ($k \geq 2^{2^{15}}$), but it seems to be an artifact of our proof technique and we expect it to hold for smaller values of k. The condition $k \geq 2^{2^{15}}$ arises only in our argument relating $\Delta(f, \mathcal{Q})$ to the Fourier mass of f above level 2 (Section 4).

The two notions of τ-resistance (namely Definition 7 and 5) are very closely related and so we have chosen to use a similar notation for both. Notice that we use a more precise notion of integrality gap which specifies the optimal fractional and integral solution (i.e. the *gap location*) and not just their ratio. Our main result regarding integrality gap mimics the result regarding NP-hardness gap:

Theorem 3. *Let k be sufficiently large and let $f : \{0,1\}^k \to \{0,1\}$ be a predicate with Fourier degree at least 3 (and hence $\Delta(f, \mathcal{Q}) > 0$).*

1. *If $\Delta(f, \mathcal{Q}) \geq 1/k^3$, then $\tau^*(f) \geq \rho(f) + \Omega(1/k^5)$.*
2. *If $\Delta(f, \mathcal{Q}) = \delta \leq 1/k^3$, then let $g \in \mathcal{Q}$ denote the unique function such that $\Delta(f, g) = \delta$.*
   (a) *If $\exists x \in \{0,1\}^k$ such that $f(x) = 1 \wedge g(x) = 0$ then $\tau^*(f) \geq \rho(f) + \Omega(1/k)$.*
   (b) *Otherwise, g is* monotonically above *f. In this case, SDP rounding of the natural Lasserre relaxation, after just one round, finds an assignment that is $\left(\rho(f) + \Omega(\frac{\varepsilon}{k^2 \log(1/\varepsilon)})\right)$-satisfying if the instance is $\rho(f) + \varepsilon$ satisfiable for any $\varepsilon \geq Ck^3\delta$ and C is an absolute constant. In particular,*
   $$\tau^*(f) \leq \rho(f) + O(k^3\delta).$$
   Moreover, $\tau^*(f) \geq \rho(f) + \Omega\left(\frac{\delta}{k^2}\right)$.

We emphasize that our results stated above give a characterization that applies to *all predicates f*. Only very few classfication results of this form, applying to all predicates are known so far. Some notable examples are the dichotomy results of Schaefer [22] and the characterization of CSPs for which there exist "robust" satisfaction algorithms, as conjectured by Guruswami and Zhou [15] and proved recently by Barto and Kozik [5].

1.2 Overview of the Proof

In this section, we provide a brief sketch of proof of Theorem 2, hiding many details however. Our starting point is a recent result of Chan [9] showing that a predicate $L : \{0,1\}^k \mapsto \{0,1\}$ is 1-resistant (i.e. approximation resistant) if $L^{-1}(1)$ is an affine translate of the orthogonal complement of a distance (at least) 3 code. We will call such a predicate a *good* predicate for this section. A useful fact is that sparse good predicates exist (i.e. $|L^{-1}(1)|$ is $O(k^2)$) and are numerous: an affine translate of the orthogonal complement of a random linear subspace of dimension $k - 2\log_2 k - O(1)$ works with probability 99%.

A predicate $f : \{0,1\}^k \to \{0,1\}$ is said to be τ-correlated with a good predicate L if a uniformly random satisfying assignment for L is also a satisfying assignment for f with probability at least τ (i.e. $|L^{-1}(1) \cap f^{-1}(1)|/|L^{-1}(1)| \geq \tau$). Given a predicate $f : \{0,1\}^k \mapsto \{0,1\}$, we observe that if f is τ-correlated with a good predicate, then f is τ-resistant. The reason is rather straightforward. Chan [9] gives a reduction showing that L is 1-resistant. We take the same reduction but pretend that the predicate used for every constraint is f instead of L and this minor modification suffices to show that f is τ-resistant. For the sake of future reference, we note that any predicate f, not identically zero, always $\Omega\left(\frac{1}{k^2}\right)$-correlates with some good predicate. This is simply because we pick an arbitrary good predicate L with $|L^{-1}(1)| \leq O(k^2)$ and translating it if necessary ensure that $L^{-1}(1) \cap f^{-1}(1) \neq \emptyset$. This gives correlation of at least $1/|L^{-1}(1)|$.

With these observations at hand, our first task is to (approximately) characterize the best possible correlation that a given predicate $f : \{0,1\}^k \mapsto \{0,1\}$ can have with a good predicate. We show that this is related to the Fourier mass of f at level 3 and above, denoted $\gamma_3(f)$, which in turn is related to the distance $\Delta(f, \mathcal{Q})$. In the range of parameters of interest, we show that f is τ-correlated with a good predicate (and hence τ-resistant) with

$$\tau \geq \rho(f) + \Omega\left(\frac{\gamma_3(f)}{k^2}\right),$$

and moreover that

$$\gamma_3(f) = \Theta(\Delta(f, \mathcal{Q})).$$

Our lower bound on $\tau(f)$ in Case (1) and Case (2b) of Theorem 2 now follow immediately from the above two claims. The proof of the first claim uses a (somewhat novel) probabilistic argument showing that a random good predicate works. The second claim follows from an unpublished result of Kindler and Safra [18]. We provide another proof, for completeness sake, which uses standard Fourier analytic techniques from the works of KKL and Friedgut [17,12]. The relation above between $\gamma_3(f)$ and $\Theta(\Delta(f, \mathcal{Q}))$ as stated above is similar, though quantitatively incomparable, to a result of Friedgut, Kalai and Naor [13] which relates the Fourier mass above level 1 to the distance from dictator (and constant) functions.

We are now left with the Case (2a) and the upper bound in Case (2b) of Theorem 2. Note that we are in the scenario where there is a function $g \in \mathcal{Q}$ with $\Delta(f, g) = \delta$ for some *tiny* δ.

We illustrate Case (2a) first. For the sake of illustration, assume that $g \equiv 0$, which amounts to saying that $\rho(f) = \delta$. As we noted, f always $\Omega\left(\frac{1}{k^2}\right)$-correlates with a good predicate and hence is $\Omega\left(\frac{1}{k^2}\right)$-resistant. Since $\rho(f) = \delta$ is *tiny*, we have $\tau(f) \geq \rho(f) + \Omega(1/k^2)$ as desired (this is a bit weaker than the bound we actually get/state in Theorem 2). The proof for Case (2a) in general is a bit tricky and we refer the reader to Section 5.1. We note that this is the case where the gap $\tau(f) - \rho(f)$ is *large* (i.e. $\Omega(1/k^2)$) even though the Fourier mass at level 3 and above is at most δ which could be as low as 2^{-k}.

Finally, we arrive at the upper bound in Case (2b). Here the algorithm is designed by using an algorithm of Charikar and Wirth [10] as a black-box. Note that we are in the scenario where there is a function $g \in \mathcal{Q}$ with $\Delta(f, g) = \delta$ for a tiny δ and moreover that f *implies* g. Given an instance of $\mathsf{MAX\ k\text{-}CSP}(f)$ that is $(\rho(f) + \varepsilon)$-satisfiable, we begin by pretending that it is an instance of $\mathsf{MAX\ k\text{-}CSP}(g)$ with the predicate f on every constraint replaced by g. Since f implies g and they are close in Hamming distance, the instance remains $(\rho(g) + \varepsilon/2)$-satisfiable as an instance of $\mathsf{MAX\ k\text{-}CSP}(g)$. For the predicate g of Fourier degree at most 2, the algorithm of Charikar and Wirth yields an assignment that is $(\rho(g) + \Omega(\varepsilon/\log(1/\varepsilon)))$-satisfying. This assignment, by itself, might be quite bad when viewed as an assignment for MAX k-CSP(f). To correct this, we re-randomize each variable with probability $1 - \frac{1}{2k}$ and show that it now serves as a $(\rho(g) + \Omega(1/k^2 \cdot \varepsilon/\log(1/\varepsilon)))$-satisfying assignment to MAX k-CSP(f).

This completes our overview. In the context of Lasserre integrality gaps and Theorem 3, our starting point is a result of the second author [26] that is analogous to Chan's NP-hardness result. The *hardness reductions* are now replaced by *integrality gap constructions*, but they are identical in spirit.

2 Preliminaries

Definition 8. *s We say that a predicate $f : \{0,1\}^k \to \{0,1\}$ τ-correlates with a predicate $g : \{0,1\}^k \to \{0,1\}$ if*

$$\frac{|f^{-1}(1) \cap g^{-1}(1)|}{|g^{-1}(1)|} \geq \tau.$$

Equivalently $\mathbb{E}_{x\in g^{-1}(1)}[f(x)] \geq \tau$.

Definition 9. *A* linear predicate *$L : \{0,1\}^k \to \{0,1\}$ corresponds to set of assignments $L^{-1}(1)$ which form a affine subspace of $\mathbb{F}_2^k$. We call such a predicate* well-distributed *if the uniform distribution on $L^{-1}(1)$ is a balanced pairwise independent distribution on $\{0,1\}^k$ i.e., $\forall i \neq j \in [k], b_1, b_2 \in \{0,1\}, \quad \mathbb{P}_{x\in L^{-1}(1)}[x_i = b_1, x_j = b_2] = 1/4$.*

The following alternate characterization of well-distributed linear predicates is known via (by now) standard arguments (see for example [11]).

Claim. Let $L : \{0,1\}^k \to \{0,1\}$ be a linear predicate such that $L^{-1}(1) = S + z$ for a subspace S of $\mathbb{F}_2^k$ and $z \in \mathbb{F}_2^k$. Then L is well-distributed if and only if $S^\perp$ forms a (linear) code of distance at least 3 over $\{0,1\}^k$.

3 A Relation to Level 3 Fourier Mass

The following theorem shows that a predicate f with *high* Fourier mass at level 3 and above (i.e. *high* value of $\gamma_3(f)$) has a *high* correlation, say τ, with a well-distributed linear predicate. Since a well-distributed linear predicate is 1-resistant, it immediately implies that f is τ-resistant. This argument is used to prove the lower bound on $\tau(f)$ in Case (1) and Case (2b) in Theorem 2.

Theorem 4. *Let $k \geq 16$ and $f : \{0,1\}^k \to \{0,1\}$ be a predicate. There exists*

$$\tau \geq \sqrt{\rho(f)^2 + \frac{\gamma_3(f)}{100k^2}} \tag{3.1}$$

such that f τ-correlates with some well-distributed linear predicate (and hence is τ-resistant as well as τ-resistant for the Lasserre hierarchy).

4 Fourier Spectrum and Closeness to $\mathcal{Q}$

In this section we show that if $\gamma_3(f)$ is sufficiently small then f is close in Hamming distance to a quadratic function $g \in \mathcal{Q}$. In fact the distance $\Delta(f, \mathcal{Q})$ is proportional to $\gamma_3(f)$ whenever $\gamma_3(f) \leq 1/k^3$ (note that we are interested in the case when $\gamma_3(f)$ is polynomially small in $\frac{1}{k}$ which is somewhat atypical situation). This is similar, though incomparable to a result of Friedgut, Kalai and Naor [13] which shows that if $\gamma_2(f)$ is a sufficiently small *constant*, then f is close to a constant function or a dictator (Boolean functions of Fourier degree at most 1). Though our result works for functions of higher Fourier degree, it requires the Fourier mass at higher levels to be polynomially small in $1/k$.

Our Fourier analytic results (Lemma 1 and and its generalization to higher degrees) essentially follow from an earlier unpublished result of Kindler and Safra [18]. In [18], the authors show that if γ_3 (resp. γ_r) is smaller than a constant depending on r, then f is close to real valued junta (cf. also [8]). A close examination of their proof allows one to deduce that f is close to a degree 2 (resp. degree $r-1$) junta. However, for completeness sake, we provide a self-contained proof of the result that we need.

Lemma 1. *Let $f : \{0,1\}^k \mapsto \{0,1\}$ be a predicate such that $\gamma_3(f) \leq 1/k^3$ and $k \geq 2^{2^{15}}$. Then*

$$\gamma_3(f) \leq \Delta(f, \mathcal{Q}) \leq C \cdot \gamma_3(f),$$

for an absolute constant C and $C = 128$ works.

We note that the lower bound above holds because a function $g \in \mathcal{Q}$ has no non-zero Fourier coefficient of degree 3 or more and hence

$$\Delta(f,g) = \|f-g\|_2^2 = \sum_{\alpha} (\hat{f}(\alpha) - \hat{g}(\alpha))^2 \geq \sum_{|\alpha| \geq 3} \hat{f}(\alpha)^2 = \gamma_3(f).$$

Our proof of the upper bound above is similar to those in the papers by Kahn, Kalai, Linial [17] and Friedgut [12].

5 Proof of Main Theorem

In this section, we collect the rest of the pieces required in the proof of Theorem 2. We first show the hardness of approximating $\mathsf{MAX\ k\text{-}CSP}(f)$ when f has good correlation with a well-distributed linear predicate (Lemma 2). Next, we show that $\mathsf{MAX\ k\text{-}CSP}(f)$ is hard to approximate when f is close to a junta g which is *not* monotonically above f (Lemma 3). These two statements suffice to prove the required lower bounds on $\tau(f)$ in Theorem 2 since we can show that f must have the appropriate correlation with a well-distributed linear predicate in cases (1) and (2b), and must be close to a $g \in \mathcal{Q}$ in case (2a). Finally, we give an approximation algorithm for the case when f is close to a $g \in \mathcal{Q}$ and $g \geq f$.

5.1 Reductions from the Hardness of Approximating Well-Distributed Linear Predicates

We now give the reductions from Chan's result [9] on the hardness of approximating well-distributed linear predicates. His result shows that a well-distributed linear predicate $L : \{0,1\}^k \to \{0,1\}$ is 1-resistant, even on $\mathsf{MAX\ k\text{-}CSP}(L)$ instances with certain uniformity properties. These properties concern what the assignments to n variables look when restricted to the k variables in a randomly chosen constraint from the instance. Recall that for an instance Φ of $\mathsf{MAX\ k\text{-}CSP}(L)$, a constraint $C \in \Phi$ is of the form $L(x_{i_1} + b_{i_1}, \ldots, x_{i_k} + b_{i_k})$. Let $\mathbf{x}_C$ denote the tuple $(x_{i_1}, \ldots, x_{i_k})$ of the variables in the constraint C and Let $\mathbf{b}_C$ denote the tuple $(b_{i_1}, \ldots, b_{i_k})$. Also, for an assignment $A : [n] \to \{0,1\}$, let $A(\mathbf{x}_C)$ denote $(A(x_{i_1}), \ldots, A(x_{i_k}))$. The following follows easily from the statement of Theorem 5.4 and the proof of Theorem 1.1 in [9].

Theorem 5 ([9]). *Let $k \geq 3$ and let $\eta, \varepsilon > 0$ be arbitrarily small constants. $L : \{0,1\}^k \to \{0,1\}$ be a well-distributed linear predicate. Then, given an instance Φ of $\mathsf{MAX\ k\text{-}CSP}(L)$ on variables $x_1, \ldots, x_n$, it is NP-hard to distinguish between the following two cases:*

Yes: *There exists an assignment $A : [n] \to \{0,1\}$ satisfying $1 - \eta$ fraction of the constraints. In fact, for any $z \in L^{-1}(1)$*

$$\frac{1-\eta}{|L^{-1}(1)|} \leq \underset{C \in \Phi}{\mathbb{P}}[A(\mathbf{x}_C) + \mathbf{b}_C = z] \leq \frac{1+\eta}{|L^{-1}(1)|}.$$

No: *For all assignments $A : [n] \to \{0,1\}$ and all $z \in \{0,1\}^k$, we have*

$$\frac{1-\varepsilon}{2^k} \leq \underset{C \in \Phi}{\mathbb{P}}[A(\mathbf{x}_C) + \mathbf{b}_C = z] \leq \frac{1+\varepsilon}{2^k}.$$

Thus, the theorem states that in the Yes case, not only are most constraint satisfied, but the tuple $A(\mathbf{x}_C) + \mathbf{b}_C$ looks almost uniformly distributed over $L^{-1}(1)$, over the choice of a random constraint $C \in \Phi$. On the other hand, in the No case, $A(\mathbf{x}_C) + \mathbf{b}_C$ looks almost uniformly distributed over *all of* $\{0,1\}^k$. In particular, this means that the fraction of satisfied constraints is at most $\left|L^{-1}(1)\right| / 2^k + \varepsilon$.

Given the above theorem, it is easy to prove that a predicate f which correlates with some well-distributed linear predicate must also be hard to approximate.

Lemma 2. *Let $k \geq 3$ and let $\eta, \varepsilon > 0$ be arbitrarily small constants. Let $f : \{0,1\}^k \to \{0,1\}$ be a predicate which τ-correlates with some well-distributed linear predicate L. Then, given an instance Φ of $\mathsf{MAX\ k\text{-}CSP}(f)$ on variables $x_1, \ldots, x_n$, it is NP-hard to distinguish between the following cases:*

Yes: *There exists an assignment $A : [n] \to \{0,1\}$ satisfying $(1 - \eta) \cdot \tau$ fraction of the constraints.*

No: *All assignments $A : [n] \to \{0,1\}$ satisfy at most $(1 + \varepsilon) \cdot \rho(f)$ fraction of the constraints.*

Now we consider the case when $f : \{0,1\}^k \to \{0,1\}$ is close to a function g with Fourier degree at most 2, but is not monotonically dominated by it i.e., when $f^{-1}(1) \cap g^{-1}(0) \neq \emptyset$. We show that such an f is $(\rho(f) + \Omega(1/k))$-resistant. In fact we prove the statement below for any function g which is a junta depending only on s variables. This is sufficient because Theorem 1 implies that a Boolean function g of degree 2 must depend on at most 4 of the k variables and hence the required result will follow easily.

Lemma 3. *Let k, s be such that $k - s \geq 3$ and let $\varepsilon > 0$ be an arbitrarily small constant. Let $g : \{0,1\}^k \to \{0,1\}$ be an s-junta and let $f : \{0,1\}^k \to \{0,1\}$ be a predicate such that $\Delta(f,g) = \delta \leq 1/(2^{s+2} \cdot k)$ and $f^{-1}(1) \cap g^{-1}(0) \neq \emptyset$. Then, given an instance Φ of* MAX k-CSP(f) *on variables $x_1, \ldots, x_n$, it is NP-hard to distinguish between the following two cases:*

Yes: *There exists an assignment $A : [n] \to \{0,1\}$ satisfying $\rho(f) + 1/(2^{s+3} \cdot k)$ fraction of the constraints.*

No: *All assignments $A : [n] \to \{0,1\}$ satisfy at most $(1+\varepsilon) \cdot \rho(f)$ fraction of the constraints in Φ.*

5.2 Proofs of Lower Bounds on $\tau(f)$

We can now prove the lower bounds on $\tau(f)$ in Theorem 2.

- Case 1: $\Delta(f, \mathcal{Q}) \geq 1/k^3$ implies by Lemma 1 that $\gamma_3 \geq 1/(12k^3)$. Theorem 4 then gives that f is τ-correlated with some well-distributed predicate for
$$\tau \geq \sqrt{\rho(f)^2 + \Omega\left(\frac{1}{k^5}\right)} \geq \rho(f) + \Omega(1/k^5)\,.$$
Lemma 2 then implies that f is τ-resistant and hence $\tau(f) \geq \rho(f) + \Omega(1/k^5)$.
- Case 2b: In this case, Lemma 1 gives $\gamma_3 \geq \Omega(\delta)$ and Theorem 4 again gives that f must τ-correlate with a well-distributed linear predicate for
$$\tau \geq \sqrt{\rho(f)^2 + \Omega\left(\frac{\delta}{k^2}\right)} \geq \rho(f) + \Omega(\delta/k^2)$$
As before, an application of Lemma 2 completes the proof.
- Case 2a: In this case $\Delta(f,g) = \delta \leq 1/k^3$ for some $g \in \mathcal{Q}$, which must be a 4-junta by Theorem 1. Then, if $f^{-1}(1) \cap g^{-1}(0) \neq \emptyset$, Lemma 3 gives that $\tau(f) \geq \rho(f) + \Omega(1/k)$.

5.3 An SDP Rounding Algorithm

We now provide an SDP rounding algorithm based on the algorithm by Charikar and Wirth [10] to prove the upper bound in case (2b) of Theorem 2. For f and δ as in case (2b), and Φ which is a $(\rho(f)+\varepsilon)$-satisfiable instance of MAX k-CSP(f), the algorithm below yields a non-trivial approximation when $\varepsilon = \Omega(k^3 \cdot \delta)$. This gives $\tau(f) \leq \rho(f) + O(k^3 \cdot \delta)$.

Håstad [25] observed that algorithm of Charikar and Wirth [10], which rounds an SDP relaxation for maximizing a *homogeneous* quadratic objective function, can in fact be used for approximating $\mathsf{MAX\ k\text{-}CSP}(g)$ for any $g : \{0,1\}^k \to \{0,1\}$ which has Fourier degree at most 2 (by rounding the standard SDP relaxation). This observation gives the following lemma.

Lemma 4. *Let $g \in \mathcal{Q}$. Then there exists a randomized polynomial time algorithm for rounding the standard SDP relaxation of MAX k-CSP(g), which given an instance Φ with SDP value $\rho(g) + \varepsilon$, outputs an assignment A satisfying at least $\rho(g) + \frac{c \cdot \varepsilon}{\log(1/\varepsilon)}$ fraction of the constraints in expectation. Here c is an absolute constant.*

We now proceed to the main theorem for this section. The proof will essentially replace an instance Φ of $\mathsf{MAX\ k\text{-}CSP}(f)$ by an appropriate instance Φ_g of $\mathsf{MAX\ k\text{-}CSP}(g)$ and use the algorithm in Lemma 4 to find an assignment A_g for Φ_g. Our assignment for Φ will be obtained from A_g by a simple transformation which trades-off the approximation factor to avoid the bad situation where A_g ends up falsifying many constraints in Φ while still satisfying many constraints in Φ_g. We show the following:

Theorem 6. *Let $f : \{0,1\}^k \to \{0,1\}$ be a predicate such that there exists another predicate $g \in \mathcal{Q}$ satisfying $g \geq f$ and $\Delta(f, g) = \delta \leq 1/k^3$. Then there exists a randomized polynomial time algorithm, which given an instance of MAX k-CSP(f) in which $\rho(f) + \varepsilon$ fraction of constraints can be satisfied for $\varepsilon = \Omega(k^3 \cdot \delta)$, finds an assignment such that $\mathbb{E}_A\left[\mathsf{val}_\Phi(A)\right] \geq \rho(f) + \frac{c \cdot \varepsilon}{8k^2 \log(\frac{1}{\varepsilon})}$.*

Acknowledgements. We thank Ryan O'Donnell for help with references. We thank the anonymous referees for their useful comments and for pointing out the connection of our Fourier Analytic Lemma with the unpublished result of Kindler and Safra. We also thank the referess for pointing out a bug in the previous proof of Lemma 3 and for suggesting a fix.

References

1. Alekhnovich, M., Ben-Sasson, E., Razborov, A.A., Wigderson, A.: Pseudorandom generators in propositional proof complexity. SIAM J. Comput. 34(1), 67–88 (2005)
2. Alekhnovich, M., Razborov, A.: Lower Bounds for Polynomial Calculus: Non-Binomial Case. In: FOCS, pp. 190–199 (2001)
3. Austrin, P., Khot, S.: A Characterization of Approximation Resistance for Even k-Partite CSPs (2012) (manuscript)
4. Austrin, P., Mossel, E.: Approximation Resistant Predicates from Pairwise Independence. Computational Complexity 18, 249–271 (2009)
5. Barto, L., Kozik, M.: Robust satisfiability of constraint satisfaction problems. In: STOC, pp. 931–940 (2012)
6. Ben-Sasson, E., Impagliazzo, R.: Random CNFs are Hard for the Polynomial Calculus. Computational Complexity 19(4), 501–519 (2010)
7. Ben-Sasson, E., Wigderson, A.: Short Proofs are Narrow – Resolution Made Simple. J. ACM 48(2), 149–169 (2001)

8. Bourgain, J.: On the Distribution of the Fourier Spectrum of Boolean Functions. Israel J. of Math., 269–276 (2002)
9. Chan, S.O.: Approximation Resistance from Pairwise Independent Subgroups. Electronic Colloquium on Computational Complexity (ECCC) 19, 110 (2012)
10. Charikar, M., Wirth, A.: Maximizing Quadratic Programs: Extending Grothendieck's Inequality. In: FOCS, pp. 54–60 (2004)
11. Chor, B., Goldreich, O., Håstad, J., Friedman, J., Rudich, S., Smolensky, R.: The Bit Extraction Problem of t-Resilient Functions. In: FOCS, pp. 396–407 (1985)
12. Friedgut, E.: Boolean Functions With Low Average Sensitivity Depend On Few Coordinates. Combinatorica 18(1), 27–35 (1998)
13. Friedgut, E., Kalai, G., Naor, A.: Boolean Functions whose Fourier Transform is Concentrated on the First Two Levels. Advances in Applied Mathematics 29(3), 427–437 (2002)
14. Georgiou, K., Magen, A., Tulsiani, M.: Optimal Sherali-Adams Gaps from Pairwise Independence. In: APPROX-RANDOM, pp. 125–139 (2009)
15. Guruswami, V., Zhou, Y.: Tight bounds on the approximability of almost-satisfiable horn sat and exact hitting set. In: SODA, pp. 1574–1589 (2011)
16. Hast, G.: Beating a Random Assignment. PhD thesis, Royal Institute of Technology, Sweden (2005)
17. Kahn, J., Kalai, G., Linial, N.: The Influence of Variables on Boolean Functions. In: FOCS, pp. 68–80 (1988)
18. Kindler, G., Safra, S.: Noise-Resistant Boolean-Functions are Juntas (2004), `http://www.cs.huji.ac.il/$\sim$gkindler/papers/Papers.htm`
19. Nisan, N., Szegedy, M.: On the degree of boolean functions as real polynomials. Computational Complexity 4, 301–313 (1994)
20. O'Donnell, R.: Lecture notes for analysis of Boolean functions (2012), `http://analysisofbooleanfunctions.org`
21. Samorodnitsky, A., Trevisan, L.: A PCP Characterization of NP with Optimal Amortized Query Complexity. In: STOC, pp. 191–199 (2000)
22. Schaefer, T.J.: The complexity of Satisfiability Problems. In: STOC, pp. 216–226 (1978)
23. Schoenebeck, G.: Linear Level Lasserre Lower Bounds for Certain k-CSPs. In: FOCS, pp. 593–602 (2008)
24. Håstad, J.: Some Optimal Inapproximability Results. J. of the ACM, 798–859 (2001)
25. Håstad, J.: On the Efficient Approximability of Constraint Satisfaction Problems. In: Surveys in Combinatorics, vol. 346, pp. 201–222. Cambridge University Press (2007)
26. Tulsiani, M.: CSP gaps and Reductions in the Lasserre Hierarchy. In: STOC, pp. 303–312 (2009)
27. Tulsiani, M., Worah, P.: LS+ Lower Bounds from Pairwise Independence. Electronic Colloquium on Computational Complexity (ECCC) 19, 105 (2012)

Approximate Nonnegative Rank Is Equivalent to the Smooth Rectangle Bound

Gillat Kol[1,⋆], Shay Moran[2,⋆⋆], Amir Shpilka[3,⋆⋆⋆], and Amir Yehudayoff[4,†]

[1] School of Mathematics, Institute for Advanced Study, Princeton NJ
gillat.kol@gmail.com
[2] Departments of Computer Science and Mathematics, Technion, Israel
shaymoran1@gmail.com
[3] Department of Computer Science, Technion, Haifa, Israel
shpilka@cs.technion.ac.il
[4] Department of Mathematics, Technion, Haifa, Israel
amir.yehudayoff@gmail.com

Abstract. We consider two known lower bounds on randomized communication complexity: The smooth rectangle bound and the logarithm of the approximate nonnegative rank. Our main result is that they are the same up to a multiplicative constant and a small additive term.

The logarithm of the nonnegative rank is known to be a nearly tight lower bound on the deterministic communication complexity. Our result indicates that proving the analogue for the randomized case, namely that the log approximate nonnegative rank is a nearly tight bound on randomized communication complexity, would imply the tightness of the information cost bound.

Another corollary of our result is the existence of a boolean function with a quasipolynomial gap between its approximate rank and approximate nonnegative rank.

1 Introduction

In this work we are mainly interested in understanding two useful techniques that were developed for proving lower bounds on randomized communication

[⋆] Partially supported by NSF grant CCF-0832797 and the Weizmann Institute of Science National Postdoctoral Award Program for Advancing Women in Science. Part of this work was done while the author was visiting the Simons Institute for the Theory of Computing, Berkeley CA, and the Technion, Israel.

[⋆⋆] The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 257575.

[⋆⋆⋆] The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 257575.

[†] Horev fellow – supported by the Taub foundation. Research is also supported by ISF and BSF.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 701–712, 2014.

complexity: The smooth rectangle bound [JK10] and the approximate nonnegative rank (see Section 1.3 for both definitions). Our main result is that although these two techniques are seemingly different, the lower bounds that may be derived from them are, more or less, equivalent. As a consequence, we are able to apply previous results regarding the smooth rectangle bound to get new results about the approximate nonnegative rank, thus providing information about two of the open problems in [Lee12] (see Corollaries 1 and 2).

We next survey the relevant lower bounds methods for randomized communication complexity. Here and below, f is a boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$. We denote by $\mathsf{D}(f)$ the deterministic communication complexity of f, and by $R_\epsilon(f)$ the randomized private coin[1] communication complexity of f with error ϵ. These and other basic definitions can be found in [KN97].

1.1 Randomized Communication Complexity Lower Bounds

Nonnegative Rank. A well-known linear algebraic lower bound on the deterministic communication complexity of f is[2] $\log \mathsf{rank}(M_f)$, where M_f is the $2^n \times 2^n$ boolean matrix given by $M_f(x,y) = f(x,y)$ [MS82]. The long standing log-rank conjecture asserts that this bound is tight, up to a polynomial overhead.

*Conjecture 1 (**log-rank conjecture, Lovász and Saks** [LS88]).* For every function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, it holds that[3]

$$\mathsf{D}(f) \leq \operatorname{polylog}(\mathsf{rank}(M_f)) .$$

Yannakakis [Yan91] introduced the notion of *nonnegative rank* to communication complexity. We say that a real matrix M is nonnegative if all its entries are nonnegative. The nonnegative rank of a nonnegative real matrix M, denoted $\mathsf{rank}^+(M)$, is the minimum natural number r such that M is the sum of r nonnegative rank-1 matrices.

The nonnegative rank is clearly at least as large as rank, that is, $\mathsf{rank}^+(M) \geq \mathsf{rank}(M)$. The nonnegative rank can be arbitrarily larger than the rank, if we allow non-boolean matrices. Indeed, for every $k \in \mathbb{N}$ there exists a matrix M such that $\mathsf{rank}(M) = 3$ and $\mathsf{rank}^+(M) \geq k$ (see [BL09]). However, if we restrict our attention to boolean matrices then no such separation between rank and rank^+ is known. The best known separation for boolean matrices is quasipolynomial[4]. This separation follows from the best known separation between the logarithm of the rank and the communication complexity

[1] For simplicity, we only consider private coin protocols. However, all the results carry over to the public coin model via Newman's Theorem [New91].

[2] Here and below rank is over the real numbers.

[3] In this text, logarithms are base two.

[4] There exists a sequence of boolean matrices $\{M_n\}_{n=1}^{n=\infty}$ such that $\log \mathsf{rank}^+(M_i) = \Omega((\log \mathsf{rank}(M_i))^\alpha)$ for some constant $\alpha > 1$.

(see discussion following Corollary 2). Moreover, determining the dependency between rank and rank^+ for boolean matrices is equivalent to solving the log-rank conjecture in communication complexity (see Theorem 1).

While we are still far from proving the log-rank conjecture (the best result in this direction [Lov13] is that $\mathsf{D}(f)$ is at most roughly $\sqrt{\mathsf{rank}(M_f)}$), a variant of the conjecture obtained by replacing the rank by the nonnegative rank is known to hold.

Theorem 1 (Log Nonnegative Rank Theorem, Lovász [Lov90]). *For every function* $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$*, it holds that*[5]

$$\mathsf{D}(f) \leq \left(\log \mathsf{rank}^+(M_f) + 1\right)\left(\log \mathsf{rank}(M_{1-f}) + 1\right). \tag{1}$$

In particular, $\mathsf{D}(f) \leq O(\log^2 \mathsf{rank}^+(M_f))$.

We study a randomized analogue of Equation (1). In the randomized setting, the notion of nonnegative rank needs to be altered to an approximate one. The ϵ-approximate nonnegative rank of a matrix M, denoted $\mathsf{rank}^+_\epsilon(M)$, is the minimum nonnegative rank of a $2^n \times 2^n$ nonnegative matrix M' so that $\|M - M'\|_\infty \leq \epsilon$, i.e., $|M(x,y) - M'(x,y)| \leq \epsilon$ for all $x, y \in \{0,1\}^n$. The ϵ-approximate rank of a matrix M, denoted $\mathsf{rank}_\epsilon(M)$, is defined similarly.

Again, clearly $\mathsf{rank}_\epsilon(M) \leq \mathsf{rank}^+_\epsilon(M)$. One can also prove, using the separation discussed earlier between rank and rank^+, that for every $k \in \mathbb{N}$ there exists a matrix M and $\epsilon > 0$ such that $\mathsf{rank}_\epsilon(M) \leq 3$ and $\mathsf{rank}^+_\epsilon(M) \geq k$. However, not much is known about the relation between rank_ϵ and rank^+_ϵ for boolean matrices.

It was shown by [Kra96] that $\mathsf{R}_\epsilon(f) \geq \log \mathsf{rank}^+_\epsilon(M_f)$. We consider the following conjecture asserting that this bound is nearly tight, as in the deterministic case:

Conjecture 2 ***(Log Approximate Nonnegative Rank Conjecture, See also TH8 in [Lee12]).*** For every sufficiently small constant $0 < \epsilon < 1$ and every function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$,

$$\mathsf{R}_\epsilon(f) \leq \text{polylog}\left(\mathsf{rank}^+_\epsilon(M_f) \cdot \mathsf{rank}^+_\epsilon(M_{1-f})\right).$$

We will later relate this conjecture to an open problem regarding the compression of communication protocols.

The Smooth Rectangle Bound. A different approach for proving lower bounds on randomized communication complexity, which we refer to as the "rectangle based method", is based on bounding from below the weight of the largest (almost) monochromatic combinatorial rectangle.

The smooth rectangle bound, suggested by [JK10], is a rectangle based method shown to be a stronger lower bound than many of the previous methods (for example, the rectangle/corruption bound, the discrepancy bound, and the γ_2 approach [LS07]). Informally speaking, the smooth rectangle bound for a

[5] We use $1 - f$ to denote the boolean function $(1 - f)(x, y) = 1 - f(x, y)$.

function f with error ϵ, denoted $\mathsf{srec}^1_\epsilon(f)$, considers assignments of weights (nonnegative real values) to combinatorial rectangles (sets of the form $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}, \mathcal{Y} \subseteq \{0,1\}^n$) satisfying:

1. For inputs $(x, y) \in f^{-1}(1)$, the total weight assigned to rectangles containing (x, y) is between $1 - \epsilon$ and 1.
2. For inputs $(x, y) \in f^{-1}(0)$, the total weight assigned to rectangles containing (x, y) is at most ϵ.

The value $\mathsf{srec}^1_\epsilon(f)$ is the logarithm of the minimum total weight assigned to all rectangles by any such assignment. A formal definition of the smooth rectangle bound can be found in Section 1.3.

Information Cost. Another recent lower bound method is based on *information cost*. The information cost of a function f with error ϵ, denoted $\mathsf{IC}_\epsilon(f)$, measures the amount of information the players must learn about each other's input while executing any protocol that computes f, with error at most ϵ [CSWY01, Bra12]. For a formal definition of $\mathsf{IC}_\epsilon(f)$, see e.g. Definition 2.1 in [KLL$^+$12]. The information cost is known to lower bound the randomized public coin communication complexity of f [BR11]. The other direction, namely whether every function with information cost I has a randomized protocol with communication complexity I (a "compressed" protocol), is yet another open problem in communication complexity [Bra12]. We state here a somewhat weaker conjecture than Open Problem 1 of [Bra12]:

*Conjecture 3 (**Weak Compression Conjecture**).* For every sufficiently small $0 < \epsilon < 1$ and every function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$,

$$\mathsf{R}_\epsilon(f) \leq \mathrm{poly}\left(\mathsf{IC}_\epsilon(f), \log(n), 1/\epsilon\right).$$

It was recently shown by [KLL$^+$12] that the information cost bound is at least as powerful as almost all the rectangle methods. This was done by showing that the relaxed partition bound is always (roughly) at most the information cost. It is easily seen (by comparing the corresponding linear-programs) that for boolean functions, the relaxed partition bound corresponds to a two-sided smooth rectangle bound, defined as the maximum between the smooth rectangle bound of f and of $1 - f$. In fact, prior to our work, the logarithm of the approximate nonnegative rank was one of the few bounds not known to be weaker than the information cost.

1.2 Our Results

Our main result is the following theorem showing that the smooth rectangle bound is almost equivalent to the logarithm of the approximate nonnegative rank.

Theorem 2 (Main). *For every* $0 < \epsilon < \frac{1}{10}$ *and a function* $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$,

$$\mathsf{srec}^1_{3\epsilon}(f) \le \log \mathsf{rank}^+_\epsilon(M_f) \le 2\mathsf{srec}^1_{\epsilon/2}(f) + \log(12n/\epsilon^2).$$

Furthermore, an additive $\log(n/\epsilon)$ *term on the right hand side is needed.*

Theorem 2 is proved in Section 2. Next, we give several corollaries of this theorem.

The Log Approximate Nonnegative Rank Conjecture and Compression. One corollary is that proving the log approximate nonnegative rank conjecture (Conjecture 2) would imply that the information cost bound is nearly tight (Conjecture 3). Formally, we prove the following corollary.

Corollary 1. *There exists a constant* $c > 0$ *such that for every sufficiently small* $0 < \epsilon < 1$,

$$\mathsf{IC}_\epsilon(f) \ge c \cdot \epsilon^2 \left(\log \mathsf{rank}^+_{4\epsilon}(M_f) - \log(3n/8\epsilon^2)\right) - 1.$$

The proof of this corollary can be found in the full version of this paper, [KMSY14].

Separating the approximate rank and the approximate nonnegative rank. It follows that the approximate nonnegative rank of the negation of the disjointness function on n bit-strings, denoted NDISJ_n, is quasipolynomial in its approximate rank, thus addressing Problem `TH9` in [Lee12]: For a small constant $\epsilon > 0$, [Raz02] proved that $\mathsf{rank}_\epsilon(\mathsf{NDISJ}_n) \le 2^{O(\sqrt{n})}$ (see also the discussion after Conjecture 42 in [LS09b]), while $\mathsf{srec}^1_{3\epsilon}(\mathsf{NDISJ}_n) \ge \Omega(n)$. By Theorem 2, $\mathsf{rank}^+_\epsilon(\mathsf{NDISJ}_n) \ge 2^{\Omega(n)}$.

Corollary 2. *If* $0 < \epsilon < 1$ *is a sufficiently small constant then for every* $n \in \mathbb{N}$,

$$\log \mathsf{rank}^+_\epsilon(M_{\mathsf{NDISJ}_n}) \ge \Omega\left(\log^2 \mathsf{rank}_\epsilon(M_{\mathsf{NDISJ}_n})\right).$$

We mention that in the non-approximate case, any gap greater than quasipolynomial between the rank and nonnegative rank will disprove the log-rank conjecture as $\mathsf{D}(f) \ge \log \mathsf{rank}^+(M_f)$. The best known gap is only $\mathsf{D}(f) \ge \Omega\left((\log \mathsf{rank}(M_f))^\alpha\right)$ for $\alpha = \log_3(6) < 2$ (Kushilevitz (unpublished), cf. [NW95]).

New upper bound on deterministic complexity. Theorem 1 implies $\mathsf{D}(f) \le O\left(\log \mathsf{rank}^+(M_f) \cdot \log \mathsf{rank}(M_f)\right)$. By combining Theorem 2 with results from [GL13] and [JK10], we devise a similar bound using the (potentially smaller) approximate nonnegative rank instead of the nonnegative rank. Thus, in order to prove the log-rank conjecture it is enough to show that $\log \mathsf{rank}(M_f) \le$ polylog$(\mathsf{rank}^+_\epsilon(M_f) + \mathsf{rank}^+_\epsilon(M_{1-f}))$.

Corollary 3. *For every function* $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$,

$$\mathsf{D}(f) \le O(\log(\mathsf{rank}^+_{1/18}(M_f) + \mathsf{rank}^+_{1/18}(M_{1-f})) \cdot \log^2 \mathsf{rank}(f)).$$

The proof of this corollary can be found in the full version of this paper, [KMSY14].

Open Problems. Consider the following stronger version of the log approximate nonnegative rank conjecture (Conjecture 2), asserting that

$$\mathsf{R}_\epsilon(f) \le \text{polylog}\left(\mathsf{rank}^+_\epsilon(M_f)\right).$$

This conjecture, if true, has two simple corollaries, that describe properties of boolean matrices and are of independent interest. One concerns the behavior of the approximate nonnegative rank when negating f, namely, that $\log \mathsf{rank}^+_\epsilon(M_{1-f})$ is at most polynomial in $\log \mathsf{rank}^+_\epsilon(M_f)$. Observe that the approximate rank satisfies this property as $\mathsf{rank}(M_{1-f}) \le \mathsf{rank}(M_f) + 1$, and so does the nonnegative rank as $\mathsf{D}(1-f) = \mathsf{D}(f)$ and $\log \mathsf{rank}^+(M_f) \le \mathsf{D}(f) \le O(\log^2 \mathsf{rank}^+(M_f))$. A second corollary concerns error reduction, namely, that we have the bound $\log \mathsf{rank}^+_\epsilon(M_f) \le (\log \mathsf{rank}^+_{1/3}(M_f))^{O(\log(1/\epsilon))}$. The approximate rank was shown to satisfy this property (it actually satisfies the stronger property $\mathsf{rank}_\epsilon(M_f) \le (\mathsf{rank}_{1/3}(M_f))^{O(\log(1/\epsilon))}$, see [Alo03, LS09a]). Both of the above corollaries are still open, and one may wish to study either of them prior to the log approximate nonnegative rank conjecture. In Section 3 we show that the method used in [Alo03, LS09a] to prove error reduction for approximate rank cannot work for the approximate nonnegative rank.

1.3 Definitions

We conclude the introduction by giving the needed formal definitions.

Definition 1 (nonnegative rank). *Let $M \in \mathbb{R}^{n\times m}$ be a matrix. M is* nonnegative *if $M(x,y) \ge 0$ for every x, y. M has* rank one *if it is of the form $M = v \otimes u$, where $v \in \mathbb{R}^n, u \in \mathbb{R}^m$ and $\otimes$ denotes tensor product (that is, $M(x,y) = v(x)u(y)$ for every x, y).*

The nonnegative rank *of a nonnegative matrix M is*

$$\mathsf{rank}^+(M) = \min\left\{ r \in \mathbb{N} \;:\; M = \sum_{i=1}^{r} M_i, \quad \forall\, i \quad M_i \textit{ is nonnegative and of rank one}\right\}.$$

The ϵ-approximate nonnegative rank *of M is*

$$\mathsf{rank}^+_\epsilon(M) = \min\left\{\mathsf{rank}^+(M') \;:\; M' \textit{ is nonnegative, } \|M - M'\|_\infty \le \epsilon\right\}.$$

Definition 2 (Smooth Rectangle Bound). *For $0 \le \epsilon < \frac{1}{2}$ and a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, the (one)* smooth rectangle bound $\mathsf{srec}^1_\epsilon(f)$ *is the logarithm of the value of the following linear program. Below, R ranges over combinatorial rectangles (sets of the form $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}, \mathcal{Y} \subseteq \{0,1\}^n$) and $R(x,y)$ is the indicator for the event $(x,y) \in R$.*

$$\begin{aligned} \min \sum_R w_R \;:\; & \forall (x,y) \in f^{-1}(1) :\; 1 - \epsilon \le \sum_R w_R R(x,y) \le 1, \\ & \forall (x,y) \in f^{-1}(0) :\; \sum_R w_R R(x,y) \le \epsilon, \\ & \forall R :\; w_R \ge 0. \end{aligned}$$

2 Proving the Equivalence, Theorem 2

The "$\mathsf{srec}^1 \le \log \mathsf{rank}^+$*" direction.* We start by upper bounding the smooth rectangle bound by the logarithms of the approximate nonnegative rank. Let $r = \mathsf{rank}^+_\epsilon(M_f)$. Let $M'' \in (\mathbb{R}^+)^{2^n \times 2^n}$ be the promised nonnegative matrix satisfying $\|M_f - M''\|_\infty \le \epsilon$ and $\mathsf{rank}^+(M'') = r$. Observe that the entries of M'' are bounded by $1 + \epsilon$. It will be convenient for us to consider the matrix $M' = \frac{1}{1+\epsilon}M''$ whose entries are bounded by 1. Observe that still $\|M_f - M'\|_\infty \le 1 - \frac{1-\epsilon}{1+\epsilon} \le 2\epsilon$ and $\mathsf{rank}^+(M') = r$. Let $M_1, \ldots, M_r \in (\mathbb{R}^+)^{2^n \times 2^n}$ be nonnegative rank-1 matrices so that $M' = \sum_{t=1}^r M_t$.

Fix $t \in [r]$ for now. Write $M_t = v \otimes u$ for two nonnegative vectors $v, u \in (\mathbb{R}^+)^{2^n}$. We may assume without loss of generality that

$$\|v\|_\infty, \|u\|_\infty \le 1, \tag{2}$$

as u, v can always be converted to such vectors for the following reason: Let $a = v(i)$ be the maximum entry in v, and let $b = u(j)$ be the maximum entry in u. Assume without loss of generality that $b \ge a$. It holds that $1 \ge M_t(i,j) = v(i)u(j) = ab$. If $b \le 1$ we are done. Otherwise, $b > 1$, and we replace v by bv and u by $\frac{1}{b}u$. Observe that now both vectors have entries in the interval $[0,1]$ (as $a \le 1/b$), and that $(bv) \otimes \left(\frac{1}{b}u\right) = v \otimes u = M_t$.

Let $K = \lceil \frac{2r}{\epsilon} \rceil$. For an integer $1 \le k \le K$, define the vector v_k in the following way: For $i \in [2^n]$, set $v_k(i) = 1/K$ if $v(i) \ge k/K$, and $v_k(i) = 0$ if $v(i) < k/K$. Define u_k similarly. Let

$$v' = \sum_{k \in [K]} v_k \quad \text{and} \quad u' = \sum_{k \in K} u_k.$$

It holds that $\|v - v'\|_\infty, \|u - u'\|_\infty \le 1/K$, as e.g. v' rounds v to the nearest integer multiple of $1/K$ from below. Let

$$M'_t = v' \otimes u' = \left(\sum_{k \in [K]} v_k\right) \otimes \left(\sum_{k' \in K} u_{k'}\right) = \sum_{k,k' \in [K]} v_k \otimes u_{k'}.$$

Using Equation (2),

$$\begin{aligned}\|M_t - M'_t\|_\infty &\le \max_{i,j \in [2^n]} \{v(i)u(j) - v'(i)u'(j)\} \\ &\le \max_{i,j \in [2^n]} \left\{v(i)u(j) - (v(i) - \tfrac{1}{K})(u(j) - \tfrac{1}{K})\right\} \\ &\le \tfrac{1}{K} \max_{i,j \in [2^n]} \{v(i) + u(j)\} \le \tfrac{2}{K}.\end{aligned}$$

Thus, we approximated M_t (with error $2/K$) by a sum of at most K^2 rectangles, each of weight $1/K^2$.

By summing over all $t \in [r]$,

$$\left\| M' - \sum_{t=1}^{r} M'_t \right\|_\infty \leq \frac{2r}{K} \leq \epsilon.$$

Thus,

$$\left\| M_f - \sum_{t=1}^{r} M'_t \right\|_\infty \leq 3\epsilon.$$

We approximated M_f (with error 3ϵ) by a sum of at most $K^2 r$ rectangles, each of weight $1/K^2$. Furthermore, for every (x, y),

$$\sum_{t=1}^{r} M'_t(x, y) \leq \sum_{t=1}^{r} M_t(x, y) = M'(x, y) \leq 1.$$

Thus, the total weight of rectangles containing (x, y) is at most 1. This means that

$$\mathsf{srec}^1_{3\epsilon}(f) \leq \log(K^2 r \cdot 1/K^2) = \log(r).$$

The "$\log \mathsf{rank}^+ \leq \mathsf{srec}^1$*" direction.* Next we show that the logarithm of the approximate nonnegative rank is not much larger than the smooth rectangle bound. Let W be such that $\mathsf{srec}^1_\epsilon(f) = \log W$, and let (w_R) be weights for rectangles satisfying the conditions of the linear program defining the smooth rectangle bound, for which $\sum_R w_R = W$. Similarly to [LLR12], we consider the probability distribution on rectangles μ, defined by $\mu(R) = w_R/W$ for all R. For every (x, y), let

$$e_{x,y} = \mathop{\mathbf{E}}_{R \sim \mu}[R(x, y)] = \sum_R \frac{w_R}{W} R(x, y),$$

where we recall that $R(x, y)$ is the indicator for $(x, y) \in R$. We get

$$|f(x, y) - W e_{x,y}| = |f(x, y) - \sum_R w_R R(x, y)| \leq \epsilon. \tag{3}$$

In other words, when R is selected according to μ, the value $W \cdot R(x, y)$ is a good estimation for $f(x, y)$.

Let $R_1, \ldots, R_k$ be independent samples from μ for $k = \lceil 2W^2 n/\epsilon^2 \rceil$. For every (x, y), Hoeffding's bound implies that

$$\Pr\left[\left|\frac{1}{k}\sum_{t=1}^{k} R_t(x, y) - e_{x,y}\right| \geq \epsilon/W\right] \leq 2e^{-2\epsilon^2 k/W^2},$$

where the probability is taken over the (independent) choices of $R_1, \ldots, R_k$. By the union bound, since $2e^{-2\epsilon^2 k/W^2} 2^{2n} < 1$, there is a choice of $R_1, \ldots, R_k$ so that for all (x, y),

$$\left|\frac{1}{k}\sum_{t=1}^{k} R_t(x, y) - e_{x,y}\right| < \epsilon/W. \tag{4}$$

Define

$$M' = \frac{W}{k} \sum_{t=1}^{k} R_t.$$

The nonnegative rank of M' is at most k. By Equations (3) and (4), for all (x, y),

$$|f(x,y) - M'(x,y)| \leq |f(x,y) - We_{x,y}| + W \left| e_{x,y} - \frac{1}{k} \sum_{t=1}^{k} R_t(x,y) \right| \leq 2\epsilon.$$

Therefore, $\|M - M'\|_\infty \leq 2\epsilon$. This means that

$$\log \mathsf{rank}^+_{2\epsilon}(M_f) \leq \log k \leq 2 \log W + \log(3n/\epsilon^2).$$

The additive $\log(n/\epsilon)$ *term is needed.* The additive $\log(n/\epsilon)$ term on the right hand side of Theorem 2 must be there as the following example shows. Let f be the equality function, that is, M_f is the $2^n \times 2^n$ identity matrix. In [Alo03, Alo09], it was shown that $\mathsf{rank}_\epsilon(M_f) \geq \Omega\left(\frac{n}{\epsilon^2 \log(1/\epsilon)}\right)$. Obviously the same lower bound holds for the ϵ-approximate nonnegative rank of M_f.

We claim that $\mathsf{srec}^1_\epsilon(f)$ is at most $\log(1/\epsilon)$. Let p_R be the distribution on rectangles of the form $R = A \times A$ defined by: Each x is in A with probability ϵ independently of other x's. Let $w_R = p_R/\epsilon$. For every (x, y), if $f(x, y) = 1$ (i.e., $x = y$), then $\mathbf{E}[R(x,y)] = \Pr[x \in A] = \epsilon$ and so $\sum_R w_R R(x,y) = 1$. If $f(x,y) = 0$, then $\mathbf{E}[R(x,y)] = \Pr[x \in A] \Pr[y \in A] = \epsilon^2$ and so $\sum_R w_R R(x,y) = \epsilon$. So w_R is a solution to the above linear program, and the corresponding value is $\sum_R w_R = 1/\epsilon$.

3 No Monotone Error Reduction

Part of the argument in [Alo03] concerns error reduction for approximate rank. It is shown that for a boolean function f, if $\mathsf{rank}_{1/3}(M_f) \leq r$ then $\mathsf{rank}_\epsilon(M_f) \leq r^{O(\log(1/\epsilon))}$. For the nonnegative case, we may ask an even easier question: Is it true that

$$\mathsf{rank}^+_\epsilon(M_f) \leq \left(\mathsf{rank}^+_{1/3}(M_f) + \mathsf{rank}^+_{1/3}(M_{1-f})\right)^{O(\log(1/\epsilon))}? \tag{5}$$

The argument in [Alo03] is based on the following observation: There is a univariate polynomial p of constant degree d, so that for every $b \in \{0, 1\}$ and for every x so that $|x - b| < 1/3$ we have $|p(x) - b| < |x - b|/2$. In other words, the polynomial p contracts around zero and around one. The error reduction follows by observing that if we point-wise apply p to the matrix approximating M_f, we get a better approximation of M_f while the rank is increased by at most a power of d.

We show that this method cannot work in the nonnegative case, in the sense that we cannot replace the polynomial p by a nonnegative polynomial.

Specifically, Proposition 1 states that a bivariate polynomial p with certain properties does not exist. Before stating the proposition we demonstrate how one could have used such a polynomial p (should it exist) to perform nonnegative error reduction.

Assume for simplicity that f is so that both $\mathsf{rank}^+_{1/4}(M_f)$ and $\mathsf{rank}^+_{1/4}(M_{1-f})$ are 1. Let M'_0 and M'_1 be nonnegative matrices of rank 1 that are (1/4)-close to M_f and M_{1-f}, respectively. Assume that we are given a bivariate polynomial $p : [0,1] \times [0,1] \to \mathbb{R}^+$ with nonnegative coefficients and finite degree so that for every $(x, y) \in [3/4, 1] \times [0, 1/4]$ we have $|1 - p(x, y)| \leq (1 - x)/2$, and for every $(x, y) \in (0, 1/4] \times [3/4, 1]$ we have $p(x, y) < x/2$. It is not hard to verify that M', defined as

$$M'(x, y) = p(M'_0(x, y), M'_1(x, y)),$$

gives a nonnegative (1/8)-approximation for M_f of constant nonnegative rank.

Proposition 1. *There is no bivariate polynomial $p : [0,1] \times [0,1] \to \mathbb{R}^+$ with nonnegative coefficients so that the followings hold: for every $(x, y) \in [3/4, 1] \times [0, 1/4]$ we have $|p(x, y) - 1| \leq 1 - x$, and for every $(x, y) \in (0, 1/4] \times [3/4, 1]$ we have $p(x, y) < x$.*

In other words, a nonnegative polynomial that contracts around $x = 1$ must be expanding around $x = 0$. Notice that we do not restrict the degree of p in the proposition above. We mention that on the line $y = 1 - x$, there is a such a polynomial p of degree four with positive coefficients (e.g. the Bernstein approximation of the step function).

Proof (Proof of Proposition 1). Assume towards a contradiction that such a polynomial p exists. Write

$$p(x, y) = yg(x, y) + h(x),$$

where g and h are polynomials with nonnegative coefficients. By assumption,

$$\forall\ 3/4 \leq x \leq 1 :\ |p(x, 0) - 1| = |h(x) - 1| \leq 1 - x. \tag{6}$$

In addition, for $0 < x \leq 1/4$ it holds that $p(x, 1) = g(x, 1) + h(x) < x$, so

$$\forall\ 0 < x \leq 1/4 :\ h(x) < x. \tag{7}$$

We claim that no such h exists. Indeed, by the above $h(1) = 1$ and $h(1/4) < 1/4$. Since h has positive coefficients, it is convex on the ray of positive real numbers, which implies

$$h(3/4) = h(1/3 \cdot 1/4 + 2/3 \cdot 1) \leq 1/3 \cdot h(1/4) + 2/3 \cdot h(1) < 1/3 \cdot 1/4 + 2/3 = 3/4.$$

This is a contradiction to that $|h(3/4) - 1| \leq 1 - 3/4$.

References

[Alo03] Alon, N.: Problems and results in extremal combinatorics–i. Discrete Mathematics 273(1-3), 31–53 (2003)

[Alo09] Alon, N.: Perturbed identity matrices have high rank: Proof and applications. Combinatorics, Probability & Computing 18(1-2), 3–15 (2009)

[BL09] Beasley, L.B., Laffey, T.J.: Real rank versus nonnegative rank. Linear Algebra and its Applications 431(12), 2330–2335 (2009); Special Issue in honor of Shmuel Friedland

[BR11] Braverman, M., Rao, A.: Information equals amortized communication. In: FOCS, pp. 748–757 (2011)

[Bra12] Braverman, M.: Interactive information complexity. In: STOC, pp. 505–524 (2012)

[CSWY01] Chakrabarti, A., Shi, Y., Wirth, A., Yao, A.C.-C.: Informational complexity and the direct sum problem for simultaneous message complexity. In: FOCS, pp. 270–278 (2001)

[GL13] Gavinsky, D., Lovett, S.: En route to the log-rank conjecture: New reductions and equivalent formulations. Electronic Colloquium on Computational Complexity (ECCC) 20, 80 (2013)

[JK10] Jain, R., Klauck, H.: The partition bound for classical communication complexity and query complexity. In: IEEE Conference on Computational Complexity, pp. 247–258 (2010)

[KLL$^+$12] Kerenidis, I., Laplante, S., Lerays, V., Roland, J., Xiao, D.: Lower bounds on information complexity via zero-communication protocols and applications. In: FOCS, pp. 500–509 (2012)

[KMSY14] Kol, G., Moran, S., Shpilka, A., Yehudayoff, A.: Approximate nonnegative rank is equivalent to the smooth rectangle boundy. Electronic Colloquium on Computational Complexity (ECCC) 21, 46 (2014)

[KN97] Kushilevitz, E., Nisan, N.: Communication complexity. Cambridge University Press (1997)

[Kra96] Krause, M.: Geometric arguments yield better bounds for threshold circuits and distributed computing. Theoretical Computer Science 156(1&2), 99–117 (1996)

[Lee12] Troy Lee's homepage. Some open problems around nonnegative rank (2012), `http://www.research.rutgers.edu/~troyjlee/open_problems.pdf`

[LLR12] Laplante, S., Lerays, V., Roland, J.: Classical and quantum partition bound and detector inefficiency. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012, Part I. LNCS, vol. 7391, pp. 617–628. Springer, Heidelberg (2012)

[Lov90] Lovász, L.: Communication complexity: A survey. In: Path, Flows, and VLSI-Networks, pp. 235–265 (1990)

[Lov13] Lovett, S.: Communication is bounded by root of rank. CoRR, abs/1306.1877 (2013)

[LS88] Lovász, L., Saks, M.E.: Lattices, möbius functions and communication complexity. In: FOCS, pp. 81–90 (1988)

[LS07] Linial, N., Shraibman, A.: Lower bounds in communication complexity based on factorization norms. In: STOC, pp. 699–708 (2007)

[LS09a] Lee, T., Shraibman, A.: An approximation algorithm for approximation rank. In: IEEE Conference on Computational Complexity, pp. 351–357 (2009)

[LS09b] Lee, T., Shraibman, A.: Lower bounds in communication complexity. Foundations and Trends in Theoretical Computer Science 3(4), 263–398 (2009)

[MS82] Mehlhorn, K., Schmidt, E.M.: Las vegas is better than determinism in vlsi and distributed computing (extended abstract). In: STOC, pp. 330–337 (1982)

[New91] Newman, I.: Private vs. common random bits in communication complexity. Information Processing Letters 39(2), 67–71 (1991)

[NW95] Nisan, N., Wigderson, A.: On rank vs. communication complexity. Combinatorica 15(4), 557–565 (1995)

[Raz02] Razborov, A.A.: Quantum communication complexity of symmetric predicates. Izvestiya of the Russian Academy of Science, Mathematics 67, 2003 (2002)

[Yan91] Yannakakis, M.: Expressing combinatorial optimization problems by linear programs. Journal of Computer and System Sciences 43(3), 441–466 (1991)

Distance Oracles for Time-Dependent Networks*

Spyros Kontogiannis[1,2] and Christos Zaroliagis[2,3]

[1] Dept. of Comp. Science & Engineering, U. Ioannina, 45110 Ioannina, Greece
kontog@cs.uoi.gr

[2] Computer Technology Institute & Press "Diophantus", 26504 Patras, Greece

[3] Dept. of Comp. Engineering & Informatics, U. Patras, 26504 Patras, Greece
zaro@ceid.upatras.gr

Abstract. We present the first approximate distance oracle for sparse directed networks with *time-dependent* arc-travel-times determined by continuous, piecewise linear, positive functions possessing the FIFO property. Our approach precomputes $(1 + \varepsilon)-$approximate distance summaries from selected landmark vertices to all other vertices in the network, and provides two sublinear-time query algorithms that deliver constant and $(1+\sigma)-$approximate shortest-travel-times, respectively, for arbitrary origin-destination pairs in the network. Our oracle is based only on the sparsity of the network, along with two quite natural assumptions about travel-time functions which allow the smooth transition towards asymmetric and time-dependent distance metrics.

1 Introduction

Distance oracles are succinct data structures encoding shortest path information among a carefully selected subset of pairs of vertices in a graph. The encoding is done in such a way that the oracle can *efficiently answer* shortest path queries for arbitrary origin-destination pairs, exploiting the preprocessed data and/or *local* shortest path searches. A distance oracle is exact (resp. approximate) if the returned distances by the accompanying query algorithm are exact (resp. approximate). A bulk of important work (e.g., [22,21,17,18,23,24,2]) is devoted to constructing distance oracles for *static* (i.e., *time-independent*), mostly undirected networks in which the arc-costs are fixed, providing trade-offs between the oracle's space and query time and, in case of approximate oracles, also of the stretch (maximum ratio, over all origin-destination pairs, between the distance returned by the oracle and the actual distance). For an overview of distance oracles for static networks, the reader is deferred to [20] and references therein.

In many real-world applications, however, the arc costs may vary as functions of time (e.g., when representing travel-times) giving rise to *time-dependent* network models. A striking example is route planning in road networks where the

* Full version available at [14]. This work was supported by EU FP7/2007-2013 under grant agreements no. 288094 (eCOMPASS) and no. 609026 (MOVESMART), and partially done while both authors were visiting the Karlsruhe Inst. of Technology.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 713–725, 2014.

travel-time for traversing an arc $a = uv$ (modelling a road segment) depends on the temporal traffic conditions while traversing uv, and thus on the departure time from its tail u. Consequently, the optimal origin-destination path may vary with the departure-time from the origin. Apart from the theoretical challenge, the time-dependent model is also much more appropriate with respect to the historic traffic data that the route planning vendors have to digest, in order to provide their customers with fast route plans. For example, TomTom's *LiveTraffic* service provides real-time estimations of average travel-time values, collected by periodically sampling the average speed of each road segment in a city, using the connected cars to the service as sampling devices. The crux is how to exploit all this historic traffic information in order to provide *efficiently* route plans that will adapt to the departure-time from the origin. Towards this direction, we consider the continuous, piecewise linear (pwl) interpolants of these sample points as *arc-travel-time functions* of the corresponding instance.

Computing a time-dependent shortest path for a triple (o, d, t_o) of an origin o, a destination d and a departure-time t_o from the origin, has been studied long time ago (see e.g., [4,11,16]). The shape of arc-travel-time functions and the waiting policy at vertices may considerably affect the tractability of the problem [16]. A crucial property is the *FIFO property*, according to which each arc-arrival-time at the head of an arc is a *non-decreasing* function of the departure-time from the tail. If *waiting-at-vertices* is forbidden and the arc-travel-time functions may be non-FIFO, then subpath optimality and simplicity of shortest paths is not guaranteed. Thus, (even if it exists) an optimal route is not computable by well known techniques (Dijkstra or Bellman-Ford) [16]. Additionally, many variants of the problem are also $\mathbf{NP}-$hard [19]. On the other hand, if arc-travel-time functions possess the FIFO property, then the problem can be solved in polynomial time by a straightforward variant of Dijkstra's algorithm (**TDD**), which relaxes arcs by computing the arc costs "on the fly", when scanning their tails. This has been first observed in [11], where the *unrestricted waiting policy* was (implicitly) assumed for vertices, along with the non-FIFO property for arcs.

The FIFO property may seem unreasonable in some application scenarios, e.g., when travellers at the dock of a train station wonder whether to take the very next slow train towards destination, or wait for a subsequent but faster train. Our motivation in this work stems from *route planning* in urban-traffic road networks where the FIFO property seems much more natural: Cars are assumed to travel according to the same (possibly time-dependent) average speed in each road segment, and overtaking is not considered as an option. Additionally, when shortest-travel-times are well defined and optimal waiting-times at nodes always exist, a non-FIFO arc with *unrestricted-waiting-at-tail* policy is equivalent to a FIFO arc in which waiting at the tail is useless [16]. Therefore, our focus in this work is on networks with FIFO arc-travel-time functions.

Until recently, most of the previous work on the time-dependent shortest path problem concentrated on computing an optimal origin-destination path providing the earliest-arrival time at destination when departing at a *given* time from the origin, and neglected the computational complexity of providing succinct representations of the entire earliest-arrival-time *functions*, for *all* departure-times from the origin. Such representations, apart from allowing rapid answers to several queries for selected origin-destination pairs but for varying departure times, would also be valuable for the construction of *distance summaries* (a.k.a. *route planning maps*, or *search profiles*) from central vertices (e.g., *landmarks* or *hubs*) towards other vertices in the network, providing a crucial ingredient for the construction of distance oracles to support real-time responses to arbitrary queries $(o, d, t_o) \in V \times V \times \mathbb{R}$.

The complexity of succinctly representing earliest-arrival-time functions was first questioned in [5,7,6], but was solved only recently in [13] which, for FIFO-abiding pwl arc-travel-time functions, showed that the problem of succinctly representing such a function for a *single origin-destination pair* has space-complexity $(1 + K) \cdot n^{\Theta(\log n)}$, where n is the number of vertices and K is the total number of breakpoints (or legs) of all the arc-travel-time functions. Polynomial-time algorithms (or even PTAS) for constructing *point-to-point* approximate distance functions are provided in [13,8]. Such approximate distance functions possess *succinct representations*, since they require only $\mathcal{O}(1 + K)$ breakpoints per origin-destination pair. It is also easy to verify that K could be substituted by the number K^* of *concavity-spoiling* breakpoints of the arc-travel-time functions (i.e., breakpoints at which the arc-travel-time slopes increase).

To the best of our knowledge, the problem of providing distance oracles for time-dependent networks with *provably* good approximation guarantees, small preprocessing-space complexity and sublinear time complexity, has not been investigated so far. Due to the hardness of providing succinct representations of exact shortest-travel-time functions, the only realistic alternative is to use approximations of these functions for the distance summaries that will be preprocessed and stored by the oracle. Exploiting a PTAS (such as that in [13]) for computing approximate distance functions, one could provide a trivial oracle with query-time complexity $Q \in \mathcal{O}(\log\log(K^*))$, at the cost of an exceedingly high space-complexity $S \in \mathcal{O}\big((1 + K^*) \cdot n^2\big)$, by storing succinct representations of all the point-to-point $(1 + \varepsilon)-$approximate shortest-travel-time functions. At the other extreme, one might use the minimum possible space complexity $S \in \mathcal{O}(n + m + K)$ for storing the input, at the cost of suffering a query-time complexity $Q \in \mathcal{O}(m + n\log(n)[1 + \log\log(1 + K_{\max})])$ (i.e., respond to each query by running **TDD** in real-time using a predecessor search structure for evaluating pwl functions)[1]. The main challenge considered in this work is to smoothly close the gap between these two extremes, i.e., to achieve a better (e.g., *sublinear*) query-time complexity, while consuming smaller space-complexity (e.g., $\mathrm{o}\big((1 + K^*) \cdot n^2\big)$) for succinctly representing travel-time *functions*, and enjoying a small (e.g., close to 1) approximation guarantee.

We present the *first* approximate distance oracle for sparse directed graphs with time-dependent arc-travel-times, which achieves all these goals. Our oracle is based only on the sparsity of the network, plus two assumptions of travel-time functions which are quite natural for route planning in road networks

[1] $K_{\max}$ denotes the maximum number of breakpoints in an arc-travel-time function.

(cf. Assumptions 1 and 2 in Section 2). It should be mentioned that: (i) even in static undirected networks, achieving a stretch factor below 2 using subquadratic space and sublinear query time, is possible only when $m \in \mathrm{o}\big(n^2\big)$, as it has been recently shown [18,2]; (ii) there is important applied work [10,3,9,15] to develop time-dependent shortest path *heuristics*, which however provide mainly empirical evidence on the success of the adopted approaches.

At a high level, our approach resembles the typical ones used in *static* and *undirected* graphs (e.g., [22,18,2]): Distance summaries from selected landmarks are precomputed and stored; fast responses to arbitrary real-time queries are provided by growing small distance balls around the origin and the destination, and then closing the gap between the prefix subpath from the origin and the suffix subpath towards the destination. However, it is not at all straightforward how this generic approach can be extended to *time-dependent* and *directed* graphs, since one is confronted with two highly non-trivial challenges: (i) handling directedness, and (ii) dealing with time-dependence, i.e., deciding the arrival-times to grow balls around vertices in the vicinity of the destination, because we simply do ***not*** know the earliest-arrival-time at destination – actually, this is what the original query to the oracle asks for. A novelty of our query algorithms, contrary to other approaches, is exactly that we achieve the approximation guarantees by growing balls only from vertices around the origin. Managing this was a necessity for our analysis since growing balls around vertices in the vicinity of the destination at the *right* arrival-time is essentially not an option.

Let U be the worst-case number of breakpoints for an $(1+\varepsilon)-$approximation of a *concave* distance function stored in our oracle, and TDP be the maximum number of time-dependent shortest path probes during their construction[2]. The following theorem summarizes our results.

Theorem 1. *For time-dependent instances compliant with Assumptions 1 and 2, a distance oracle is provided storing $(1+\varepsilon)-$approximate distance functions from landmarks, which are uniformly and independently selected with probability ρ, to all other vertices, and uses a recursion depth (budget) r in the query algorithm, guaranteeing expected values of: (i) preprocessing space $\mathcal{O}\big(\rho n^2(1+K^*)U\big)$; (ii) preprocessing time $\mathcal{O}\big(\rho n^2(1+K^*)\log(n)\log\log(K_{\max})TDP\big)$; (iii) query time $\mathcal{O}\left(\left(\frac{1}{\rho}\right)^{r+1}\log\left(\frac{1}{\rho}\right)\log\log(K_{\max})\right)$. The guaranteed stretch is $1+\varepsilon\frac{(1+\frac{\varepsilon}{\psi})^{r+1}}{(1+\frac{\varepsilon}{\psi})^{r+1}-1}$, where ψ is a fixed constant depending on the characteristics of the arc-travel-time functions, but is independent of the network size.*

Note that, apart from the choice of landmarks, our algorithms are deterministic. Due to space limitations, proofs and a table with solid examples of the oracle's space/query-time/stretch trade-offs can be found in the full version [14].

[2] As proved in [14], U and TDP are independent of the network size n.

2 Ingredients and Overview of Our Approach

Our input is provided by a directed graph $G = (V, A)$ with n vertices and m arcs. Every arc $uv \in A$ is equipped with a periodic, continuous, piecewise-linear (pwl) *arc-travel-time* (a.k.a. *arc-delay*) function $D[uv] : \mathbb{R} \rightarrow \mathbb{R}_{>0}$, such that $\forall k \in \mathbb{Z}, \forall t_u \in [0, T),\ D[uv](k \cdot T + t_u) = D[uv](t_u)$ is the arc-travel-time of uv when the departure-time from u is $k \cdot T + t_u$. $D[uv]$ is represented succinctly as a continuous pwl function, by K_{uv} breakpoints describing its projection to $[0, T)$. $K = \sum_{uv \in A} K_{uv}$ is the number of breakpoints to represent all the arc-delay functions in the network, and $K_{\max} = \max_{uv \in A} K_{uv}$. K^* is the number of *concavity-spoiling* breakpoints, i.e., the ones in which the arc-delay slopes increase. Clearly, $K^* \leq K$, and $K^* = 0$ for *concave* pwl functions. The space to represent the entire network is $\mathcal{O}(n + m + K)$. The *arc-arrival* function $Arr[uv](t_u) = t_u + D[uv](t_u)$ represents arrival-times at v, depending on the departure-times t_u from u. For any $(o, d) \in V \times V$, $\mathcal{P}_{o,d}$ is the set of $od-$paths, and $\mathcal{P} = \cup_{(o,d)} \mathcal{P}_{o,d}$. For a path $p \in \mathcal{P}$, $p_{x \rightsquigarrow y}$ is its subpath from (the first appearance of) vertex x until (the subsequent first appearance of) vertex y. For any pair of paths $p \in \mathcal{P}_{o,v}$ and $q \in \mathcal{P}_{v,d}$, $p \bullet q$ is the $od-$path produced as the concatenation of p and q at v. For any path (represented as a sequence of arcs) $p = \langle a_1, a_2, \cdots, a_k \rangle \in \mathcal{P}_{o,d}$, the *path-arrival* function is the composition of the constituent arc-arrival functions: $\forall t_o \in [0, T),\ Arr[p](t_o) = Arr[a_k](Arr[a_{k-1}](\cdots(Arr[a_1](t_o))\cdots))$. The *path-travel-time* function is $D[p](t_o) = Arr[p](t_o) - t_o$. The *earliest-arrival-time* and *shortest-travel-time* functions from o to d are: $\forall t_o \in [0, T), Arr[o, d](t_o) = \min_{p \in \mathcal{P}_{o,d}} \{Arr[p](t_o)\}$ and $D[o, d](t_o) = Arr[o, d](t_o) - t_o$. Finally, $SP[o, d](t_o)$ (resp. $ASP[o, d](t_o)$) is the set of shortest (resp., with stretch-factor at most $(1 + \varepsilon)$) $od-$paths for a given departure-time t_o.

Facts of the FIFO Property. We consider *networks* $(G = (V, A), (D[a])_{a \in A})$ with continuous arc-delay functions, possessing the *FIFO* (a.k.a. *non-overtaking*) property, according to which all arc-arrival-time functions are non-decreasing:

$$\forall t_u, t'_u \in \mathbb{R}, \forall uv \in A, t_u > t'_u \Rightarrow Arr[uv](t_u) \geq Arr[uv](t'_u) \tag{1}$$

The FIFO property is *strict*, if the above inequality is strict. The FIFO property implies that: (i) the slope of any arc-delay function is greater than -1; (ii) the slope of any path-delay or shortest-travel-time function is greater than -1. The *strict* FIFO property implies *subpath optimality* of shortest paths. For formal statements and proofs of these facts, see [14].

Towards a Time-Dependent Distance Oracle. Our approach for providing a time-dependent distance oracle is inspired by the generic approach for general *undirected* graphs under *static* travel-time metrics. However, we have to tackle the two main challenges of *directedness* and *time-dependence*. Notice that together these two challenges imply an *asymmetric* distance metric which also *evolves* with time. Consequently, to achieve a smooth transition from the static and undirected world towards the time-dependent and directed world, we have to quantify the *degrees of asymmetry and evolution* in our metric. Towards this direction, we make two assumptions on the kind of shortest-travel-time functions in the network. Both assumptions are quite natural and justified by a thorough investigation of historic traffic data for the city of Berlin, kindly provided to us by TomTom [12] (see [14] for a more detailed justification). The first assumption, called *Bounded Travel-Time Slopes*, asserts that the partial derivatives of the shortest-travel-time functions between any pair of origin-destination vertices are bounded in a given fixed interval $[\Lambda_{\min}, \Lambda_{\max}]$.

Assumption 1 (Bounded Travel-Time Slopes). *There are* constants $\Lambda_{\min} > -1$ *and* $\Lambda_{\max} \geq 0$ *s.t.:* $\forall (o,d) \in V \times V,\ \forall t_1 < t_2,\ \frac{D[o,d](t_1) - D[o,d](t_2)}{t_1 - t_2} \in [\Lambda_{\min}, \Lambda_{\max}]$.

The second assumption, called *Bounded Opposite Trips*, asserts that for any given departure time, the shortest-travel-time from o to d is not more than a *constant* $\zeta \geq 1$ times the shortest-travel-time in the opposite direction (but not necessarily along the same path).

Assumption 2 (Bounded Opposite Trips). *There is a* constant $\zeta \geq 1$ *such that:* $\forall (o,d) \in V \times V,\ \forall t \in [0,T),\ D[o,d](t) \leq \zeta \cdot D[d,o](t)$.

As we show in Section 4, the parameters $\Lambda_{\max}$ and ζ allow us to quantify the degree of asymmetry and evolution in time in our distance metric and achieve the aforementioned smooth transition. Another assumption we make and which can be easily guaranteed is that the maximum out-degree is bounded by 2.

Overview of Our Approach. We follow (at a high level) the typical approach adopted for the construction of approximate distance oracles in the static case. In particular, we start by selecting a subset $L \subset V$ of *landmarks*, i.e., vertices which will act as reference points for our distance summaries. For our oracle to work, several ways to choose L would be acceptable. Nevertheless, for the sake of the analysis we assume that this is done by deciding for each vertex randomly and independently with probability $\rho \in (0,1)$ whether it belongs to L. After having L fixed, our approach is deterministic. We start by constructing (concurrently, per landmark) and storing the *distance summaries*, i.e., all landmark-to-vertex $(1+\varepsilon)-$approximate travel-time functions, in time $\mathrm{o}\big((1+K^*)n^2\big)$ and consuming space $\mathrm{o}\big((1+K^*)n^2\big)$ which is indeed asymptotically optimal w.r.t. the required approximation guarantee (cf. Section 3). Then, we provide two approximation algorithms for arbitrary queries $(o,d,t_o) \in V \times V \times [0,T)$. The first (**FCA**) is a simple *sublinear*-time constant-approximation algorithm (cf. Section 4). The second (**RQA**) is a recursive algorithm growing small **TDD** outgoing balls from vertices in the vicinity of the origin, until either a satisfactory approximation guarantee is achieved, or an upper bound r on the depth of the recursion (the *recursion budget*) has been exhausted. **RQA** finally responds with a $(1+\sigma)-$approximate travel-time to the query in *sublinear* time, for any constant $\sigma > \varepsilon$ (cf. Section 4). As it is customary in the distance oracle literature, the query times of our algorithms concern the determination of (upper bounds on) shortest-travel-time from o to d. An actual path guaranteeing this bound can be reported in additional time that is linear in the number of its arcs.

3 Preprocessing Distance Summaries

We now demonstrate how to construct the preprocessed information that will comprise the *distance summaries* of the oracle, i.e., all landmark-to-vertex shortest-travel-time functions. If there exist $K^* \geq 1$ *concavity-spoiling* breakpoints among the arc-delay functions, then we do the following: For each of them (which is a departure-time t_u from the tail u of an arc $uv \in A$) we run a variant of **TDD** with root (u, t_u) on the *reverse network* $(\overleftarrow{G} = (V, A, (\overleftarrow{D}[a])_{a \in A})$, where $\overleftarrow{D}[uv]$ is the delay of arc uv, measured now as a function of the arrival-time t_v at the tail v. The algorithm proceeds backwards both along the connecting path (from the destination towards the origin) and in time. As a result, we compute all *latest-departure-times* from landmarks that allow us to determine the images (i.e., projections to appropriate departure-times from all possible origins) of concavity-spoiling breakpoints. For each landmark, we repeat the procedure described in the rest of this section for every $K^* + 1$ subinterval of $[0, T)$ determined by *consecutive* images of concavity-spoiling breakpoints. Within each subinterval all arc-travel-time functions are concave, as required in our analysis.

We must construct in polynomial time, for all $(\ell, v) \in L \times V$, succinctly represented upper-bounding $(1+\varepsilon)-$approximations $\mathit{\Delta}[\ell, v] : [0, T) \to \mathbb{R}_{>0}$ of the shortest-travel-time functions $D[\ell, v] : [0, T) \to \mathbb{R}_{>0}$. An algorithm providing such functions in a *point-to-point* fashion was proposed in [13]. For each landmark $\ell \in L$, it has to be executed n times so as to construct all the required landmark-to-vertex approximate functions. The main idea of that algorithm is to keep sampling the travel-time axis of the unknown function $D[\ell, v]$ at a logarithmically growing scale, until its slope becomes less than 1. It then samples the departure-time axis via bisection, until the required approximation guarantee is achieved. All the sample points (in both phases) correspond to breakpoints of a lower-approximating function. The upper-approximating function has at most twice as many points. The number of breakpoints returned may be suboptimal, given the required approximation guarantee: even for an affine shortest-travel-time function with slope in $(1, 2]$ it would require a number of points logarithmic in the ratio of max-to-min travel-time values from ℓ to v, despite the fact that we could avoid all intermediate breakpoints for the upper-approximating function.

Our solution is an improvement of the approach in [13] in two aspects: (i) it computes *concurrently* all the required approximate distance functions from a given landmark, at a cost equal to that of a single (worst-case with respect to the given origin and all possible destinations) point-to-point approximation of [13]; (ii) within every subinterval of consecutive images of concavity-spoiling breakpoints, it provides asymptotically optimal space per landmark, which is also independent of the network size per landmark-vertex pair, implying that the required preprocessing space per vertex is $\mathcal{O}(|L|)$. This is also claimed in [13], but it is actually true only for their second phase (the bisection). For the first phase of their algorithm, there is no such guarantee. Even for a linear arc-travel-time function, the first phase of that algorithm would still require a number of samples which is logarithmic in the max-to-min travel-time ratio.

Our algorithm, in order to achieve a concurrent one-to-all construction of upper-bounding approximations from a given landmark $\ell \in L$, is purely based on bisection. This is done because the departure-time axis is common for all these unknown functions $(D[\ell, v])_{v\in V}$. In order for this technique to work, despite the fact that the slopes may be greater than one, a crucial ingredient is an *exact closed-form estimation* of the worst-case absolute error that we provide. This helps our construction to indeed consider only the necessary sampling points as breakpoints of the corresponding (concurrently constructed) shortest travel-time functions. It is mentioned that this guarantee could also be used in the first phase of the approximation algorithm in [13], in order to discard all unnecessary sampling points from being actual breakpoints in the approximate functions.

In a nutshell, we construct two continuous pwl-approximations of the *unknown* shortest-travel-time function $D[\ell, v] : [0, T) \to \mathbb{R}_{>0}$, an upper-bounding approximate function $\overline{D}[\ell, v]$ (playing the role of $\Delta[\ell, v]$) and a lower-bounding approximate function $\underline{D}[\ell, v]$. Our construction guarantees that the exact function is always "sandwiched" between these two approximations. For a given landmark $\ell \in L$ and a subinterval $[t_s, t_f) \subseteq [0, T)$ of departure times from ℓ, in which all the (unknown) shortest-travel-time functions from ℓ are concave, the algorithm proceeds as follows (details are provided in [14]): The current subinterval $[t_s, t_f)$ is bisected in the middle $t_m = \frac{t_s+t_f}{2}$. The result of this bisection is for the lower-approximating function $\underline{D}[\ell, v]$ to be augmented by the new breakpoint t_m, for all still *active* (having not yet met their required approximation guarantee) destination vertices v w.r.t. $[t_s, t_f)$. Our next step is, for each $v \in V$, to check whether the upper-approximating function $\overline{D}[\ell, v]$, consisting of the lower-envelope of the tangents of $D[\ell, v]$ at t_s, t_m and t_f, i.e., at most five breakpoints for the subinterval $[t_s, t_f)$, is already a $(1 + \varepsilon)-$approximation of $\underline{D}[\ell, v]$ within $[t_s, t_m)$ and $[t_m, t_f)$. Each destination vertex that is already satisfied by the current approximation becomes *inactive* for the subsequent subintervals. If any of the two subintervals still has active destination nodes, it is recursively bisected.

$L[\ell, v]$ and $U[\ell, v]$ denote the numbers of breakpoints for $\underline{D}[\ell, v]$ and $\overline{D}[\ell, v]$, $U = \max_{\ell,v}\{U[\ell, v]\}$, and TDP is the number of shortest-path probes during a bisection. By construction it holds that $U[\ell, v] \leq 2 \cdot L[\ell, v]$ (for an explanation see [14]). The expected number of landmarks is $\mathbb{E}\{|L|\} = \rho n$. It is then easy to deduce the required time and space complexity of our entire preprocessing.

Theorem 2. *The preprocessing has expected space/time complexities* $\mathbb{E}\{\mathcal{S}\} \in \mathcal{O}\big(\rho n^2(1 + K^*)U\big)$ *and* $\mathbb{E}\{\mathcal{P}\} \in \mathcal{O}\big(\rho n^2 \log(n) \log\log(K_{\max})(1 + K^*)TDP\big)$.

U and TDP are independent of n (cf. [14]), so we treat them as constants. If all arc-travel-time functions are concave, i.e., $K^* = 0$, then we achieve subquadratic preprocessing space and time $\forall\rho \in \mathcal{O}(n^{-\alpha})$, where $0 < \alpha < 1$. Real data (e.g., TomTom's traffic data for the city of Berlin [12]) demonstrate that: (i) only a small fraction of the arc-travel-time functions exhibit non-constant behaviour; (ii) for the vast majority of these non-constant-delay arcs, their functions are either concave, or can be very tightly approximated by a typical *concave* bell-shaped pwl function. It is only a tiny subset of critical arcs (e.g., bottleneck road segments) for which it would be meaningful to consider non-concave behaviour. Therefore, $K^* \in \mathrm{o}(n)$ is the typical case. E.g., assuming $K^* \in \mathcal{O}(\mathrm{polylog}(n))$, we can fine-tune ρ and the parameters σ, r (cf. Section 4) so as to achieve *subquadratic* preprocessing space and time. In particular, for $K^* \in \mathcal{O}(\log(n))$ and $K_{\max} \in \mathcal{O}(1)$, $\forall \gamma > \frac{1}{2}$, $\mathbb{E}\{\mathcal{S}\} \in \mathcal{O}\big(n^{2-\varepsilon/(\gamma\psi)} \log(n)\big)$ and $\mathbb{E}\{\mathcal{P}\} \in \mathcal{O}\big(n^{2-\varepsilon/(\gamma\psi)} \log^2(n)\big)$, where $\psi = \psi(\zeta, \Lambda_{\max})$ is a constant that will be specified in Theorem 3. More details are provided in [14].

4 Query Algorithms

Constant-Approximation Query Algorithm. Our next step towards a distance oracle is to provide a fast query algorithm providing constant approximations to the actual shortest-travel-time values of arbitrary queries $(o, d, t_o) \in V \times V \times [0, T)$. Here we propose such a query algorithm, called *Forward Constant Approximation* (**FCA**), which grows an outgoing ball $B_o \equiv B[o](t_o) = \{x \in V : D[o, x](t_o) < D[o, \ell_o](t_o)\}$ around (o, t_o) by running **TDD**, until either d or the closest landmark $\ell_o \in \arg\min_{\ell \in L}\{D[o, \ell](t_o)\}$ is scanned. We call $R_o = D[o, \ell_o](t_o)$ the *radius* of B_o. **FCA** returns either the exact travel-time value, or the approximate travel-time value via ℓ_o. Figure 1 gives an overview of the whole idea. The pseudocode is provided in [14].

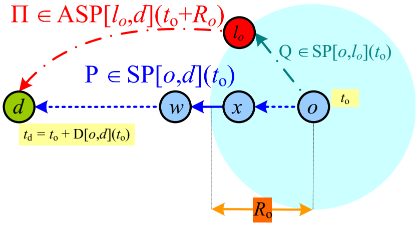


Fig. 1. The rationale of **FCA**. The dashed (blue) path is a shortest $od-$path for query (o, d, t_o). The dashed-dotted (green and red) path is the via-landmark $od-$path indicated by the algorithm, if the destination vertex is out of the origin's **TDD** ball.

Correctness. The next theorem demonstrates that **FCA** returns $od-$paths whose travel-times are constant approximations to the shortest travel-times.

Theorem 3. $\forall (o, d, t_o) \in V \times V \times [0, T)$, **FCA** *returns either an exact path* $P \in SP[o, d](t_o)$, *or a via-landmark od−path* $Q \bullet \Pi$, *s.t.* $Q \in SP[o, \ell_o](t_o)$, $\Pi \in ASP[\ell_o, d](t_o + R_o)$, *and* $D[o, d](t_o) \leq R_o + \Delta[\ell_o, d](t_o + R_o) \leq (1+\varepsilon) \cdot D[o, d](t_o) + \psi \cdot R_o \leq (1+\varepsilon+\psi) \cdot D[o, d](t_o)$, *where* $\psi = 1 + \Lambda_{\max}(1+\varepsilon)(1+2\zeta+\Lambda_{\max}\zeta) + (1+\varepsilon)\zeta$.

Note that **FCA** is a generalization of the 3−approximation algorithm in [2] for symmetric (i.e., $\zeta = 1$) and time-independent (i.e., $\Lambda_{\min} = \Lambda_{\max} = 0$) network instances, the only difference being that the stored distance summaries we consider are $(1+\varepsilon)-$approximations of the actual shortest-travel-times. Observe that our algorithm smoothly departs, through the parameters ζ and $\Lambda_{\max}$, towards both *asymmetry* and *time-dependence* of the travel-time metric.

Complexity. The main cost of **FCA** is to grow the ball $B_o = B[o](t_o)$ by running **TDD**. Therefore, what really matters is the number of vertices in B_o, since the maximum out-degree is 2. L is chosen randomly by selecting each vertex v to become a landmark independently of other vertices, with probability $\rho \in (0,1)$. Clearly $\mathbb{E}\{|B_o|\} = 1/\rho$, and moreover (as a geometrically distributed random variable), $\forall k \geq 1$,$\mathbb{P}\{|B_o| > k\} = (1-\rho)^k \leq e^{-\rho k}$. By setting $k = (1/\rho)\ln(1/\rho)$ we conclude that: $\mathbb{P}\{|B_o| > (1/\rho)\ln(1/\rho)\} \leq \rho$. Since the maximum out-degree is 2, **TDD** will relax at most $2k$ arcs. Hence, for the query-time complexity $\mathcal{Q}_{FCA}$ of **FCA** we conclude that $\mathbb{E}\{\mathcal{Q}_{FCA}\} \in \mathcal{O}((1/\rho)\ln(1/\rho)\log\log(K_{\max}))$, and $\mathbb{P}\left\{\mathcal{Q}_{FCA} \in \Omega\big((1/\rho)\ln^2(1/\rho)\log\log(K_{\max})\big)\right\} \in \mathcal{O}(\rho)$.

$\boldsymbol{(1+\sigma)-}$**Approximate Query Algorithm.** The *Recursive Query Algorithm* (**RQA**) improves the approximation guarantee of the chosen *od*−path provided by **FCA**, by exploiting carefully a number (called the *recursion budget*) of recursive accesses to the preprocessed information, each of which produces (via a call to **FCA**) another candidate *od*−path sol_i. The crux of our approach is the following: We assure that, unless the required approximation guarantee has already been reached by a candidate solution, the recursion budget must be exhausted and the sequence of radii of the consecutive balls that we grow recursively is lower-bounded by a *geometrically increasing* sequence. We prove that this sequence can only have a *constant* number of elements, since the sum of all these radii provides a lower bound on the shortest-travel-time that we seek.

A similar approach was proposed for *undirected* and *static* sparse networks [2], in which a number of recursively growing balls (up to the recursion budget) is used in the vicinities of *both* the origin *and* the destination nodes, before eventually applying a constant-approximation algorithm to close the gap, so as to achieve improved approximation guarantees.

In our case the network is both directed and time-dependent. Due to our ignorance of the exact arrival time at the destination, it is difficult (if at all possible) to grow incoming balls in the vicinity of the destination node. Hence, our only choice is to build a recursive argument that grows outgoing balls in the vicinity of the origin, since we only know the requested departure-time from it. This is exactly what we do: So long as we have not discovered the destination node within the explored area around the origin, and there is still some remaining recursion budget, we "guess" (by exhaustively searching for it) the next node w_k along the (unknown) shortest *od*−path. We then grow a new out-ball from the new center $(w_k, t_k = t_o + D[o, w_k](t_o))$, until we reach the closest landmark-vertex ℓ_k to it, at distance $R_k = D[w_k, \ell_k](t_k)$. This new landmark offers an alternative *od*−path $sol_k = P_{o,k} \bullet Q_k \bullet \Pi_k$ by a new application of **FCA**, where $P_{o,k} \in SP[o, w_k](t_o)$, $Q_k \in SP[w_k, \ell_k](t_k)$, and $\Pi_k \in ASP[\ell_k, d](t_k + R_k)$ is the approximate suffix subpath provided by the distance oracle. Observe that sol_k uses a *longer* optimal prefix-subpath P_k which is then completed with a shorter approximate suffix-subpath $Q_k \bullet \Pi_k$. The pseudocode is provided in [14]. Figure 2 provides an overview of **RQA**'s execution.

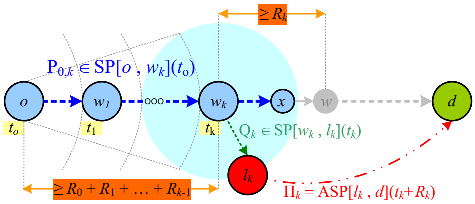

≥ R_k
$P_{0,k} \in SP[o , w_k](t_o)$
o
w_1
ooo
w_k
x
w
d
t_o
t_1
t_k
$Q_k \in SP[w_k , l_k](t_k)$
l_k
≥ $R_0 + R_1 + \ldots + R_{k-1}$
$\Pi_k = ASP[l_k , d](t_k + R_k)$

Fig. 2. Overview of the execution of **RQA**

Correctness & Quality. The correctness of **RQA** implies that the algorithm always returns some $od-$path. This is true due to the fact that it either discovers the destination node d as it explores new nodes in the vicinity of the origin node o, or it returns the shortest of the approximate $od-$paths $sol_0, \ldots, sol_r$ via one of the closest landmarks $\ell_o, \ldots, \ell_r$ to "guessed" nodes $w_0 = o, w_1, \ldots, w_r$ along the shortest $od-$path $P \in SP[o, d](t_o)$, where r is the recursion budget. Since the preprocessed distance summaries stored by the oracle provide approximate travel-times corresponding to actual paths from landmarks to vertices in the graph, it is clear that **RQA** always implies an $od-$path whose travel-time does not exceed the alleged upper bound on the actual distance.

Our next task is to study the quality of the provided stretch $1 + \sigma$ guaranteed by **RQA**. Let $\delta > 0$ be a parameter such that $\sigma = \varepsilon + \delta$ and recall the definition of ψ from Theorem 3. In [14] it is shown that the sequence of ball radii grown from vertices of the shortest $od-$path $P[o, d](t_o)$ by the recursive calls of **RQA** is lower-bounded by a *geometrically increasing* sequence. The next theorem shows that **RQA** indeed provides $(1 + \sigma)-$approximate distances in response to arbitrary queries $(o, d, t_o) \in V \times V \times [0, T)$.

Theorem 4. *For the stretch of* **RQA** *the following hold:*

1. *If* $r = \left\lceil \frac{\ln\left(1+\frac{\varepsilon}{\delta}\right)}{\ln\left(1+\frac{\varepsilon}{\psi}\right)} \right\rceil - 1$ *for* $\delta > 0$*, then,* **RQA** *guarantees a stretch* $1 + \sigma = 1 + \varepsilon + \delta$.
2. *For a given recursion budget* $r \in \mathbb{N}$, **RQA** *guarantees stretch* $1 + \sigma$*, where* $\sigma = \sigma(r) \leq \frac{\varepsilon \cdot (1+\varepsilon/\psi)^{r+1}}{(1+\varepsilon/\psi)^{r+1} - 1}$.

Note that for time-independent, undirected-graphs (for which $\Lambda_{\min} = \Lambda_{\max} = 0$ and $\zeta = 1$) it holds that $\psi = 2 + \varepsilon$. If we equip our oracle with *exact* rather than $(1 + \varepsilon)-$approximate landmark-to-vertex distances (i.e., $\varepsilon = 0$), then in order to achieve $\sigma = \delta = \frac{2}{t+1}$ for some positive integer t, our recursion budget r is *upper bounded* by $\frac{\psi}{\delta} - 1 = t$. This is exactly the amount of recursion required by the approach in [2] to assure the same approximation guarantee. That is, at its one extreme ($\Lambda_{\min} = \Lambda_{\max} = 0$, $\zeta = 1$, $\psi = 2$) our approach matches the bounds in [2] for the same class of graphs, without the need to grow balls from both the origin and destination vertices. Moreover, our approach allows for a *smooth* transition from static and undirected-graphs to directed-graphs with FIFO arc-delay functions. The required recursion budget now depends not only on the targeted approximation guarantee, but also on the degree of asymmetry (the value of $\zeta \geq 1$) and the steepness of the shortest-travel-time functions (the value of $\Lambda_{\max}$) for the time-dependent case. It is noted that we have recently become aware of an improved bidirectional approximate distance oracle for static undirected graphs [1] which outperforms [2] in the stretch-time-space tradeoff.

Complexity. It only remains to determine the query-time complexity $\mathcal{Q}_{RQA}$ of **RQA**. This is provided by the following theorem.

Theorem 5. *For networks having* $|A|/|V| \in \mathcal{O}(1)$*, the expected running time of* **RQA** *is* $\mathbb{E}\left\{\mathcal{Q}_{RQA}\right\} \in \mathcal{O}\big((1/\rho)^{r+1} \cdot \ln(1/\rho) \cdot \log\log(K_{\max})\big)$*, and it holds that:*

$$\mathbb{P}\left\{\mathcal{Q}_{RQA} \in \mathcal{O}\left(\left(\frac{\ln(n)}{\rho}\right)^{r+1} \cdot \left[\ln\ln(n) + \ln\left(\frac{1}{\rho}\right)\right] \cdot \log\log(K_{\max})\right)\right\} \in 1 - \mathcal{O}\big(\tfrac{1}{n}\big) \ .$$

Continuing the discussion in the paragraph following Theorem 2, we can fine-tune the parameters σ, r so as to achieve, along with subquadratic space and preprocessing time, sublinear query-time complexity $\mathbb{E}\left\{\mathcal{Q}_{RQA}\right\} \in \mathcal{O}\big(n^{1/(2\gamma)}\log(n)\big)$, $\forall\gamma > \frac{1}{2}$. More details (and examples) are provided in [14].

References

1. Agarwal, R.: The space-stretch-time trade-off in distance oracles (July 2013) (manuscript)
2. Agarwal, R., Godfrey, P.: Distance oracles for stretch less than 2. In: Proceedings of the 24th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA 2013), pp. 526–538. ACM-SIAM (2013)
3. Batz, G.V., Geisberger, R., Sanders, P., Vetter, C.: Minimum time-dependent travel times with contraction hierarchies. ACM Journal of Experimental Algorithmics 18 (2013)
4. Cooke, K., Halsey, E.: The shortest route through a network with time-dependent intermodal transit times. Journal of Mathematical Analysis and Applications 14(3), 493–498 (1966)
5. Dean, B.C.: Continuous-time dynamic shortest path algorithms. Master's thesis, Massachusetts Institute of Technology (1999)
6. Dean, B.C.: Algorithms for minimum-cost paths in time-dependent networks with waiting policies. Networks 44(1), 41–46 (2004)
7. Dean, B.C.: Shortest paths in fifo time-dependent networks: Theory and algorithms. Technical report, MIT (2004)
8. Dehne, F., Masoud, O.T., Sack, J.-R.: Shortest paths in time-dependent fifo networks. Algorithmica 62(1-2), 416–435 (2012)
9. Delling, D.: Time-Dependent SHARC-Routing. Algorithmica 60(1), 60–94 (2011); Special Issue: European Symposium on Algorithms (2008)
10. Delling, D., Wagner, D.: Time-Dependent Route Planning. In: Ahuja, R.K., Möhring, R.H., Zaroliagis, C.D. (eds.) Robust and Online Large-Scale Optimization. LNCS, vol. 5868, pp. 207–230. Springer, Heidelberg (2009)
11. Dreyfus, S.E.: An appraisal of some shortest-path algorithms. Operations Research 17(3), 395–412 (1969)
12. eCOMPASS Project (2011-2014), `http://www.ecompass-project.eu`
13. Foschini, L., Hershberger, J., Suri, S.: On the complexity of time-dependent shortest paths. Algorithmica 68(4), 1075–1097 (2014); Preliminary version in ACM-SIAM SODA (2011)

14. Kontogiannis, S., Zaroliagis, C.: Distance oracles for time dependent networks. eCOMPASS Technical Report (eCOMPASS-TR-025) / ArXiv Report (arXiv.org > cs > arXiv:1309.4973) (September 2013)
15. Nannicini, G., Delling, D., Liberti, L., Schultes, D.: Bidirectional A* Search on Time-Dependent Road Networks. Networks 59, 240–251 (2012)
16. Orda, A., Rom, R.: Shortest-path and minimum delay algorithms in networks with time-dependent edge-length. Journal of the ACM 37(3), 607–625 (1990)
17. Patrascu, M., Roditty, L.: Distance oracles beyond the Thorup–Zwick bound. In: Proc. of 51th IEEE Symp. on Found. of Comp. Sci. (FOCS 2010), pp. 815–823 (2010)
18. Porat, E., Roditty, L.: Preprocess, set, query! In: Demetrescu, C., Halldórsson, M.M. (eds.) ESA 2011. LNCS, vol. 6942, pp. 603–614. Springer, Heidelberg (2011)
19. Sherali, H.D., Ozbay, K., Subramanian, S.: The time-dependent shortest pair of disjoint paths problem: Complexity, Models, and Algorithms. Networks 31(4), 259–272 (1998)
20. Sommer, C.: Shortest-path queries in static networks. ACM Computing Surveys 46 (2014)
21. Sommer, C., Verbin, E., Yu, W.: Distance oracles for sparse graphs. In: Proc. of 50th IEEE Symp. on Found. of Comp. Sci. (FOCS 2009), pp. 703–712 (2009)
22. Thorup, M., Zwick, U.: Approximate distance oracles. J. of ACM 52, 1–24 (2005)
23. Wulff-Nilsen, C.: Approximate distance oracles with improved preprocessing time. In: Proc. of 23rd ACM-SIAM Symp. on Discr. Alg. (SODA 2012) (2012)
24. Wulff-Nilsen, C.: Approximate distance oracles with improved query time. arXiv abs/1202.2336 (2012)

Efficient Indexing of Necklaces and Irreducible Polynomials over Finite Fields

Swastik Kopparty[1,⋆], Mrinal Kumar[2,⋆⋆], and Michael Saks[3,⋆⋆⋆]

[1] Department of Computer Science and Department of Mathematics, Rutgers University
[2] Department of Computer Science, Rutgers University
[3] Department of Mathematics, Rutgers University

Abstract. We study the problem of *indexing* necklaces, and give the first polynomial time algorithm for this problem. Specifically, we give a $\mathsf{poly}(n, \log |\Sigma|)$-time computable bijection between $\{1, \ldots, |\mathcal{N}|\}$ and the set $\mathcal{N}$ of all necklaces of length n over a finite alphabet Σ.

Our main application is to give an explicit indexing of all irreducible polynomials of degree n over the finite field $\mathbb{F}_q$ in time $\mathsf{poly}(n, \log q)$ (with $n \log q$ bits of advice). This has applications in pseudorandomness, and answers an open question of Alon, Goldreich, Håstad and Peralta [2].

1 Introduction

An *indexing* of a finite set S is a bijection from the set $\{1, \ldots, |S|\}$ to S. An *indexing* of a language L is an algorithm A which takes as input a length parameter n and an index j and outputs $A^n(j)$, so that for each n:

- A^n maps the set $\{1, \ldots, |L^n|\}$ bijectively to L^n, where L^n is the set of length n words of L.
- If $j > |L^n|$ then $A^n(j)$ returns **too large**.

A *reverse-indexing* of L is a bijection from L^n to $\{1, \ldots, |L^n|\}$. An indexing of reverse-indexing is *efficient* if it runs in time $\mathsf{poly}(n)$. We can formulate these problems for any combinatorial structure, such as permutations, graphs, partitions, etc. by using standard efficient encodings of such structures by strings.

This paper gives the first efficient algorithm for indexing *necklaces*, where a necklace of length n over $[q]$ is an equivalence class of $[q]^n$ under cyclic permutation. As a consequence, we show the existence of polynomial size circuits for indexing *irreducible polynomials over finite fields* (answering a problem of Alon, Goldreich, Håstad and Peralta [2]). This latter problem is the polynomial analogue of the problem of "giving a formula for the n-bit primes" (with a little advice).

[⋆] Research supported in part by a Sloan Fellowship and NSF grant CCF-1253886.
[⋆⋆] Research supported in part by NSF grant CCF-1253886.
[⋆⋆⋆] Research supported in part by NSF grants CCF-0832787 and CCF-1218711.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 726–737, 2014.

Indexing, Enumeration, Counting, Ranking and Unranking. Indexing is closely related to *counting* and *enumeration*. A counting algorithm for L inputs n and outputs $|L^n|$, and an enumeration algorithm lists the elements of L^n without repetition. Such algorithms are *efficient* if they run in time $\mathsf{poly}(n)$ (for counting) or $|L^n| \cdot \mathsf{poly}(n)$ (for enumeration). There is well developed complexity theory for counting problems, starting with Valiant [22]. Many basic combinatorial identities (e.g. expressing the number objects of a given type of size n in terms of the number of objects of smaller sizes) give efficient counting algorithms. The enumeration problem for combinatorial structures has also received a large amount of attention, see, for example [4,13,16,18].

Indexing and reverse-indexing are even more closely related to the *unranking* and *ranking* problems. For the ranking (resp. unranking) problem, e.g. [15], the combinatorial objects being indexed have a prespecified order (such as lexicographic order) and the goal is to compute a reverse-indexing (resp. indexing) that is consistent with that order.

Counting and enumeration can be easily reduced to indexing. Conversely, for many structures, such as subsets, permutations, set partitions, integer partitions and trees, the known counting algorithms can be modified to give efficient indexing algorithms, e.g., when the counting problem is solved by a recurrence relation that has a bijective proof.

However, it seems that not all combinatorial counting arguments lead to efficient indexing algorithms. For example, we can count the orbits of a group action on a set using the Burnside counting lemma, but we don't know a general way to efficiently index orbits (where the indexing algorithm represents an orbit by some (arbitrary) representative). The problem of indexing necklaces is perhaps the simplest non-trivial problem of this type.

Main Results. We give efficient algorithms for indexing and reverse-indexing necklaces:

Theorem 1. *There are* $\mathsf{poly}(n \log q)$*-time algorithm for indexing and reverse-indexing necklaces of* $\{1, \dots, q\}^n$.

Using a known correspondence [10] between necklaces and irreducible polynomials over finite fields, we also index irreducible polynomials.

Theorem 2. *There is a* $\mathsf{poly}(n \log q)$*-time algorithm (using* $n \log q$ *bits of advice) for indexing irreducible polynomials of degree* n *over the finite field* $\mathbb{F}_q$.

This is the polynomial analogue of the problem of finding a "formula for the primes" (which is wide open, even with advice) and answers a question of [2].

Note that there is no known deterministic algorithm without advice for producing even a single irreducible polynomial of degree n (for general q) in time $\mathsf{poly}(n \log q)$. Our result shows that with a little advice, we can produce not just one, but all irreducible polynomials. For constant q it is known how to construct one irreducible polynomial in $\mathsf{poly}(n)$) time without advice, and our indexing algorithm needs just $\mathsf{poly}(\log n)$ bits of advice.

We prove Theorem 2 in Section 3. Other proofs will be in the full paper.

We mention a few applications of our algorithms; see the full paper for details.

The indexing algorithm for irreducible polynomials can be used to reduce the advice needed (from exponential to logarithmic) in a classical ϵ-biased set construction from [2] based on linear-feedback shift register sequences.

The indexing algorithm for irreducible polynomials can also be used to make the explicit subspace designs of [11] very explicit (with small advice).

Indexing a set immediately gives a method for randomness-optimal sampling of the set. Sampling algorithms for irreducible polynomials have several uses, such as polynomial identity testing and string fingerprinting. Agrawal and Biswas [1] constructed a family of nearly-coprime polynomials, to obtain a randomness-efficient black-box polynomial identity test. The string fingerprinting algorithm by Rabin [17] interprets a string as the coefficients of a polynomial and reduces it mod a random irreducible polynomial. Our indexing gives a way to reduce randomness in these constructions (at the cost of some advice).

We give the first $\mathsf{poly}(n)$ time algorithm for computing any given entry of the $k \times 2^n$ generator matrix matrix or the $(2^n - k) \times 2^n$ parity check matrix of BCH codes for all values of the designed distance (this is the standard notion of strong explicitness for error-correcting codes). Earlier, it was only known how to compute this entry explicitly for very small values of the designed distance (which is usually the setting where BCH codes are used).

Related Work. There is an extensive literature on enumeration algorithms for combinatorial objects (see the books [4,12,13,16,18]). These references include discussions of necklaces and the ranking/unranking problems for various combinatorial objects.

The lexicographically smallest element of a rotation class is called a *Lyndon word*. Algorithmically, the problem of enumerating/indexing necklaces is essentially equivalent to the problem of enumerating/indexing Lyndon words. Following a long line of work [5–9,19,20], we now know linear time enumeration algorithms for Lyndon words/necklaces.

Our indexing/reverse indexing algorithms for necklaces also gives and efficient ranking/unranking of the lexicographic order on Lyndon words, which was noted as an open problem in [14].

Andoni, Goldberger, McGregor and Porat [3] studied a problem that is an approximate version of reverse indexing of necklaces. They gave a randomized algorithm for producing short fingerprints of strings, such that the fingerprints of rotations of a string are determined by the fingerprint of the string itself.

2 An Algorithm for Indexing Necklaces

Overview. Given an acyclic directed graph D on vertex set V and distinguished subsets S and T of nodes, there is a straightforward algorithm to index the set of paths that start in S and end in T: Fix an arbitrary ordering on the nodes, and consider the induced lexicographic ordering on paths (i.e. path $P_1P_2\ldots$ is less than path $Q_1Q_2\ldots$ if $P_i < Q_i$ where i is the least integer such that $P_i \neq Q_i$).

Our indexing function maps index j to the jth path from S to T in lexicographic order. There is a simple dynamic program which computes for each node v, the number $N(v)$ of paths from v to T. Let $v_1, \ldots, v_r$ be the nodes of S in order. On input j, we find the first source v_i such $N(v_1) + \cdots + N(v_i) \geq j$; if there is no such source then the index j is larger than the number of paths being indexed. Otherwise, v_i is the first node of the desired path, and we proceed inductively by replacing S by the set of children of v_i.

This approach can be used to index a set S of strings where S is the set of strings of length n accepted by a given deterministic automaton with $\mathsf{poly}(n)$ states. From this we build a layered directed graph with a single source at level 0, and n additional layers each containing copies of all of the states which represents the computation of the automaton over time, and whose source-to-sink paths correspond to strings of length n accepted by the automaton.

This suggests the following approach to indexing necklaces. For each equivalence class of strings (necklace) identify a canonical representative string of the class (such as the lexicographically smallest representative). Then build an automaton B which, given string y, determines whether y is a canonical representative of its class. By the preceding paragraph, this would be enough to index all of the canonical representatives, which is equivalent to indexing equivalence classes.

In fact, we are able to implement this approach provided that $q = 2$ and n is prime. However, we have not been able to make it work in general. For this we need another approach, which uses finite automata in a more involved way.

First some notation. For a string y, $\mathsf{Rot}^i(y)$ is the string obtained from y by cyclically rotating i positions to the right. $\mathsf{Orbit}(y)$ (also called the *equivalence class of* y is the set of all rotations of y). A string y is *periodic with period* p if it can be written as ${y_1}^q$ for some $y_1 \in \Sigma^p$ and $q = \frac{n}{p}$. The *fundamental period* of y, $FP(x)$ is its smallest period. Note that $FP(y) = |\mathsf{Orbit}(y)|$.

If E is an orbit and x is a string, we say that $E < x$ if E has at least one string $y {<_{\mathsf{lex}}} x$, where $<_{\mathsf{lex}}$ denotes lexicographic order. (Notice that if x and y are strings then we might have both that the orbit of x is less than y and the orbit of y is less than x). Let $\mathcal{C}_x$ be the set of orbits that are less than x.

Observation. If $x {<_{\mathsf{lex}}} y$ then $\mathcal{C}_x \subseteq \mathcal{C}_y$.

Our goal is to design an algorithm which, given string x, returns the size of $\mathcal{C}_x$. From this we can solve the indexing problem using binary search: our indexing maps j to the orbit of the largest x such that $|\mathcal{C}_x| < j$. It is easy to see that this map is a bijection. A reverse indexing can be constructed similarly.

Thus it is enough to give an algorithm for computing $|\mathcal{C}_x|$.

We define:

$G_{x,p} = \bigcup_{E \in \mathcal{C}_x : |E| = p} E$ and $G_{x,\leq p} = \bigcup_{E \in \mathcal{C}_x : |E| \text{ divides } p} E$.

As we will see, $|\mathcal{C}_x|$ can be computed exactly as a function of $|G_{x,p}|$ (for various p). Furthermore, if we can count $|G_{x,\leq p}|$ for each p then using Möbius inversion we can count $|G_{x,p}|$ for each p. The main component of our algorithm is a subroutine that given x and p, computes $|G_{x,\leq p}|$. This subroutine builds an automaton with $n^{O(1)}$ states, that accepts an input string y if and only if (1) the orbit of y has size dividing p and (2) $\mathsf{Orbit}(y) < x$. The desired quantity $|G_{x,\leq p}|$, is the number of y accepted by this automaton (and can be computed in polynomial time via a dynamic program).

Some Notation and Preliminaries

A string is said to be *good* if it is an element of G_x. We denote by $G_{x,p}$ the set of all good strings of fundamental period p and by $G_{x,\leq p}$ the set of all good strings with period p.

For a set of strings W, $\mathsf{Prefix}(W)$ (resp. $\mathsf{Suffix}(W)$,$\mathsf{Substring}(W)$) denotes the set of prefixes, (resp., suffixes, contiguous substrings) of all strings in W (including the empty string ϵ). For string x, we denote $\mathsf{Prefix}(\{x\})$ by $\mathsf{Prefix}(x)$, etc.

An automaton will refer to a deterministic finite state machine where every edge is labeled by an element of the alphabet[1]. For an automaton A, $L(A)$, $Q(A), s(A), F(A)$ and δ_A denote, respectively, the language accepted by A, the set of states of A, the start state of A, the set of final or accepting states of A and the transition function of A.

We state (without proof) some basic facts about periodic strings.

Fact 1. Let y be a string of length n. Then, $|\mathsf{Orbit}(y)| = FP(y)$ is a divisor of n. Thus y can be written as $y_1{}^{\frac{n}{FP(y)}}$ for an aperiodic string $y_1 \in \Sigma^{FP(y)}$ and the fundamental period of a string is a divisor of any period of the string.

Reduction to Computing $|G_{x,\leq p}|$. We now reduce computing $|\mathcal{C}_x|$ to computing $|G_{x,\leq p}|$ (for various p).

Lemma 3. *For all $x \in \Sigma^n$, $|\mathcal{C}_x| = \sum_{y \in G_{x,\leq n}} \frac{1}{|\mathsf{Orbit}(y)|} = \sum_{y \in G_{x,\leq n}} \frac{1}{FP(y)}$.*

The sum on the right hand side can be split on the basis of the period of y. From Lemma 3 and Fact 1, we have the following lemma.

Lemma 4. *For all $x \in \Sigma^n$, $|\mathcal{C}_x| = \sum_{i|n} |G_{x,i}|/i$.*

So, to count $|\mathcal{C}_x|$ efficiently, it suffices to compute $|G_{x,i}|$ efficiently for each $i|n$. Now, just from the definitions, we have the following lemma.

Lemma 5. *For all $x \in \Sigma^n$, $|G_{x,\leq p}| = \sum_{i|p} |G_{x,i}|$.*

From the Möbius Inversion Formula, we have the following equality.

Lemma 6.

$$|G_{x,p}| = \sum_{i|p} \mu\left(\frac{p}{i}\right) |G_{x,\leq i}|$$

Lemma 6 implies that it suffices to compute $|G_{x,\leq p}|$ efficiently for every divisor p of n. In the next few sections, we will focus on this sub-problem and design an

[1] In our context, the automaton size will usually be at least as large as the size of the input on which it is designed to run. Thus we could also have referred to automata as “read-once branching programs” .

efficient algorithm for this problem. We will first describe the algorithm when the alphabet is binary, and then generalize to larger alphabets.

Computing $|G_{x,\leq n}|$ Efficiently for the Binary Alphabet
We present an efficient algorithm which on input x outputs $|G_{x,\leq n}|$. We restrict to $\Sigma = \{0,1\}$ (and handle the case of larger Σ later).

Lemma 7. *There is a deterministic algorithm that takes as input a deterministic automaton A and an integer n, runs in time $\mathsf{poly}(|A|, n)$ and outputs $|L(A) \cap \{0,1\}^n|$, the number of strings of length n accepted by A.*

Proof. Consider the natural (edge-labelled) directed graph associated with the automaton A. The number of strings of length n accepted by A is equal to the number of walks of length n from the start state to an accept state, which is the sum of appropriate entries of the n^{th} power of the adjacency matrix of the graph and is computable in time polynomial in the size of the graph and n.

For a string x of length n we construct an automaton A_x of size $n^{O(1)}$ such that $L(A_x) \cap \{0,1\}^n = G_{x,\leq n}$. Lemma 7 lets us compute $|G_{x,\leq n}|$ in time $n^{O(1)}$.

For strings x, y, $y <_{\mathsf{lex}} x$ if and only if there is an $i \in \{1, 2, \ldots, n-1\}$ such that $x_{i+1} > y_{i+1}$ and $y_j = x_j$ for $j \leq i$. For binary strings, $x_{i+1} = 1$ and $y_{i+1} = 0$. We define the *witness set for x* to be the set $L_x = \{s0 : s1 \text{ is a prefix of } x\}$. Summarizing the previous discussion we have:

Observation 8. *For $x, y \in \{0,1\}^n$, we have $y <_{\mathsf{lex}} x$ if and only if some prefix of y lies in L_x.*

We will now generalize this observation to strings under rotation. For strings x, y, when is $\mathsf{Orbit}(y) < x$? Recall that $\mathsf{Orbit}(y) < x$ if for some $y' \in \mathsf{Orbit}(y)$, we have $y' <_{\mathsf{lex}} x$. From Observation 8, we know that this happens if and only if some $y' \in \mathsf{Orbit}(y)$ has some prefix w in L_x. Rotating back to y, two situations can arise. Either y contains w as a contiguous substring, or w appears as a "split substring" wrapped around the end of y. In the latter case, y has a prefix w_1 and a suffix w_2 such that $w_2 w_1 = w \in L_x$.

Recall that $G_{x,\leq n}$ is the set of y with $\mathsf{Orbit}(y) < x$. Thus, every string $y \in G_{x,\leq n}$ either has a contiguous substring as a witness, or it has a witness which is wrapped around its end. Let us separate these two cases out.

Definition 1. *For a string $x \in \{0,1\}^n$,*

$$G^c_{x,\leq n} = \{y \in \{0,1\}^n : y \textit{ contains a string in } L_x \textit{ as a contiguous substring }\}$$

$$G^w_{x,\leq n} = \{y \in \{0,1\}^n : y \textit{ has a prefix } w_1 \textit{ and suffix } w_2 \textit{ such that } w_2 w_1 \in L_x\}$$

From the discussion in the paragraph above, we have the following observation:

Observation 9.

$$G_{x,\leq n} = G^c_{x,\leq n} \cup G^w_{x,\leq n}$$

So, in order to construct an automaton A_x for which $L(A_x) \cap \{0,1\}^n = G_{x,\leq n}$, it suffices to construct automata A^i_x for $i \in \{c, w\}$ recognizing G^i_x. The size of A_x would be at most $|A^w_x| \times |A^c_x|$. In the next two sections, we will give efficient algorithms to construct small automata A^w_x and A^c_x, given x as input.

Remark 1. In the definition 1, if $w_2 = \epsilon$ or $w_1 = \epsilon$, then $y \in G^c_{x,\leq n} \cap G^w_{x,\leq n}$. Hence, $G^c_{x,\leq n}$ and $G^w_{x,\leq n}$ do not partition the set $G_{x,\leq n}$ and such strings y will be accepted by both the automata A^w_x and A^c_x and hence A_x.

Note that we will be designing algorithms to produce automata. We will never actually run these automata; we will eventually feed the automaton A_x into the algorithm of Lemma 7 in order to determine $|G_{x,\leq n}|$.

Constructing Automaton A^c_x Efficiently. We now design a poly-time algorithm which on input $x \in \{0,1\}^n$, outputs an automaton A^c_x.

Definition 2 (Automaton A^c_x). *The automaton A^c_x is given as a tuple $(Q(A^c_x), s(A^c_x), F(A^c_x), \delta_{A^c_x})$, with*

- *Set of states $Q(A^c_x) = \{s_u : u \in \mathsf{Prefix}(L_x)\}$*
- *Start state $s(A^c_x) = s_\epsilon$*
- *Set of final states $F(A^c_x) = \{s_u : u \in L_x\}$*
- *Transition function $\delta_{A^c_x}$ defined as:*
  1. *for every $s_t \in F$ and $b \in \{0,1\}$, $\delta_{A^c_x}(s_t, b) = s_t$*
  2. *for every $s_t \notin F$ and $b \in \{0,1\}$, such that tb has a suffix in L_x, $\delta_{A^c_x}(s_t, b) = s_u$ where u is the longest such suffix*
  3. *for every $s_t \notin F$ and $b \in \{0,1\}$, such that tb has no suffix in L_x, $\delta_{A^c_x}(s_t, b) = s_u$, where u is the longest suffix of tb in $\mathsf{Prefix}(L_x)$*

We will now argue that $L(A^c_x) \cap \{0,1\}^n = G^c_x$. We will first prove the following claim.

Claim. For every state $s_t \in Q(A^c_x)$ and $z \in \{0,1\}^*$, if A^c_x reaches the state s_t on input z, then one of the following is true:

- $t \in L_x$ and t is a contiguous substring of z
- $t \notin L_x$ and t is the longest suffix of z in $\mathsf{Prefix}(L_x)$

The claim above implies that a string is accepted if and only if it has a contiguous substring in L_x. In particular, we get the correctness of the construction.

Lemma 10. $L(A^c_x) \cap \{0,1\}^n = G^c_x$

For $x \in \{0,1\}^n$, the sets L_x and $\mathsf{Prefix}(L_x)$ can be constructed in time polynomial in n. Therefore, the automaton A^c_x can be constructed in time polynomial in n.

Constructing A_x^w Efficiently. We design an efficient algorithm that constructs an automaton A_x^w satisfying $L(A_x^w) \cap \{0,1\}^n = G_x^w$, i.e., A_x^w accepts strings y that have a prefix b and a suffix a such that $ab \in L_x$. Intuitively, A_x^w is obtained by combining two automata $A_x^{w,\mathsf{P}}$, which detects the prefix and $A_x^{w,\mathsf{S}}$ which detects the suffix. Due to the lack of space, the constructions and analyses of the automata $A_x^{w,\mathsf{P}}$, $A_x^{w,\mathsf{S}}$ and A_x^w are omitted from this version of the paper. Lemma 11 summarizes the property we need on the strings accepted by A_x^w.

Lemma 11. *A string z is accepted by A_x^w if and only if, it has a prefix β and a suffix α, possibly empty, such that $\alpha\beta \in L_x$.*

The Lemma implies that that $L(A_x^w) \cap \{0,1\}^n = G_x^w$.

Putting Things Together. From the constructions, it is clear that the size of the automaton A_x^w and A_x^c is polynomial in the size of L_x and hence polynomial in $n = |x|$. Moreover, the product automaton of A_x^w and A_x^c can be constructed in time polynomial in n. This observation, along with the Lemma 7 implies:

Lemma 12. *There is an algorithm which takes as input a string x in $\{0,1\}^n$ and outputs the size of $G_{x,\leq n}$ in time polynomial in n.*

Computing $|G_{x,\leq p}|$ Efficiently. In this section, we will show that for every $p|n$, we can compute the quantity $|G_{x,\leq p}|$ efficiently. The algorithm will be a small variation of our algorithm for computing $|G_{x,\leq n}|$ from the previous section. Let p be a divisor of n with $p < n$. Every string $y \in G_{x,\leq p}$ is of the form $a^{\frac{n}{p}}$ for some $a \in \{0,1\}^p$, and every string in $\mathsf{Orbit}(y)$ is of the form $(\mathsf{Rot}^i(a))^{\frac{n}{p}}$, for some $i \leq p$. Let us write the string x as $x_1 x_2 \ldots x_{\frac{n}{p}}$ where for each i, x_i is of length exactly p. We will now try to characterize the strings in $G_{x,\leq p}$. From the definitions, $y = a^{\frac{n}{p}} \in G_{x,\leq p}$ if and only if there is a rotation i such that $(\mathsf{Rot}^i(a))^{\frac{n}{p}}$ has a prefix in L_x, which holds if and only if there is an $i < p$ such that one of the following is true.

- $\mathsf{Rot}^i(a) < x_1$ in lexicographic order, or
- there is j, $0 < j < \frac{n}{p}$, such that $\mathsf{Rot}^i(a) = x_1 = x_2 = x_3 = \ldots = x_i$ and $\mathsf{Rot}^i(a) < x_{i+1}$ in lexicographic order.

The strings $y = a^{\frac{n}{p}}$ for which a has a rotation which is less than x_1 in lexicographic order are exactly the strings of the form $c^{\frac{n}{p}}$ with $c \in G_{x_1,\leq p}$. Via the algorithm of the previous subsection, there is a polynomial in n time algorithm which outputs an automaton recognizing $G_{x_1,\leq p}$. The only strings which satisfy the second condition are of the form $c^{\frac{n}{p}}$, where c is a rotation of x_1 and $x_1 < x_{i+1}$ in lexicographic order. There are at most $|\mathsf{Orbit}(x_1)|$ such strings, and we can count them directly given x. This leads to:

Algorithm for Computing $|G_{x,\leq p}|$:

1. Write x as $x = x_1 x_2 \dots x_{\frac{n}{p}}$ where $|x_i| = p \forall i \in [\frac{n}{p}]$
2. Construct an automaton A_{x_1} such that $L(A_{x_1}) \cap \{0,1\}^p = G_{x_1,\leq p}$
3. Let M be the number of strings of length p accepted by A_{x_1}
4. If there is an $0 < i < \frac{n}{p}$ such that $x_1 = x_2 = x_3 = \dots x_i$ and $x_1 < x_{i+1}$ in lexicographic order, and $x_1 \notin L(A_{x_1})$, then output $M + |\mathsf{Orbit}(x_1)|$, else output M.

From the construction in Section 2 and Lemma 12, it follows that we can construct A_{x_1} and count M in time polynomial in n. We thus have:

Lemma 13. *For any divisor p of n and string $x \in \{0,1\}^n$, we can compute the size of the set $G_{x,\leq p}$ in time polynomial in n.*

We now have all the ingredients for the proof of the following theorem.

Theorem 14. *There is an algorithm for indexing necklaces of length n over the alphabet $\{0,1\}$, which runs in time $n^{O(1)}$.*

Proof. The proof simply follows by plugging together the conclusions of Lemma 4, Lemma 5, Lemma 6, Lemma 7 and Lemma 13.

It is not difficult to see that the indexing algorithm can be used to obtain a reverse indexing algorithm as well and hence, we also obtain a special case of Theorem 1 for the binary alphabet.

Indexing Necklaces over Large Alphabets. We now turn to the case of general alphabets Σ of size q. A straightforward generalization of the algorithm for binary alphabets, runs in time polynomial in n and q. Our goal here is to improve the running time to $\mathsf{poly}(n, \log q)$. The idea is to represent the elements in Σ by binary strings of length $t \overset{\text{def}}{=} \lceil \log q \rceil$. Let $\mathsf{Bin} : \Sigma \to \{0,1\}^t$ be an injective map whose image is the set Γ of q lexicographically smallest strings in $\{0,1\}^t$. Extend this to a map $\mathsf{Bin} : \Sigma^n \to \{0,1\}^{tn}$ in the natural way. We now use Bin to convert our indexing/counting problems over the large alphabet Σ to a related problem over $\{0,1\}$.

For $x \in \Sigma^n$, we have $\mathsf{Bin}(\mathsf{Rot}^i(x)) = \mathsf{Rot}^{ti}(\mathsf{Bin}(x))$. For an orbit $E \subseteq \Sigma^n$ and $x \in \{0,1\}^{tn}$, we say $E < x$ if some element $z \in E$ satisfies $\mathsf{Bin}(z) <_{\mathsf{lex}} x$.

For $x \in \Sigma^n$, we define $\mathcal{C}_x$, $G_{x,p}$ and $G_{x,\leq p}$ as before. We count $G_{x,\leq n}$ using:

$$|G_{x,\leq n}| = |\{y \in \{0,1\}^{tn} \mid y \in \Gamma^n, \exists i < n \text{ s.t. } \mathsf{Rot}^{it}(y) <_{\mathsf{lex}} x\}|.$$

It is easy to efficiently produce an automaton A_0 such that $L(A_0) \cap \{0,1\}^{tn} = \Gamma^n$. As we will describe below, the methods of the previous section can be easily adapted to efficiently produce an automaton A_x such that

$$L(A_x) \cap \{0,1\}^{tn} = \{y \in \{0,1\}^{tn} \mid \exists i < n \text{ s.t. } \mathsf{Rot}^{it}(y) <_{\mathsf{lex}} x\}.$$

The following lemma will be crucial in the design of this automaton.

Lemma 15. *Let $y \in \{0,1\}^{tn}$. There exists $i < n$ such that $\mathsf{Rot}^{it}(y) <_{\mathsf{lex}} x$ if and only if at least one of the following events occurs:*

1. *there exists $w \in L_x$ such that w appears a contiguous substring of y starting at a coordinate j with $j \equiv 0 \mod t$ (where the coordinates of x are $0, 1, \ldots, (tn-1)$).*
2. *there exist strings w_1, w_2 such that $w_1 w_2 \in L_x$, w_2 is a prefix of y, w_1 is a suffix of y, and $|w_1| \equiv 0 \mod t$.*

The construction of A_x now follows easily via the techniques used in the binary case. The main addition is that one needs to remember the current coordinate mod t, which can be done by blowing up the number of states of the automaton by a factor t. Intersecting the accepted sets of A_x and A_0 gives us our desired automaton which allows us to count $|G_{x,\leq n}|$. This easily adapts to also count $|G_{x,\leq p}|$ for each $p \mid n$. Putting everything together, we get:

Theorem 16. *There is an $\mathsf{poly}(n, \log|\Sigma|)$-time indexing algorithm for necklaces of length n over Σ.*

3 Indexing Irreducible Polynomials

Let q be a prime power, and $\mathbb{F}_q$ be the finite field of q elements. For $n > 0$, let $I_{q,n}$ denote the set of monic, irreducible polynomials of degree n in $\mathbb{F}_q[T]$.

Theorem 17. *For every q, n as above, there is an algorithm that runs in $\mathsf{poly}(n, \log q)$ time, takes $n \log q$ bits of advice, and indexes $I_{q,n}$.*

Proof. Let $P(T) \in I_{q,n}$. Note that $P(T)$ has all its roots in the field $\mathbb{F}_{q^n}$. Let $\alpha \in \mathbb{F}_{q^n}$ be one of the roots of $P(T)$. Then we have that $\alpha, \alpha^q, \ldots, \alpha^{q^{n-1}}$ are all distinct, and $P(T) = \prod_{i=0}^{n-1}(T - \alpha^{q^i})$.

Conversely, if we take $\alpha \in \mathbb{F}_{q^n}$ such that $\alpha, \alpha^q, \ldots, \alpha^{q^{n-1}}$ are all distinct, then the polynomial $P(T) = \prod_{i=0}^{n-1}(T - \alpha^{q^i})$ is in $I_{q,n}$.

Define an action of $\mathbb{Z}_n$ on $\mathbb{F}_{q^n}^*$ as follows: for $k \in \mathbb{Z}_n$ and $\alpha \in (\mathbb{F}_{q^n})^*$, define $k[\alpha] = \alpha^{q^k}$. This action partitions $\mathbb{F}_{q^n}^*$ into orbits. By the above discussion, $I_{q,n}$ is in one-to-one correspondence with the orbits of this action with size exactly n. Thus it suffices to index these orbits.

Let g be a generator of the the multiplicative group $(\mathbb{F}_{q^n})^*$. Define a map (bijection) $E : \mathbb{Z}_{q^n-1} \to \mathbb{F}_{q^n}^*$ by $E(a) = g^a$.. Via this bijection, we have an action of $\mathbb{Z}_n$ on $\mathbb{Z}_{q^n-1}$, where for $k \in \mathbb{Z}_n$ and $a \in \mathbb{Z}_{q^n-1}$, $k[a] = q^k \cdot a$.

Now represent elements of $\mathbb{Z}_{q^n-1}$ by integers in $\{0, 1, \ldots, q^n - 2\}$. Define $\Sigma = \{0, 1, \ldots, q-1\}$. For $a \in \mathbb{Z}_{q^n-1}$, consider its base-q expansion $a_\sigma \in \Sigma^n$. This gives us a bijection between $\mathbb{Z}_{q^n-1}$ and $\Sigma^n \setminus \{(q-1, \ldots, q-1)\}$. Via this bijection, we get an action of $\mathbb{Z}_n$ on $\Sigma^n \setminus \{(q-1, \ldots, q-1)\}$. This action is precisely the standard rotation action!

The Indexing Algorithm:
Input: q (a prime power), $n \geq 0$, $i \in [|I_{q,n}|]$
Advice: A description of $\mathbb{F}_q$ and an irreducible polynomial $F(T) \in \mathbb{F}_q[T]$ of degree n, whose root is a generator g of $(\mathbb{F}_{q^n})^*$ (a.k.a. primitive polynomial).

1. Let $\Sigma = \{0, 1, \ldots, q-1\}$.
2. Use i to index an necklace $\sigma \in \Sigma^n \setminus \{(q-1, q-1, \ldots, q-1)\}$ with fundamental period exactly n (via our main theorem).
3. View σ as the base q expansion of an integer $a \in \{0, 1, \ldots, q^n - 2\}$.
4. Use $F(T)$ to construct the finite field $\mathbb{F}_{q^n}$ and the element $g \in \mathbb{F}_{q^n}^*$. (This can be done by setting $\mathbb{F}_{q^n} = \mathbb{F}_q[T]/F(T)$, and taking the class of the element T in that quotient to be the element g.)
5. Set $\alpha = g^a$.
6. Set $P(T) = \prod_{i=0}^{n-1}(T - \alpha^{q^i})$.
7. Output $P(T)$.

For constant q, this algorithm can be made to work with $\mathsf{poly}(\log n)$ advice. Indeed, one can construct the finite field $\mathbb{F}_{q^n}$ in $\mathsf{poly}(q, n)$ time, and a wonderful result of Shoup [21] constructs a set of $q^{\mathsf{poly}(\log n)}$ elements in $\mathbb{F}_{q^n}$, one of which is guaranteed to be a generator. The advice is then the index of an element of this set which is a generator.

4 Open Problems

We conclude with some open problems.

1. Can the orbits of group actions be indexed in general? One formulation of this problem is as follows: Let G be a finite group acting on a set X, both of size $\mathsf{poly}(n)$. Suppose G and its action on X are given as input explicitly. For a finite alphabet Σ, consider the action of G on Σ^X (by permuting coordinates according to the action on X). Can the orbits of this action be indexed? Can they be reverse-indexed?
2. Let G be the symmetric group S_n. Consider its action on $\{0,1\}^{\binom{[n]}{2}}$, where G acts by permuting coordinates. The orbits of this action correspond to the isomorphism classes of n-vertex graphs. Can these orbits be indexed?
   More ambitiously, can these orbits be reverse-indexed? This would imply that graph isomorphism is in P.
3. It would be interesting to explore the complexity theory of indexing and reverse-indexing. Which languages can be indexed efficiently? Can this be characterized in terms of known complexity classes?

References

1. Agrawal, M., Biswas, S.: Primality and identity testing via chinese remaindering. J. ACM 50(4), 429–443 (2003)
2. Alon, N., Goldreich, O., Håstad, J., Peralta, R.: Simple construction of almost k-wise independent random variables. Random Struct. Algorithms 3(3), 289–304 (1992)

3. Andoni, A., Goldberger, A., McGregor, A., Porat, E.: Homomorphic fingerprints under misalignments: sketching edit and shift distances. In: STOC, pp. 931–940 (2013)
4. Arndt, J.: Matters computational. Springer (2011)
5. Berstel, J., Pocchiola, M.: Average cost of duval's algorithm for generating lyndon words. Theoretical Computer Science 132(1), 415–425 (1994)
6. Cattell, K., Ruskey, F., Sawada, J., Serra, M., Miers, C.R.: Fast algorithms to generate necklaces, unlabeled necklaces, and irreducible polynomials over gf(2). Journal of Algorithms 37(2), 267–282 (2000)
7. Duval, J.-P.: Génération d'une section des classes de conjugaison et arbre des mots de lyndon de longueur bornée. Theoretical Computer Science 60(3), 255–283 (1988)
8. Fredricksen, H., Kessler, I.: An algorithm for generating necklaces of beads in two colors. Discrete Mathematics 61(2), 181–188 (1986)
9. Fredricksen, H., Maiorana, J.: Necklaces of beads in k colors and k-ary de bruijn sequences. Discrete Mathematics 23(3), 207–210 (1978)
10. Golomb, S.W.: Irreducible polynomials, synchronizing codes, primitive necklaces and cyclotomic algebra. In: Conference on Combinatorial Math. and Its Applications, pp. 358–370 (1969)
11. Guruswami, V., Kopparty, S.: Explicit subspace designs. Electronic Colloquium on Computational Complexity (ECCC) 20, 60 (2013)
12. Knuth, D.E.: Art of Computer Programming. Fascicle 4, The: Generating All Trees–History of Combinatorial Generation, vol. 4. Addison-Wesley Professional (2006)
13. Kreher, D.L., Stinson, D.R.: Combinatorial algorithms: generation, enumeration, and search, vol. 7. CRC Press (1999)
14. Martínez, C., Molinero, X.: An efficient generic algorithm for the generation of unlabelled cycles. Mathematics and Computer Science III, pp. 187–197. Springer (2004)
15. Myrvold, W., Ruskey, F.: Ranking and unranking permutations in linear time. Information Processing Letters 79(6), 281–284 (2001)
16. Nijenhuis, A., Wilf, H.S.: Combinatorial algorithms for computers and calculators. In: Computer Science and Applied Mathematics, 2nd edn., vol. 1, Academic Press, New York (1978)
17. Rabin, M.O.: Fingerprinting by Random Polynomials. Center for Research in Computing Techn., Aiken Computation Laboratory, Univ. (1981)
18. Ruskey, F.: Combinatorial generation. Draft of a book (2003), `http://www.1stworks.com/ref/RuskeyCombGen.pdf`
19. Ruskey, F., Savage, C., Wang, T.M.Y.: Generating necklaces. Journal of Algorithms 13(3), 414–430 (1992)
20. Ruskey, F., Sawada, J.: An efficient algorithm for generating necklaces with fixed density. SIAM Journal on Computing 29(2), 671–684 (1999)
21. Shoup, V.: Searching for primitive roots in finite fields. In: STOC, pp. 546–554 (1990)
22. Valiant, L.G.: The complexity of computing the permanent. Theor. Comput. Sci. 8, 189–201 (1979)

Coloring Relatives of Interval Overlap Graphs via On-line Games[⋆]

Tomasz Krawczyk[1,⋆⋆] and Bartosz Walczak[1,2,⋆⋆,⋆⋆⋆]

[1] Theoretical Computer Science Department,
Faculty of Mathematics and Computer Science, Jagiellonian University,
Kraków, Poland
{krawczyk,walczak}@tcs.uj.edu.pl
[2] École Polytechnique Fédérale de Lausanne, Switzerland

Abstract. The main goal of this paper is to formalize and explore a connection between chromatic properties of graphs defined by geometric representations and competitivity analysis of on-line algorithms. This connection became apparent after the recent construction of triangle-free geometric intersection graphs with arbitrarily large chromatic number due to Pawlik et al. We show that any on-line graph coloring problem gives rise to a class of *game graphs*, which in many cases have a natural representation by geometric objects. As a consequence, problems of estimating the chromatic number of graphs with geometric representations are reduced to finding on-line coloring algorithms that use few colors or proving that such algorithms do not exist.

We use this framework to derive upper and lower bounds on the maximum possible chromatic number in terms of the clique number in the following classes of graphs: rectangle overlap graphs, subtree overlap graphs and interval filament graphs. These graphs generalize interval overlap graphs (also known as circle graphs)—a well-studied class of graphs with chromatic number bounded by a function of the clique number. Our bounds are absolute for interval filament graphs and asymptotic of the form $(\log\log n)^{f(\omega)}$ for rectangle and subtree overlap graphs. In particular, we provide the first construction of geometric intersection graphs with bounded clique number and with chromatic number asymptotically greater than $\log\log n$. Moreover, with some additional assumptions on the geometric representation, the bounds obtained are tight.

1 Introduction

Graphs represented by geometric objects have been attracting researchers for many reasons, ranging from purely aesthetic to practical ones. A problem which has been extensively studied for this kind of graphs is that of proper coloring: given a family of objects, one wants to color them with few colors so that any two

[⋆] Full version of this paper is available on-line at arXiv:1402.2437.

[⋆⋆] Supported by Polish National Science Center grant 2011/03/B/ST6/01367.

[⋆⋆⋆] Supported by Swiss National Science Foundation grant 200020-144531.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 738–750, 2014.

objects generating an edge of the graph obtain distinct colors. It finds practical applications in areas like channel assignment, map labeling, and VLSI design.

We write χ, ω and n to denote the chromatic number, the clique number (maximum size of a clique), and the number of vertices of a graph under consideration, respectively. If $\chi = \omega$ holds for a graph G and all its induced subgraphs, then G is *perfect.* A class of graphs is *χ-bounded* or *near-perfect* if their chromatic number is bounded by some function of their clique number.

Any finite family of sets $\mathcal{F}$ gives rise to two graphs with vertex set $\mathcal{F}$: the *intersection graph*, whose edges connect pairs of intersecting members of $\mathcal{F}$, and the *overlap graph*, whose edges connect pairs of members of $\mathcal{F}$ that *overlap*, that is, intersect but are not nested. Ranging over all families $\mathcal{F}$ of sets of a particular kind, for example, having a specific geometric shape, we obtain various classes of intersection and overlap graphs. Prototypical examples are *interval graphs* and *interval overlap graphs*, which are intersection and overlap graphs, respectively, of families of closed intervals in $\mathbb{R}$. Interval overlap graphs are isomorphic to *circle graphs*—intersection graphs of chords of a circle.

Interval graphs are well known to be perfect. Interval overlap graphs are no longer perfect, but they are near-perfect, which was shown by Gyárfás [7]. Currently the best upper and lower bounds on the maximum chromatic number of an interval overlap graph with clique number ω are $O(2^{\omega})$ [9] and $\Omega(\omega \log \omega)$ [8]. The exponential gap between them remains open for almost 30 years.

A family of sets $\mathcal{F}$ is *clean* if no set in $\mathcal{F}$ is contained in two other overlapping sets in $\mathcal{F}$. Overlap graphs of clean families are themselves called *clean.* Kostochka and Milans [10] proved that clean interval overlap graphs satisfy $\chi \leqslant 2\omega - 1$.

Intervals in $\mathbb{R}$ are naturally generalized by axis-parallel rectangles in $\mathbb{R}^2$ and by subtrees of a tree, which give rise to the following classes of graphs:

- *chordal graphs*—intersection graphs of families of subtrees of a tree, originally defined as graphs containing no induced cycles of length greater than three,
- *subtree overlap graphs*—overlap graphs of families of subtrees of a tree,
- *rectangle graphs* and *rectangle overlap graphs*—intersection graphs and overlap graphs, respectively, of families of axis-parallel rectangles in the plane.

Chordal graphs are perfect. Rectangle graphs are near-perfect: Asplund and Grünbaum [1] showed that they satisfy $\chi = O(\omega^2)$. Rectangle overlap graphs are no longer near-perfect: Pawlik et al. [12] gave a construction of triangle-free rectangle overlap graphs with chromatic number $\Theta(\log \log n)$. Actually, it produces graphs that we call *interval overlap game graphs.* They form a subclass of rectangle overlap graphs and of subtree overlap graphs, which implies that subtree overlap graphs are not near-perfect either. Interval overlap game graphs play an important role in this paper, but their definition requires some preparation, so it is postponed until Section 4. It is proved in [11] that triangle-free rectangle overlap graphs have chromatic number $O(\log \log n)$.

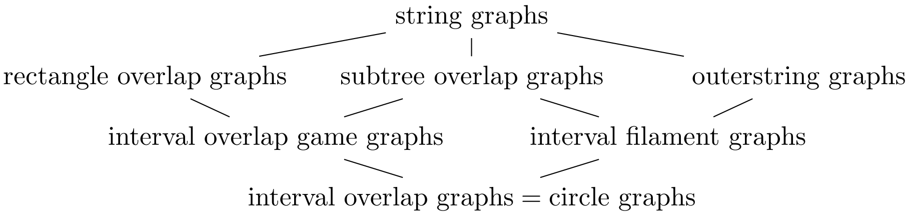
Interval filament graphs are intersection graphs of *interval filaments*, which are continuous non-negative functions defined on closed intervals with value zero on the endpoints. They were introduced in [5] as a generalization of interval overlap graphs, polygon-circle graphs, chordal graphs and co-comparability graphs.

They form a subclass of subtree overlap graphs [2]. On the other hand, for any collection $\mathcal{S}$ of subtrees of a tree T intersecting a common path in T, the overlap graph of $\mathcal{S}$ is an interval filament graph [2]. An interval filament graph is *domain-non-overlapping* if it is an intersection graph of interval filaments with non-overlapping domains.

Outerstring graphs are intersection graphs of curves in a halfplane with one endpoint on the boundary. Every interval filament graph is an outerstring graph.

String graphs are intersection graphs of arbitrary curves in the plane. Every graph of any class considered above is a string graph. For example, a rectangle overlap graph can be represented as an intersection graph of boundaries of rectangles, and the overlap graph of a family of subtrees of a tree T can be represented as the intersection graph of closed curves encompassing these subtrees in a planar drawing of T. The best known upper bound on the chromatic number of string graphs is $(\log n)^{O(\log \omega)}$ due to Fox and Pach [4].

The following diagram illustrates the inclusions between most of the classes defined above:



Here is the summary of the results of this paper. In what follows, we write O_ω and Θ_ω to denote the asymptotics with ω fixed as a constant.

Theorem 1. (1) *Every interval filament graph satisfies* $\chi = O\bigl(2^\omega \binom{\omega+1}{2}\bigr)$.
(2) *Every domain-non-overlapping interval filament graph satisfies* $\chi \leqslant \binom{\omega+1}{2}$.
(3) *There are domain-non-overlapping interval filament graphs with* $\chi = \binom{\omega+1}{2}$.

Theorem 2. (1) *Every subtree overlap graph safisfies* $\chi = O_\omega((\log\log n)^{\binom{\omega}{2}})$.
(2) *Every clean subtree overlap graph satisfies* $\chi = O_\omega((\log\log n)^{\omega-1})$.
(3) *There are clean subtree overlap graphs with* $\chi = \Theta_\omega((\log\log n)^{\omega-1})$.

Theorem 3. (1) *Every rectangle overlap graph satisfies* $\chi = O_\omega((\log\log n)^{\omega-1})$.
(2) *Every clean rectangle overlap graph satisfies* $\chi = O_\omega(\log\log n)$.

Theorem 3 for $\omega = 2$ was proved in [11]. The construction of triangle-free rectangle overlap graphs with chromatic number $\Theta(\log\log n)$ due to Pawlik et al. [12] shows that the bound of Theorem 3 (2) is asymptotically tight. It also proves Theorem 2 (3) for $\omega = 2$, which we comment on in Section 4. Theorem 2 (3) provides the first construction of string graphs with bounded clique number and with chromatic number asymptotically greater than $\log\log n$. Theorem 2 (2) and (3) implies that for every $k \geqslant 2$, there are string graphs with clique number k that do not have a K_k-free coloring (i.e. a coloring of vertices such that every color class induces a K_k-free subgraph) with fewer than $\Theta_k(\log\log n)$ colors.

Theorem 1 (1) asserts in particular that interval filament graphs are χ-bounded. This is also implied by a very recent result of Rok and Walczak [13] that outerstring graphs are χ-bounded. That result is proved using different techniques, and the bound resulting from that proof is enormous.

The proofs of the upper bounds in Theorems 1–3 are constructive—they can be used to design polynomial-time algorithms that given a graph and its geometric representation, produce a proper coloring with the claimed number of colors.

All our proofs heavily depend on the correspondence between on-line graph coloring games and off-line colorings of so-called *game graphs*, which originates from considerations in [11,12] and which we formalize in the next section. This approach is the only one known to give upper bounds better than single logarithmic on the chromatic number in those classes of string graphs with bounded clique number that do not allow a constant bound. We believe that its further exploration will result in upper bounds of the form $\chi = O((\log\log n)^{f(\omega)})$ for broad classes of graphs with geometric representation that are not χ-bounded.

2 On-line Graph Coloring Games and Game Graphs

The *on-line graph coloring game* is played by two deterministic players: Presenter and Algorithm. It is played in rounds. In each round, Presenter introduces a new vertex of the graph and defines whether or not it has an edge to each of the vertices presented before. In the same round, Algorithm colors this vertex keeping the property that the coloring is proper. Imposing additional restrictions on Presenter's moves gives rise to many possible variants of the on-line graph coloring game. Typical kinds of such restrictions look as follows:

(i) Presenter builds a graph G from a specific class.
(ii) Presenter builds a mapping $\mu\colon V(G) \to \mathcal{C}$ called a *representation* of G in some class of objects $\mathcal{C}$, and the edges of G are defined from μ.
(iii) Presenter builds relations $R_1, \dots, R_r$ on $V(G)$ defining the edges of G.
(iv) There can be some restrictions relating μ, $R_1, \dots, R_r$, and the order in which the vertices are presented.

The decisions of both players are irrevocable. That is, Presenter cannot change the graph, the representation or the relations once they have been set, and Algorithm cannot change colors once they have been assigned. The goal of Algorithm is to keep using as few colors as possible, while Presenter wants to force Algorithm to use as many colors as possible. The *value* of such a game is the minimum number c such that Algorithm has a strategy to color any graph that can be presented using at most c colors or, equivalently, the maximum number c such that Presenter has a strategy to force Algorithm to use at least c colors regardless of how Algorithm responds.

For convenience, we call any variant of the on-line graph coloring game simply an *on-line game*, and any coloring strategy of Algorithm simply an *on-line algorithm*. We denote by $\prec$ the order in which the vertices are presented. Now, we explain the crucial concept of our paper—game graphs.

Let G be an on-line game with representation μ in a class $\mathcal{C}$ and relations $R_1, \ldots, R_r$. An *n-round presentation scenario* in G is a graph G with representation $\mu\colon V(G) \to \mathcal{C}$, relations $R_1, \ldots, R_r$ on $V(G)$, and an order $\prec$ on $V(G)$, which can possibly have been presented in G after n rounds so that $\prec$ is the order of presentation. We define the class of *game graphs* associated with G as follows. A graph G is a *game graph* of G if there exist a rooted forest F on $V(G)$, a mapping $\mu\colon V(G) \to \mathcal{C}$, and relations $R_1, \ldots, R_r$ on $V(G)$ such that

(i) for every $v \in V(G)$, the subgraph $G[V(P_v)]$ of G induced on the vertices of the path P_v in F from a root to v, the representation μ restricted to $V(P_v)$, the relations $R_1, \ldots, R_r$ restricted to $V(P_v)$, and the order $\prec$ of vertices along P_v give a valid $|V(P_v)|$-round presentation scenario in G,

(ii) if $uv \in E(G)$, then u is an ancestor of v or v is an ancestor of u in F.

For two distinct vertices u and v of a game graph, we write $u \prec v$ to denote that u is an ancestor of v in F. Therefore, the relations $\prec$ in the on-line game and in the game graph correspond to each other the same way as $R_1, \ldots, R_r$. A game graph can be viewed as a union of several presentation scenarios in which some (not necessarily all) common prefixes of these scenarios have been identified.

Lemma 1. *If there is an on-line algorithm using at most c colors in an on-line game* G*, then every game graph of* G *has chromatic number at most c.*

Proof (sketch). To color a game graph properly, it is enough to run the on-line algorithm on the subgraph induced on every path in F from a root to a leaf. □

We say that a strategy of Presenter in a game G is *finite* if the number of all presentation scenarios that can occur in the game when Presenter plays according to this strategy, for all possible responses of Algorithm, is finite.

Lemma 2. *If Presenter has a finite strategy to force Algorithm to use at least c colors in an on-line game* G*, then there exists a game graph of* G *with chromatic number at least c. Moreover, the number of vertices of this graph is equal to the number of presentation scenarios possible with this strategy.*

Proof (sketch). On the set S of all presentation scenarios possible with Presenter's considered strategy, we define a relation $\prec$ so that $u \prec v$ if and only if the scenario u is an initial part of the scenario v (after only a part of the vertices of v have been presented). Clearly, $\prec$ is the ancestor-descendant relation of a tree F on S whose root is the empty scenario. There is a unique way of defining a game graph G on S so that the presentation scenario induced on each path P_v, as explained in condition (i) above, is isomorphic to v. If G has a proper coloring with fewer than c colors, then Algorithm can use it to beat Presenter's considered strategy using fewer than c colors, which is a contradiction. □

Here is how Lemmas 1 and 2 are typically used. To provide an upper bound on the chromatic number of graphs of some class $\mathcal{G}$, we show that each graph in $\mathcal{G}$ is a game graph of an appropriately chosen on-line game, and we find an on-line algorithm in this game using few colors. To construct graphs of some class $\mathcal{G}$ with large chromatic number, we show that every game graph of an appropriately chosen on-line game is isomorphic to a graph in $\mathcal{G}$, and we find a finite strategy of Presenter in this game forcing Algorithm to use many colors.

We use this approach to prove the results of the paper. First, we reduce Theorems 1–3 to claims about game graphs of appropriately chosen on-line games. Then, to prove these claims, we devise strategies for Algorithm and Presenter in these games and apply Lemmas 1 and 2 accordingly.

3 Interval Filament Graphs

The following lemma reduces the general problem of coloring interval filament graphs to the problem for domain-non-overlapping interval filament graphs.

Lemma 3. *The vertices of every interval filament graph can be partitioned into $O(2^{\omega})$ classes each containing interval filaments with non-overlapping domains.*

The *incomparability graph* of a set P partially ordered by a relation $<$ is the graph with vertex set P and edge set consisting of pairs of $<$-incomparable elements of P. A graph is a *co-comparability graph* if it is the incomparability graph of some partially ordered set. Consider an on-line game $\mathsf{COCO}(k)$ on the class of co-comparability graphs with clique number at most k presented with their ordering relation in the *up-growing* manner. That is, Presenter builds a co-comparability graph G and an order $<$ on $V(G)$, in each round defining the relation $<$ between the new vertex and the vertices presented before, so that

(i) G is the incomparability graph of $V(G)$ with respect to the order $<$,
(ii) every vertex of G is $<$-maximal at the moment it is presented,
(iii) $\omega(G) \leqslant k$, that is, the width of $V(G)$ with respect to $<$ is at most k.

Lemma 4. *A graph G is a game graph of $\mathsf{COCO}(k)$ if and only if G is isomorphic to a domain-non-overlapping interval filament graph and $\omega(G) \leqslant k$.*

Proof (sketch). Let G be a domain-non-overlapping interval filament graph with $\omega(G) \leqslant k$. The inclusion order on the domains of members of $V(G)$ defines a forest F on $V(G)$ in which u is an ancestor of v if and only if $\mathrm{dom}(u) \supset \mathrm{dom}(v)$. We define a relation $<$ on $V(G)$ so that $u < v$ if $\mathrm{dom}(u) \supset \mathrm{dom}(v)$ and $u \cap v = \emptyset$. Consider the path P_v in F from a root to a vertex v. It is easy to check that the graph $G[V(P_v)]$, the order $<$ restricted to $V(P_v)$, and the order $\prec$ of vertices along P_v form a valid $|V(P_v)|$-round presentation scenario in $\mathsf{COCO}(k)$.

Now, let G be a game graph of $\mathsf{COCO}(k)$ with underlying forest F and relation $<$. Let L denote the set of leaves of F and $L(u)$ denote the set of leaves of F that are descendants of a vertex u in F. We assign pairwise disjoint intervals $[x_v, y_v] \subset \mathbb{R}$ to the leaves $v \in L$ so that the order of these intervals corresponds to the depth-first search order of the leaves. For $v \in L$, let P_v denote the path in F from a root to v. The graph $G[V(P_v)]$ is the incomparability graph of the order $<$ restricted to $V(G)$, so it can be represented as an intersection graph of continuous functions $[x_v, y_v] \to (0, \infty)$ so that whenever $u_1, u_2 \in V(P_v)$ and $u_1 < u_2$, the function representing u_1 lies above the function representing u_2, see [6]. This way each vertex $u \in V(G)$ is drawn as a continuous function $\bigcup_{v \in L(u)}[x_v, y_v] \to (0, \infty)$, and G is isomorphic to the intersection graph of these functions. It is not difficult to see that these functions can be extended to interval filaments with non-overlapping domains without changing their intersection graph. □

Theorem 4 (Felsner [3]). *The value of the game* $\mathsf{COCO}(k)$ *is* $\binom{k+1}{2}$*. That is, there is an on-line coloring algorithm using at most* $\binom{k+1}{2}$ *colors, and there is a finite strategy of Presenter forcing Algorithm to use* $\binom{k+1}{2}$ *colors in* $\mathsf{COCO}(k)$*.*

Theorem 1 (2) and (3) follows from Theorem 4, Lemma 4, and Lemmas 1 and 2 (respectively). Theorem 1 (1) follows from Theorem 1 (2) and Lemma 3.

4 Rectangle and Subtree Overlap Graphs

The following theorem, whose proof we omit, reduces the general problem of coloring overlap graphs to the problem for clean overlap graphs.

Theorem 5. *Let G be an overlap graph. If every clean induced subgraph H of G with $\omega(H) \leqslant j$ satisfies $\chi(H) \leqslant \alpha_j$ for $2 \leqslant j \leqslant \omega(G)$, then $\chi(G) \leqslant \prod_{j=2}^{\omega(G)} 2\alpha_j$.*

First, we introduce the on-line game corresponding to clean rectangle overlap graphs, and we define interval overlap game graphs. Let $\mathcal{I}$ denote the set of closed intervals in $\mathbb{R}$. Let $\ell(x)$ and $r(x)$ denote the left and the right endpoint of an interval $x \in \mathcal{I}$, respectively. Consider an on-line game $\mathsf{IOV}(k)$, in which Presenter builds an interval overlap graph G and a representation $\mu\colon V(G) \to \mathcal{I}$ so that

(i) μ is the overlap model of G ($xy \in E(G)$ if and only if $\mu(x)$ and $\mu(y)$ overlap),
(ii) if $x, y \in V(G)$ and x is presented before y, then $\ell(\mu(x)) < \ell(\mu(y))$,
(iii) the family of intervals $\{\mu(x)\colon x \in V(G)\}$ is clean, that is, there are no $x, y, z \in V(G)$ such that $\mu(x)$ and $\mu(y)$ overlap and $\mu(z) \subset \mu(x) \cap \mu(y)$,
(iv) $\omega(G) \leqslant k$.

As a consequence of the definition of a game graph, a graph G is a game graph of $\mathsf{IOV}(k)$ if there exist a rooted forest F on $V(G)$ and a mapping $\mu\colon V(G) \to \mathcal{I}$ such that the following conditions, corresponding to the four above, are satisfied:

(i) $xy \in E(G)$ if and only if $x \prec y$ or $y \prec x$ and $\mu(x)$ overlaps $\mu(y)$,
(ii) if $x, y \in V(G)$ and $x \prec y$, then $\ell(\mu(x)) < \ell(\mu(y))$,
(iii) there are no $x, y, z \in V(G)$ with $x \prec y \prec z$ such that $\mu(x)$ and $\mu(y)$ overlap and $\mu(z) \subset \mu(x) \cap \mu(y)$,
(iv) $\omega(G) \leqslant k$.

A graph is an *interval overlap game graph* if it is a game graph of $\mathsf{IOV}(k)$ for some k. The characterization above (without condition (iv)) was used in [11] as the definition of an interval overlap game graph.

Lemma 5 (Krawczyk, Pawlik, Walczak [11]). *Every interval overlap game graph is a clean rectangle overlap graph. The vertices of every clean rectangle overlap graph can be partitioned into $O_\omega(1)$ classes so that the subgraph induced on each class is an interval overlap game graph.*

It is proved in [11] that triangle-free interval overlap game graphs (and so, by Lemma 5, triangle-free clean rectangle overlap graphs) satisfy $\chi = O(\log\log n)$. That proof essentially comes down to an on-line algorithm using $O(\log r)$ colors in r rounds of the game $\mathsf{IOV}(2)$, the trick with heavy-light decomposition that we explain later, and the application of Lemma 1. We are going to generalize this to game graphs of $\mathsf{IOV}(k)$ and thus to clean rectangle overlap graphs with any fixed clique number. On the other hand, it is proved in [12] that Presenter has a strategy to force Algorithm to use m colors in 2^{m-1} rounds of the game $\mathsf{IOV}(2)$, and this strategy has $2^{2^{O(m)}}$ presentation scenarios. Hence Lemma 2 implies that there are triangle-free interval overlap game graphs (and therefore triangle-free clean rectangle overlap graphs) with chromatic number $\Omega(\log\log n)$.

Now, we are going to define the on-line game corresponding to clean subtree overlap graphs. Let G be a clean subtree overlap graph with underlying tree T.That is, $V(G)$ is a clean family of subtrees of T whose overlap graph is G. To avoid confusion with vertices of G, we call vertices of T *nodes*. We make T a rooted tree by choosing an arbitrary node r as the root. For every $S \in V(G)$, we define r_S to be the unique node in S that is closest to r in T. We call the nodes r_S *subtree roots*. We can assume without loss of generality that all subtree roots are pairwise distinct. Now, we construct a rooted forest F on $V(G)$ as follows. A subtree $S \in V(G)$ is a root of F if the path from r to r_S in T contains no other subtree roots. Otherwise, the parent of S in F is the subtree $S' \in V(G)$ such that $r_{S'}$ is the last subtree root before r_S on the path from r to r_S in T.

Overlap graphs of subtrees intersecting a common path of the underlying tree are isomorphic to interval filament graphs [2]. Hence for every path P in F starting at a root of F, the graph $G[V(P)]$ is isomorphic to an interval filament graph. We will color G properly using the on-line approach of Lemma 1. For each path P in F starting at a root, we will simulate an on-line algorithm on $G[V(P)]$ presenting the vertices in their order along P. This will work correctly if the algorithm always assigns the same color to each subtree $S \in V(G)$, regardless of the choice of P. This will be the case when the presentation scenarios up to the point when S is presented are identical for all paths passing through S. However, this cannot be guaranteed using the representation of $G[V(P)]$ by interval filaments. For example, for some two overlapping subtrees $S_1, S_2 \in V(G)$ lying on the common part of two paths P_1 and P_2, we may need to represent S_1 and S_2 by interval filaments whose domains are nested if we continue along P_1, but overlap if we continue along P_2. If the algorithm makes use of the representation, then the colorings it generates on P_1 and P_2 may be inconsistent.

We overcome this difficulty providing a more abstract description of G, which we then use to define the on-line game. For distinct subtrees $x, y \in V(G)$, let $x \prec y$ denote that x is an ancestor of y in F. We partition the relation $\prec$ on $V(G)$ into three relations $\Supset$, $\between$ and $\|$ as follows:

- $x \Supset y$ if $x \prec y$ and the subtree x contains the subtree y,
- $x \between y$ if $x \prec y$ and the subtrees x and y overlap,
- $x \| y$ if $x \prec y$ and the subtrees x and y are disjoint.

This implies the following, for $x, y, z \in V(G)$:

(A1) if $x \Supset y$ and $y \Supset z$, then $x \Supset z$,
(A2) if $x \Supset y$ and $y \between z$, then $x \Supset z$ or $x \between z$,
(A3) if $x \between y$ and $y \Supset z$, then $x \between z$ or $x \parallel z$,
(A4) if $x \parallel y$ and $y \prec z$, then $x \parallel z$.

We define an on-line game $\mathsf{ABS}(k)$ in which Presenter builds a graph G together with relations $\Supset$, $\between$ and $\parallel$, in each round defining the relations $\Supset$, $\between$ and $\parallel$ between the new vertex and the vertices presented before, so that

(i) $\Supset$, $\between$ and $\parallel$ partition the order of presentation $\prec$ and satisfy (A1)–(A4),
(ii) $xy \in E(G)$ if and only if $x \between y$ or $y \between x$,
(iii) $\omega(G) \leqslant k$.

Lemma 6. *A graph G is a game graph of* $\mathsf{ABS}(k)$ *if and only if G is isomorphic to a clean subtree overlap graph. If G is a game graph of* $\mathsf{ABS}(k)$ *and the relation $\prec$ is a total order on $V(G)$, then G is isomorphic to an interval filament graph.*

Proof (sketch). We have just argued that every clean subtree overlap graph with clique number at most k is a game graph of $\mathsf{ABS}(k)$. Now, suppose that G is a game graph of $\mathsf{ABS}(k)$ with underlying forest F. Let T be a tree with

$$\begin{aligned} V(T) &= \{r\} \cup \{u_x \colon x \in V(G)\} \cup \{v_x \colon x \in V(G)\}, \\ E(T) &= \{ru_x \colon x \text{ is a root of } F\} \cup \{u_x u_y \colon xy \in E(F)\} \cup \{u_x v_x \colon x \in V(G)\}. \end{aligned}$$

For $x \in V(G)$, let $V_x = \{u_x, v_x\} \cup \{u_y \colon x \Supset y \text{ or } x \between y\} \cup \{v_y \colon x \Supset y\}$. It is not difficult to prove that $\{V_x \colon x \in V(G)\}$ is a clean family of node sets of subtrees of T whose overlap graph is isomorphic to G. If $\prec$ is a total order on $V(G)$, then the nodes u_x form a path in T, which implies, by the result of [2], that the overlap graph of $\{V_x \colon x \in V(G)\}$ is isomorphic to an interval filament graph. □

The game $\mathsf{IOV}(k)$ is more restricted for Presenter than $\mathsf{ABS}(k)$, in the sense that every presentation scenario in the former can be translated into a presentation scenario in the latter. Indeed, let G be a graph presented in $\mathsf{IOV}(k)$ together with representation $\mu \colon V(G) \to \mathcal{I}$ and order of presentation $\prec$. We can define relations $\Supset$, $\between$ and $\parallel$ on $V(G)$ just as before:

- $x \Supset y$ if $x \prec y$ and the interval $\mu(x)$ contains the interval $\mu(y)$,
- $x \between y$ if $x \prec y$ and the intervals $\mu(x)$ and $\mu(y)$ overlap,
- $x \parallel y$ if $x \prec y$ and the intervals $\mu(x)$ and $\mu(y)$ are disjoint.

Clearly, the conditions (i)–(iii) of $\mathsf{ABS}(k)$ are satisfied. This and Lemma 6 imply that every interval overlap game graph is a clean subtree overlap graph.

We are going to prove that game graphs of $\mathsf{ABS}(k)$ have chromatic number $O((\log\log n)^{k-1})$ and those of $\mathsf{IOV}(k)$ have chromatic number $O(\log\log n)$. Then, the same bounds on the chromatic number of clean rectangle and subtree overlap graphs (respectively) will follow from Lemmas 6 and 5 (respectively).

The general idea is that we provide on-line algorithms in $\mathsf{ABS}(k)$ and $\mathsf{IOV}(k)$ using few colors, and then use Lemma 1 to derive an upper bound on the chromatic number of their game graphs. However, since Presenter has a strategy in $\mathsf{IOV}(2)$ to force Algorithm to use $\Omega(\log r)$ colors in r rounds, direct application of Lemma 1 to the game graph cannot succeed if the rooted forest F underlying the game graph contains long paths. To solve this problem, we use the technique of *heavy-light decomposition* due to Sleator and Tarjan [14].

Let G be a game graph of $\mathsf{ABS}(k)$ or $\mathsf{IOV}(k)$ with underlying forest F. We call an edge uv of F, where v is a child of u, *heavy* if the subtree of F rooted at v contains more than half of the vertices of the subtree of F rooted at u and *light* otherwise. Every vertex of F has a heavy edge to at most one of its children, so the heavy edges induce in F a collection of paths, called *heavy paths.*

Lemma 7 (Sleator, Tarjan [14]). *Every path in F from a root to a leaf contains at most $\lfloor \log_2 n \rfloor$ light edges.*

Let $b = \lfloor \log_2 n \rfloor + 1$. For each heavy path P, by Lemma 6, the graph $G[V(P)]$ is an interval filament graph, so by Theorem 1 (1), it can be colored properly using $O_k(1)$ colors. We use the same set of colors on each heavy path, so that we use $O_k(1)$ colors in total. It easily follows from the definitions of $\mathsf{ABS}(k)$ and $\mathsf{IOV}(k)$ that induced subgraphs of their game graphs are also their game graphs. Hence we are going to color the subgraph of G induced on each color class separately by an on-line algorithm, using $O_k((\log b)^{k-1})$ colors in $\mathsf{ABS}(k)$ and $O_k(\log b)$ colors in $\mathsf{IOV}(k)$. Formally, we define on-line games $\mathsf{ABS}(k, b)$ and $\mathsf{IOV}(k, b)$ like $\mathsf{ABS}(k)$ and $\mathsf{IOV}(k)$, respectively, but with one additional constraint:

(v) there is a partition of $V(G)$ into at most b blocks of vertices consecutive in the order $\prec$ such that no edge of G connects vertices in the same block.

It follows from Lemma 7 and the definition of b that the game graph of $\mathsf{ABS}(k)$ or $\mathsf{IOV}(k)$ induced on each color class as explained above is a game graph of $\mathsf{ABS}(k, b)$ or $\mathsf{IOV}(k, b)$, respectively. We are going to prove the following.

Lemma 8. *There is an on-line $O_k((\log b)^{k-1})$-coloring algorithm in $\mathsf{ABS}(k, b)$.*

Lemma 9. *There is an on-line $O_k(\log b)$-coloring algorithm in $\mathsf{IOV}(k, b)$.*

For the next part of this section, we are in the setting of Lemma 8: a graph G with relations $\Supset$, $\between$ and $\parallel$ is presented in the game $\mathsf{ABS}(k, b)$. We are to color G properly using $O_k((\log b)^{k-1})$ colors on-line. Whatever we show for $\mathsf{ABS}(k, b)$ applies also to $\mathsf{IOV}(k, b)$, as it is more restricted for Presenter. The proof of Lemma 9 will differ only in the last part, where the use of a direct argument instead of induction will allow us to reduce the number of colors to $O_k(\log b)$.

As the vertices of G are presented, we classify them as *primary* or *secondary* according to the following on-line rule: if there are $x, y \in V(G)$ such that y is primary, $x \between y, z$ and $y \Supset z$, then z is secondary; otherwise z is primary. Let P be the set of primary vertices, built on-line during the game. For every $y \in P$, let $S(y)$ be the set containing y and all secondary vertices z for which y is the rightmost vertex with the property that $y \Supset z$ and there is x with $x \between y, z$, also built on-line during the game. The sets $S(p)$ for $p \in P$ partition the entire $V(G)$.

First, we reduce on-line coloring of G to on-line colorings of $G[P]$ and $G[S(p)]$. Then, we show how to color $G[P]$ on-line using $O_k(\log b)$ colors. Finally, we apply induction to deduce an on-line $O_k((\log b)^{k-1})$-coloring algorithm for the whole G.

Lemma 10. *If $G[P]$ can be properly colored on-line using at most a colors and $G[S(p)]$ can be properly colored on-line using at most c colors for each $p \in P$, then G can be properly colored on-line using at most $2ac$ colors.*

Proof (sketch). Any independent set I in $G[P]$ satisfies the following:

(i) If $p, q \in I$, $p \prec q$, $x \in S(p)$, $y \in S(q)$, and $xy \in E(G)$, then $x \between q$.

(ii) For every $q \in I$, there is at most one $p \in I$ such that $p \prec q$ and there is $x \in S(p)$ with $x \between q$.

For $p \in P$, let $\phi(p)$ denote the color of p in an on-line proper coloring of $G[P]$ using colors $1, \dots, a$. By (ii), for each color $i \in \{1, \dots, a\}$, the set $I_i = \{p \in P \colon \phi(p) = i\}$ can be further colored on-line using two colors so as to distinguish any $p, q \in I_i$ such that $p \prec q$ and there is $x \in S(p)$ with $x \between q$. Let ψ be such a 2-coloring of each I_i using colors 1 and 2. For $p \in P$ and $x \in S(p)$, let $\xi(x)$ denote the color of x in an on-line proper coloring of $G[S(p)]$ using colors $1, \dots, c$. We color each vertex $x \in S(p)$ by the triple $(\phi(p), \psi(p), \xi(x))$. Hence we use at most $2ac$ colors. This is a proper coloring, because if $p, q \in I_i$, $x \in S(p)$, $y \in S(q)$, and $xy \in E(G)$, then $xq \in E(G)$ by (i), so $\psi(p) \neq \psi(q)$. □

First-fit is the on-line algorithm coloring the graph properly with color set $\{1, 2, \dots\}$ so that when a new vertex is presented, it is assigned the minimum color that has not been used on any of its neighbors presented before.

Theorem 6 (Folklore). *First-fit uses at most $\lfloor \log_2 n \rfloor + 1$ colors on any forest with n vertices presented in any order.*

Lemma 11. *$G[P]$ can be properly colored on-line using $O_k(\log b)$ colors.*

Proof (sketch). We partition the vertices of $G[P]$ on-line into sets $P_1, \dots, P_k$ so that the following holds for any $x, y, z \in P_i$ and $1 \leqslant i \leqslant k$:

$$\text{if } x \between y \prec z, \text{ then } x \parallel z \text{ or } y \parallel z. \qquad (*)$$

To this end, we use the following two observations:

(i) If x, y, z do not satisfy $(*)$, then neither do x, y, y' for any y' with $y \prec y' \prec z$,

(ii) If x, y, z are in P and do not satisfy $(*)$, then $y \between z$.

At the time when a vertex $z \in P$ is presented, consider the set Y of all vertices $y \in P$ for which there exists $x \in P$ such that x, y, z do not satisfy $(*)$. It follows from (i) and (ii) that $Y \cup \{z\}$ is a clique in $G[P]$, hence $|Y| \leqslant k - 1$. We add z to that of the sets $P_1, \dots, P_k$ which is disjoint from Y.

We color each $G[P_i]$ by first-fit using a separate set of colors $\{1^i, 2^i, \dots\}$. We will prove that this way we use $O(\log b)$ colors on each set P_i. This will imply that we have used $O_k(\log b)$ colors on the entire P. Let $(c_i)^i$ be the maximum color used by first-fit on P_i. Let R_i denote the set of vertices in P_i that have no neighbor to the right in $G[P_i]$. The following is a not difficult consequence of $(*)$:

(iii) Each member of $P_i \setminus R_i$ has at most one neighbor to the right in $G[P_i \setminus R_i]$.

In particular, $G[P_i \setminus R_i]$ is a forest. Clearly, the colors of vertices in $P_i \setminus R_i$ do not depend on the colors of vertices in R_i. In particular, if we ran first-fit only on the graph $G[P_i \setminus R_i]$, then we would obtain exactly the same colors on the vertices in $P_i \setminus R_i$. Since there is a vertex in P_i with color $(c_i)^i$, there must be a vertex in $P_i \setminus R_i$ with color $(c_i - 1)^i$. This is enough to conclude, by Theorem 6, that $c_i \leqslant \lfloor \log_2 |P_i| \rfloor + 2$. We can additionaly show that each edge-free block of consecutive vertices in P_i can contain at most one vertex with color greater than 1^i. An analogous argument then yields $c_i \leqslant \lfloor \log_2 b \rfloor + 3$. □

Lemma 12. *Let $p \in P$. There is $x \in V(G)$ such that $x \between s$ for every $s \in S(p)$.*

Proof (Lemma 8). The proof goes by induction on k. The case $k = 1$ is trivial, so assume $k \geqslant 2$. It follows from Lemma 12 that $\omega(G[S(p)]) \leqslant k-1$ for every $p \in P$. Therefore, by the induction hypothesis, $G[S(p)]$ can be properly colored on-line using $O_k((\log b)^{k-2})$ colors. This and Lemma 11 imply that the assumptions of Lemma 10 are satisfied with $a = O_k(\log b)$ and $c = O_k((\log b)^{k-2})$. Hence we conclude that G can be properly colored on-line using $O_k((\log b)^{k-1})$ colors. □

Proof (Lemma 9). Consider one of the sets $S(p)$ built during the game. At each point of the game, by Lemma 12, there is an interval $x \prec p$ that overlaps every interval in $S(p)$. Hence the intervals in $S(p)$ have non-empty intersection and $\omega(G[S(p)]) \leqslant k-1$. Define a partial order $<$ on $S(p)$ so that $s_1 < s_2$ whenever $\ell(s_1) < \ell(s_2)$ and $r(s_1) \geqslant r(s_2)$. It follows that $G[S(p)]$ is the incomparability graph of $S(p)$ with respect to $<$. Moreover, the set $S(p)$ is built in the up-growing manner with respect to $<$. Therefore, by Theorem 4, the graph $G[S(p)]$ can be properly colored on-line using $\binom{k}{2}$ colors. This and Lemma 11 imply that the assumptions of Lemma 10 are satisfied with $a = O_k(\log b)$ and $c = \binom{k}{2}$. Hence we conclude that G can be properly colored on-line using $O_k(\log b)$ colors. □

Theorems 2 (1), (2) and 3 now follow from Theorem 5, Lemmas 6 and 5 (respectively), Lemmas 8 and 9 (respectively), Lemma 1, and $b = \lfloor \log_2 n \rfloor + 1$.

The following allows us to conclude that the coloring algorithm of clean subtree overlap graphs presented above is asymptotically optimal.

Lemma 13. *For $k, m \geqslant 1$, Presenter has a finite strategy to force Algorithm to use at least m^{k-1} colors in the game $\mathsf{ABS}(k)$. Moreover, the number of presentation scenarios for all possible responses of Algorithm is $2^{2^{O_k(m)}}$.*

Theorem 2 (3) now follows from Lemmas 13, 2 and 6.

References

1. Asplund, E., Grünbaum, B.: On a colouring problem. Math. Scand. 8, 181–188 (1960)
2. Enright, J., Stewart, L.: Subtree filament graphs are subtree overlap graphs. Inform. Process. Lett. 104(6), 228–232 (2007)
3. Felsner, S.: On-line chain partitions of orders. Theoret. Comput. Sci. 175(2), 283–292 (1997)

4. Fox, J., Pach, J.: Applications of a new separator theorem for string graphs. Combin. Prob. Comput. 23(1), 66–74 (2014)
5. Gavril, F.: Maximum weight independent sets and cliques in intersection graphs of filaments. Inform. Process. Lett. 73(5-6), 181–188 (2000)
6. Golumbic, M.C., Rotem, D., Urrutia, J.: Comparability graphs and intersection graphs. Discrete Math. 43(1), 37–46 (1983)
7. Gyárfás, A.: On the chromatic number of multiple interval graphs and overlap graphs. Discrete Math. 55(2), 161–166 (1985); Corrigendum: Discrete Math. 62(3), 333 (1986)
8. Kostochka, A.: On upper bounds for the chromatic numbers of graphs. Trudy Inst. Mat. 10, 204–226 (1988)
9. Kostochka, A., Kratochvíl, J.: Covering and coloring polygon-circle graphs. Discrete Math. 163(1-3), 299–305 (1997)
10. Kostochka, A., Milans, K.: Coloring clean and K_4-free circle graphs. In: Pach, J. (ed.) Thirty Essays on Geometric Graph Theory, pp. 399–414. Springer (2012)
11. Krawczyk, T., Pawlik, A., Walczak, B.: Coloring triangle-free rectangle overlap graphs with $O(\log \log n)$ colors, arXiv:1301.0541 (submitted)
12. Pawlik, A., Kozik, J., Krawczyk, T., Lasoń, M., Micek, P., Trotter, W.T., Walczak, B.: Triangle-free geometric intersection graphs with large chromatic number. Discrete Comput. Geom. 50(3), 714–726 (2013)
13. Rok, A., Walczak, B.: Outerstring graphs are χ-bounded. In: 30th Annual Symposium on Computational Geometry (SoCG 2014), arXiv:1312.1559 (to appear, 2014)
14. Sleator, D.D., Tarjan, R.E.: A data structure for dynamic trees. J. Comput. System Sci. 26(3), 362–391 (1983)

Superpolynomial Lower Bounds for General Homogeneous Depth 4 Arithmetic Circuits*

Mrinal Kumar[1] and Shubhangi Saraf[2,**]

[1] Department of Computer Science, Rutgers University
[2] Department of Computer Science and Department of Mathematics,
Rutgers University

Abstract. In this paper, we prove superpolynomial lower bounds for the class of homogeneous depth 4 arithmetic circuits. We give an explicit polynomial in VNP of degree n in n^2 variables such that any homogeneous depth 4 arithmetic circuit computing it must have size $n^{\Omega(\log \log n)}$.

Our results extend the works of Nisan-Wigderson [13] (which showed superpolynomial lower bounds for homogeneous depth 3 circuits), Gupta-Kamath-Kayal-Saptharishi and Kayal-Saha-Saptharishi [4, 7] (which showed superpolynomial lower bounds for homogeneous depth 4 circuits with bounded bottom fan-in), Kumar-Saraf [9] (which showed superpolynomial lower bounds for homogeneous depth 4 circuits with bounded top fan-in) and Raz-Yehudayoff and Fournier-Limaye-Malod-Srinivasan [3,14] (which showed superpolynomial lower bounds for multilinear depth 4 circuits). Several of these results in fact showed exponential lower bounds.

The main ingredient in our proof is a new complexity measure of *bounded support* shifted partial derivatives. This measure allows us to prove exponential lower bounds for homogeneous depth 4 circuits where all the monomials computed at the bottom layer have *bounded support* (but possibly unbounded degree/fan-in), strengthening the results of Gupta et al and Kayal et al [4, 7]. This new lower bound combined with a careful "random restriction" procedure (that transforms general depth 4 homogeneous circuits to depth 4 circuits with bounded support) gives us our final result.

1 Introduction

Proving lower bounds for explicit polynomials is one of the most important open problems in the area of algebraic complexity theory. Valiant [17] defined the classes VP and VNP as the algebraic analog of the classes P and NP, and showed that proving superpolynomial lower bounds for the Permanent would suffice in separating VP from VNP. Despite the amount of attention received by the problem, we still do not know any superpolynomial (or even *quadratic*) lower bounds for general arithmetic circuits. This absence of progress on the general

[*] The full version of the paper is available on ECCC at `http://eccc.hpi-web.de/report/2013/181/`

[**] Research supported by NSF grant CCF-1350572

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 751–762, 2014.

problem has led to a lot of attention on the problem of proving lower bounds for restricted classes of arithmetic circuits. The hope is that an understanding of restricted classes might lead to a better understanding of the nature of the more general problem, and the techniques developed in this process could possibly be adapted to understand general circuits better. Among the many restricted classes of arithmetic circuits that have been studied with this motivation, *bounded depth* circuits have received a lot of attention.

In a striking result, Valiant et al [18] showed that any n variate polynomial of degree $\text{poly}(n)$ which can be computed by a polynomial sized arithmetic circuit of arbitrary depth can also be computed by an arithmetic circuit of depth $O(\log^2 n)$ and size $\text{poly}(n)$. Hence, proving superpolynomial lower bounds for circuits of depth $\log^2 n$ is as hard as proving lower bounds for general arithmetic circuits. In a series of recent works, Agrawal-Vinay [1], Koiran [8] and Tavenas [16] showed that the depth reduction techniques of Valiant et al [18] can in fact be extended much further. They essentially showed that in order to prove superpolynomial lower bounds for general arithmetic circuits, it suffices to prove strong enough lower bounds for just *homogeneous depth 4* circuits. In particular, to separate VNP from VP, it would suffice to focus our attention on proving strong enough lower bounds for homogeneous depth 4 circuits.

The first superpolynomial lower bounds for homogeneous circuits of depth 3 were proved by Nisan and Wigderson [13]. Their main technical tool was the use of the *dimension of partial derivatives* of the underlying polynomials as a complexity measure. For many years thereafter, progress on the question of improved lower bounds stalled. In a recent breakthrough result on this problem, Gupta, Kamath, Kayal and Saptharishi [4] proved the first superpolynomial ($2^{\Omega(\sqrt{n})}$) lower bounds for homogeneous depth 4 circuits when the fan-in of the product gates at the bottom level is bounded (by $\sqrt{n}$). This result was all the more remarkable in light of the results by Koiran [8] and Tavenas [16] which showed that $2^{\omega(\sqrt{n}\log n)}$ lower bounds for this model would suffice in separating VP from VNP. The results of Gupta et al were further improved upon by Kayal Saha and Sapthrashi [7] who showed $2^{\Omega(\sqrt{n}\log n)}$ lower bounds for the model of homogeneous depth 4 circuits when the fan-in of the product gates at the bottom level is bounded (by $\sqrt{n}$). Thus even a slight asymptotic improvement in the exponent of either of these bounds would imply lower bounds for general arithmetic circuits!

The main tool used in both the papers [4] and [7] was the notion of the dimension of *shifted partial derivatives* as a complexity measure, a refinement of the Nisan-Wigderson complexity measure of dimension of partial derivatives.

In spite of all this exciting progress on homogeneous depth 4 circuits with bounded bottom fanin (which suggests that possibly we might be within reach of lower bounds for much more general classes of circuits) these results give almost no non trivial (not even super linear) lower bounds for general homogeneous depth 4 circuits (with no bound on bottom fanin). Indeed the only lower bounds we know for general homogeneous depth 4 circuits are the slightly superlinear lower bounds by Raz using the notion of elusive functions [15].

Thus nontrivial lower bounds for the class of general depth 4 homogeneous circuits seems like a natural and basic question left open by these works, and strong enough lower bounds for this model seems to be an important barrier to overcome before proving lower bounds for more general classes of circuits.

In this direction, building upon the work in [4, 7], Kumar and Saraf [9, 10] proved superpolynomial lower bounds for depth 4 circuits with unbounded bottom fan-in but *bounded top fan-in.* For the case of *multilinear* depth 4 circuits, superpolynomial lower bounds were first proved by Raz and Yehudayoff [14]. These lower bounds were recently improved in a paper by Fournier, Limaye, Malod and Srinivasan [3]. The main technical tool in the work of Fournier et al was the use of the technique of *random restrictions* before using shifted partial derivatives as a complexity measure. By setting a large collection of variables at random to zero, all the product gates with high bottom fan-in got set to zero. Thus the resulting circuit had bounded bottom fanin and then known techniques of shifted partial derivatives could be applied. This idea of random restrictions crucially uses the multilinearity of the circuits, since in multilinear circuits high bottom fanin means *many* distinct variables feeding in to a gate, and thus if a large collection of variables is set at random to zero, then with high probability that gate is also set to zero.

Our Results: In this paper, we prove the first superpolynomial lower bounds for general homogeneous depth 4 circuits with no restriction on the fan-in, either top or bottom. The main ingredient in our proof is a new complexity measure of *bounded support* shifted partial derivatives. This measure allows us to prove exponential lower bounds for homogeneous depth 4 circuits where all the monomials computed at the bottom layer have only few variables (but possibly large degree/fan-in). This exponential lower bound combined with a careful "random restriction" procedure that allows us to transform general depth 4 homogeneous circuits to this form gives us our final result. We now formally state our results.

Our main theorem is stated below.

Theorem 1. *There is an explicit family of homogeneous polynomials of degree n in n^2 variables in* VNP *which requires homogeneous $\Sigma\Pi\Sigma\Pi$ circuits of size $n^{\Omega(\log\log n)}$ to compute it.*

We prove our lower bound for the family of Nisan-Wigderson polynomials NW_d which is based upon the idea of Nisan-Wigderson designs. We give the formal definition in Section 3.

As a first step in the proof of Theorem 1, we prove an exponential lower bound on the top fan-in of any homogeneous $\Sigma\Pi\Sigma\Pi$ circuit where every product gate at the bottom level has at most $O(\log n)$ distinct variables feeding into it. Let homogeneous $\Sigma\Pi\Sigma\Pi^{\{s\}}$ circuits denote the class of homogeneous $\Sigma\Pi\Sigma\Pi$ circuits where every product gate at the bottom level has at most s distinct variables feeding into it (i.e. has support at most s).

Theorem 2. *There exists a constant $\beta > 0$, and an explicit family of homogeneous polynomials of degree n in n^2 variables in* VNP *such that any homogeneous $\Sigma\Pi\Sigma\Pi^{\{\beta \log n\}}$ circuit computing it must have top fan-in at least $2^{\Omega(n)}$.*

Observe that since homogeneous $\Sigma\Pi\Sigma\Pi^{\{s\}}$ circuits are a more general class of circuits than homogeneous $\Sigma\Pi\Sigma\Pi$ circuits with bottom fan-in at most s, our result strengthens the results of of Gupta et al and Kayal et al [4, 7] when $s = O(\log n)$.

We prove Theorem 1 by applying carefully chosen random restrictions to both the polynomial family and to any arbitrary homogeneous $\Sigma\Pi\Sigma\Pi$ circuit and showing that with high probability the circuit simplifies into a homogeneous $\Sigma\Pi\Sigma\Pi$ circuit with bounded bottom support while the polynomial (even after the restriction) is still rich enough for Theorem 2 to hold. Our results hold over every field.

Recent Related Work: Recently, in an independent work, superpolynomial lower bounds for depth 4 homogeneous circuits were also shown by Limaye, Saha and Srinivasan [12]. They proved an $n^{\Omega(\log n)}$ lower bound on the size of homogeneous depth 4 circuits computing the Determinant of an $n \times n$ matrix. They also achieved a similar bound for the Iterated Matrix Multiplication polynomial. Their proof uses a different variation of shifted partial derivatives as their complexity measure- instead of bounding the support of the monomials used in the shift, they use projections to a particular set of randomly chosen monomials after shifting. Their proof doesn't proceed via first proving lower bounds for homogeneous depth 4 circuits with bounded bottom support, and thus the proof of Theorem 2 that we give here is the only proof we know of this result (which also works over all fields - see next paragraph).

In a subsequent independent work, Kayal, Limaye, Saha and Srinivasan [6] showed exponential lower bounds for homogeneous depth 4 circuits over the field of real numbers. This result combines the use of "bounded support shifts" along with the use of *random projections*. This proof does proceed via first proving lower bounds for depth 4 circuits for bounded bottom support, and over the field of real numbers they are able to prove exponential lower bounds for this model as well.

Organization of the Paper: The rest of the paper is organized as follows. In Section 2, we provide a high level overview of the proof. In Section 3, we introduce some notations and preliminary notions used in the paper. In Section 4, we sketch a proof of Theorem 2. In Section 5, we describe the effects of the random restriction procedure on the circuit and the polynomial. In Section 6, we provide a sketch of the proof of Theorem 1. In the absence of sufficient space, we skip some of the details. We refer the interested reader to the full version of the paper on ECCC [10].

2 Proof Overview

Our proof is divided into two parts. In the first part we show a $2^{\Omega(n)}$ lower bound for homogeneous $\Sigma\Pi\Sigma\Pi$ circuits whose *bottom support* is at most $O(\log n)$. To the best of our knowledge, even when the bottom support is 1, none of the earlier lower bound techniques sufficed for showing nontrivial lower bounds for this model. Thus a new complexity measure was needed. We consider the measure of *bounded support* shifted partial derivatives, a refinement of the measure of shifted partial derivatives used in several recent works [3, 4, 7, 9, 10]. For this measure, we show that the complexity of the NW_d polynomial (an explicit polynomial in VNP) is *high* whereas any subexponential sized homogeneous depth 4 circuit with bounded bottom support has a much smaller complexity measure. Thus for any depth 4 circuit to compute the NW_d polynomial, it must be large – we show that it must have exponential top fan-in. Thus we get an exponential lower bound for bounded bottom support homogeneous $\Sigma\Pi\Sigma\Pi$ circuits. We believe this result might be of independent interest.

In the second part we show how to "reduce" any $\Sigma\Pi\Sigma\Pi$ circuit that is not too large to a $\Sigma\Pi\Sigma\Pi$ circuit with bounded bottom support. This reduction basically follows from a random restriction procedure that sets some of the variables feeding into the circuit to zero. At the same time we ensure that when this random restriction procedure is applied to NW_d, the polynomial does not get affected very much, and still has large complexity.

We could have set variables to zero by picking the variables to set to zero independently at random. The problem with this approach is that we do not know how to analyze the effect of this simple randomized procedure on NW_d[1]. Thus we define a slightly more refined random restriction procedure which keeps the NW_d polynomial hard and at the same time makes the $\Sigma\Pi\Sigma\Pi$ circuit one of bounded bottom support. We remark that it is the choice of these random restrictions that lead to a lower bound of $n^{\Omega(\log\log n)}$ as opposed to $n^{\Omega(\log n)}$.

3 Preliminaries and Notations

Arithmetic Circuits: An arithmetic circuit over a field $\mathbb{F}$ and a set of variables $x_1, x_2, \ldots, x_N$ is an directed acyclic graph whose internal nodes are labelled by the field operations and the leaf nodes are labelled by the variables or field elements. The nodes with fan-out zero are called the output gates and the nodes with fan-in zero are called the leaves. In this paper, we always assume that there is a unique output gate in the circuit. The *size* of the circuit is the number of nodes in the underlying graph and the *depth* of the circuit is the length of the longest path from the root to a leaf. We call a circuit *homogeneous* if the polynomial computed at every node is a homogeneous polynomial. By a $\Sigma\Pi\Sigma\Pi$ circuit or a depth 4 circuit, we mean a circuit of depth 4 with the top layer and the third layer only have sum gates and the second and the bottom layer have only product gates. In this paper, we confine ourselves to working with homogeneous depth 4 circuits. A homogeneous polynomial P of degree n in N variables, which is computed by a homogeneous $\Sigma\Pi\Sigma\Pi$ circuit can be written as

$$P(x_1, x_2, \ldots, x_N) = \sum_{i=1}^{T} \prod_{j=1}^{d_i} Q_{i,j}(x_1, x_2, \ldots, x_N) \qquad (1)$$

[1] This strategy was shown to work with some change in parameters and a more careful analysis in [6] and [11].

Here, T is the top fan-in of the circuit. Since the circuit is homogeneous, we know that for every $i \in \{1, 2, 3, \ldots, T\}$, $\sum_{j=1}^{d_i} \deg(Q_{i,j}) = n$. By the support of a monomial α, we refer to the set of variables which have a positive degree in α. In this paper, we also study the class of homogeneous $\Sigma\Pi\Sigma\Pi$ circuits such that for every i, j, every monomial in $Q_{i,j}$ has bounded support. We now formally define this class.

Homogeneous $\Sigma\Pi\Sigma\Pi^{\{s\}}$ Circuits: A homogeneous $\Sigma\Pi\Sigma\Pi$ circuit in Equation 1, is said to be a $\Sigma\Pi\Sigma\Pi^{\{s\}}$ circuit if every product gate at the bottom level has support at most s. Observe that there is no restriction on the bottom fan-in except that implied by the restriction of homogeneity.

Shifted Partial Derivatives: In this paper we use a variant of the notion of *shifted partial derivatives* which was introduced in [5] and has subsequently been the complexity measure used to to prove lower bounds for various restricted classes of depth four circuits and formulas(for example in [3, 4, 7, 9, 10]). For a field $\mathbb{F}$, an N variate polynomial $P \in \mathbb{F}[x_1, \ldots, x_N]$ and a positive integer r, we denote by $\partial^r P$, the set of all partial derivatives of order equal to r of P. For a polynomial P and a monomial γ, we denote by $\partial_\gamma(P)$ the partial derivative of P with respect to γ. We now reproduce the formal definition from [4].

Definition 3 (Order-r ℓ-Shifted Partial Derivatives). *For an N variate polynomial $P \in \mathbb{F}[x_1, x_2, \ldots, x_N]$ and positive integers $r, \ell \geq 0$, the space of order-r ℓ-shifted partial derivatives of P is defined as*

$$\langle \partial^r P \rangle_\ell \overset{def}{=} \mathbb{F}\text{-}span\{ \prod_{i \in [N]} {x_i}^{j_i} \cdot g : \sum_{i \in [N]} j_i = \ell, g \in \partial^r P\} \tag{2}$$

In this paper, we introduce the variation of *bounded support* shifted partial derivatives as a complexity measure. The basic difference is that instead of shifting the partial derivatives by all monomials of degree ℓ, we shift the partial derivatives only by only those monomials of degree ℓ which have support(the number of distinct variables which have non-zero degree in the monomial) exactly equal to m. We now formally define the notion of support-m degree-ℓ shifted partial derivatives of order-r of a polynomial, which for the rest of the paper, we refer by (m, ℓ, r)-shifted partial derivatives.

Definition 4 ((m, ℓ, r)-Shifted Partial Derivatives). *For an N variate polynomial $P \in \mathbb{F}[x_1, x_2, \ldots, x_N]$ and positive integers $r, \ell, m \geq 0$, the space of support-m degree-ℓ shifted partial derivatives of order-r of P is defined as*

$$\langle \partial^r P \rangle_{(\ell, m)} \overset{def}{=}$$
$$\mathbb{F}\text{-}span\{\prod_{i \in S} {x_i}^{j_i} \cdot g : S \subseteq [N], |S| = m, \sum_{i \in S} j_i = \ell, j_i \geq 1, g \in \partial^r P\}$$

The following property follows from the definition above.

Lemma 5. *For any two multivariate polynomials P and Q in $\mathbb{F}[x_1, x_2, \ldots, x_N]$ and any positive integers r, ℓ, m, and scalars α and β*

$$\mathsf{Dim}(\langle \partial^r (\alpha P + \beta Q) \rangle_{(\ell, m)}) \leq \mathsf{Dim}(\langle \partial^r P \rangle_{(\ell, m)}) + \mathsf{Dim}(\langle \partial^r Q \rangle_{(\ell, m)})$$

For any linear or affine space V over a field $\mathbb{F}$, we use $\mathsf{Dim}(V)$ to represent the dimension of V over $\mathbb{F}$. We use the dimension of the space $\langle \partial^r P \rangle_{(\ell,m)}$ which we denote by $\mathsf{Dim}(\langle \partial^r P \rangle_{(\ell,m)})$ as the measure of complexity of a polynomial.

Nisan-Wigderson Polynomials: We show our lower bounds for a family of polynomials in VNP which were used for the first time in the context of lower bounds in [7]. The construction is based upon the intuition that over any field, any two distinct low degree polynomials do not agree at too many points. For the rest of this paper, we assume n to be of the form 2^k for some positive integer k. Let $\mathbb{F}_n$ be a field of size n. For the set of $N = n^2$ variables $\{x_{i,j} : i, j \in [n]\}$ and $d < n$, we define the degree n homogeneous polynomial NW_d as

$$NW_d = \sum_{\substack{f(z)\in\mathbb{F}_n[z]\\ deg(f)\leq d-1}} \prod_{i\in[n]} x_{i,f(i)}$$

From the definition, we can observe the following properties of NW_d.

1. The number of monomials in NW_d is exactly n^d.
2. Each of the monomials in NW_d is multilinear.
3. Each monomial corresponds to evaluations of a univariate polynomial of degree at most $d-1$ at all points of $\mathbb{F}_n$. Thus, any two distinct monomials agree in at most $d-1$ variables in their support.

For any $S \subseteq [n]$ and each $f \in \mathbb{F}_n[z]$, we define the monomial $m_f^S = \prod_{i\in S} x_{i,f(i)}$ and $m_f = \prod_{i\in[n]} x_{i,f(i)}$ We also define the set $\mathcal{M}^S$ to represent the set of monomials $\{x_{i_1,j_1} \cdot x_{i_2,j_2} \cdot x_{i_3,j_3} \cdots x_{i_{|S|},j_{|S|}} : i_1 < i_2 \ldots < i_{|S|} \in S$ and $\forall t \in [|S|], j_t \in [n]\}$. Clearly, $NW_d = \sum_{\substack{f(z)\in\mathbb{F}_n[z]\\ deg(f)\leq d-1}} m_f$.

Monomial Ordering and Distance: We also use the notion of a monomial being an extension of another as defined below.

Definition 6. *A monomial θ is said to be an extension of a monomial $\tilde{\theta}$, if θ divides $\tilde{\theta}$.*

In this paper, we imagine our variables to be coming from a $n \times n$ matrix $\{x_{i,j}\}_{i,j\in[n]}$. We also consider the following total order on the variables. $x_{i_1,j_1} > x_{i_2,j_2}$ if either $i_1 < i_2$ or $i_1 = i_2$ and $j_1 < j_2$. This total order induces a lexicographic order on the monomials. For a polynomial P, we use the notation $\mathsf{Lead\text{-}Mon}(P)$ to indicate the leading monomial of P under this monomial ordering.

We use the following notion of distance between two monomials which was also used in [2].

Definition 7 (Monomial Distance). *Let m_1 and m_2 be two monomials over a set of variables. Let S_1 and S_2 be the multiset of variables in m_1 and m_2 respectively, then the distance $\Delta(m_1, m_2)$ between m_1 and m_2 is the min$\{|S_1| - |S_1 \cap S_2|, |S_2| - |S_1 \cap S_2|\}$ where the cardinalities are the order of the multisets.*

In this paper, we invoke this definition only for multilinear monomials of the same degree. In this special case, we have the following crucial observation.

Observation 8. *Let α and β be two multilinear monomials of the same degree which are at a distance Δ from each other. If Supp(α) and Supp(β) are the supports of α and β respectively, then $|Supp(\alpha)| - |Supp(\alpha) \cap Supp(\beta)| = |Supp(\beta)| - |Supp(\alpha) \cap Supp(\beta)| = \Delta$.*

4 Lower Bounds for $\Sigma\Pi\Sigma\Pi^{\{O(\log n)\}}$ Circuits

In this section, we sketch the outline of the proof of Theorem 2. We refer the interested reader to the full version of the paper [10] for the complete proof. We show an exponential lower bound on the top fan-in for homogeneous $\Sigma\Pi\Sigma\Pi$ circuits such that every product gate at the bottom has a bounded number of variables feeding into it. We use the dimension of the span of (m, ℓ, r)-shifted partial derivatives as the complexity measure. Our lower bound holds for the NW_d polynomial. The proof has two major components. In the first part, we obtain an upper bounded on the complexity of the circuit. Then, we obtain a lower bound on the complexity of the NW_d polynomial. Comparing the two then implies our lower bound. The bound holds for NW_d for any $d = \delta n$, where δ is a constant such that $0 < \delta < 1$.

4.1 Complexity of Homogeneous Depth 4 $\Sigma\Pi\Sigma\Pi^{\{s\}}$ Circuits

Let C be a homogeneous $\Sigma\Pi\Sigma\Pi^{\{s\}}$ circuit computing the NW_d polynomial. We now state an upper bound on the complexity of a product gate in such a circuit. The proof is fairly straightforward, and we refer the reader to [10] for details. The bound on the complexity of the circuit follows from the subadditivity of the complexity measure.

Lemma 9. *Let $Q = \prod_{i=1}^{n} Q_i$ be a product gate at the second layer from the top in a homogeneous $\Sigma\Pi\Sigma\Pi^{\{s\}}$ circuit computing a homogeneous degree n polynomial in N variables. For any positive integers m, r, s, ℓ satisfying $m + rs \leq \frac{N}{2}$ and $m + rs \leq \frac{\ell}{2}$,*

$$\mathsf{Dim}(\langle \partial^r Q \rangle_{(\ell,m)}) \leq poly(nrs)\binom{n+r}{r}\binom{N}{m+rs}\binom{\ell+n-r}{m+rs}$$

For a homogeneous $\Sigma\Pi\Sigma\Pi$ circuit where each of the bottom level product gates is of support at most s, Lemma 9 immediately implies the following upper bound on the complexity of the circuit due to subadditivity from Lemma 5.

Corollary 10. *Let $C = \sum_{j=1}^{T}\prod_{i=1}^{n} Q_{i,j}$ be a a homogeneous $\Sigma\Pi\Sigma\Pi^{\{s\}}$ circuit computing a homogeneous degree n polynomial in N variables. For any m, r, s, ℓ satisfying $m + rs \leq \frac{N}{2}$ and $m + rs \leq \frac{\ell}{2}$,*

$$\mathsf{Dim}(\langle \partial^r C \rangle_{(\ell,m)}) \leq T \times poly(nrs)\binom{n+r}{r}\binom{N}{m+rs}\binom{\ell+n-r-1}{m+rs-1}$$

4.2 Lower Bound on the Complexity of the NW_d Polynomial

We now outline the approach to obtain a lower bound on the complexity of the NW_d polynomial. For this, we first observe that distinct partial derivatives of the NW_d polynomial are *far* from each other in some sense and then show that shifting such partial derivatives gives us a lot of distinct shifted partial derivatives. Recall that we defined the set $\mathcal{M}^S$ to represent the set $\{x_{i_1,j_1} \cdot x_{i_2,j_2} \cdot x_{i_3,j_3} \cdots x_{i_{|S|},j_{|S|}} : i_1 < i_2 \ldots < i_{|S|} \in S$ and $\forall t \in [|S|], j_t \in [n]\}$. We start with the following observation.

Lemma 11. *For any positive integer r such that $n - r > d$ and $r < d - 1$, the set $\{\partial_\alpha(NW_d) : \alpha \in \mathcal{M}^{[r]}\}$ consists of $|\mathcal{M}^{[r]}| = n^r$ nonzero distinct polynomials.*

It can be observed that for any $\alpha \neq \beta \in \mathcal{M}^{[r]}$, the leading monomials of $\partial_\alpha(NW_d)$ and $\partial_\beta(NW_d)$ are multilinear monomials of at a distance at least $n-r-d$ from each other. We exploit this structure in order to show that shifting the polynomials in the set $\{\partial_\alpha(NW_d) : \alpha \in \mathcal{M}^{[r]}\}$ by monomials of support m and degree ℓ results in many linearly independent shifted partial derivatives. We crucially use the following simple lemma.

Lemma 12. *Let α and β be two distinct multilinear monomials of equal degree such that the distance between them is Δ. Let S_α and S_β be the set of all monomials obtained by shifting α and β respectively with monomials of degree ℓ and support exactly m over N variables. Then $|S_\alpha \cap S_\beta| \leq \binom{N-\Delta}{m-\Delta}\binom{\ell-1}{m-1}$.*

For any monomial α and positive integers ℓ, m, we denote by $S_{\ell,m}(\alpha)$ the set of all shifts of $\partial_\alpha NW_d$ by monomials of degree ℓ and support m. More formally,

$$S_{\ell,m}(\alpha) = \{\gamma \cdot \partial_\alpha(NW_d) : \gamma = \prod_{i \in U} x_i^{j_i}, U \subseteq [N], |U| = m, \sum_{i \in U} j_i = \ell, j_i \geq 1\}$$

also, let

$$LM_{\ell,m}(\alpha) = \{\mathsf{Lead\text{-}Mon}(f) : f \in S_{\ell,m}(\alpha)\}$$

An application of Lemma 12 to the NW_d polynomial gives us the following lemma.

Lemma 13. *For any positive integers r, m and ℓ such that $n - r > d$ and $r < d - 1$, let α and β be two distinct monomials in $\mathcal{M}^{[r]}$. Then $|S_{\ell,m}(\alpha) \cap S_{\ell,m}(\beta)| \leq \binom{N-(n-d-r)}{m-(n-d-r)}\binom{\ell-1}{m-1}$.*

We now obtain a lower bound on the dimension of the span of (m, ℓ, r)-shifted partial derivatives of the NW_d polynomial. For this, we use the following proposition from [4], the proof of which is a simple application of Gaussian elimination.

Proposition 14 ([4]). *For any field $\mathbb{F}$, let $\mathcal{P} \subseteq \mathbb{F}[z]$ be any finite set of polynomials. Then,*

$$\mathsf{Dim}(\mathbb{F}\text{-}span(\mathcal{P})) = |\{\mathsf{Lead\text{-}Mon}(f) : f \in \mathbb{F}\text{-}span(\mathcal{P})\}|$$

Therefore, in order to lower bound $\mathsf{Dim}(\langle\partial^r NW_d\rangle_{(\ell,m)})$, it would suffice to get a lower bound on the size of the set $\bigcup_\alpha LM_{\ell,m}(\alpha)$, where the union is over all monomials α of degree equal to r. To achieve this, we first obtain a lower bound on the size of the set $\bigcup_{\alpha\in\mathcal{M}^{[r]}} LM_{\ell,m}(\alpha)$. The bound is formally given by the lemma below. The proof follows via an application of the principle of inclusion-exclusion. We refer the reader to the full version of this paper [10] for more details.

Lemma 15. *Let $d = \delta n$ for any constant $0 < \delta < 1$. Let ℓ, m, r be positive integers such that $n - r > d$, $r < d - 1$, $m \le N$, $m = \theta(N)$ and for $\phi = \frac{N}{m}$, r satisfies $r \le \frac{(n-d)\log\phi \pm O(\phi\frac{(n-d-r)^2}{N})}{\log n + \log\phi}$. Then,*

$$\mathsf{Dim}(\langle\partial^r NW_d\rangle_{(\ell,m)}) \ge 0.5n^r \binom{N}{m}\binom{\ell-1}{m-1}$$

4.3 Top Fan-in Lower Bound

Comparing the bounds in complexity given by Lemma 15 and Corollary 10 gives us a lower bound on the top fan-in of any homogeneous $\Sigma\Pi\Sigma\Pi^{\{\beta\log n\}}$ (for some constant β) that computes the NW_d polynomial, where $d = \delta n$ for some constant δ between 0 and 1. We formally state the result below and refer the reader to [10] for more details.

Theorem 16. *Let $d = \delta n$ for any constant $0 < \delta < 1$. There exists a constant β such that all homogeneous $\Sigma\Pi\Sigma\Pi^{\{\beta\log n\}}$ circuits which compute the NW_d polynomial have top fan-in at least $2^{\Omega(n)}$.*

5 Random Restrictions

The strategy now, is to define an appropriate random restriction procedure and show that with a non-zero probability, all the large support product gates in the bottom level of the circuit get set to zero while the complexity of the polynomial remains large enough. For the lack of space we refer the reader to the full version of the paper [10] for details. The two main statements we need in order to complete the proof are enumerated below. The lemma below summarizes that any restriction $R_\epsilon(NW_d)$ of NW_d obtained as the outcome of our random restriction procedure still remains hard with respect to $\Sigma\Pi\Sigma\Pi^{\{O(\log n)\}}$ circuits.

Lemma 17. *Let $d = \delta n$ for any constant δ such that $0 < \delta < 1$. Then, there exist constants ϵ, β such that any homogeneous $\Sigma\Pi\Sigma\Pi^{\{\beta\log n\}}$ circuit computing the $R_\epsilon(NW_d)$ polynomial for any random restriction R_ϵ has top fan-in is at least $2^{\Omega(n)}$.*

The proof of the lemma is essentially the same as the proof of Theorem 16 and we skip the details to the full version of this paper.

The following lemma summarizes that under our random restriction procedure, all the product gates with large support vanish with a high probability.

Lemma 18 (Random restriction on $\Sigma\Pi\Sigma\Pi$ circuit). *Let $\epsilon > 0$ and $\beta > 0$ be constants. Then there exists $\rho > 0$ such that if C is a $\Sigma\Pi\Sigma\Pi$ circuit of size at most $n^{\rho \log\log n}$, then with probability $> 9/10$, all the monomials computed at the bottom layer which have support at least $\beta \log n$ have some variable set to 0 by R_ϵ.*

6 Lower Bounds for NW_d

In this section, we state our main theorem and give a sketch of the proof. The proof is very similar to proof of Theorem 16 and follows via comparing the complexities of the polynomial and the circuit after random restrictions.

Theorem 19. *Let $d = \delta n$ for any constant δ such that $0 < \delta < 1$. Any homogeneous $\Sigma\Pi\Sigma\Pi$ circuit computing the NW_d must have size at least $n^{\Omega(\log\log n)}$.*

Proof. For every value of δ, such that $0 < \delta < 1$, choose the parameters $\epsilon = \tilde{\epsilon}, \beta = \tilde{\beta}$ such that Lemma 17 is true for $\tilde{d} = \delta n$. Now, let us choose a constant $\rho = \tilde{\rho}$ such that Lemma 18 holds. Now, let C be a homogeneous $\Sigma\Pi\Sigma\Pi$ circuit computing the $NW_{\tilde{d}}$ polynomial. If the number of bottom product gates of C was at least $n^{\tilde{\rho} \log\log n}$, then C has large size and we are done. Else, let us now apply a random restriction R_ϵ to the circuit. By the choice of parameters, Lemma 18 holds and so with probability 0.9 every bottom product gate in C with support larger than $\tilde{\beta} \log n$ is set to zero. After a restriction, the circuit computes $R_{\tilde{\epsilon}}(NW_{\tilde{d}})$. So, now we are in the case when we have a small support homogeneous circuit of depth four computing some random restriction of the $NW_{\tilde{d}}$ polynomial and then, by Lemma 17 above, the top fan-in of $R_{\tilde{\epsilon}}(C)$ must be at least $2^{\Omega(n)}$. Hence, any homogeneous $\Sigma\Pi\Sigma\Pi$ circuit computing $NW_{\tilde{d}}$ must have size at least $n^{\Omega(\log\log n)}$.

References

1. Agrawal, M., Vinay, V.: Arithmetic circuits: A chasm at depth four. In: Proceedings of the 49th Annual FOCS, pp. 67–75 (2008)
2. Chillara, S., Mukhopadhyay, P.: Depth-4 lower bounds, determinantal complexity: A unified approach. CoRR, abs/1308.1640v3 (2013)
3. Fournier, H., Limaye, N., Malod, G., Srinivasan, S.: Lower bounds for depth 4 formulas computing iterated matrix multiplication. In: STOC 2014 (to appear, 2014)
4. Gupta, A., Kamath, P., Kayal, N., Saptharishi, R.: Approaching the chasm at depth four. In: Proceedings of CCC (2013)
5. Kayal, N.: An exponential lower bound for the sum of powers of bounded degree polynomials. ECCC 19, 81 (2012)
6. Kayal, N., Limaye, N., Saha, C., Srinivasan, S.: An exponential lower bound for homogeneous depth four arithmetic formulas. ECCC (2014)
7. Kayal, N., Saha, C., Saptharishi, R.: A super-polynomial lower bound for regular arithmetic formulas. ECCC 20, 91 (2013)

8. Koiran, P.: Arithmetic circuits: The chasm at depth four gets wider. Theoretical Computer Science 448, 56–65 (2012)
9. Kumar, M., Saraf, S.: The limits of depth reduction for arithmetic formulas: It's all about the top fan-in. In: STOC 2014 (to appear, 2014)
10. Kumar, M., Saraf, S.: Superpolynomial lower bounds for general homogeneous depth 4 arithmetic circuits. ECCC (2013)
11. Kumar, M., Saraf, S.: On the power of homogeneous depth 4 arithmetic circuits. ECCC (2014)
12. Limaye, N., Saha, C., Srinivasan, S.: Super-polynomial lower bounds for depth-4 homogeneous arithmetic formulas. In: STOC 2014 (to appear, 2014)
13. Nisan, N., Wigderson, A.: Lower bounds on arithmetic circuits via partial derivatives. In: Proceedings of the 36th Annual FOCS, pp. 16–25 (1995)
14. Raz, R., Yehudayoff, A.: Lower bounds and separations for constant depth multilinear circuits. In: Conference on Computational Complexity, pp. 128–139 (June 2008)
15. Raz, R.: Elusive functions and lower bounds for arithmetic circuits. Theory of Computing 6(1), 135–177 (2010)
16. Tavenas, S.: Improved bounds for reduction to depth 4 and depth 3. In: Chatterjee, K., Sgall, J. (eds.) MFCS 2013. LNCS, vol. 8087, pp. 813–824. Springer, Heidelberg (2013)
17. Valiant, L.: Completeness classes in algebra. In: Proceedings of the 11th Annual STOC, STOC 1979, pp. 249–261. ACM, New York (1979)
18. Valiant, L., Skyum, S., Berkowitz, S., Rackoff, C.: Fast parallel computation of polynomials using few processors. SIAM Journal of Computation 12(4), 641–644 (1983)

Testing Forest-Isomorphism in the Adjacency List Model

Mitsuru Kusumoto[1,⋆] and Yuichi Yoshida[1,2,⋆⋆]

[1] Preferred Infrastructure, Inc
mkusumoto@preferred.jp
[2] National Institute of Informatics
yyoshida@nii.ac.jp

Abstract. We consider the problem of testing if two input forests are isomorphic or are far from being so. An algorithm is called an ε-tester for forest-isomorphism if given an oracle access to two forests G and H in the adjacency list model, with high probability, accepts if G and H are isomorphic and rejects if we must modify at least εn edges to make G isomorphic to H. We show an ε-tester for forest-isomorphism with a query complexity polylog(n) and a lower bound of $\Omega(\sqrt{\log n})$. Further, with the aid of the tester, we show that every graph property is testable in the adjacency list model with polylog(n) queries if the input graph is a forest.

1 Introduction

In *property testing*, we want to design an efficient algorithm that distinguishes the case in which the input object satisfies some property or is "far" from satisfying it [11]. In particular, an object is called *ε-far* from a property P if we have to modify an ε-fraction of the input to make it satisfy P. A (randomized) algorithm is called an *ε-tester* for a property P if it accepts objects satisfying P and rejects objects that are ε-far from P with high probability (say 2/3).

Graph property testing is one of the major topics in property testing, and many properties are known to be testable in sublinear time or even in constant time (in the input size). See [5] for surveys. In order to design sublinear-time testers, we have to define how to access the input graph, as just reading the entire graph requires linear time. The model used here is the *adjacency list model* [9]. In this model, the input graph $G = (V, E)$ is represented by an adjacency list and we are given an oracle access $\mathcal{O}_G$ to it. We have two types of queries. The first query, called a *degree query*, specifies a vertex v, and the oracle $\mathcal{O}_G$ returns the degree of v. The second query, called a *neighbor query*, specifies a vertex v and an index i, and the oracle $\mathcal{O}_G$ returns the i-th neighbor of v. A graph G is called *ε-far*

[⋆] JST, ERATO, Kawarabayashi Large Graph Project.

[⋆⋆] Supported by JSPS Grant-in-Aid for Research Activity Start-up (24800082), MEXT Grant-in-Aid for Scientific Research on Innovative Areas (24106003), and JST, ERATO, Kawarabayashi Large Graph Project.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 763–774, 2014.

from a property P if we must add or remove at least εm edges for it to satisfy the property P, where m is the number of edges. In contrast to other models such as the adjacency matrix and the bounded-degree models, only a few properties are known to be efficiently testable in the adjacency list model. For examples, testing triangle-freeness, k-colorability for a constant k, and bipartiteness requires $\Omega(\sqrt{n})$ queries [2,3,9], where n is the number of vertices.

A graph G is called *isomorphic* to another graph H if there is a bijection $\pi : V(G) \to V(H)$ such that $(u, v) \in E(G)$ if and only if $(\pi(u), \pi(v)) \in E(H)$. In this paper, we consider the problem of testing if the input graph G is isomorphic to a fixed graph H, or if it is *H-isomorphic.* We assume that the (unknown) input graph G has the same number of vertices as H. The problem of deciding if a graph is isomorphic to H is fundamental and theoretically important. For example, the problem is one of the rare problems that is neither known to be in **P** nor **NP-Complete**. This motivates us to consider H-isomorphism in the property testing literature. A *graph property* refers to a property that is closed under taking isomorphism. Then, H-isomorphism can be identified as the simplest graph property such that every graph property can be expressed as a union of H-isomorphisms. Owing to these observations, Newman and Sohler [10] showed that every graph property is testable in the bounded-degree model if the input graph is a (bounded-degree) planar graph. This connection also holds for the adjacency list model, which motivates us to consider H-isomorphism in the adjacency list model.

If we assume that the input graph is an arbitrary graph possibly containing $\Omega(n^2)$ edges, testing H-isomorphism in the adjacency list model requires $\Omega(\sqrt{n})$ queries [4]. To investigate efficient testers for H-isomorphism, we restrict the input graph: We assume that the input graph and H are forests with the same number of vertices n. Note that we have no assumption on the degree as opposed to the bounded-degree model. To avoid uninteresting technicalities, we modify the definition of ε-farness as follows: Instead of using the number of edges in G to measure the distance, we say that a forest G is *ε-far* from isomorphic to a forest H if we must add or remove εn edges to transform G to H.[1]

With these definitions, we refer to the problem of testing the property of being isomorphic to a fixed forest as *testing forest-isomorphism.* The main result of this paper is as follows.

Theorem 1.1. *In the adjacency list model, we can test forest-isomorphism with* $\operatorname{polylog}(n)$ *queries.*

Indeed, in our proof, we show that we can test forest-isomorphism even if both graphs are given as oracle accesses.

Further, we show a lower bound for testing forest-isomorphism.

Theorem 1.2. *In the adjacency list model, testing forest-isomorphism requires* $\Omega(\sqrt{\log n})$ *queries.*

[1] Indeed, we often assume that the input graph contains $\Omega(n)$ edges in the adjacency list model. Thus, our definition of ε-farness for forests and the definition of ε-farness in the adjacency list model with the assumption are identical up to a constant multiplicative factor.

As a corollary of Theorem 1.1, we show the following general result.

Theorem 1.3. *In the adjacency list model, given an oracle access to a forest, we can test any graph property with* $\mathrm{polylog}(n)$ *queries.*

Techniques. We state a proof sketch of our main theorem, Theorem 1.1. Given a tree G, by removing εn edges from G, we can obtain a graph G' with the following property for some $s = s(\varepsilon)$. Each connected component of G' is either (i) a tree of maximum degree at most s, or (ii) a tree consisting of a (unique) root vertex of degree more than s and subtrees of size at most s.

The first step in our algorithm is providing an oracle access $\mathcal{O}_{G'}$ to G' using the oracle access $\mathcal{O}_G$ to G. We call $\mathcal{O}_{G'}$ the *partitioning oracle*. In particular, if we specify a vertex v and an index i, the oracle $\mathcal{O}_{G'}$ returns whether the i-th edge incident to v in G is still alive in G'. By carefully designing the construction of G', we can answer the query with $O(s^2)$ queries to $\mathcal{O}_G$.

Suppose that we have an oracle access $\mathcal{O}_{G'}$ to G'. Since we can deal with trees of type (i) using existing algorithms in the bounded-degree model, let us elaborate on trees of type (ii). For a tree T of type (ii), we can associate a tuple $(d, c_1, \ldots, c_{t(s)})$ with it, where $t(s)$ is the number of possible trees of maximum degree at most s and size at most s. Note that $t(s)$ depends only on ε. Here, d is the degree of the root vertex of T, and c_i is the number of subtrees of the i-th type in T. Though we cannot exactly compute the tuple, given the root vertex of T, we can approximate it well using $\mathcal{O}_{G'}$. Since G' consists of trees of type (ii), we can associate a multiset of tuples with G'. We call it the *sketch* of G'. Though we cannot exactly compute the sketch, we can approximate it to some extent. The query complexity becomes $\mathrm{polylog}(n)$ since we want to approximate d to within the multiplicative factor of $1 + \varepsilon$ and d can be up to n.

If G and H are isomorphic, then sketches associated with G' and H' must be the same. Our claim is that, if G' and H' are ε-far from being isomorphic, then their sketches are also far. Further, we will show that the distance between two sketches can be computed via maximum matching in the bipartite graph such that each vertex in the left part corresponds to a tree in G' and each vertex in the right part corresponds to a tree in H'. Since we can approximate sketches well and then approximate the size of the maximum matching from them, we obtain a tester for forest-isomorphism.

Related Works. There are two major models on the representation of graphs. In the *dense graph model*, a graph $G = (V, E)$ is given as an oracle $\mathcal{O}_G : V \times V \to \{0, 1\}$. Given two vertices $u, v \in V$, the oracle returns whether u and v are connected in G. A graph is called *ε-far* from a property P if we must add or remove at least εn^2 edges for it to satisfy P.

In the dense graph model, many properties such as triangle-freeness and k-colorability are known to be testable in constant time [6]. Indeed, Alon et al. [1] obtained the characterization of constant-time testable properties using Szemerédi's regularity lemma. As for graph isomorphism, Fischer and Matsliah [4] showed that testing H-isomorphism can be carried out with $\widetilde{\Theta}(\sqrt{n})$ queries. If both G and H are given as oracle accesses, then we need $\Omega(n)$ queries, and we can test with $\widetilde{O}(n^{5/4})$ queries. We can trivially test forest-isomorphism: If a graph is isomorphic to a forest H, then it has at most n edges. If a graph is ε-far from being isomorphic to H, then it has at least $\varepsilon n^2 - n$ edges (otherwise, we can remove all edges and then add new edges to make H). Thus, we can distinguish the two cases only by estimating the number of edges up to, say $\frac{\varepsilon n^2}{2}$.

In the *bounded-degree model* with a degree bound d, a graph $G = (V, E)$ is given as an oracle $\mathcal{O}_G : V \times [d] \to V \cup \{\bot\}$, where $[d] = \{1, \ldots, d\}$ and $\bot$ is a special symbol. Given a vertex $v \in V$ and an index $i \in [d]$, the oracle returns the i-th neighbor of v. If there is no such neighbor, then the oracle returns $\bot$.

Many properties are known to be testable in constant time [7] and several general conditions of constant-time testability are shown [10,12]. Hassidim et al. [8] introduced the concept of the partitioning oracle to test minor closed properties. Our partitioning oracle is similar to theirs, but their oracle provides an oracle access to the graph that is determined by its internal random coin whereas ours provides an oracle access to a graph that is deterministically determined. As for graph isomorphism, it is known that H-isomorphism is testable in constant time when H is hyperfinite [10]. Here, a graph is *hyperfinite* if by removing εn edges, we can decompose the graph into connected components of size at most $f(\varepsilon)$ for some function f.

Organization. In Section 2, we give notations and definitions used throughout the paper. In Section 3, we introduce the partitioning oracle. Using the partitioning oracle, it suffices to consider the case where each tree in the input graph is either a bounded-degree tree or a tree consisting of a high-degree root and subtrees of small sizes. In Section 4, we consider the case in which every tree in the input graph is the latter type and the degrees of roots are within a small interval. We deal with the general case and prove Theorem 1.1 in Section 5. Due to limitations of space, some proofs in Section 3, 4, 5 are presented in the full version. The proof of Theorem 1.3 and the lower bound is also given in the full version.

2 Preliminaries

For an integer n, we denote by $[n]$ the set $\{1, 2, \ldots, n\}$ and denote by $\mathbb{N}_{<n}$ (resp. $\mathbb{N}_{\leq n}$) the set $\{0, 1, \ldots, n-1\}$ (resp. $\{0, 1, \ldots, n\}$).

Let $G = (V, E)$ be a graph. For a vertex v, $\deg_G(v)$ denotes the *degree* of v. We omit the subscript if it is clear from the context. For a set of vertices $S \subseteq V$, $G[S]$ denotes the subgraph *induced* by S. For graphs G and H with the same number of vertices, the *distance* $d(G, H)$ between G and H is defined as the minimum number of edges that need to be added or removed to make G isomorphic to H. Formally,

$$\begin{aligned} d(G,H) = \min_{\pi}(&\#\{(u,v) \in E(G) \mid (\pi(u),\pi(v)) \notin E(H)\} \\ &+ \#\{(u,v) \notin E(G) \mid (\pi(u),\pi(v)) \in E(H)\}), \end{aligned}$$

where π is over bijections from $V(G)$ to $V(H)$. We extend the definition of $d(G,H)$ for the case in which G and H have different number of vertices by adding a sufficient number of isolated vertices. For a graph G and an integer k, let $G+kv$ be the graph consisting of G and k isolated vertices. If $|V(G)| > |V(H)|$, we define $d(G,H) = d(G, H+(|V(G)|-|V(H)|)v)$. Similarly, if $|V(G)| < |V(H)|$, we define $d(G,H) = d(G+(|V(H)|-|V(G)|)v, H)$.

For an integer $s \geq 1$, we call a tree T an *s-rooted tree* if T contains a (unique) vertex v with $\deg(v) \geq s+1$ such that each subtree of v contains at most s vertices. The vertex v is called the *root vertex* of T and is denoted by $\mathrm{root}(T)$. We call a tree T an *s-bounded-degree tree* if every vertex in T has a degree of at most s. We call a tree T an *s-tree* if it is an s-rooted tree or an s-bounded-degree tree. To designate a union of trees, we use the term "forest." For example, an *s-rooted forest* means a disjoint union of s-rooted trees.

3 Partitioning Oracle

In this section, we show that, for any $\varepsilon > 0$, there exists $s = s(\varepsilon)$ such that we can partition any forest into an s-forest by removing at most εn edges. Then, we show that we can provide an oracle access to the s-forest, which we call the *partitioning oracle*. All missing proofs are given in the full version. We refer to a vertex with degree more than s in the original graph G as a *high-degree* vertex.

Lemma 3.1 (Partitioning oracle). *Suppose that we have an oracle access $\mathcal{O}_G$ to a forest G in the adjacency list model. Then for every $\varepsilon > 0$, we can provide an oracle access $\mathcal{O}'_G$ to a graph G' with the following properties:*

1. *G' is an s-forest for some $s = s_{3.1}(\varepsilon)$. G' depends only on G and ε.*
2. *G' is obtained from G by removing at most εn edges.*
3. *Let V_h be high-degree vertices in G. Then, each tree in G' contains at most one vertex from V_h.*

The oracle $\mathcal{O}'_G$ supports alive-edge queries*: Given a vertex v and an integer i, the oracle returns whether the i-th edge incident to v in G still exists in G'. For each alive-edge query, the oracle issues $O(1/\varepsilon^2)$ queries to $\mathcal{O}_G$. The output of $\mathcal{O}'_G$ is deterministically calculated. Moreover, if G and H are isomorphic and $\Psi : V(G) \to V(H)$ is an isomorphism, $\mathcal{O}'_G(e) = \mathcal{O}'_H(\Psi(e))$ holds for every edge $e \in E(G)$.*

Proof. We set $s = \frac{11}{\varepsilon}$. If the degree of a vertex is at most s, we call it *low-degree*. Let V_h and V_l be the sets of high-degree and low-degree vertices in G, respectively. We call a connected component in $G[V_l]$ *large* if it has more than s vertices and *small* otherwise. From the definition, there are at most $2n/s$ high-degree vertices in G and at most n/s large components in $G[V_l]$.

We first give a polynomial-time algorithm that outputs an s-forest from the input forest G. First, we remove edges (u,v) with $u,v \in V_h$ from G. Owing to this, the resulting graph can be seen as a bipartite graph, where the left part is V_h and the right part consists of components in $G[V_l]$. Now for each small component C in $G[V_l]$, if it is adjacent to two or more vertices in V_h, we remove all the edges connecting C and V_h. Further, we remove all the edges between large components in $G[V_l]$ and V_h. We define G' as the resulting graph. As every subtree of each high-degree vertex is small, G' is an s-forest. Since each connected component of G' contains at most one high-degree vertex, the third property holds. Further, since any large small-degree connected component is not connected to a high-degree vertex, the first property holds. The total number of removed edges is at most $|V_h| + 2|V_h| + (n/s + 2|V_h|) = \varepsilon n$. Thus, the second property also holds.

To provide an oracle access to G' for an edge $e = (v, w)$, we perform the BFS from v and w. See the full version for the details. □

Since our construction of G' is deterministic and we remove at most εn edges, the following corollary holds.

Corollary 3.2. *Let G and H be two forests of n vertices, and G' and H' be the graphs obtained from G and H by the partitioning oracle with a parameter $\frac{\varepsilon}{4}$, respectively. If $d(G, H) = 0$, then $d(G', H') = 0$ holds. If $d(G, H) \geq \varepsilon n$, then $d(G', H') \geq \varepsilon n/2$ holds.* □

Thus, we can preprocess the graph using the partitioning oracle, and it is sufficient to show that we can test isomorphism between two s-forests. We consider s-bounded-degree forests and s-rooted forests separately. Therefore, we construct a tester for the isomorphism of each corresponding tree in G' and H'. To test isomorphism between s-bounded-degree forests, we use a technique from [10]. We will develop a technique to test isomorphism between s-rooted forests in G' and H' under some conditions in the next section.

One technical issue of the partitioning oracle is that we cannot obtain the exact degree $\deg_{G'}(v)$ of a vertex v in G' since $\deg_G(v)$ can be up to n. Instead of computing the exact degree, we approximate the degree by randomly sampling incident edges as follows: Choose $i \in [\deg_G(v)]$ uniformly at random and apply the alive-edge query to the i-th incident edge. For a parameter $q \geq 1$, repeat this q times. Then, count the number of existing edges. Let c be this count. We use the value $\frac{c \deg_G(v)}{q}$ as an approximation to $\deg_{G'}(v)$ and denote it by $\widetilde{\deg}_{G',q}(v)$. The standard argument using Chernoff's bound gives the following lemma.

Lemma 3.3. *Let G' be the graph obtained from a graph G by the partitioning oracle. For any $\delta, \tau \in (0, 1)$ and a vertex v, there exists a polynomial $q = q_{3.3}(\delta, \tau)$ such that* $\Pr[|\widetilde{\deg}_{G',q}(v) - \deg_{G'}(v)| \leq \delta \deg_G(v)] \geq 1 - \tau$.

There is another issue of the partitioning oracle. If most parts of edges incident to a high-degree vertex v (i.e., a vertex with degree more than s) are removed by the partitioning oracle, the approximation $\widetilde{\deg}_{G',q}(v)$ may have a considerably large relative error. However, we can ensure that the number of such high-degree vertices v is sufficiently small. To make the argument more formal, for an integer $R > s$, we call a vertex v *R-bad* if $R \cdot \max(\deg_{G'}(v), 1) \leq \deg_G(v)$. Otherwise, we call v *R-good.* Note that an R-bad vertex must satisfy $\deg_G(v) \geq R > s$.

Thus, an R-bad vertex must be a high-degree vertex in G. Further, we call an s-rooted tree *R-bad* (resp. *R-good*) if the root vertex is R-bad (resp. R-good). Then, the number of vertices in R-bad s-rooted trees is bounded as follows.

Lemma 3.4. *Let G' be the s-forest obtained from a graph G by the partitioning oracle. For any $R > s$, the number of vertices in R-bad s-rooted trees of G' is at most $\frac{4sn}{R}$.*

By Lemma 3.4, random vertex sampling does not pick up any R-bad vertex with high probability if R is chosen sufficiently large. In Section 4, assuming that every s-rooted tree is R-good in the input graph, we will construct a tester for forest-isomorphism. In Section 5, combining Lemma 3.4 and the tester given in Section 4, we will construct a tester for any s-forest.

For later use, we define auxiliary procedures on s-rooted trees. First, the following lemma is useful.

Lemma 3.5. *Given a vertex $v \in V(G')$ in an s-rooted tree T, there is an algorithm that finds a root vertex* $\mathrm{root}(T)$ *with query complexity* $O(\mathrm{poly}(s))$.

Proof. Perform a BFS in G' starting from the vertex v until we find a high-degree vertex. The third property of Lemma 3.1 guarantees that we can find the high-degree vertex and it is $\mathrm{root}(T)$. □

Let $\mathcal{T}(s) = \{T^{(1)}, T^{(2)}, \ldots, T^{(t(s))}\}$ be the family of all rooted trees with at most s vertices, where $t(s) = |\mathcal{T}(s)|$. For an s-rooted tree T, let $\mathsf{Freq}(T)$ be the $t(s)$-dimensional vector whose i-th coordinate is the number of subtrees of $\mathrm{root}(T)$ isomorphic to $T^{(i)}$. As the root vertex uniquely exists in an s-rooted tree T, there is a unique $t(s)$-dimensional vector corresponding to T.

Since the degree of a root vertex can be up to n, we cannot exactly compute $\mathsf{Freq}(T)$. Instead, we approximate $\mathsf{Freq}(T)$ by randomly sampling subtrees in T. Given the root vertex v of an s-rooted tree T, we can define a procedure $\widetilde{\mathsf{Freq}_q(v)}$ to approximate $\mathsf{Freq}(T)$. Due to limitations of space, we give the procedure $\widetilde{\mathsf{Freq}}$ in the full version. Chernoff's bound guarantees the following.

Lemma 3.6. *For $s \geq 1$ and $\delta, \tau \in (0,1)$, there exists a polynomial $q = q_{3.6}(s, \delta, \tau)$ such that for any s-rooted tree T, $|\mathsf{Freq}(T)[i] - \widetilde{\mathsf{Freq}}_q(\mathrm{root}(T))[i]| \leq \delta \deg_G(v)$ for all $i \in [t(s)]$ with probability at least $1 - \tau$.*

It is also useful to approximate the number of vertices in an s-rooted tree. For an s-rooted tree T, we can define a procedure $\widetilde{\mathsf{Size}}$ that approximates $|V(T)|$ by randomly sampling subtrees of T and computing the number of vertices in the subtrees. Here, if T is guaranteed to be R-good for some $R > s$, the following holds. Again, we give the procedure $\widetilde{\mathsf{Size}}$ in the full version.

Lemma 3.7. *For any $s, R \geq 1$ and $\delta, \tau \in (0,1)$, there exists a polynomial $q = q_{3.7}(s, \delta, \tau)$ such that, for any R-good s-rooted tree T, $|\widetilde{\mathsf{Size}}_{G',q,R}(\mathrm{root}(T)) - |V(T)|| \leq \delta |V(T)|$ holds with probability at least $1 - \tau$. The expected number of queries issued by the procedure $\widetilde{\mathsf{Size}}$ is $O(\mathrm{poly}(s, R, \delta, \tau))$.*

4 When All Root Vertices Have Similar Degrees

In this section and the next section, we assume that we read the input graphs G and H through the partitioning oracle. Thus, we are allowed to use alive-edge queries and the procedures $\widetilde{\deg}$, $\widetilde{\mathsf{Freq}}$, and $\widetilde{\mathsf{Size}}$. Further, we assume that s is a constant that depends only on ε.

We consider the case in which the root of all components have similar degrees. Formally, we assume that each component in G and H is R-good s-rooted tree and that the degree of each s-rooted tree in G and H is greater than B and at most γB. Here, $B (> s)$ is an integer that can be up to $O(n)$ and $s, \gamma \geq 1$ is an arbitrary constant. We call such a forest an *R-good s-rooted forest with root degrees in* $(B, \gamma B]$. In this section, we will show that there is a forest-isomorphism tester for R-good s-rooted forest with root degrees in $(B, \gamma B]$ whose query complexity is a polynomial in γ and R.

With the tester given in this section, we can construct a tester for the general case as follows. After applying the partitioning oracle, the graph becomes a disjoint union of an s-bounded-degree forest and an s-rooted forest. We partition the s-rooted forest into several groups by the root degree. First, we ignore all the R-bad s-rooted trees from the graph. Since the number of R-bad trees is sufficiently small for a large R from Lemma 3.4, this does not affect so much. Second, if $\deg(\mathrm{root}(T))$ is greater than $O(\gamma^i)$ and at most $O(\gamma^{i+1})$, we consider that a tree T is in the i-th group. Note that there are $O(\log n)$ groups. Then we apply the isomorphism tester of this section to each group. If input graphs G and H are isomorphic, the tester must return YES (isomorphic) for all the groups. In contrast, if G and H are ε-far from isomorphic, there must exist a group such that the tester returns NO (not isomorphic) for the group. Here, there is one technical issue: The number of vertices in such a group might be different.

We resolve this issue. We assume that $n := |V(G)|$ and $n' := |V(H)|$ might be slightly different and the algorithm does not know the exact values of n and n' but know their approximations. Formally, we assume that our algorithm will be given a value $\tilde{n} \geq 1$, an approximation to n and n', and $\eta \in (0, 1)$ with $\frac{\tilde{n}}{n}, \frac{\tilde{n}}{n'} \in [1 - \eta, 1]$.

We can prove the following lemma.

Lemma 4.1. *Suppose that we are given $\varepsilon' > 0$, $\tilde{n} \geq 1$, $\gamma \geq 1$, $R, B > s$, $\tau \in (0, 1)$ and we can access s-forests G and H through the partitioning oracle, where $n = |V(G)|$ and $n' = |V(H)|$ might be different. Then, there exists $\eta = \eta_{4.1}(s, \varepsilon', \gamma, \tau, R) > 0$ with the following property. If G and H are R-good s-rooted forests with root degrees in $(B, \gamma B]$ with $\frac{\tilde{n}}{n}, \frac{\tilde{n}}{n'} \in [1 - \eta, 1]$, then there exists an algorithm that tests if $d(G, H) = 0$ or $d(G, H) \geq \varepsilon' \tilde{n}$ with probability at least $1 - \tau$. Assuming that s is constant, the query complexity is a polynomial in $R, \gamma, \varepsilon', \tau$ and does not depend on $B, \tilde{n}$. Further, denote by $q^{4.1}_{\mathrm{random}}(s, \gamma, \varepsilon', \tau)$ the number of random vertex queries the algorithm invokes. Then, $q^{4.1}_{\mathrm{random}}$ is a polynomial in $\gamma, \varepsilon', \tau$.*

Due to limitations of space, we provide the proof of Lemma 4.1 in the full version. In this section, we write an overview of the proof.

Since $\mathsf{Freq}(T)$ maps to a unique $t(s)$-dimensional vector corresponding to an s-rooted tree T, there is a unique multiset of vectors corresponding to an s-rooted forest G. For a $t(s)$-dimensional vector $\mathbf{w} \in \mathbb{N}_{<n}^{t(s)}$, let $\Psi_G[\mathbf{w}]$ be the number of s-rooted trees T in G such that $\mathsf{Freq}(T) = \mathbf{w}$. Note that Ψ_G can be seen as the sketch of G. Clearly, G is isomorphic to H if and only if $\Psi_G[\mathbf{w}] = \Psi_H[\mathbf{w}]$ for all $\mathbf{w}$. We use this property to create a tester. Since it is impossible to compute Ψ_G exactly, we resort to approximate it. We choose an integer $k \geq 1$, and divide each axis of the $t(s)$-dimensional space into k segments to make $k^{t(s)}$ cells. We then estimate the number of s-rooted trees in each cell. We call this estimation the *sketch* of G. We focus on computing the sketch.

For an integer $k \geq 1$, we define intervals $I_i = [\frac{\tilde{n}i}{(1-\eta)k}, \frac{\tilde{n}(i+1)}{(1-\eta)k})$ $(i \in \mathbb{N}_{<k})$. Note that, for every $0 \leq i \leq n-1$, there exists a unique interval I_j with $i \in I_j$. For a vector $\mathbf{u} \in \mathbb{N}_{<k}^{t(s)}$, let $\mathrm{Cell}(\mathbf{u})$ be the corresponding cell formed by intervals $I_{\mathbf{u}[1]}, \ldots, I_{\mathbf{u}[t(s)]}$. Further, for a vector $\mathbf{w} \in [0,n]^{t(s)}$, we define $\mathbf{Round}(\mathbf{w}) = \mathbf{u}$, where $\mathbf{u} \in \mathbb{N}_{<k}^{t(s)}$ is such that $\mathrm{Cell}(\mathbf{u}) \ni \mathbf{w}$.

For a vector $\mathbf{u} \in \mathbb{N}_{<k}^{t(s)}$, we approximate the number of s-rooted trees T in G with $\mathsf{Freq}(T) \in \mathrm{Cell}(\mathbf{u})$ by the following algorithm $\widetilde{\mathsf{Sketch}}$.

Algorithm 1. returns a map $\Phi : \mathbb{N}_{<k}^{t(s)} \to [0,n]$, given integers q_{loop}, q_{freq}, q_{size},R,k, a real $\tilde{n}$ and an R-good s-rooted forest G with root degrees in $(B, \gamma B]$ through the partitioning oracle. Here, $\Phi(\mathbf{u})$ is an approximation to the number of s-rooted trees T with $\mathsf{Freq}(T) \in \mathrm{Cell}(\mathbf{u})$.

```
procedure $\widetilde{\mathsf{Sketch}}_{q_{\mathrm{loop}},q_{\mathrm{freq}},q_{\mathrm{size}},R,k}(G)$
    Set $\Phi(\mathbf{u}) = 0$ for all $\mathbf{u} \in \mathbb{N}_{<k}^{t(s)}$
    for $j = 1, \ldots, q_{\mathrm{loop}}$ do
        Choose a vertex $u \in V(G)$ uniformly at random
        Perform a BFS from $u$ to find a root vertex $v$.
        $\mathbf{u} = \mathbf{Round}(\widetilde{\mathsf{Freq}}_{q_{\mathrm{freq}}}(v))$
        $\Phi(\mathbf{u}) = \Phi(\mathbf{u}) + 1/\widetilde{\mathsf{Size}}_{G,q_{\mathrm{size}},R}(v)$
    return $\frac{\tilde{n}}{q_{\mathrm{loop}}}\Phi$
```

To create a forest-isomorphism tester, we first compute the sketches of G and H by the algorithm $\widetilde{\mathsf{Sketch}}$, and then, we compute the minimum matching between the sketches. Here, the minimum matching is defined as the min-cost flow of complete bipartite graphs where vertices correspond to the cells of the sketches and the weight of an edge is the L1 distance between two cells of the sketches in the $t(s)$-dimensional space. Since the L1 distance in the $t(s)$-dimensional space corresponds to the number of different subtrees in s-rooted trees, we can prove that the a minimum cost matching between the sketches is a good approximation to $d(G,H)$ with high probability. Thus, it suffices to compute the sketches of G and H and the minimum cost matching between them. Note that we do not have to make any query to G and H to compute the minimum cost matching.

5 General Case

In this section, we prove Theorem 1.1. Missing proofs and procedures of this section are given in the full version. Again G and H denote the graphs given through the partitioning oracle and s is constant. For an integer $L \geq 1$, we call $G_1, \ldots, G_L \subseteq G$ *a partition of* G if each G_i is a union of connected components in G and G is a disjoint union of $G_1, \ldots, G_L$. The following lemma allows us to consider each part in the partition separately.

Lemma 5.1. *Let $L \geq 1$ be an integer and $G_1, \cdots, G_L$ (resp. $H_1, \cdots, H_L$) be any partition of G (resp. H). Then, for any $\beta_1, \cdots, \beta_L \geq 0$ summing up to 1, the following holds: For any $\varepsilon > 0$, if $d(G, H) \geq \varepsilon n$, there exists $i \in [L]$ such that $d(G_i, H_i) \geq \beta_i \varepsilon n$ holds.*

To construct a tester for the isomorphism of s-forests, we first give a partition of an s-forest and apply Lemma 5.1. Then we test the isomorphism of each corresponding partition of G and H. That is, we check $d(G_i, H_i) = 0$ or $d(G_i, H_i) \geq \beta_i \varepsilon n$ for each i. Here, if $d(G, H) = 0$, all parts of the partition in G and H are isomorphic, so all the tests must output YES (with high probability). If $d(G, H) \geq \varepsilon n$, there must be an index i where the test outputs NO. To provide oracle accesses to G_i and H_i, we estimate the size of $V(G_i)$ and $V(H_i)$ by random sampling. If they are sufficiently far, we immediately return NO. If they are sufficiently small, we simply ignore G_i and H_i. Otherwise, we can provide the oracle accesses to G_i and H_i that costs for each query at most $\mathrm{poly}(L)$ queries to G and H. Using this access, we test whether $d(G_i, H_i) \geq \beta_i \varepsilon n$.

To provide a partition of an s-forest, we introduce a new notion. For $\alpha, \gamma \geq 1$, $\mu > 0$, and a tree T, we say that T is *on the* (α, γ, μ)*-boundary*, if there exists an integer $i \geq 1$ with $1 - \mu \leq \deg(\mathrm{root}(T))/(\alpha\gamma^i) \leq 1 + \mu$. We denote by $B_{\alpha,\gamma,\mu}(G)$ the number of vertices in the trees of G that are on the (α, γ, μ)-boundary. For $\lambda > 0$, we call α (γ, μ, λ)*-good with respect to* G if $B_{\alpha,\gamma,\mu}(G) < \lambda n$. We can show that, if we choose α from $[1, \gamma]$ at random, α is (γ, μ, λ)-good with high probability.

Lemma 5.2. *Suppose that α is chosen from $[1, \gamma]$ uniformly at random. Then, for $\gamma \geq 2$, $\mu \in (0, 1/3)$, and $\lambda \in (0, 1)$, α is (γ, μ, λ)-good with respect to G with probability at least $1 - \frac{4\gamma\mu}{\lambda}$.*

We consider a partition of an s-forest G. Let α, γ, μ, and R be values chosen later. Let $G^{[0]}_{s,\alpha,\gamma,\mu,R}$ be the maximal s-bounded-degree forest in G and $G^{[1]}_{s,\alpha,\gamma,\mu,R}$ be the union of R-good s-rooted trees with root degree in $(s, \alpha\gamma]$ that are not on the (α, γ, μ)-boundary in G. Similarly, for $2 \leq i \leq L$, where $L = \lceil \log n / \log \gamma \rceil$, let $G^{[i]}_{s,\alpha,\gamma,\mu,R}$ be the union of R-good s-rooted trees with root degree in $(\alpha\gamma^{i-1}, \alpha\gamma^i]$

that are not on the (α, γ, μ)-boundary in G. Finally, let $G^{[L+1]}_{s,\alpha,\gamma,\mu,R}$ be the remaining trees that are not assigned to any partition so far. That is, $G^{[L+1]}$ is the union of trees that are R-bad or on the (α, γ, μ)-boundary in G. We omit the subscript of $G^{[i]}_{s,\alpha,\gamma,\mu,R}$ if it is clear from the context. Note that we can write $G = G^{[0]} \cup G^{[1]} \cup \cdots \cup G^{[L+1]}$. We use the same notion for the other graph H.

We define a procedure that, given a vertex $v \in V(G)$, returns i with $v \in G^{[i]}$ as follows. Our procedure first determines if v is in an s-bounded-degree tree by performing a BFS from v until we visit $O(s)$ vertices. If we cannot find a high-degree vertex, $v \in G^{[0]}$. Otherwise, for a parameter $q \geq 1$, we invoke $\widetilde{\deg}_q(\mathrm{root}(v))$ and return an appropriate output. We call this procedure $\mathsf{Which}_q(v)$.

Here, the technical issue is that the procedure Which may output a wrong value. We show that Which outputs the correct value with high probability for any partition of G except for $G^{[L+1]}$ and that the size of $G^{[L+1]}$ is sufficiently small.

Lemma 5.3. *For any $\tau \in (0, 1)$ and $R \geq 1$, there exists a polynomial $q = q_{5.3}(\gamma, \mu, R, \tau)$ such that the procedure $\mathsf{Which}_q(v)$ outputs a correct value with probability $1 - \tau$ for $v \in V(G^{[0]}_{s,\alpha,\gamma,\mu,R}) \cup \cdots \cup V(G^{[L]}_{s,\alpha,\gamma,\mu,R})$.*

Lemma 5.4. *For any $\gamma \geq 2$ and $\lambda \in (0, 1)$, there exist $R = O(s/\lambda)$, $\mu = O(\lambda/\gamma)$ such that if α is chosen from $[1, \gamma]$ uniformly at random, $|V(G^{[L+1]}_{s,\alpha,\gamma,\mu,R})| \leq \lambda n$ holds with probability $1 - O(1)$.*

Using the procedure Which, we can approximate the number of vertices in $G^{[i]}$ by random sampling. For $i \in \mathbb{N}_{\leq L}$, we denote by $\mathsf{Size}_{q_{\mathrm{loop}}, q_{\mathrm{which}}}(G, i)$ the algorithm that samples q_{loop} vertices uniformly at random, and applies $\mathsf{Which}_{q_{\mathrm{which}}}$ for each sampled vertex, and then approximates $|V(G^{[i]})|$. By Chernoff's bound, we obtain the following lemma.

Lemma 5.5. *For any $\delta, \tau \in (0, 1)$ and parameters α, γ, μ, and R, there exist polynomials $q_{\mathrm{loop}} = q_{\mathrm{loop}5.5}(\delta, \tau)$ and $q_{\mathrm{which}} = q_{\mathrm{which}5.5}(\delta, \tau)$ such that the following holds: For any $\lambda \in (0, 1)$ with $|V(G^{[L+1]}_{s,\alpha,\gamma,\mu,R})| \leq \lambda n$, $|\mathsf{Size}_{q_{\mathrm{loop}}, q_{\mathrm{which}}}(G, i) - |V(G^{[i]}_{s,\alpha,\gamma,\mu,R})|| \leq (\lambda + \delta)n$ with probability $1 - \tau$.* □

Further, using the procedure Which, we can provide oracle accesses to $G^{[i]}$ for $i \in \mathbb{N}_{i \leq L}$. Let $\mathsf{Random}_q(G, i)$ denote the procedure that repeats itself to pick up a vertex v in G uniformly at random and invokes the procedure $\mathsf{Which}_q(v)$ and returns v if the returned value of Which is i.

Lemma 5.6. *For every $\delta, \tau \in (0, 1)$ and parameters α, γ, μ, and R, there exist polynomials $q = q_{5.6}(\delta, \tau)$ and $\lambda = \lambda_{5.6}(\delta, \tau)$ such that the following holds for every $i \in \mathbb{N}_{\leq L}$: If $|V(G^{[i]})| \geq \delta n$ and $|V(G^{[L+1]})| \leq \lambda n$, the procedure $\mathsf{Random}_q(G, i)$ outputs a vertex of $G^{[i]}$ uniformly at random by invoking the procedure Which_q at most $O(1/(\delta\tau))$ times with probability $1 - \tau$.*

The sketch of the proof of Theorem 1.1 is as follows. As we mentioned, it suffices to create an isomorphism tester between $G^{[i]}$ and $H^{[i]}$ for each $i \in \mathbb{N}_{\leq L}$.

First, set $\gamma = 2s$ and choose $\alpha \in [1, \gamma]$ uniformly at random. From Lemma 5.4, $|V(G^{[L+1]})|$ and $|V(H^{[L+1]})|$ are small with high probability. Thus, we can apply the procedures Which, Size and Random to the input graphs. From Lemmas 5.3, 5.5, and 5.6, these procedures output the correct value with sufficiently high probability. Using the procedure Size, we can test if $|V(G^{[i]})|$ and $|V(H^{[i]})|$ are large and sufficiently close. Then, we can test forest-isomorphism between $G^{[i]}$ and $H^{[i]}$ (with high probability) by providing oracle accesses to $G^{[i]}$ and $H^{[i]}$ through the procedure Random. For $i = 0$, we use a method proposed by [10] with a little modification. For $1 \leq i \leq L$, we use the method in Section 4. Here, every parameter depends on polylog(n) assuming that s is constant. Thus, the query complexity of our forest-isomorphism tester is polylog(n) in total. See the full version for the detailed description of the tester for forest-isomorphism.

References

1. Alon, N., Fischer, E., Newman, I., Shapira, A.: A combinatorial characterization of the testable graph properties: It's all about regularity. SIAM Journal on Computing 39(1), 143–167 (2009)
2. Alon, N., Kaufman, T., Krivelevich, M., Ron, D.: Testing triangle-freeness in general graphs. SIAM Journal on Discrete Mathematics 22(2), 786–819 (2008)
3. Ben-Eliezer, I., Kaufman, T., Krivelevich, M., Ron, D.: Comparing the strength of query types in property testing: the case of testing k-colorability. In: SODA 2008: Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1213–1222 (2008)
4. Fischer, E., Matsliah, A.: Testing graph isomorphism. SIAM Journal on Computing 38(1), 207–225 (2008)
5. Goldreich, O.: Introduction to testing graph properties. In: Goldreich, O. (ed.) Property Testing. LNCS, vol. 6390, pp. 105–141. Springer, Heidelberg (2010)
6. Goldreich, O., Goldwasser, S., Ron, D.: Property testing and its connection to learning and approximation. Journal of the ACM 45(4), 653–750 (1998)
7. Goldreich, O., Ron, D.: Property testing in bounded degree graphs. Algorithmica 32(2), 302–343 (2002)
8. Hassidim, A., Kelner, J.A., Nguyen, H.N., Onak, K.: Local graph partitions for approximation and testing. In: FOCS 2009: Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science, pp. 22–31 (2009)
9. Kaufman, T., Krivelevich, M., Ron, D.: Tight bounds for testing bipartiteness in general graphs. SIAM Journal on Computing 33(6), 1441–1483 (2004)
10. Newman, I., Sohler, C.: Every property of hyperfinite graphs is testable. SIAM Journal on Computing 42(3), 1095–1112 (2013)
11. Rubinfeld, R., Sudan, M.: Robust characterizations of polynomials with applications to program testing. SIAM Journal on Computing 25(2), 252–271 (1996)
12. Tanigawa, S., Yoshida, Y.: Testing the supermodular-cut condition. Algorithmica, 1–11 (2013)

Parameterized Approximation Schemes Using Graph Widths[⋆]

Michael Lampis

Research Institute for Mathematical Sciences (RIMS), Kyoto University
mlampis@kurims.kyoto-u.ac.jp

Abstract. Combining the techniques of approximation algorithms and parameterized complexity has long been considered a promising research area, but relatively few results are currently known. In this paper we study the parameterized approximability of a number of problems which are known to be hard to solve exactly when parameterized by treewidth or clique-width. Our main contribution is to present a natural randomized rounding technique that extends well-known ideas and can be used for both of these widths. Applying this very generic technique we obtain approximation schemes for a number of problems, evading both polynomial-time inapproximability and parameterized intractability bounds.

1 Introduction

Approximation algorithms and parameterized complexity are two of the most popular ways of dealing with NP-hard optimization problems. Nevertheless, the two sets of techniques are usually treated independently. It's therefore a very natural question whether combining the techniques of both theories can be used to obtain algorithmic results which are out of reach for each one of them separately. This has often been identified as a promising research field (see [17] for a survey), but its development has so far been somewhat limited. The goal of this paper is to add some results in this area by designing parameterized approximation schemes for problems which are both parameterized intractable (W-hard) and hard to approximate in polynomial time (APX-hard).

The problems we will focus on are optimization problems on graphs of bounded treewidth or clique-width. These two graph widths are of central importance to parameterized complexity theory. At the same time, they play a significant role in the design of approximation algorithms, since subroutines employing them are often used as building blocks of larger algorithms. Therefore, understanding the extent to which we can efficiently approximate problems which remain W-hard for these widths is of potentially great importance from several points of view.

Let us begin by stating a result representative of the aims of this paper.

[⋆] Research partially supported by a Scientific Grant-in-Aid from Ministry of Education, Culture, Sports, Science and Technology of Japan. See http://arxiv.org/abs/1311.2466 for the full version of this paper.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 775–786, 2014.

Theorem 1 (Partial Statement).
There exists a randomized $(1+\epsilon)$*-approximation algorithm for* Max Cut *running in time* $(\log n/\epsilon)^{O(w)}n^{O(1)}$*, where* w *is the input graph's clique-width.*

Max Cut is of course a problem of central importance in the contexts of both approximability and parameterized complexity. It is APX-hard (so an approximation ratio of $1+\epsilon$ is probably impossible in polynomial time) and W-hard parameterized by clique-width (so the fastest exact algorithm probably needs time roughly n^w). Our main point here is that using a *parameterized approximation* approach we can evade these lower bounds, leading to a $(1+\epsilon)$-approximation running in time only $(\log n)^{O(w)}$, that is, an FPT approximation scheme. More generally, the goal of this paper is to provide, in a *uniform* way, similar approximation (or bicriteria approximation) schemes for a diverse set of W-hard and APX-hard graph problems. The problems for which we will provide algorithms are the following: Max Cut, Edge Dominating Set, Bounded Degree Deletion, Capacitated Dominating Set, Capacitated Vertex Cover, Equitable Coloring and Graph Balancing. For most of these problems we are able to provide equally efficient algorithms for both treewidth and clique-width (a detailed description of all results is given further below).

Paper Overview: In this paper we adopt a generic technique that is a variation of the standard dynamic programming used for treewidth and clique-width. We observe that for a number of problems which are parameterized intractable for these widths, the hardness intuitively stems from the fact that large *integers* need to be stored in the dynamic programming table. These integers are usually calculated simply by *adding* previously calculated entries (all the problems listed above fall into this general category, though for some this is not obvious). We want to shrink the table, and thus speed up the algorithms, by storing these integers approximately.

The basic idea we use is very natural. We fix a parameter $\delta > 0$ and represent all integers in $\{1,\dots,n\}$ by rounding them to the closest integer power of $(1+\delta)$. If δ is not too small ($\delta = \Omega(\frac{1}{\log^c n})$) the natural dynamic programming table's size is dramatically reduced from n^w to $(\log n)^{O(w)}$. The obvious obstacle to this approach, however, is that during the process of running a dynamic programming algorithm on the approximate values the rounding errors will propagate and potentially pile up to a large error. How can we keep the errors under control?

In this paper we suggest a very natural randomized rounding approach to this problem. The main contribution is to show that this rounding idea can be seamlessly incorporated into the standard dynamic programming techniques of treewidth and clique-width to give efficient approximation schemes. In order to make the transition from exact to approximate algorithms as cleanly as possible we separate the analysis into two parts. First, we introduce an abstract model of computation, called Approximate Addition Trees, which captures the essence of the rounding ideas we described. We fully analyze the approximation performance of these Trees and prove some general approximation theorems. Then, relying on this analysis we give a series of approximation algorithms using clique-width and treewidth. It's worth stressing at this point that all the algorithms we will present follow the standard dynamic programming mold that should be very familiar to readers accustomed to graph widths. The important difference is that their analysis, in addition to standard methods, also crucially relies on our results on Approximate Addition Trees (which we can use as a black box). Thus, by abstracting away the Addition Trees, our technique can be viewed as a natural extension of well-known ideas. The hope is that this modularization will allow our technique to be easily reused and eventually become part of the standard graph width toolkit.

Thus, what is left to describe is the workings of Approximate Addition Trees. This is, expectedly, the most technical part of the paper. As we will see, there do exist some important special cases where a complicated analysis can be avoided (notably, when the input tree is balanced) and there is some value in these cases since they can help make some algorithms deterministic. However, in order to obtain the more interesting results of this paper we need an analysis of full generality. In other words, we need to establish an approximation theorem that works for all Addition Trees without making any special assumptions about their structure. Our main technical contribution is that we do establish such a result and this allows us to analyze all the algorithms of this paper in terms of Addition Trees. We thus present a robust, unified technique that works for both treewidth and clique-width (and potentially other similar graph widths), without relying on any non-trivial width-specific properties.

Summary of Results: Let us now formally state the algorithmic results presented in this paper. Full problem definitions are given further below and in the full version of this paper. In the following theorems, $n^{O(1)}$ factors are omitted from the running times.

Theorem 1. *Given an n-vertex graph $G(V, E)$, a clique-width expression with w labels, and an error parameter $\epsilon > 0$, there exist randomized algorithms which, with high probability, achieve the following:*

- *Produce a solution to* Max Cut *with size at least* $\frac{\text{OPT}}{1+\epsilon}$ *in time* $(\log n/\epsilon)^{O(w)}$.
- *Produce an approximate solution to* Edge Dominating Set *with size at most* $(1+\epsilon)\text{OPT}$ *in time* $(\log n/\epsilon)^{O(w)}$.
- *Given an integer k, either decide (correctly) that G does not admit an* Equitable Coloring *with k colors or produce a valid k-coloring where the ratio of the sizes of any two color classes is at most* $(1+\epsilon)$ *in time* $(\log n/\epsilon)^{O(k)}2^{kw}$.
- *Given an integer Δ find a set of vertices that is at most as large as the optimal solution for* Bounded Degree Deletion *to degree Δ and whose deletion makes the maximum degree at most* $(1+\epsilon)\Delta$, *in time* $(\log n/\epsilon)^{O(w)}$.
- *Given a capacity for each vertex, find a* Capacitated Dominating Set *of size at most* OPT, *such that all but at most ϵn vertices are dominated, in time* $(\log n/\epsilon)^{O(w)}$.

In addition, if instead of a clique-width expression we are given a tree decomposition of width w, there exist deterministic algorithms, with the same running times, achieving all the above.

Theorem 2. *Given an n-vertex graph $G(V, E)$, a tree decomposition of width w, and an error parameter $\epsilon > 0$, there exist deterministic algorithms which achieve the following:*

- *Produce an approximate solution with cost at most* $(1 + \epsilon)$OPT *for* Graph Balancing, *in time* $(\log n)^{O(w)}$.
- *Given a capacity for each vertex, find a* Capacitated Dominating Set *(or* Capacitated Vertex Cover*) of size at most* OPT, *such that no capacity is violated by a factor of* $(1 + \epsilon)$ *or more, in time* $(\log n)^{O(w)}$.

All algorithms are given in Section 3 and in the full version of this paper.

Previous Work: Max Cut was shown to be W-hard when parameterized by clique-width in [12]. The problem is known to be APX-hard in general [19]. In Edge Dominating Set we want to select the smallest possible set of edges such that all edges share an endpoint with a selected edge. This problem is also APX-hard and W-hard for clique-width [12]. Let us also mention that all other considered problems are both W-hard for treewidth and APX-hard in general [11,23,9,2,8]. More details are given in the full version of this paper.

Very few FPT approximation schemes are currently known. For an overview of the most important results see the survey by Marx [17]. The same paper gives an FPT approximation scheme for Max Vertex Cover parameterized by the size of the cover. This is extended in [22] to an FPT approximation scheme for Max Cover. See also [5] for FPT approximation schemes for related covering problems. Sum Edge Multicoloring is a rare example of a problem currently known to admit an FPT approximation scheme parameterized by treewidth [16].

In this paper we focus on problems parameterized by treewidth or clique-width. For an introduction to these notions see [4,7,10]. It was initially believed that problems solvable on trees are almost always FPT parameterized by treewidth. Gradually, many exceptions were discovered. Most relevant to our purposes are problems which are solvable in polynomial time for constant treewidth, but not FPT. Some examples of such problems when parameterized by treewidth (in addition to the problems we consider in this paper) are given in the following papers: [1,15,18,11,20,21,13].

Preliminaries and Notation: We use boldface to denote vectors, for example $\boldsymbol{d}$. Sometimes a vector $\boldsymbol{s} \in S^k$ for S a set and $k \in \mathbb{N}$ will also be viewed as a function from $\{1, \dots, k\}$ (or some other convenient set of size k) to S, and vice-versa.

We use standard graph theoretic notation. For an undirected graph $G(V, E)$ and $X \subseteq V$ we denote by $G[X]$ the graph induced by X. We will use the standard notion of "nice" tree decompositions (see the survey by Bodlaender and Koster [4]). We will also use the notion of clique-width (see [7,10]). The set of graphs of clique-width w is the set of vertex-labelled graphs with w labels which can be constructed inductively using the following four operations: Introduce, Union, Join, and Rename. When dealing with clique-width, we assume that a clique-width expression is given in the input. We view it as a rooted binary tree where each node is labeled with its corresponding operation.

2 Approximate Addition Trees

In this section we describe an abstract model of computation which one may naturally call Addition Trees. In such a Tree each node calculates a value that is the sum of the values of its children. We also define an Approximate version of these trees, where each node *probabilistically* rounds calculated values to integer powers of $(1+\delta)$, for some parameter $\delta > 0$. These trees capture the rounding scheme that will be the heart of the algorithms of the next section. Our goal is to prove that the values of Approximate and Exact Addition Trees are almost always very close, even if δ is not too small (we want $\delta = \Omega(1/\log^c n)$). We require δ to be in this range, because in the end the algorithms of the next section will run in time roughly $(\log n/\delta)^w$. Thus, if we allow δ to become inverse polynomial in n (which would make this section easy), we will get algorithms as slow as the trivial exact ones.

Intuitively, there are two extreme cases to consider here. First, if a tree is balanced (that is, it has logarithmic height), it is not hard to establish that rounding errors cannot pile up too badly (Theorem 3). Somewhat surprisingly, this easy case is already sufficient to obtain several non-trivial algorithmic results, because tree decompositions can always be balanced reasonably well (more details are given in the next section). However, to obtain the more interesting results of this paper we need to deal with clique-width, where the input decomposition cannot in general be balanced. Therefore, we have to deal with general Addition Trees.

Our proof strategy then is to move on to a second extreme case: caterpillars. Here the height of the tree is large, but we know that one operand of each addition has no previously accumulated error. Despite this, this is actually a pretty hard case. To see why, intuitively one can think of the accumulated error at each level of the tree as a random variable, since the rounding performed on each step is randomized. The error has some probability of increasing and some of decreasing, depending on how we round, but it changes by at most a factor of $(1+\delta)$ in each step. So, if we look at its logarithm (with base $(1+\delta)$) it can (randomly) increase or decrease by at most 1. Thus, the process we are trying to analyze is akin to a memoryless random walk on the real line. We want to prove that the end result of the walk after n steps is with high probability contained in an area of size only roughly $1/\delta = \text{poly}(\log n)$. Such a statement cannot be shown with standard tools, such as Chernoff bounds, because the result they give is too weak (they give concentration in an area of size roughly $\sqrt{n}$). Instead, we need to use moment-generating functions to derive a problem-specific concentration bound that takes into account our algorithm's tendency to "self-correct". Because of this special tendency (Lemma 1), our random walk is much more strongly concentrated around its expectation than usual random walks.

Thus, eventually we establish (in Lemma 2) that the approximation error is small in the caterpillar case, with high probability. Once this has been shown we can extend the same ideas to prove a sufficiently good approximation theorem for general trees by performing induction on the "balanced height" of the tree (Theorem 4).

We remark that the only parts of this section needed to follow the results of the next one are Definitions 1,2 and Theorems 3,4. Let us now proceed to give full details.

Definition 1. *An Addition Tree is a full rooted binary tree T where we associate with each leaf l a non-negative integer* input x_l *and with each node v a non-negative integer* value y_v*. The inputs are given with T. The value of each node is calculated as follows:*

1. *For each leaf l we set $y_l := x_l$*
2. *For each internal node v with two children u_1, u_2 whose values have already been calculated we set $y_v := y_{u_1} + y_{u_2}$.*

Definition 2. *An Approximate Addition Tree with parameter δ is a full rooted binary tree T where we associate with each leaf l a non-negative integer* input x_l *and with each node v a non-negative* approximate value z_v*. The approximate value of each node is calculated as follows:*

1. *For each leaf l we set $z_l := x_l$*
2. *For each internal node v with two children u_1, u_2 we set $z_v := z_{u_1} \oplus z_{u_2}$, where the $\oplus$ operation is defined below.*

Let $a_v := z_{u_1} + z_{u_2}$. We will call a_v the initial approximate value *of v.*

We use $\oplus$ to denote the following operation: for two non-negative numbers x_1, x_2 we define $x_1 \oplus x_2 := 0$ if $x_1 = x_2 = 0$. Otherwise, select a real number $r \in (0,1)$ uniformly at random and set $x_1 \oplus x_2 := (1+\delta)^{\lfloor \log_{(1+\delta)}(x_1+x_2)+r \rfloor}$.

Since it is not hard to see that for any node v for which $y_v = 0$ an Approximate Addition Tree will also have $z_v = 0$, we are only concerned with the approximation error for nodes where $y_v \neq 0$. Therefore, in the remainder we will implicitly assume that we are talking about a tree where for all v, $y_v > 0$, because sub-trees with value 0 can be ignored.

Definition 3. *Let v be a node of an Addition Tree, y_v its (positive) value and z_v its approximate value calculated if we view the tree as an Approximate Addition Tree. Then the error λ_v is defined as $\lambda_v := \log_{(1+\delta)} \frac{z_v}{y_v}$.*

Informally, λ_v measures how many "$(1+\delta)$ intervals" off our approximation is from the correct interval.

Before we go on, let us make an easy observation that will be sufficient to handle an important special case.

Theorem 3. *If an Approximate Addition Tree has maximum depth h then for all nodes v we always have $|\lambda_v| \leq h+1$. Therefore, if $\delta < \frac{\epsilon}{3h}$ then for all v we have* $\max\{\frac{z_v}{y_v}, \frac{y_v}{z_v}\} < 1+\epsilon$.

As a consequence of Theorem 3 we get that in trees of height $O(\log n)$ we can set $\delta = \Theta(1/\log n)$ and get error at most $1+\epsilon$ everywhere *deterministically.*

The main intuitive observation that we will use to give an approximation guarantee can be summarized as follows: the process by which the initial approximation a_v is calculated is “self-correcting”.

Lemma 1. *Consider an Approximate Addition Tree with parameter* $\delta < \frac{1}{2}$*. Let* v *be an internal node with two children* u_1, u_2*. If* $\max\{|\lambda_{u_1}|, |\lambda_{u_2}|\} < \frac{1}{4\delta}$ *then* $|\log_{(1+\delta)} \frac{a_v}{y_{u_1}+y_{u_2}}| \leq \max\{|\lambda_{u_1}|, |\lambda_{u_2}|\} - \frac{1}{20}\delta p_v |\lambda_{u_1} - \lambda_{u_2}|$.

Informally, Lemma 1 states that the maximum error in a node tends to decrease by a value that is proportional to the absolute difference of the errors of its two children. Lemma 1 will be our main tool in proving that with high probability the values of an Approximate Addition Tree are not too far from those of the corresponding exact tree. We will proceed by induction, starting from a simple case of path-like trees (caterpillars).

Lemma 2. *If* T *is a caterpillar and* $\lambda \in (\frac{2}{\sqrt{\delta}}, \frac{1}{4\delta})$ *we have* $\mathbf{Pr}\left[\exists v \in T: \ |\lambda_v| > \lambda\right] \leq 2n^2 e^{-\frac{\lambda\sqrt{\delta}}{20}}$.

After extending this to arbitrary trees, we are led to the main theorem of this section

Theorem 4. *Let* T *be an Approximate Addition Tree on* n *nodes with parameter* $\delta \in (0, \frac{1}{2})$*. There exists a fixed constant* $C > 0$ *such that for all* $\epsilon \in (0, \frac{1}{8})$ *and sufficiently large* n*, if* $\delta < \frac{\epsilon^2}{C \log^6 n}$ *we have*

$$\mathbf{Pr}\left[\exists v \in T : \max\{\frac{z_v}{y_v}, \frac{y_v}{z_v}\} > 1+\epsilon\right] \leq n^{-\log n}$$

3 Approximation Schemes

We are now ready to use the results of the previous section to design some approximation algorithms. As mentioned, we only need Definitions 1,2 and Theorems 3, 4. Let us first describe the general technique.

An important property of all the problems we consider is that they admit an exact dynamic program that takes time (roughly) n^w. These dynamic programs calculate a set of w integers on each node of the decomposition by adding previously calculated values or values read from the graph. The idea is to reuse this dynamic program, but round all values to integer powers of $(1+\delta)$ and perform all additions with the $\oplus$ operation. Note that, we are in fact able to also handle some other auxiliary operations besides addition (such as comparisons). It will, however, be important that our dynamic programs avoid using subtractions (because then we lose the approximation guarantee) and part of our effort is devoted to achieving this.

How do we analyze the approximation ratio of such an algorithm? On a high level, the strategy is to inductively construct Addition Trees such that their

Approximate version correspond to our algorithm's table and their exact version to actual solutions. We then argue that if our algorithm's table has a large error, we have a bad Approximate Addition Tree, which only happens with very low probability.

What remains to say is what values we select for the parameter δ. Here we have two choices. In the general case, we set δ to the value dictated by Theorem 4. This works quite well essentially all the time, as we can guarantee that with high probability the Addition Trees we consider during the analysis of our algorithm will have almost correct values. We can thus obtain randomized approximation schemes for both clique-width and treewidth with the promised running time of roughly $(\log n/\epsilon)^{O(w)}$. This is the general technique we consider to be the main contribution of this paper.

Nevertheless, as mentioned there exists an important special case where things can become simpler. It is known that for any graph of treewidth w there exists a tree decomposition of width $O(w)$ and only logarithmic (in n) height [3]. Thus, another approach available to us is to use this theorem to first balance the decomposition and then rely on Theorem 3, instead of the more general Theorem 4. Unfortunately, this argument is specific to treewidth (similar balancing results for clique-width are much more inefficient and would lead to an unreasonable running time of roughly $(\log n)^{2^w}$ [6]). It also fails to speed up even the treewidth algorithms much, because of the blow-up in w. Despite these shortcomings, we still believe there may be some value in this treewidth-specific balancing trick and mention it as an alternative approach because it allows us to get rid of randomization in this case. To the best of our knowledge, this is the first application of tree decomposition balancing theorems to approximation algorithms.

In the rest of this section we describe the algorithms for Max Cut and Edge Dominating Set. Algorithms for Equitable Coloring, Capacitated Dominating Set, Capacitated Vertex Cover, Bounded Degree Deletion, and Graph Balancing are given in the full version of this paper.

3.1 Max Cut

In this section we attack the Max Cut problem parameterized by clique-width. In Max Cut we are given a graph $G(V, E)$ and are asked to find a partition of V into $L, R \subseteq V,\ L = V \setminus R$ such that the number of edges with exactly one endpoint in L is maximized.

We will follow the straightforward dynamic program for this problem, as used for example in [12]. The idea is to describe a partial solution by keeping track of the intersection of each label set with L. The first difference is of course that we will round all values to save time. The second (somewhat counter-intuitive) difference is that for each label set we will keep track of the size of its intersection with *both* L and R. In the exact program this would be wasteful, since one of these two numbers can be calculated from the other, using the (known) size of the whole label set. However, here we are trying to implement a dynamic program that relies only on additions.

Dynamic Program. As mentioned, we view the clique-width expression as a rooted binary tree. First, define the set $B := \{0\} \cup \{(1+\delta)^j \mid j \in \mathbb{N}\}$. Informally, B is the set of rounded values that may appear in our table. Even though B is infinite, whenever the algorithm produces an entry with value larger than $(1+\epsilon)n^2$ we simply drop that entry. It will not be hard to see that this will not affect the analysis if all entries are within a $(1+\epsilon)$ factor of being correct (then nothing will be dropped). Observe that the size of B then becomes $\log_{(1+\delta)}(n^2) = \text{poly}(\log n/\epsilon)$, if we set δ according to Theorem 4.

The dynamic programming table D_i for a node i is a subset of $B^w \times B^w \times B$. The informal meaning is that an entry $(\boldsymbol{l}, \boldsymbol{r}, c) \in D_i$ if and only if there exists a partition of the subgraph of G represented by the sub-tree rooted at i into L_i, R_i such that: first, for all $l \in \{1, \ldots, w\}$ the number of vertices in L_i (respectively R_i) with label l is roughly $\boldsymbol{l}(l)$ (respectively $\boldsymbol{r}(l)$); and second, the number of edges with exactly one endpoint in L_i is roughly c.

A dynamic programming algorithm can now be formulated in a straightforward way. It is easy to fill the table for initial nodes. For Rename and Union nodes the exact dynamic program would perform some addition, which we now replace with the $\oplus$ operation. For example, consider a Rename node with labels $l_1 \to l_2$. For each entry $(\boldsymbol{l}, \boldsymbol{r}, c)$ in the child's table we create an entry $(\boldsymbol{l'}, \boldsymbol{r'}, c)$ in the current node as follows: we set $\boldsymbol{l'}(l_1) := 0$, $\boldsymbol{l'}(l_2) := \boldsymbol{l}(l_1) \oplus \boldsymbol{l}(l_2)$ and $\boldsymbol{l'}$ the same as $\boldsymbol{l}$ for other labels to make the vector $\boldsymbol{l'}$ of a new entry (similarly for $\boldsymbol{r'}$). In the same way, for a Union node, for each entry $(\boldsymbol{l_1}, \boldsymbol{r_1}, c_1)$ in the first child's table and for each entry $(\boldsymbol{l_2}, \boldsymbol{r_2}, c_2)$ in the second child's table we construct an entry $(\boldsymbol{l_1} \oplus \boldsymbol{l_2}, \boldsymbol{r_1} \oplus \boldsymbol{r_2}, c_1 \oplus c_2)$, where $\oplus$ is applied component-wise for vectors.

For Join nodes with labels l_1, l_2 we do something similar. For each entry $(\boldsymbol{l}, \boldsymbol{r}, c)$ in the child's table construct a new vector with the same $\boldsymbol{l}, \boldsymbol{r}$ and $c' := c \oplus (\boldsymbol{l}(l_1) \cdot \boldsymbol{r}(l_2) + \boldsymbol{l}(l_2) \cdot \boldsymbol{r}(l_1))$. Note that if the elements of $\boldsymbol{l}, \boldsymbol{r}$ are known with error at most $(1+\epsilon)$ then the second term of this addition is known with error at most $(1+\epsilon)^2 \approx 1 + 2\epsilon$.

Analysis. First, observe that because the running time of the algorithm is polynomial in the size of the tables, the algorithm clearly achieves the running time stated in Theorem 1. We only have to prove its approximation guarantee.

Any node i of the clique-width expression defines a subgraph G_i of G (the graph produced by the sub-expression rooted at i). A partial solution is simply a restriction of a solution L, R for G to the vertices of G_i. The *signature* of a partial solution is a vector of $2w + 1$ integers that would represent this solution in an exact dynamic programming table. In this case, it contains the exact size of the intersections of L, R with each label set and the size of the cut in G_i. To keep things simpler, we will only be concerned with the $2w$ values that represent the intersection. As mentioned, if we can get these right, the algorithm also gets the size of the cut right within a factor of roughly $(1 + 2\epsilon)$.

Consider a partial solution and its signature. Our strategy is to inductively define a mapping that gives for each such signature a collection of $2w$ Addition Trees. The exact results of these trees will be the values of the signature (that is, the sizes of the intersections of the two sets with each label). We will then establish (by induction) that there exists an entry in our algorithm's table whose values follow the same distribution as those Trees, if they are viewed as Approximate Addition Trees. It will follow that for every partial solution there exists, with high probability, an entry with roughly the same values. The converse statement can be shown with essentially the same ideas.

The above can clearly be done for initial nodes, where all values stored are 0 or 1, so our algorithm stores them exactly. The relevant Addition Trees are of height 0. Assume inductively that we have established the correspondence up to a certain height in the clique-width expression. Let us then consider a Rename node i with labels $l_1 \to l_2$ and a child j. Consider a partial solution to G_i. This partial solution has some corresponding partial solution in G_j. By the inductive hypothesis, there exists an entry in $(\boldsymbol{l}, \boldsymbol{r}, c) \in D_j$ corresponding to this solution. In particular, there exist Approximate Addition Trees whose results have the same distribution as $\boldsymbol{l}(l_1)$ and $\boldsymbol{l}(l_2)$ and whose exact values are the same as the corresponding values in the partial solution to G_j. Consider the Tree obtained from these by adding a new root and making the old roots its children. This Tree now follows the same distribution as the value $\boldsymbol{l'}(l_2)$ calculated by our algorithm. Its exact value is the value we would get in the exact signature (since the same was true for the sub-trees). Reasoning in the same way about the other coordinates we have completed the inductive step in this case.

The cases of Join and Union nodes can be handled with similar inductive arguments as above. We can thus establish that for any valid partial solution signature there exists an entry of the table that approximately corresponds to it. It is also not hard to use inductive reasoning to also establish the converse. We now select the entry in the root's table that has maximum c. With high probability, it must correspond to a solution with approximately the same cut size. Retracing the steps of the dynamic programming we can turn this entry into an actual cut.

3.2 Edge Dominating Set

Let us now give a dynamic programming algorithm for Edge Dominating Set. Here, things are a little trickier, because it's not immediately obvious that using subtractions can be avoided. We will use the following equivalent version of the problem: we are looking for a minimum-size set of vertices S such that S is a vertex cover of G and $G[S]$ has a perfect matching. It is not hard to see that this is the same problem (intuitively, S is the set of vertices incident on an edge of the edge dominating set) because an optimal edge dominating set is also a maximal matching.

We define the dynamic programming table D_i for a node i as a subset of $(B \cup \{F\})^w \times B^w$, where F is a special symbol. Let G_i be the corresponding subgraph. Fix a solution to the problem in G, that is a vertex cover S and a matching of its vertices M.

Informally, the intended meaning of an entry $(\boldsymbol{s}, \boldsymbol{c}) \in D_i$ is the following: $\boldsymbol{s}$ tells us how many vertices we have selected in S from each label set (with

$\boldsymbol{s}(l) = F$ meaning we have selected *all* of the set), while $\boldsymbol{c}$ tells us how many of the selected vertices have already been matched. In the calculations below, when we perform arithmetic operations on a value $\boldsymbol{s}(l)$ that is equal to F we substitute it with the size of the set V_l of vertices with label l.

Let us now describe the algorithm. Initial nodes are easy. For Rename and Union nodes we just have to add (component-wise) appropriate entries, taking care to maintain F values if possible. The interesting case is Join nodes.

Let i be a join node with labels $l_1 \leftrightarrow l_2$ and child j. For each entry $(\boldsymbol{s}, \boldsymbol{c}) \in D_j$ do the following. Let V_{l_1}, V_{l_2} be the set of vertices with label l_1, l_2 respectively in G_i. If $\boldsymbol{s}(l_1) \neq F$ and $\boldsymbol{s}(l_2) \neq F$ then ignore this entry because this partial solution is not a vertex cover of G_i. Otherwise, for each $m \in \{0, \ldots, \min(|V_{l_1}|, |V_{l_2}|)\}$ calculate a vector $\boldsymbol{c_m}$ which is identical to $\boldsymbol{c}$ except that $\boldsymbol{c_m}(l_1) = \boldsymbol{c}(l_1) \oplus m$ and $\boldsymbol{c_m}(l_2) = \boldsymbol{c}(l_2) \oplus m$. Informally, we are selecting how many of the join edges will eventually be used in the matching M. If we have $\boldsymbol{c_m}(l_1) > (1+\epsilon)\boldsymbol{s}(l_1)$ or $\boldsymbol{c_m}(l_2) > (1+\epsilon)\boldsymbol{s}(l_2)$ ignore $\boldsymbol{c_m}$. Otherwise, add $(\boldsymbol{s}, \boldsymbol{c_m})$ to D_i.

Once the root's table has been calculated we select the entry $(\boldsymbol{s}, \boldsymbol{c})$ such that $\sum_l \boldsymbol{s}(l)$ is minimized, among entries where $\boldsymbol{c}(l) \geq \boldsymbol{s}(l)/(1+\epsilon)$. Retracing the steps of the dynamic programming we then obtain a vertex cover S. In polynomial time we can calculate a maximum matching in $G[S]$. This is our initial candidate solution. If some vertex of S is unmatched we add one of the edges connecting it to $V \setminus S$. We now have an edge dominating set.

We can again rely on an inductive analysis using Approximate Addition Trees, as in the case of Max Cut, to prove that S has size close to optimal and we have a set of edges M such that almost all vertices of S are incident to a unique edge of M. The only point of difference is when solutions are dropped because we are trying to select too many of the new edges in M. In this case, we leave enough "slack" in our comparisons so that if a solution is erroneously dropped we can extract an Approximate Addition Tree with high error, something that can only happen with very low probability.

References

1. Ben-Zwi, O., Hermelin, D., Lokshtanov, D., Newman, I.: Treewidth governs the complexity of target set selection. Discrete Optimization 8(1), 87–96 (2011)
2. Betzler, N., Bredereck, R., Niedermeier, R., Uhlmann, J.: On bounded-degree vertex deletion parameterized by treewidth. Discrete Applied Mathematics 160(1), 53–60 (2012)
3. Bodlaender, H.L., Hagerup, T.: Parallel algorithms with optimal speedup for bounded treewidth. SIAM J. Comput. 27(6), 1725–1746 (1998)
4. Bodlaender, H.L., Koster, A.M.: Combinatorial optimization on graphs of bounded treewidth. The Computer Journal 51(3), 255–269 (2008)
5. Bonnet, E., Paschos, V.T.: Parameterized (in)approximability of subset problems. CoRR, abs/1310.5576 (2013)
6. Courcelle, B., Kanté, M.M.: Graph operations characterizing rank-width and balanced graph expressions. In: Brandstädt, A., Kratsch, D., Müller, H. (eds.) WG 2007. LNCS, vol. 4769, pp. 66–75. Springer, Heidelberg (2007)

7. Courcelle, B., Makowsky, J.A., Rotics, U.: Linear time solvable optimization problems on graphs of bounded clique-width. Theory of Computing Systems 33(2), 125–150 (2000)
8. Dom, M., Lokshtanov, D., Saurabh, S., Villanger, Y.: Capacitated domination and covering: A parameterized perspective. In: Grohe, M., Niedermeier, R. (eds.) IWPEC 2008. LNCS, vol. 5018, pp. 78–90. Springer, Heidelberg (2008)
9. Ebenlendr, T., Krcál, M., Sgall, J.: Graph balancing: a special case of scheduling unrelated parallel machines. In: Teng, S.-H. (ed.) SODA, pp. 483–490. SIAM (2008)
10. Espelage, W., Gurski, F., Wanke, E.: How to solve NP-hard graph problems on clique-width bounded graphs in polynomial time. In: Brandstädt, A., Van Le, B. (eds.) WG 2001. LNCS, vol. 2204, pp. 117–128. Springer, Heidelberg (2001)
11. Fellows, M.R., Fomin, F.V., Lokshtanov, D., Rosamond, F., Saurabh, S., Szeider, S., Thomassen, C.: On the complexity of some colorful problems parameterized by treewidth. Information and Computation 209(2), 143–153 (2011)
12. Fomin, F.V., Golovach, P.A., Lokshtanov, D., Saurabh, S.: Algorithmic lower bounds for problems parameterized with clique-width. In: Charikar, M. (ed.) SODA, pp. 493–502. SIAM (2010)
13. Golovach, P.A., Thilikos, D.M.: Paths of bounded length and their cuts: Parameterized complexity and algorithms. Discrete Optimization 8(1), 72–86 (2011)
14. Harland, J., Manyem, P.: Proceedings of the Theory of Computing 2008. Proc. Fourteenth Computing: The Australasian Theory Symposium (CATS 2008), Wollongong, NSW, Australia, January 22-25. CRPIT, vol. 77. Australian Computer Society (2008)
15. Lampis, M.: Parameterized maximum path coloring. In: Marx, D., Rossmanith, P. (eds.) IPEC 2011. LNCS, vol. 7112, pp. 232–245. Springer, Heidelberg (2012)
16. Marx, D.: Minimum sum multicoloring on the edges of planar graphs and partial k-trees. In: Persiano, G., Solis-Oba, R. (eds.) WAOA 2004. LNCS, vol. 3351, pp. 9–22. Springer, Heidelberg (2005)
17. Marx, D.: Parameterized complexity and approximation algorithms. Comput. J. 51(1), 60–78 (2008)
18. Meeks, K., Scott, A.: The parameterised complexity of list problems on graphs of bounded treewidth. CoRR, abs/1110.4077 (2011)
19. Papadimitriou, C.H., Yannakakis, M.: Optimization, approximation, and complexity classes. Journal of Computer and System Sciences 43(3), 425–440 (1991)
20. Samer, M., Szeider, S.: Tractable cases of the extended global cardinality constraint. In: Harland and Manyem [14], pp. 67–74
21. Samer, M., Szeider, S.: Constraint satisfaction with bounded treewidth revisited. J. Comput. Syst. Sci. 76(2), 103–114 (2010)
22. Skowron, P., Faliszewski, P.: Approximating the MaxCover problem with bounded frequencies in FPT time. CoRR, abs/1309.4405 (2013)
23. Szeider, S.: Not so easy problems for tree decomposable graphs. CoRR, abs/1107.1177 (2011)

FPTAS for Weighted Fibonacci Gates and Its Applications

Pinyan Lu[1], Menghui Wang[2,⋆], and Chihao Zhang[3,⋆⋆]

[1] Microsoft Research
pinyanl@microsoft.com
[2] University of Wisconsin-Madison
menghui@cs.wisc.edu
[3] Shanghai Jiao Tong University
chihao.zhang@gmail.com

Abstract. Fibonacci gate problems have severed as computation primitives to solve other problems by holographic algorithm [5] and play an important role in the dichotomy of exact counting for Holant and CSP frameworks [6]. We generalize them to weighted cases and allow each vertex function to have different parameters, which is a much boarder family and #P-hard for exactly counting. We design a fully polynomial-time approximation scheme (FPTAS) for this generalization by correlation decay technique. This is the first deterministic FPTAS for approximate counting in the general Holant framework without a degree bound. We also formally introduce holographic reduction in the study of approximate counting and these weighted Fibonacci gate problems serve as computation primitives for approximate counting. Under holographic reduction, we obtain FPTAS for other Holant problems and spin problems. One important application is developing an FPTAS for a large range of ferromagnetic two-state spin systems. This is the first deterministic FPTAS in the ferromagnetic range for two-state spin systems without a degree bound. Besides these algorithms, we also develop several new tools and techniques to establish the correlation decay property, which are applicable in other problems.

1 Introduction

Holant is a refined framework for counting problems [5,6,8], which is more expressive than previous frameworks such as counting constraint satisfaction problems (CSP) in the sense that they can be simulated using Holant instances. In this paper, we consider a generalization called weighted Holant problems. A weighted Holant is an extension of a Holant problem where each edge e is assigned an activity λ_e, and if it is chosen it contributes to the partition function a factor of λ_e.

[⋆] This work was partially performed when the author was an undergraduate student at Shanghai Jiao Tong University.

[⋆⋆] The author is supported by NSF of China (61033002, ANR 61261130589)

[⋆⋆⋆] The full version of this paper is available at http://arxiv.org/abs/1402.4370.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 787–799, 2014.

Given a graph $G(V,E)$, a family of node functions $\mathcal{F} = \{F_v | v \in V\}$, and edge weights $\Lambda = \{\lambda_e | e \in E\}$, the partition function for a weighted Holant instance $\Omega(G,\mathcal{F},\Lambda)$ is the summation of the weights over all configurations $\sigma : E \to \{0,1\}$, specifically the value of $\sum_\sigma \left(\prod_{e\in E} \lambda_e(\sigma(e)) \prod_{v\in V} F_v\left(\sigma|_{E(v)}\right)\right)$. We use Holant$(\mathcal{F},\Lambda)$ to denote the class of Holant problems where all functions are taken from $\mathcal{F}$ and all edge weights are taken from Λ. For example, consider the Perfect Matching problem on G. This problem corresponds to attaching the Exact-One function on every vertex of G — for each 0-1 edge assignment, the product $\prod_{v\in V} F_v(\sigma|_{E(v)})$ evaluates to 1 when the assignment is a perfect matching, and 0 otherwise, thereby summing over all 0-1 edge assignments gives us the number of perfect matchings in G. If we use the At-Most-One function at each vertex, then we can count all matchings, including those that are not perfect.

A symmetric function F can be expressed by $[f_0, f_1, \ldots, f_k]$, where f_i is the value of F on inputs of hamming weight i. The above mentioned Exact-One and At-Most-One functions are both symmetric and can be expressed as $[0,1,0,0,\ldots]$ and $[1,1,0,0,\ldots]$ respectively. A Fibonacci function F is a symmetric function $[f_0, f_1, \ldots, f_k]$, satisfying that $f_i = cf_{i-1} + f_{i-2}$ for some constant c. For example, the parity function $[a,b,a,b,\ldots]$ is a special Fibonacci function with $c = 0$. If there are no edge weights (or equivalently all the weights are equal to 1) and all the node functions are Fibonacci functions with a same parameter c, we have a polynomial time algorithm to compute the partition function exactly [5]. These problems also form the base for a family of holographic algorithms, where other interesting problems can be reduced to the Fibonacci gate problems [5]. Furthermore, this family of functions is interesting not only because of its tractability, but also because it essentially captures almost all tractable Holant problems with all unary functions available [6,8].

If we allow edges to have non-trivial weights or each function to have different parameters in Fibonacci gates, then the exact counting problem becomes #P-hard [6,8]. Nevertheless, it is interesting to study the problem in the approximation setting. In contrast to the exact counting setting, the approximability of Holant problem is much less understood. In this paper, we study approximate counting for weighted Fibonacci gate problems.

Another closely related and well-studied model is spin systems. In this paper, we focus on two-state spin systems. An instance of a spin system is a graph $G(V,E)$. A configuration $\sigma : V \to \{0,1\}$ assigns every vertex one of the two states. The contributions of local interactions between adjacent vertices are quantified by a matrix $A = \begin{bmatrix} A_{0,0} & A_{0,1} \\ A_{1,0} & A_{1,1} \end{bmatrix} = \begin{bmatrix} \beta & 1 \\ 1 & \gamma \end{bmatrix}$, where $\beta, \gamma \geq 0$. The partition function is defined by $Z_A(G) = \sum_{\sigma\in\{0,1\}^V} \prod_{(u,v)\in E} A_{\sigma(u),\sigma(v)}$.

There has been a lot of studies on the approximability of the partition function in terms of parameters β and γ. The problem is exactly solvable in polynomial time if $\beta\gamma = 1$. When $\beta\gamma < 1$, the system is called anti-ferromagnetic and we have a complete understanding of its approximability: there is a uniqueness boundary, above which there is an FPTAS [27,15,23,16] and below which it is NP-hard [24,25,9].

The story is different in ferromagnetic range $\beta\gamma > 1$. Jerrum and Sinclair [13] gave an FPRAS for Ising model ($\beta = \gamma > 1$) based on Markov Chain Monte Carlo (MCMC) method and lately Goldberg et al. extended that to all $\beta\gamma > 1$ plane. However, these algorithms are all randomized. Can we design a deterministic FPTAS for it as that for anti-ferromagnetic range? Indeed, this is an interesting and important question in general and many effort has been made for derandomizing MCMC based algorithms. For instance, there is an FPRAS for counting matchings [12] but FPTAS is only known for graphs of bounded degree [2]. The situation is similar in computing permanent of nonnegative matrix, although an FPRAS is known [14], the current best deterministic algorithm can only approximate the permanent with an exponential large factor [18]. To the best of our knowledge, no deterministic FPTAS was previously known for two-state spin systems in ferromagnetic range. In particular, the correlation decay technique, the main tool to design FPTAS in anti-ferromagnetic range, cannot directly apply.

1.1 Our Results

The main results of this paper are a number of FPTAS's for computing the partition function of different Holant problems and spin systems.

Weighted Fibonacci Gates. We design an FPTAS for weighted Fibonacci gates when the parameters satisfy certain conditions. We have several theorems to cover different ranges. In Theorem 1, we prove that for any fixed choice of other parameters, we can design an FPTAS as long as the edge weights are close enough to 1. This result demonstrates a smooth transition from the unweighted case to weighted ones in terms of approximation.

Another interesting range is that we have an FPTAS for the whole range as long as the Fibonacci parameter c is reasonably large (no less than a constant 1.17) and edge weights are no less than 1 (which means all the edges prefer to be chosen) (Theorem 2). We also allow different nodes to have functions with different parameter c, which contrasts the exact counting setting where a uniform parameter on each node is crucial to have a polynomial time algorithm.

Ferromagnetic Two-State Spin Systems. We design an FPTAS for a large range of ferromagnetic two state spin systems. This is the first deterministic FPTAS in the ferromagnetic range for two-state spin systems without a degree bound. To describe the tractable range, we present a monotonically increasing function $\Gamma : [1, \infty] \to \mathbb{R}$ with $\Gamma(1) = 1$ and $\Gamma(x) \le x$. We have an FPTAS for a ferromagnetic spin system $\begin{bmatrix} \beta & 1 \\ 1 & \gamma \end{bmatrix}$ as long as $\gamma \le \Gamma(\beta)$ or $\beta \le \Gamma(\gamma)$ (Theorem 4). The exact formula of Γ is complicated and we do not spend much effort to optimize it. However, it already enjoys a nice property in that $\lim_{x\to+\infty} \frac{\Gamma(x)}{x} = 1$. This means that although the range does not cover the Ising model ($\beta = \gamma$), it gets relatively close to that in infinity. We also have similar results for two-spin system with external fields.

Other Holant Problems. We can extend our FPTAS to functions $[f_0, f_1, \ldots, f_d]$ with form $f_{i+2} = af_{i+1} + bf_i$ for a range of parameters. This is a much bigger family than Fibonacci gates, since Fibonacci gates corresponds to $b = 1$.

1.2 Our Techniques

Our main approach for designing FPTAS's is the correlation decay technique introduced in [1] and [27]. While the general framework is standard, it is highly non-trivial to design a recursive computational structure and especially to prove the property of exponential correlation decay for a specific problem.

A powerful technique we use is to apply a potential function to amortize the decay rate, which was introduced and used in many circumstances [22,15,23,16,20]. Besides this, to enrich the toolkit, we introduce several new techniques to design and analysis the recursive computational structure. We believe that these techniques can find their applications in other problems.

Working with Dangling Edges. The recursive computational structure for spin problems usually relates a marginal probability of a vertex to that of its neighbors. In Holant problems, we work with assignments and marginal probabilities on edges. Since an edge has two ends, it has two sets of neighbors, which complicates things a lot. In this paper, we instead work on instances with dangling edges, that is, a half edge with neighbors only on one end, and then reduce regular instances to dangling instances. This technique works for any Holant problems and we believe that it is the right structure to work with in the Holant framework. Indeed, the idea has later been successfully used in [17].

Computation Tree with Bounded Degrees. The correlation decay property only directly implies an FPTAS for systems with bounded degrees. One exception is the anti-ferromagnetic two-state spin systems, where a stronger notion of computationally efficient correlation decay is introduced [15]. In this paper, we also establish the computationally efficient correlation decay for systems with unbounded degree, but via a different approach. Thanks to the unique property of Fibonacci functions, we can decompose a node into several nodes with constant degrees. Thus, at each step of our computation tree, we only involve constant many sub-instances even if the degree of the original system is not bounded.

Bounding Range of Variables. After we get a recursion system, the main task is to prove the correlation decay property. This is usually achieved by proving that a certain amortized decay rate, which is a function of several variables, is less than one for any choice of these variables in their domain. Thus if one can prove a smaller domain for each vairable, the analysis becomes easier. Some naive implementation of this idea already appeared in approximate counting of coloring problems [10,20]. In this paper, we develop this idea much further. We divide sub-instances involved in the computation tree into two classes: deep ones for which we can get a much better estimation of their range and shallow ones for which we can compute their value without error. Then we can either compute the exact value or we can safely assume that it is within a smaller domain, which enables us to prove the correlation decay property easier.

Holographic Reduction. We formally introduce holographic reduction in the study of approximate counting. We use weighted Fibonacci gate problems as computational primitives for approximate counting and design holographic algorithms for other problems based on them. In particular, we use the FPTAS for Fibonacci gates to obtain an FPTAS for ferromagnetic two-state spin systems. It is noteworthy that the correlation decay property does not generally hold for ferromagnetic two-state spin systems. So we cannot do a similar argument to get the FPTAS in the spin world directly. Moreover, the idea of holographic reduction can apply to any Holant problems, which extends known counting algorithms (both exact and approximate, both deterministic and randomized) to a broader family of problems. Indeed, the other direction of holographic reduction is also used in our algorithm. We design an exact algorithm for shallow sub-instances of Fibonacci instance by a holographic reduction to the spin world.

1.3 Related Works

Most previous studies of the Holant framework are for exact counting, and a number of dichotomy theorems were proved [8,11,3]. Holographic reduction was introduced by Valiant in holographic algorithms [26,4], which is later also used to prove hardness result of counting problems [5,8,7].

For some special Holant problems such as counting (perfect) matchings, their approximate versions are well studied [2,12,14]. In particular, [2] gave an FPTAS to count matchings but only for graphs with bounded degrees. It is relatively less studied in the general Holant framework in terms of approximate counting except for two recent work: [28] studied general Holant problems but only for planar graph instances with a bounded degree; [21] gives an FPRAS for several Holant problems. Another well-known example is the "sub-graph world" in [13]. It is indeed a weighted Holant problem with Fibonacci functions of $c = 0$, for which an FPRAS was given. In that paper, holographic reduction was also implicitly used, which extends the FPRAS to the Ising model.

Most previous study for FPTAS via correlation decay is on the spin systems. It was extremely successful in the anti-ferromagnetic two-spin system [27,15,23,16]. It is also used in multi-spin systems [10,20].

2 Preliminaries

A weighted Holant instance $\Omega = (G(V,E), \{F_v | v \in V\}, \{\lambda_e | e \in E\})$ is a tuple. $G(V,E)$ is a graph. F_v is a function with arity d_v: $\{0,1\}^{d_v} \to \mathbb{R}^+$, where d_v is the degree of v and $\mathbb{R}^+$ denotes non-negative real numbers. Edge weight λ_e is a mapping $\{0,1\} \to \mathbb{R}^+$. A configuration σ is a mapping $E \to \{0,1\}$ and gives a weight $w_\Omega(\sigma) = \prod_{e\in E} \lambda_e(\sigma(e)) \prod_{v\in V} F_v(\sigma \mid_{E(v)})$, where $E(v)$ denotes the incident edges of v. The counting problem on the instance Ω is to compute the partition function: $Z(\Omega) = \sum_\sigma \left(\prod_{e\in E} \lambda_e(\sigma(e)) \prod_{v\in V} F_v(\sigma \mid_{E(v)})\right)$.

We can represent each function F_v by a vector in $(\mathbb{R}^+)^{2^{d_v}}$, or a tensor in $((\mathbb{R}^+)^2)^{\otimes d_v}$. This is also called a *signature*. A symmetric function F can be expressed by $[f_0, f_1, \ldots, f_k]$, where f_j is the value of F on inputs of hamming weight j. For example, the equality function is $[1, 0, \ldots, 0, 1]$. Edge weight is a unary function, which can be written as $[\lambda_e(0), \lambda_e(1)]$. Since we do not care about global scale factor, we always normalize that $\lambda_e(0) = 1$ and use the notation $\lambda_e = \lambda_e(1)$ as a real number.

A Holant problem is parameterized by a set of functions $\mathcal{F}$ and edge weights Λ. We denote by $\mathrm{Holant}(\mathcal{F}, \Lambda)$ the following computation problem .

Definition 1. *Given a set of functions $\mathcal{F}$ and edge weights Λ, we denote by* $\mathrm{Holant}(\mathcal{F}, \Lambda)$ *the following computation problem.*
Input: *A Holant instance $\Omega = (G(V,E), \{F_v | v \in V\}, \{\lambda_e | e \in E\})$, where $F_v \in \mathcal{F}$ and $\lambda_e \in \Lambda$;*
Output: *The partition function $Z(\Omega)$.*

The weights of configurations also give a distribution over all possible configurations:

$$\mathbb{P}_\Omega(\sigma) = \frac{w_\Omega(\sigma)}{Z(\Omega)} = \frac{1}{Z(\Omega)} \prod_{e \in E} \lambda_e(\sigma(e)) \prod_{v \in V} F_v(\sigma\mid_{E(v)}).$$

This defines the marginal probability of each edge $e_0 \in E$.

$$\mathbb{P}_\Omega(\sigma(e_0) = 0) = \frac{\sum_{\sigma:\sigma(e_0)=0} \left(\prod_{e \in E} \lambda_e(\sigma(e)) \prod_{v \in V} F_v(\sigma\mid_{E(v)})\right)}{Z(\Omega)}.$$

Similarly, we can define the marginal probability of a subset of edges. Let $E_0 \subset E$ and $e_1, e_2, \ldots, e_{|E_0|}$ be an enumeration of the edges in E_0. Then we can define $\sigma(E_0) = \sigma(e_1)\sigma(e_2)\cdots\sigma(e_{|E_0|})$ as a Boolean string of length $|E_0|$. Let $\alpha \in \{0,1\}^{|E_0|}$, we define

$$\mathbb{P}_\Omega(\sigma(E_0) = \alpha) = \frac{\sum_{\sigma:\sigma(e_i)=\alpha_i, i=1,2,\ldots,|E_0|} \left(\prod_{e \in E} \lambda_e(\sigma(e)) \prod_{v \in V} F_v(\sigma\mid_{E(v)})\right)}{Z(\Omega)}.$$

We denote the partial summation as

$$Z(\Omega, \sigma(E_0) = \alpha) = \sum_{\sigma:\sigma(e_i)=\alpha_i} \left(\prod_{e \in E} \lambda_e(\sigma(e)) \prod_{v \in V} F_v(\sigma\mid_{E(v)}) \right).$$

We define a dangling instance Ω^D of $\mathrm{Holant}(\mathcal{F}, \Lambda)$ also as a tuple $(G(V, E \cup D), \{F_v | v \in V\}, \{\lambda_e | e \in E\})$, where $G(V, E \cup D)$ is a graph with dangling edges D. A dangling edge can be viewed as a half edge, with one end attached to a regular vertex in V and the other end dangling (not considered as a vertex). A dangling instance Ω^D is the same as a Holant instance except for these dangling edges. In $G(V, E \cup D)$ each node is assigned a function in $\mathcal{F}$ (we do not consider "dangling" leaf nodes at the end of a dangling edge among these), each regular edge in E is assigned a weight from Λ and we always assume that there is no weight on a dangling edge in this paper. A dangling instance can be also viewed as a regular instance by attaching a vertex with function $[1, 1]$ at the dangling end of each dangling edge. We can define the probability distribution and marginal probabilities just as for regular instance. In particular, we shall use dangling instance Ω^e with single dangling edge e extensively in this paper. For that, we define $R(\Omega^e) = \frac{\mathbb{P}_{\Omega^e}(\sigma(e)=1)}{\mathbb{P}_{\Omega^e}(\sigma(e)=0)}$.

3 Statement of Main Results

A symmetrical function $[f_0, f_1, \ldots, f_d]$ is called a (generalized) Fibonacci function if there exists a constant c such that $f_{i+2} = cf_{i+1}+f_i$, where $i = 0, 1, \cdots, d-2$. We denote this family of function as $\mathcal{F}_c$, the Fibonacci functions with parameter c. We use $\mathcal{F}_c^{p,q}$ to denote a subfamily of $\mathcal{F}_c$ such that $f_{i+1} \geq pf_i$ and $f_{i+1} \leq qf_i$ for all $i = 0, 1, \cdots, d-1$. When the upper bound q is not given, we simply write $\mathcal{F}_c^p$. We use $\mathcal{F}_{c_1,c_2}^{p,q}$ to denote $\bigcup_{c_1 \leq c \leq c_2} \mathcal{F}_c^{p,q}$. We use $\Lambda_{\lambda_1,\lambda_2}$ to denote the set of edge weights λ_e such that $\lambda_1 \leq \lambda_e \leq \lambda_2$.

Here is a list of FPTAS's we get:

Theorem 1. *For any $c > 0$ and $p > 0$, there exists $\lambda_1(p, c) < 1$ and $\lambda_2(p, c) > 1$ such that there is an FPTAS for* $\text{Holant}(\mathcal{F}_c^p, \Lambda_{\lambda_1(p,c),\lambda_2(p,c)})$.

Theorem 2. *Let $p > 0$. Then there is an FPTAS for* $\text{Holant}(\mathcal{F}_{1.17,+\infty}^p, \Lambda_{1,+\infty})$.

Theorem 3. *Let $\lambda > 0$ and $c \geq 2.57$. There is an FPTAS for* $\text{Holant}(\mathcal{F}_c^{c/2,c+2/c}, \Lambda_{\lambda,+\infty})$.

Under a holographic reduction with base $\begin{bmatrix} 1 & t \\ \rho & -\frac{t}{\rho} \end{bmatrix}$, we have the following transformation. Let $\lambda > 0$, $\rho \geq 1$, $t(1-\lambda) > 0$, and $|t| \leq 1$. Let $\beta = \frac{1+\lambda\rho^2}{t(1-\lambda)}$ and $\gamma = \frac{t(1+\lambda\rho^{-2})}{1-\lambda}$. The two spin problem with edge function $\begin{bmatrix} \beta & 1 \\ 1 & \gamma \end{bmatrix}$ and external field μ is equivalent to $\text{Holant}(\mathcal{F}_{\rho-\frac{1}{\rho}}, \Lambda_{\lambda,\lambda})$, where $\mathcal{F}_{\rho-\frac{1}{\rho}}$ is a set of Fibonacci functions with with parameter $c = \rho - \frac{1}{\rho}$ and the one of arity n has form $f_k = \rho^k + \mu t^n(-\rho)^{-k}$. Through this reduction, we can transform Theorem 1-3 to the following FPTAS for ferromagnetic two-state spin system .

Theorem 4. *There is a continuous curve $\Gamma(\beta)$ defined on $[1, +\infty)$ such that (1) $\Gamma(1) = 1$; (2) $1 < \Gamma(\beta) < \beta$ for all $\beta > 1$; and (3) $\lim_{\beta\to+\infty} \frac{\Gamma(\beta)}{\beta} = 1$. There is an FPTAS for the two-state spin system with local interaction matrix $\begin{bmatrix} \beta & 1 \\ 1 & \gamma \end{bmatrix}$ and external field $\mu \leq 1$ if $\beta\gamma > 1$ and $\gamma \leq \Gamma(\beta)$.*

4 Computation Tree Recursion

In the exact polynomial time algorithm for Fibonacci gates without edge weights, one crucial property of a set of Fibonacci functions with a fixed parameter is

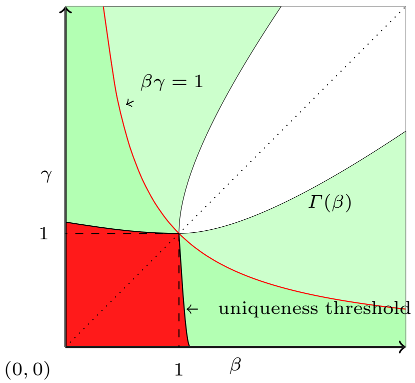


Fig. 1. This figure illustrates the rough shape of $\Gamma(\cdot)$ when there is no external field. It also includes anti-ferromagnetic range. Parameters (β, γ) admit FPTAS in green region and hard to approximate in red region.

that it is closed when two nodes are connected together [5]. This is no longer true if we have non-trivial edge weights or when different Fibonacci function have different parameters. However, we can still use the special property of a Fibonacci function to decompose a vertex, which is the key property for all FPTAS algorithms in our paper.

Let $\Omega = (G(V,E), \{F_v | v \in V\}, \{\lambda_e | e \in E\})$ be an instance of Holant $(\mathcal{F}^{p,q}_{c_1,c_2}, \Lambda_{\lambda_1,\lambda_2})$, $v \in V$ be a vertex of the instance with degree $d_1 + d_2$ $(d_1, d_2 \geq 1)$ and $e_1, e_2, \ldots, e_{d_1+d_2}$ be its incident edges. We can construct a new Holant instance Ω': Ω' is the same as Ω except that v is decomposed into two vertices v', v''. $e_1, e_2, \ldots, e_{d_1}$ are connected to v' and $e_{d_1+1}, e_{d_1+2}, \ldots, e_{d_1+d_2}$ are connected to v''. There is a new edge e connecting v' and v''. If the function on the original v is $[f_0, f_1, \ldots, f_{d_1+d_2}]$, a Fibonacci function with parameter c, then the function on v' is $[f_0, f_1, \ldots, f_{d_1}]$ and the function on v'' is $[1, 0, 1, c \ldots]$, also a Fibonacci function with parameter c. The edge weight on the new edge e is 1. The functions on all other nodes and edge weights on all other edges (except the new e) remain the same as that in Ω. We use the following notation to denote this decomposition operation

$$\Omega' = D(\Omega, v, \{e_1, e_2, \ldots, e_{d_1}\}, \{e_{d_1+1}, e_{d_1+2}, \ldots, e_{d_1+d_2}\}).$$

Using the special property of Fibonacci function, we have the following lemma.

Lemma 1. *Let $\Omega' = D(\Omega, v, E_1, E_2)$. Then $Z(\Omega) = Z(\Omega')$ and for all $e \in E$, $\mathbb{P}_\Omega(\sigma(e) = 0) = \mathbb{P}_{\Omega'}(\sigma(e) = 0)$.*

Let Ω^e be a dangling instance of Holant$(\mathcal{F}^{p}_{c_1,c_2}, \Lambda_{\lambda_1,\lambda_2})$. Let v be the attaching vertex of the dangling edge e and $e_1, e_2, \ldots, e_d$ be other incident edges of v. We compute $R(\Omega^e)$ by smaller instances depending on d. If $d = 0$, then $R(\Omega^e)$ can be computed directly. If $d = 1$, we construct a smaller dangling instance Ω^{e_1} by removing e_0 and v from G and make e_1 be the new dangling edge and remove its weight.

$$R(\Omega^e) = \frac{f_1 + \lambda_{e_1} f_2 R(\Omega^{e_1})}{f_0 + \lambda_{e_1} f_1 R(\Omega^{e_1})}. \tag{1}$$

If $d \geq 2$, we use the above lemma to decompose the vertex v into v' and v'' and let e and e_1 connect to v'' and the remaining edges connect to v'. We use e' to denote the edge between v' and v''. By removing e and v'' from Ω' , we get a dangling instance Ω^{e',e_1} with two dangling edges e', e_1.

$$\begin{aligned}
R(\Omega^e) &= \frac{Z(\Omega^e, \sigma(e) = 1)}{Z(\Omega^e, \sigma(e) = 0)} \\
&= \frac{\lambda_{e_1} Z(\Omega^{e',e_1}, \sigma(e'e_1) = 01) + Z(\Omega^{e',e_1}, \sigma(e'e_1) = 10) + c\lambda_{e_1} Z(\Omega^{e',e_1}, \sigma(e'e_1) = 11)}{Z(\Omega^{e',e_1}, \sigma(e'e_1) = 00) + \lambda_{e_1} Z(\Omega^{e',e_1}, \sigma(e'e_1) = 11)} \\
&= \frac{\lambda_{e_1} \mathbb{P}_{\Omega^{e',e_1}}(\sigma(e'e_1) = 01) + \mathbb{P}_{\Omega^{e',e_1}}(\sigma(e'e_1) = 10) + c\lambda_{e_1} \mathbb{P}_{\Omega^{e',e_1}}(\sigma(e'e_1) = 11)}{\mathbb{P}_{\Omega^{e',e_1}}(\sigma(e'e_1) = 00) + \lambda_{e_1} \mathbb{P}_{\Omega^{e',e_1}}(\sigma(e'e_1) = 11)}.
\end{aligned}$$

In the above recursion, the marginal probability of the original instance is written as that of smaller instances but with two dangling edges. In order to continue the recursive process, we need to convert them into instances with single dangling edge. This can be done by pinning one of the two dangling edges, or just leaving one of the edges free (in which case the dangling end of the free edge can be treated as a regular vertex with signature $[1, 1]$).

We use $\textsc{Pin}_{e,x}(\Omega)$ to denote the new instance obtained by pinning the edge e of the instance Ω to x.

There are many choices in deciding which edge to pin, and to what state the edge is pinned to. Each choice leads to different recursions and consequently have an impact on the following analysis. Here we give an example which is used in the proof of Theorem 1 and Theorem 3. In the proof of Theorem 2, we use a different one.

Set $\Omega^{e'} = \textsc{Pin}_{e_1,0}(\Omega^{e',e_1})$, $\Omega^{e_1} = \textsc{Pin}_{e',0}(\Omega^{e',e_1})$ and $\widetilde{\Omega}^{e_1} = \textsc{Pin}_{e',1}(\Omega^{e',e_1})$. By the definitions, we have $\mathbb{P}_{\Omega^{e'}}(\sigma(e') = 0) = \mathbb{P}_{\Omega^{e',e_1}}(\sigma(e') = 0|\sigma(e_1) = 0)$, $\mathbb{P}_{\Omega^{e_1}}(\sigma(e_1) = 0) = \mathbb{P}_{\Omega^{e',e_1}}(\sigma(e_1) = 0|\sigma(e') = 0)$, and $\mathbb{P}_{\widetilde{\Omega}^{e_1}}(\sigma(e_1) = 0) = \mathbb{P}_{\Omega^{e',e_1}}(\sigma(e_1) = 0|\sigma(e') = 1)$. Given these relation and the fact that

$$\mathbb{P}_{\Omega^{e',e_1}}(\sigma(e'e_1) = 00) + \mathbb{P}_{\Omega^{e',e_1}}(\sigma(e'e_1) = 01) +$$

$$\mathbb{P}_{\Omega^{e',e_1}}(\sigma(e'e_1) = 10) + \mathbb{P}_{\Omega^{e',e_1}}(\sigma(e'e_1) = 11) = 1.$$

We can solve these marginal probabilities and substitute these into the above recursion to get

$$R(\Omega^e) = \frac{\lambda_{e_1} R(\Omega^{e_1}) + R(\Omega^{e'}) + c\lambda_{e_1} R(\Omega^{e'}) R(\widetilde{\Omega}^{e_1})}{1 + \lambda_{e_1} R(\Omega^{e'}) R(\widetilde{\Omega}^{e_1})} \tag{2}$$

If e' and e_1 are in different connected components of Ω^{e',e_1}, then the marginal probability of e_1 is independent of e' and as a result $R(\widetilde{\Omega}^{e_1}) = R(\Omega^{e_1})$. So in this case, we have

$$R(\Omega^e) = \frac{\lambda_{e_1} R(\Omega^{e_1}) + R(\Omega^{e'}) + c\lambda_{e_1} R(\Omega^{e'}) R(\Omega^{e_1})}{1 + \lambda_{e_1} R(\Omega^{e'}) R(\Omega^{e_1})} \tag{3}$$

Starting from an dangling instance Ω^e, we can compute $R(\Omega^e)$ by one of (1), (2) and (3) recursively. We note that if $\Omega^e \in \mathrm{Holant}(\mathcal{F}^{p,q}_{c_1,c_2}, \Lambda_{\lambda_1,\lambda_2})$, the instances involved in the recursion are also in the same family. We define three functions according to these three recursions:

$$h(x) = \frac{f_1 + \lambda_{e_1} f_2 x}{f_0 + \lambda_{e_1} f_1 x}, g(x,y,z) = \frac{\lambda_{e_1} y + x + c\lambda_{e_1} xz}{1 + \lambda_{e_1} xz}, \hat{g}(x,y) = \frac{\lambda_{e_1} y + x + c\lambda_{e_1} xy}{1 + \lambda_{e_1} xy}.$$

By expanding this recursion, we get a computation tree recursion to compute $R(\Omega^e)$. We need one more step to compute the marginal probability of an edge in a regular instance. This can be done similarly and we have the following lemma.

Lemma 2. *If we can ϵ approximate $R(\Omega^e)$ for any dangling instance Ω^e of* $\mathrm{Holant}(\mathcal{F}^{p,q}_{c_1,c_2}, \Lambda_{\lambda_1,\lambda_2})$ *in time poly$(n, \frac{1}{\epsilon})$, we can also ϵ approximate the marginal probability of any edge of a regular instance of* $\mathrm{Holant}(\mathcal{F}^{p,q}_{c_1,c_2}, \Lambda_{\lambda_1,\lambda_2})$ *in time poly$(n, \frac{1}{\epsilon})$.*

5 Algorithm

The procedure from marginal probabilities to partition function is rather standard and we have the following lemma.

Lemma 3. *If for any $\epsilon > 0$ and any Ω^e of* $\mathrm{Holant}\left(\mathcal{F}^{p,q}_{c_1,c_2}, \Lambda_{\lambda_1,\lambda_2}\right)$*, we have a deterministic algorithm to get $\widehat{P}$ in time poly $\left(n, \frac{1}{\epsilon}\right)$ such that $|\widehat{P} - \mathbb{P}_{\Omega^e}(\sigma(e) = 0)| \leq \epsilon$, we have an FPTAS for* $\mathrm{Holant}(\mathcal{F}^{p,q}_{c_1,c_2}, \Lambda_{\lambda_1,\lambda_2})$.

Before we use the computation tree recursion to compute the marginal probability, we need the following lemma to handle shallow instances separately. We denote by $SP(\Omega^e)$ the longest simple path containing e in G.

Lemma 4. *Let L be a constant. We have a polynomial time algorithm to compute $R(\Omega^e)$ for all Ω^e of* $\mathrm{Holant}(\mathcal{F}^{p}_{c_1,c_2}, \Lambda_{\lambda_1,\lambda_2})$ *with $SP(\Omega^e) \leq L$.*

The proof of the above Lemma uses holographic reduction to spin world and makes use of the self-avoiding walk tree [27] for two-state spin systems. The length of the longest simple path is the same as the depth of the self-avoiding walk tree. See the full version for more details.

Now we give out formal procedure to estimate $\mathbb{P}_{\Omega^e}(\sigma(e) = 0)$. Since there is a one to one relation between $\mathbb{P}_{\Omega^e}(\sigma(e) = 0)$ and $R(\Omega^e)$, we can define our recursion on $R(\Omega^e)$, and at the final step we convert $R(\Omega^e)$ back to $\mathbb{P}_{\Omega}(\sigma(e) = 0)$. Let bounds R_1, R_2 and depth L be obtained for the family of dangling instance in the sense that for any dangling instance with $SP(\Omega^e) \geq L$, we have $R(\Omega^e) \in [R_1, R_2]$. Formally, for $t \geq 0$, the quantity $R^t(\Omega^e)$ is recursively defined as follows:

- If $SP(\Omega^e) \leq 2L$, we compute $R^t(\Omega^e) = R(\Omega^e)$ by Lemma 4.
- Else If $t = 0$, let $R^0(\Omega^e) = R_1$.
- Else If$t > 0$, use one of the recursion to get $\tilde{R}^t(\Omega^e) = \hat{g}(R^{t-1}(\Omega^{e'}), R^{t-1}(\Omega^{e_1}))$, $\tilde{R}^t(\Omega^e) = h(R^{t-1}(\Omega^{e_1}))$, or $\tilde{R}^t(\Omega^e) = g(R^{t-1}(\Omega^{e'}), R^{t-1}(\Omega^{e_1}), R^{t-1}(\widetilde{\Omega}^{e_1})$. Return the median of $R_1, \tilde{R}^t(\Omega^e), R_2$: $R^t(\Omega^e) = Med(R_1, \tilde{R}^t(\Omega^e), R_2)$.

There are three possible recursions and we define four amortized decay rates:

$$\alpha_1(x) = \frac{\Phi(x)\left|\frac{\mathrm{d}\,h}{\mathrm{d}\,x}\right|}{\Phi(h(x))}, \quad \alpha_3(x,y) = \frac{\left|\frac{\partial \hat{g}}{\partial x}\right|\Phi(x)}{\Phi(\hat{g}(x,y))}, \qquad \alpha_4(x,y) = \frac{\left|\frac{\partial \hat{g}}{\partial y}\right|\Phi(y)}{\Phi(\hat{g}(x,y))},$$

$$\alpha_2(x,y,z) = \frac{1}{\Phi(g(x,y,z))}\left(\left|\frac{\partial g}{\partial x}\right|\Phi(x) + \left|\frac{\partial g}{\partial y}\right|\Phi(y) + \left|\frac{\partial g}{\partial z}\right|\Phi(z)\right),$$

where $\Phi(\cdot)$ is a potential function.

Definition 2. *We call a function $\Phi : (0, +\infty) \to (0, +\infty)$* nice *if there is some function $f : [1, +\infty) \to (0, +\infty)$ such that for any $c \geq 1$ and $x, y > 0$ with $\frac{x}{c} \leq y \leq cx$, we have $\frac{\Phi(x)}{\Phi(y)} \leq f(c)$.*

Lemma 5. *Let bounds R_1, R_2 and depth L be obtained for dangling instances of* $\mathrm{Holant}(\mathcal{F}^{p,q}_{c_1,c_2}, \Lambda_{\lambda_1,\lambda_2})$ *such that for any dangling instance with $SP(\Omega^e) \geq L$, we have $R(\Omega^e) \in [R_1, R_2]$. If there exist a nice function $\Phi(\cdot)$ and a constant $\alpha < 1$ such that $\alpha_1(x) \leq \alpha$ for all $x \in [R_1, R_2]$, $\alpha_2(x,y,z) \leq \alpha$ for all $x, y, z \in [R_1, R_2]$, $\alpha_3(x,y) \leq \alpha$ for all $x \in [R_1, R_2]$, and $\alpha_4(x,y) \leq \alpha$ for all $y \in [R_1, R_2]$. Then there is an FPTAS for* $\mathrm{Holant}(\mathcal{F}^{p,q}_{c_1,c_2}, \Lambda_{\lambda_1,\lambda_2})$.

To obtain the FPTAS for the Fibonacci gates (Theorem 1-3), we make use of this Lemma 5. In order to apply Lemma 5, we need to establish two things: the bounds R_1, R_2 and the amortized decay rates. There two parts are technically involved and omitted here due to space limitation. The complete proof can be found in the full version of the current paper [19].

References

1. Bandyopadhyay, A., Gamarnik, D.: Counting without sampling: Asymptotics of the log-partition function for certain statistical physics models. Random Structures & Algorithms 33(4), 452–479 (2008)
2. Bayati, M., Gamarnik, D., Katz, D., Nair, C., Tetali, P.: Simple deterministic approximation algorithms for counting matchings. In: Proceedings of STOC, pp. 122–127. ACM (2007)

3. Cai, J.-Y., Guo, H., Williams, T.: A complete dichotomy rises from the capture of vanishing signatures. In: Proceedings of STOC, pp. 635–644. ACM (2013)
4. Cai, J.-Y., Lu, P.: Holographic algorithms: From art to science. Journal of Computer and System Sciences 77(1), 41–61 (2011)
5. Cai, J.-Y., Lu, P., Xia, M.: Holographic algorithms by Fibonacci gates and holographic reductions for hardness. In: Proceedings of FOCS, pp. 644–653 (2008)
6. Cai, J.-Y., Lu, P., Xia, M.: Holant problems and counting CSP. In: Proceedings of STOC, pp. 715–724 (2009)
7. Cai, J.-Y., Lu, P., Xia, M.: Holographic algorithms with matchgates capture precisely tractable planar #CSP. In: Proceedings of FOCS, pp. 427–436. IEEE (2010)
8. Cai, J.-Y., Lu, P., Xia, M.: Computational complexity of Holant problems. SIAM Journal on Computing 40(4), 1101–1132 (2011)
9. Galanis, A., Stefankovic, D., Vigoda, E.: Inapproximability of the partition function for the antiferromagnetic Ising and hard-core models. arXiv preprint arXiv:1203.2226 (2012)
10. Gamarnik, D., Katz, D.: Correlation decay and deterministic FPTAS for counting colorings of a graph. Journal of Discrete Algorithms 12, 29–47 (2012)
11. Huang, S., Lu, P.: A dichotomy for real weighted Holant problems. In: Proceedings of CCC, pp. 96–106. IEEE (2012)
12. Jerrum, M., Sinclair, A.: Approximating the permanent. SIAM Journal on Computing 18(6), 1149–1178 (1989)
13. Jerrum, M., Sinclair, A.: Polynomial-time approximation algorithms for the Ising model. SIAM Journal on Computing 22(5), 1087–1116 (1993)
14. Jerrum, M., Sinclair, A., Vigoda, E.: A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. Journal of the ACM 51(4), 671–697 (2004)
15. Li, L., Lu, P., Yin, Y.: Approximate counting via correlation decay in spin systems. In: Proceedings of SODA, pp. 922–940. SIAM (2012)
16. Li, L., Lu, P., Yin, Y.: Correlation decay up to uniqueness in spin systems. In: Proceedings of SODA, pp. 67–84 (2013)
17. Lin, C., Liu, J., Lu, P.: A simple FPTAS for counting edge covers. In: Proceedings of SODA, pp. 341–348 (2014)
18. Linial, N., Samorodnitsky, A., Wigderson, A.: A deterministic strongly polynomial algorithm for matrix scaling and approximate permanents. Combinatorica 20(4), 545–568 (2000)
19. Lu, P., Wang, M., Zhang, C.: FPTAS for weighted Fibonacci gates and its applications. arXiv preprint arXiv:1402.4370 (2014)
20. Lu, P., Yin, Y.: Improved FPTAS for multi-spin systems. In: Raghavendra, P., Raskhodnikova, S., Jansen, K., Rolim, J.D.P. (eds.) RANDOM 2013 and APPROX 2013. LNCS, vol. 8096, pp. 639–654. Springer, Heidelberg (2013)
21. McQuillan, C.: Approximating Holant problems by winding. arXiv preprint arXiv:1301.2880 (2013)
22. Restrepo, R., Shin, J., Tetali, P., Vigoda, E., Yang, L.: Improved mixing condition on the grid for counting and sampling independent sets. Probability Theory and Related Fields 156(1-2), 75–99 (2013)
23. Sinclair, A., Srivastava, P., Thurley, M.: Approximation algorithms for two-state anti-ferromagnetic spin systems on bounded degree graphs. In: Proceedings of SODA, pp. 941–953. SIAM (2012)
24. Sly, A.: Computational transition at the uniqueness threshold. In: Proceedings of FOCS, pp. 287–296. IEEE (2010)

25. Sly, A., Sun, N.: The computational hardness of counting in two-spin models on d-regular graphs. In: Proceedings of FOCS, pp. 361–369. IEEE (2012)
26. Valiant, L.G.: Holographic algorithms. SIAM Journal on Computing 37(5), 1565–1594 (2008)
27. Weitz, D.: Counting independent sets up to the tree threshold. In: Proceedings of STOC, pp. 140–149. ACM (2006)
28. Yin, Y., Zhang, C.: Approximate counting via correlation decay on planar graphs. In: Proceedings of SODA, pp. 47–66. SIAM (2013)

Parameterized Algorithms to Preserve Connectivity*

Manu Basavaraju[1], Fedor V. Fomin[1], Petr Golovach[1], Pranabendu Misra[2],
M. S. Ramanujan[1], and Saket Saurabh[1,2]

[1] University of Bergen
{Manu.Basavaraju,fomin,Petr.Golovach,Ramanujan.Sridharan}@ii.uib.no
[2] The Institute of Mathematical Sciences, Chennai, India
{pranabendu,saket}@imsc.res.in

Abstract. We study the following family of connectivity problems. For a given λ-edge connected (multi) graph $G = (V, E)$, a set of links L such that $G + L = (V, E \cup L)$ is $(\lambda + 1)$-edge connected, and a positive integer k, the questions are

Augmentation Problem: whether G can be augmented to a $(\lambda + 1)$-edge connected graph by adding at most k links from L; or

Deletion Problem: whether it is possible to preserve $(\lambda + 1)$-edge connectivity of graph $G + L$ after deleting at least k links from L.

We obtain the following results.

- An $9^k|V|^{O(1)}$ time algorithm for a weighted version of the augmentation problem. This improves over the previous best bound of $2^{O(k \log k)}|V|^{O(1)}$ given by Marx and Vegh [*ICALP 2013*]. Let us remark that even for $\lambda = 1$, the best known algorithm so far due to Nagamochi [*DAM 2003*] runs in time $2^{O(k \log k)}|V|^{O(1)}$.
- An $2^{O(k)}|V|^{O(1)}$ algorithm for the deletion problem thus establishing that the problem is fixed-parameter tractable (FPT). Moreover, we show that the problem admits a kernel with $12k$ vertices and $3k$ links when the graph G has odd-connectivity and a kernel with $O(k^2)$ vertices and $O(k^2)$ links when G has even-connectivity.

Our results are based on a novel connection between augmenting sets and the Steiner Tree problem in an appropriately defined auxiliary graph.

1 Introduction

In connectivity augmentation problems, the input is a (multi) graph and the objective is to increase edge or vertex connectivity by adding the minimum number (weight) of additional edges, called links. This problem was first studied

* The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013)/ERC Grant Agreement n. 267959, from Bergen Research Foundation and University of Bergen under project BEATING HARDNESS BY PREPROCESSING n.806572, and from PARAPPROX, ERC starting grant n. 306992.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 800–811, 2014.

by Eswaran and Tarjan [4] who showed that increasing the edge connectivity of a given graph to 2 by adding minimum number of links (also called an augmenting set) is polynomial time solvable. Subsequent work in [11,5] showed that this problem is also polynomial time solvable for any given target value of edge connectivity to be achieved. However, if the set of links is restricted, that is, there are pairs of vertices in the graph which do not constitute a link, or if the links have (non-identical) weights on them, then the problem of computing the minimum size (or weight) augmenting set is NP-complete [4].

It is interesting to note that the vertex version of the problem is substantially more difficult even when the set of links which can be added is unrestricted, and is only known to be polynomial time for the special case when the connectivity of the graph is required to increased by 1, with the general case still open [10]. Furthermore, the special case of edge connectivity augmentation where the objective is to increase the edge connectivity of a graph by 1 by adding edges from a *restricted* set is also known to be NP-complete [6]. In this paper, we will focus on the parameterized complexity aspects of the problems, Weighted Edge Connectivity Augmentation By One (w-Aug-One) and Deletion with λ connectivity.

The first problem is defined as follows.

w-Aug-One **Parameter:** k
Input: Graph $G = (V, E)$ which is λ-edge connected, set of links $\mathcal{L}$, integer k, weight function w on $\mathcal{L}$, $p \in \mathbb{R}$.
Question: Is there a link set F such that $w(F) \leq p$, $|F| \leq k$ and $(V, E \cup F)$ is $\lambda + 1$-edge connected?

Our second problem is

Deletion with λ connectivity **Parameter:** k.
Input: $(G, \mathcal{L}, k)$ where G is a λ-edge connected, $\mathcal{L}$ is a set of edges, called links, $G + \mathcal{L}$ is $(\lambda + 1)$-edge connected, and k a positive integer.
Question: Is there a set of k links in $\mathcal{L}$ whose deletion from $G + \mathcal{L}$ maintains $(\lambda + 1)$-edge connectivity?

The first parameterized algorithm for the connectivity augmentation problem was considered by Nagamochi [9], who gave an $2^{O(k \log k)}|V|^{O(1)}$ algorithm for w-Aug-One in the case when the weights on the links are identical and λ is odd. Guo and Uhlmann [7] gave a kernel with $O(k^2)$ vertices and links for the same case. Most recently, Marx and Vegh [8] studied the problem in its full generality and gave a kernel with $O(k)$ vertices, $O(k^3)$ links and weights of $(k^6 \log k)$ bit integers. This as well as other previous results lead to an algorithm with running time $2^{O(k \log k)}|V|^{O(1)}$, even for unweighted version of the problem. In this paper, we obtain the first single exponential algorithm for this problem. In particular, we have the following theorem.

Theorem 1. w-Aug-One *is solvable in time* $9^k|V|^{O(1)}$.

Intuitively, Deletion with λ connectivity seems to be harder than augmentation problems. For example, if we want to augment a tree T to a 2-edge connected graph, the number of links we have to add is at least half the number of leaves of T. By handling degree two vertices in appropriate way, this almost immediately brings us to a polynomial kernel. But when we want to delete links from $\mathcal{L}$ in order to keep a minimum 2-edge connected supergraph of T, there is no lower bounds on the number of deleted links, and this makes the problem much harder. Similar difference can be observed for graphs of larger connectivity. Not surprisingly, to solve Deletion with λ connectivity, we need completely different ideas. However, we prove the following theorem.

Theorem 2. Deletion with λ connectivity *is solvable in time* $2^{O(k)}|V|^{O(1)}$.

In other words, we establish that Deletion with λ connectivity is FPT. The next natural question following the establishment of parameterized tractability of a problem is if this problem admits a "polynomial kernel". That is, in polynomial time can we construct an equivalent instance of size and parameter bounded by a polynomial function in k? We answer this question affirmatively.

Theorem 3. Deletion with λ connectivity *admits a kernel with* $12k$ *vertices and* $3k$ *links for odd* λ *and a kernel with* $O(k^2)$ *vertices and* $O(k^2)$ *links for even* λ*, where* λ *is the connectivity of the input graph* G.

Our Approach. The result of Dinits et al. [2] allows us to assume without loss of generality that for the w-Aug-One problem, $\lambda = 1$ and G is a tree or $\lambda = 2$ and G is a cactus graph. Similarly, we can assume for the Deletion with λ connectivity problem that $\lambda = 1$ and G is a tree (we refer to this problem as Tree Augmentation) or $\lambda = 2$ and G is a cactus graph (we refer to this problem as Cactus Augmentation).

We then define an auxiliary graph which we call a *link-terminal intersection graph*, corresponding to a given instance of the problem and show that augmenting sets for the given instance exactly correspond to Steiner trees in the link-intersection graph for a specified set of terminals. This structural lemma forms the backbone of both our FPT algorithms. In the case of the w-Aug-One problem, the set of vertices in the input graph corresponding to leaves or degree-2 vertices is chosen as the set of terminals for the auxiliary Steiner Tree instance. However, the Steiner Tree problem is not FPT when parameterized by the solution size. Thus, for our purposes we can only use those parameterization for which Steiner Tree is known to be FPT. In our context we use the following known results about Steiner Tree: an algorithm with running time $2^{|T|}|V(G)|^{O(1)}$ [3], where T is the set of input terminals and an algorithm with running time $2^{O(t)}|V(G)|^{O(1)}$ [1], where t is the treewidth of the input graph. In the w-Aug-One problem, the number of links in any augmenting set is at least half the number of leaves or degree-2 vertices in the input graph. We thus obtain a Steiner Tree instance where the number of terminals is bounded linearly in the parameter, following which we can invoke known results to give an algorithm. However, the situation is much more complicated in the case of the Deletion with λ connectivity problem. Here, we design preprocessing rules based on several non-trivial structural lemmas regarding No instances of

Table 1. The table gives a summary of our results, where the $O^*()$ notation hides polynomial factors. The constants hidden by the $O(.)$ notation in the kernel sizes are independent of λ.

Problem Name	Algorithm	Kernel
w-Aug-One	$O^*(9^k)$	$O(k)$ vertices and $O(k^3)$ links [8]
Deletion with λ connectivity (odd λ)	$O^*(8^k)$	$12k$ vertices and $3k$ links
Deletion with λ connectivity (even λ)	$O^*(2^{O(k)})$	$O(k^2)$ vertices and $O(k^2)$ links

the problem. We then show that applying these preprocessing rules exhaustively results in an equivalent instance where the lengths of the cycles in the graph $G \setminus \mathcal{L}$ is bounded linearly in the parameter. In our final step, we show that this property, along with those proved beforehand, translates to a linear bound on the tree width of the link-terminal intersection graph, where it suffices to compute an optimum Steiner tree. Finally, we combine these steps along with a known algorithm for Steiner Tree on bounded tree width graphs to obtain our algorithm for Deletion with λ connectivity. The techniques we use for the kernels for Deletion with λ connectivity depend on the parity of the connectivity λ. In the case when λ is odd, we construct an alternate auxiliary graph with the set of links as the vertex set and prove two properties regarding this graph. The first is that this graph has a constant degeneracy and the second is that a vertex in this graph corresponds to an augmenting set in the input instance. Combining these two properties, we are able to conclude that if the size of the instance exceeds a certain constant factor of the parameter, then the input instance is a Yes instance, thus obtaining a linear kernel. For the case when λ is even, we introduce an additional preprocessing rule to the ones already used and show that these rules together suffice to bound the size of a given instance quadratically in the parameter.

2 Preliminaries

Definition 1. *Given a graph $G = (V, E)$, a spanning subgraph H of G is called a* **cactus** *if it is 2-edge connected and every edge belongs to exactly one cycle. Equivalently, H can be written as the union of a set $\mathcal{C}=\{C_1, \ldots, C_\ell\}$ where each C_i is a cycle, the graph induced on their union is connected and every edge in this graph belongs to exactly one cycle. Note that the subgraph H can have multi-edges. A cycle of length 2 in a cactus is called a* ***2-circuit****.*

Definition 2. *A set Z of t edges in a graph is called a* ***t-cut*** *if deleting the edges of Z disconnects the graph and for every subset $Z' \subset Z$, the graph remains connected after deleting the edges in Z'.*

The following observation is a consequence of the definition of a t-cut and will be used crucially in several of our proofs.

Observation 4. *Every 1-cut of a tree is a bridge in a tree. The 2-cuts in a cactus are contained in a cycle, i.e., if $\{e, f\}$ is a 2-cut, then both the edges e and f belong to the same cycle C of the cactus $\mathcal{C}$.*

Definition 3. *We say that a link* covers *a bridge in a tree (2-cut in a cactus) if the end points of the link are in distinct connected components obtained by deleting the bridge (respectively 2-cut). Note that since every 2-cut is contained with in a cycle, the vertices of the cycle will be in two distinct connected components.*

Tree Decompositions. A *tree decomposition* of a graph $G = (V, E)$ is a pair (M, β) where M is a rooted tree and $\beta : V(M) \to 2^V$, such that :

1. $\bigcup_{t \in V(M)} \beta(t) = V$.
2. For each edge $(u, v) \in E$, there is a $t \in V(M)$ such that both u and v belong to $\beta(t)$.
3. For each $v \in V$, the nodes in the set $\{t \in V(M) \mid v \in \beta(t)\}$ form a connected subtree of M.

Let (M, β) be a tree decomposition of a graph G. The *width* of (M, β) is $max\{|\beta(t)| - 1 \mid t \in V(M)\}$.

For ease of description, we reformulate the Deletion with λ connectivity problem also as an augmentation problem as follows. Given an instance $(G, \mathcal{L}, k)$ of Deletion with λ connectivity where G is λ edge-connected, the objective is to find a set of $|\mathcal{L}| - k$ links to add to the graph G such that its edge connectivity is $\lambda + 1$. We refer to the reformulated version as the $m - k$-Augmentation by One problem. Here $m = |\mathcal{L}|$. When G is a tree or cactus, we refer to it as the $m - k$-Tree Augmentation or $m - k$-Cactus Augmentation problem respectively. Due to space constraints, missing proofs have been moved to the appended full version of the paper.

3 Cactus Augmentation and Steiner Trees

In the first subsection, we introduce link-intersection graphs and prove certain properties of these graphs. In the following subsection, we prove our main structural result relating Steiner trees in these graphs and solutions for our problem. This result will be a vital part of our algorithms.

3.1 Link-Intersection Graphs

Definition 4. Projection of a Link onto a Cactus. *Consider a cactus $\mathcal{C}$ and a link (u, v). Let $a_1, \dots, a_r$ be the cut vertices of the cactus (other than u and v) which lie on every u-v path in the cactus and let $C_1, \dots, C_{r+1}$ be the cycles which contain the segments u-a_1, a_1-a_2,..., a_r-v respectively of this path. Then, we say that the pair $< u, a_1 >$ is the* projection *of the link (u, v) onto the cycle C_1, the pair $< a_r, v >$ is the projection of the link (u, v) onto the cycle C_{r+1} and* *the pair $< a_i, a_{i+1} >$ is the projection of the link onto the cycle C_{i+1}. Note that if u and v lie in the same cycle of the cactus, then the projection is simply the pair $< u, v >$. If the pair $< x, y >$ is a projection of a link onto a cycle of the cactus, then we say that the link is* ***projectively incident*** *to x,y and this cycle and a link (x, y) is said to be* **properly incident** *on x and y. We say that the projection of a link e on to the cycle C is* non-trivial *if it is projectively incident to exactly two vertices of the cycle.*

Definition 5. Crossing Pairs. *Consider distinct vertices x_1, x_2, y_1, y_2 which lie on the same cycle in a cactus. We say that the pair $< x_1, y_1 >$* crosses *the pair $< x_2, y_2 >$ if any path from x_2 to y_2 contains exactly one of vertices from $\{x_1, y_1\}$ as an internal vertex. Observe that the crossing relation is symmetric and hence we say that the pairs $< x_1, y_1 >$ and $< x_2, y_2 >$ are* **crossing**. *Let e be a link that is incident (possibly projectively incident) on the vertices x_1 and y_1 of the cycle, then we say the link $e = (x_1, y_1)$ crosses the pair $< x_2, y_2 >$. If both the pairs correspond to links $e = (x_1, y_1)$ and $f = (x_2, y_2)$, then we say that the links e and f cross. A set L of links is said to be* **laminar** *if it does not contain a pair of links which cross.*

We now give an equivalent definition (to Definition 3) of covering 2-cuts using the notion of crossing pairs. For ease of description, we will be working with this definition from now on.

Definition 6. Covering a 2-cut. *Let $e = (a_1, b_1)$ and $f = (a_2, b_2)$ be the edges of a cycle C such that the vertices appear as a_1, b_1, a_2, b_2 in a fixed ordering of the vertices of the cycle. Note that b_1 can be the same as a_2 and a_1 can be the same as b_2. We say that the link $d = (x, y)$ covers the 2-cut $\{e, f\}$ if it satisfies one of the following properties: (i) $\{x, y\} = \{a_1, b_1\}$, (ii) $\{x, y\} = \{a_2, b_2\}$, (iii) the link d crosses one of the pairs $< a_1, a_2 >$ or $< b_1, b_2 >$.*

Observation 5. *Let $\mathcal{C}$ be a cactus and let e be a link that covers the 2-cut formed by consecutive edges incident to a vertex v on a cycle of the cactus. Then, the link e is projectively incident to the vertex v.*

Definition 7. Intersection Graph with Respect to a Cactus. *Given a graph G, a cactus $\mathcal{C}$ and the set $\mathcal{L}$ of links, we define the intersection graph $H_{\mathcal{C}}$ of the cactus as follows. We have a vertex for every link and the vertices corresponding to 2 links e_1 and e_2 are adjacent if*

- *there is a vertex in the cactus to which both these links are projectively incident or*
- *there are crossing pairs $< x_1, y_1 >$ and $< x_2, y_2 >$ which are projections of e_1 and e_2 respectively on the same cycle of the cactus.*

We define the **link-terminal intersection graph** *$I_{\mathcal{C},\mathcal{L}}$ as the graph obtained from $H_{\mathcal{C}}$ by adding to it the vertex set of G (these new vertices are called* **terminals** *and the other vertices,* **link-vertices***) and making each vertex adjacent to the vertices of $H_{\mathcal{C}}$ which correspond to the links projectively incident to this vertex.*

3.2 Relating Augmenting Sets to Steiner Trees

In the following lemma, we show the equivalence between solving our problem and the Steiner Tree problem on the link-intersection graphs.

Lemma 1. *Given a cactus $\mathcal{C}$ and a set $\mathcal{L}$ of links, consider the link-terminal intersection graph $I_{\mathcal{C},\mathcal{L}}$ of the cactus. Let T be the terminals in $I_{\mathcal{C},\mathcal{L}}$ and let X be the vertices in $I_{\mathcal{C},\mathcal{L}}$ which correspond to degree-2 vertices in the cactus. Then, a set S of links is an augmenting set for this cactus if and only if $I_{\mathcal{C},\mathcal{L}}[S \cup X]$ is connected.*

Proof. (proof sketch) Consider the forward direction and let S be a set of links which is an augmenting set for this cactus. We prove that every pair of link-vertices in $I_{\mathcal{C},\mathcal{L}}$ which correspond to a pair of links with a non-trivial projection in a particular cycle C appear in the same connected component of $I_{\mathcal{C},\mathcal{L}}[S \cup T]$. Observe that every vertex of X lies in a unique cycle and furthermore, there must be a link in S properly incident on it. Since such a link must have a non-trivial projection in this cycle, we conclude that every vertex in X which lie in the same cycle of $\mathcal{C}$ are in the same connected component of $I_{\mathcal{C},\mathcal{L}}[S \cup T]$.

Now, since every cut vertex belongs to all cycles it is incident on, the link-vertices corresponding to links projectively incident on these cycles and the terminals corresponding to the vertices of these cycles belong to the same component in $I_{\mathcal{C},\mathcal{L}}[S \cup T]$. Since connectivity is an equivalence relation we can infer that all the link-vertices of S which are incident on all cycles and all the terminals are in the same component in $I_{\mathcal{C},\mathcal{L}}[S \cup T]$, thus proving the forward direction.

For the proof of the converse direction, we first show that if $I_{\mathcal{C},\mathcal{L}}[S \cup X]$ is connected, then $I_{\mathcal{C},\mathcal{L}}[S \cup T]$ is also connected. The proof for this claim follows from the fact that for any cut-vertex v in $T \setminus X$ any path between 2 degree-2 vertices in the subcactus on either side of this cut-vertex in $I_{\mathcal{C},\mathcal{L}}[S \cup T]$ should pass through a link-vertex incident on the terminal corresponding to this cut vertex which proves the claim.

Now, suppose that S is not an augmenting set for the cactus and let e_1, e_2 be a 2-cut in a cycle of the cactus which is not covered by S. Then the link-vertices from two components formed by this two cut are never adjacent in $I_{\mathcal{C},\mathcal{L}}$ by definition and they are disjoint. This will imply that the terminals will appear in more than one component in $I_{\mathcal{C},\mathcal{L}}$. □

4 Single Exponential Algorithms for $m-k$-Augmentation

The main objective of this section is to prove the following lemma.

Lemma 2. *There is an algorithm for $m-k$* Cactus Augmentation *running in time $O^*(2^{O(k)})$.*

For this, we give a set of preprocessing rules using which we will eventually obtain a bound on the treewidth of the link-intersection graph.

4.1 Bounding the Cycle Lengths and Tree-Width of the Link-Intersection Graph

We give a set of polynomial time preprocessing rules, whose exhaustive application enables us to prove the following lemma. Due to space constraints, we give a full description of the reduction rules in the full version.

Lemma 3. *There is a polynomial time algorithm that takes as an input an instance* $(\mathcal{C}, \mathcal{L}, k)$ *and either concludes that the given instance is a* Yes *instance or returns an equivalent instance* $(\mathcal{C}', \mathcal{L}', k')$ *such that every cycle in* $\mathcal{C}$ *has length at most* $67k$ *and has at most* $71k$ *links projectively incident on it.*

We now bound the treewidth of the link-intersection graph constructed from instances returned by the above lemma.

Lemma 4. *Consider an instance* $(\mathcal{C}, \mathcal{L}, k)$ *returned by Lemma 3. Then, there is an algorithm that in polynomial time either returns a tree-decomposition of* $I_{\mathcal{C},\mathcal{L}}$ *of width* $O(k)$ *or correctly concludes that the given instance is a* Yes *instance.*

Proof. Let $I_{\mathcal{C},\mathcal{L}} = (V, E)$ and consider the rooted block tree of the given cactus and let M be the tree whose vertices correspond to the cycles of the cactus and 2 vertices are adjacent if the block corresponding to one is the parent of the block corresponding to the other in the rooted block tree. Let V_M be the vertices of M and for every $v \in V_M$ let C_v be the corresponding cycle in the cactus. We now define the bags $\beta : V_M \to V$ as follows.

$$\beta(v) = \{u \in V | u \in C_v \text{ or } u \in \mathcal{L} \text{ and } u \text{ is projectively incident on } C_v\}$$

We claim that (M, β) is indeed a tree-decomposition of $I_{\mathcal{C},\mathcal{L}}$. Clearly, every link and every terminal appear in some bag. Furthermore, for every terminal, there is a bag which contains this terminal and all links which are projectively incident on this terminal. Now, consider an edge in $I_{\mathcal{C},\mathcal{L}}$ between 2 links. They are adjacent because they are both projectively incident on some vertex in which case there is a bag which contains both these links or they cross, which implies that they are both projectively incident on some cycle, in which case the corresponding bag contains both these vertices. Hence we conclude that every edge of $I_{\mathcal{C},\mathcal{L}}$ is contained in some bag. Finally, observe that any terminal appears only in those bags whose corresponding cycles contain this terminal and they are by definition connected. Similarly, a link only appears in those bags whose corresponding cycles have a non-trivial projection of this link, which by definition form a path in the tree M. This concludes the proof that (M, β) is indeed a tree decomposition of $I_{\mathcal{C},\mathcal{L}}$.

By Lemma 3, we have that every cycle in the cactus has length at most $67k$ and that the number of links projectively incident on a cycle is at most $71k$. Therefore, every cycle has at most $71k$ links incident on it implying that every bag in the tree-decomposition has size at most $138k$, thus concluding the proof of the lemma. □

We are now ready to prove Lemma 2 by giving an algorithm for $m-k$ Cactus Augmentation as follows. We first apply Reduction Rules 1-5 on the given instance and then use the algorithm of Lemma 4 to either conclude that the given instance is a Yes instance or obtain a tree decomposition for the link-intersection graph, whose width is bounded linearly in the parameter. By Lemma 1, it suffices to compute a minimum Steiner tree in $I_{\mathcal{C},\mathcal{L}}$ with the set of degree-2 vertices of the cactus as the terminals. We then solve the problem by using the algorithm for Steiner Tree given by [1] running in time $O^*(2^{O(tw)})$, which translates to an $O^*(2^{O(k)})$ algorithm for $m-k$ Cactus Augmentation.

Dinits et al. [2] showed that the general problem of $m-k$-augmentation by one reduces in polynomial time either to $m-k$ Cactus Augmentation or to $m-k$ Tree Augmentation without an increase in the parameter. Therefore, Theorem 2 follows from Lemma 2 and Lemma 6 (proved in the next section).

5 Polynomial Kernels for Unweighted $m-k$-Augmentation

5.1 A Linear Kernel for (m-k)-tree-Augmentation Problem

Definition 8. *In a rooted tree, the depth of any node in a tree is the length of the shortest path from the root the node. The depth of the root is* 0 *and the depth of any other node is one greater than the depth of its parent. The depth of any edge $f = ab \in E(T)$ is defined as $depth(f) = \min(depth(a), depth(b))$. The depth of a link $e = xy \in \mathcal{L}$ is the depth of the least common ancestor of x and y.*

Definition 9. *A graph is called* 2*-degenerate if all of its induced subgraphs have a vertex of degree at most 2. An ordering π of the vertices of a graph H is called a 2-degenerate ordering if for any vertex $v = \pi(i)$, the degree of v in $H[V_i]$ is at most 2.*

Lemma 5. *Let (G, T, k) be the given instance of the problem, with the additional property that every cut is covered by at least two links from $\mathcal{L}$. Then there exists an augmenting set for the tree T using at most $\lfloor 2|\mathcal{L}|/3 \rfloor$ links.*

Proof. We construct an auxiliary graph H whose vertex set is the set of links. And the depth of a vertex in H is the same as the depth of the corresponding link. The edge set is defined as follows. Let σ_E be an ordering of the edges of the tree in the non-increasing order of their depths. We process the edges according to this ordering. For an edge (a, b) in the tree, let V_{ab} be the set of vertices corresponding to the links covering the edge (a, b). If there exists an edge between a pair of vertices in V_{ab}, we do nothing. Otherwise V_{ab} is an independent set at this stage. Let ψ_{ab} be an ordering of V_{ab} in increasing order of their depths. We put an edge between the first and the second vertex of the list, i.e., $(\psi_{ab}(1), \psi_{ab}(2)) \in E(H)$. We do this for all the edges of the tree, processing each according to the list σ_E and obtain the graph H.

Note that for every edge(cut) in T, there exists an edge in H. Therefore it is easy to see that a vertex cover of H corresponds to a set of links that covers all the edges(cuts) of T. Thus if there exists an independent set of size k in H, then there exists an augmenting set of size $|\mathcal{L}| - k$ for T. We complete the proof by showing that the graph H we constructed is 2-degenerate, which implies the presence of an independent size of size at least $\lceil |\mathcal{L}|/3 \rceil$ in H which in turn implies an augmenting set of size at most $|\mathcal{L}| - \lceil |\mathcal{L}|/3 \rceil = \lfloor 2|\mathcal{L}|/3 \rfloor$ for T.

Claim. The graph H is 2-degenerate.

Proof. Let ϕ_H be the ordering of the vertices of H in the increasing order of their depths. We claim that this ordering is a 2-degenerate ordering. Suppose not and let $H[V_i]$ (The graph induced by the first i vertices in the ordering ϕ_H) be an induced subgraph which has minimum degree greater than 2. Let e the link corresponding to the vertex $\phi_H(i)$. Let the path covered by this link $e = v$ in T be $P = \{x, v_1, v_2, \ldots, v_r, y\}$. Let z be the least common ancestor of x and y. Let some three of the links corresponding to the neighbours of $\phi_H(i)$ in $H[V_i]$ be f_1, f_2, f_3. Note that all these links have depth at most as that of e since they appeared before e in the ordering ϕ_H. Since all three links are adjacent to e in H, they cover an edge in the path P. Combining the observations in the previous 2 sentences, we conclude that the least common ancestor of the endpoints of any of these links is either z or is an ancestor of z and all three of them cover an edge incident on z. Since there are only two edges of the path P incident on z, at least two of the links in f_1, f_2, f_3 have to cover one of these edges. Without loss of generality assume that f_1 and f_2 cover the edge (z, v_i) where v_i is in the subpath P' of P between x and z. Since there are edges (e, f_1) and (e, f_2) in H, there are two distinct edges, say (a, b) and (c, d) in the path P' that are responsible for the edges (e, f_1) and (e, f_2) in H respectively. Let the depth of (a, b) be greater than that of (c, d). Then when we were building the graph H, we would have added the edge (e, f_1) to H when processing the edge (a, b). Observe that the edge (c, d) is covered by both e and f_1. Therefore, the edge (c, d) could not have been responsible for the addition of the edge (e, f_2) in H, a contradiction. Hence we conclude that the graph H is 2-degenerate. □

Given Lemma 5, if $|\mathcal{L}| \geq 3k$, then the given instance is a YES instance. Therefore, we may assume that $|\mathcal{L}| \leq 3k$. Observe that the number of vertices of the tree with a link incident on it is bounded by $2|\mathcal{L}|$. This set includes all the leaves of the tree. Furthermore, the implies that the number of vertices of the tree with degree at least 3 is also bounded by $2|\mathcal{L}|$. By short-circuiting every degree-2 path in the tree with no links incident on the internal vertices (adding an edge between the endpoints of the path and removing the internal vertices of the path), we may assume that every degree-2 vertex in the tree has a link incident on it. Therefore, we have that $n \leq 4|\mathcal{L}| = 12k$. To conclude the proof of the kernel, we need to convert any given instance into one with the property assumed by the statement of Lemma 5. Observe that if a single link covers a bridge in the tree, then this link must be part of every augmenting set and hence we get an equivalent instance when we contract all edges covered by any such link and removing this link, with no change in the number of links to be excluded, k.

Finally, we infer the following lemma from the bound of $3k$ on the link set.

Lemma 6. *There is an algorithm for $m-k$ Tree Augmentation running in time $O^*(8^k)$.*

5.2 A Quadratic Kernel for $(m-k)$ Cactus Augmentation

By introducing a further reduction rule to go along with the ones introduced in the previous section, we obtain the following result.

Lemma 7. *Let $(\mathcal{C}, \mathcal{L}, k)$ be an instance of $m-k$ Cactus Augmentation. There is a polynomial time algorithm that takes $(\mathcal{C}, \mathcal{L}, k)$ as input and either correctly concludes that this instance is a Yes instance or returns an equivalent instance with $O(k^2)$ vertices and links.*

Kernels for General $m-k$ Augmentation. Given an instance of $m-k$-Augmentation by One, we apply the algorithm of [2] and reduce it in polynomial time to an equivalent instance of either $m-k$-Tree Augmentation or $m-k$-Cactus Augmentation. We then apply the apply the appropriate kernelization algorithm and obtain an equivalent kernelized instance $(G, \mathcal{L}, k)$ of $m-k$-Tree Augmentation or $m-k$-Cactus Augmentation. In the former case, make λ copies of each tree edge and in the latter case, make $\lambda/2$ copies of each cactus edge and return the instance $(G', \mathcal{L}, k)$ where G' is the graph thus constructed. This is an equivalent instance of $m-k$-Augmentation by One which has the same number of vertices and links.

6 Improved Algorithms for k-Augmentation

Theorem 6. *The* Weighted Edge Connectivity Augmentation of a Cactus *problem can be solved in time $O^*(9^k)$.*

Proof. Given an instance $(G = (V, E), \mathcal{C}, \mathcal{L}, k)$, we construct the link-terminal intersection graph I_G. Let X be the vertices of I_G which correspond to degree-2 vertices of $\mathcal{C}$. By Lemma 1, we have that a set S of links is an augmenting set if and only if $I_G[S \cup X]$ has a connected component containing X. Therefore, we conclude that a min-cost augmenting set of size at most k exactly corresponds to a min-cost Steiner tree of size at most $k + |X|$ in I_G containing the set X. The dynamic programming algorithm of Dreyfus and Wagner [3] can be modified to compute the min-cost Steiner tree of size at most k in time $O^*(3^{|X|})$. Since we already know that any augmenting set of links has size at least $|X|/2$, we return No if $|X| > 2k$. Hence, we may assume that $|X| \leq 2k$ and therefore, we have an algorithm for Min Cost augmentation of a cactus running in time $O^*(9^k)$. □

Since the problem of augmenting a tree can be reduced to augmenting a cactus simply by replacing every tree edge with a pair of parallel edges, we have the following corollary.

Corollary 1. *The* Weighted Edge Connectivity Augmentation of a Tree *problem can be solved in time* $O^*(9^k)$.

Once again, using the result of Dinits et al. referred to in earlier sections, w-Aug-One can, in polynomial time be reduced to Weighted Edge Connectivity Augmentation of a Tree or Weighted Edge Connectivity Augmentation of a Cactus depending on the parity of the connectivity of the given graph, hence proving Theorem 1.

References

1. Bodlaender, H.L., Cygan, M., Kratsch, S., Nederlof, J.: Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) ICALP 2013, Part I. LNCS, vol. 7965, pp. 196–207. Springer, Heidelberg (2013)
2. Dinits, E., Karzanov, A., Lomonosov, M.: On the structure of a family of minimal weighted cuts in graphs. In: Fridman, A. (ed.) Studies in Discrete Mathematics, Nauka, Moscow, pp. 290–306 (1976)
3. Dreyfus, S.E., Wagner, R.A.: The steiner problem in graphs. Networks 1(3), 195–207 (1971)
4. Eswaran, K., Tarjan, R.: Augmentation problems. SIAM Journal on Computing 5(4), 653–665 (1976)
5. Frank, A.: Augmenting graphs to meet edge-connectivity requirements. SIAM Journal on Discrete Mathematics 5(1), 25–53 (1992)
6. Frederickson, G.N., JáJá, J.: Approximation algorithms for several graph augmentation problems. SIAM J. Comput. 10(2), 270–283 (1981)
7. Guo, J., Uhlmann, J.: Kernelization and complexity results for connectivity augmentation problems. Networks 56(2), 131–142 (2010)
8. Marx, D., Végh, L.A.: Fixed-parameter algorithms for minimum cost edge-connectivity augmentation. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) ICALP 2013, Part I. LNCS, vol. 7965, pp. 721–732. Springer, Heidelberg (2013)
9. Nagamochi, H.: An approximation for finding a smallest 2-edge-connected subgraph containing a specified spanning tree. Discrete Applied Mathematics 126(1), 83–113 (2003); 5th Annual International Computing and combinatorics Conference
10. Vegh, L.: Augmenting undirected node-connectivity by one. SIAM Journal on Discrete Mathematics 25(2), 695–718 (2011)
11. Watanabe, T., Nakamura, A.: Edge-connectivity augmentation problems. Journal of Computer and System Sciences 35(1), 96–144 (1987)

Nonuniform Graph Partitioning with Unrelated Weights

Konstantin Makarychev[1] and Yury Makarychev[2,⋆]

[1] Microsoft Research
[2] Toyota Technological Institute at Chicago

Abstract. We give a bi-criteria approximation algorithm for the Minimum Nonuniform Partitioning problem, recently introduced by Krauthgamer, Naor, Schwartz and Talwar (2014). In this problem, we are given a graph $G = (V, E)$ on n vertices and k numbers $\rho_1, \dots, \rho_k$. The goal is to partition the graph into k disjoint sets $P_1, \dots, P_k$ satisfying $|P_i| \leq \rho_i n$ so as to minimize the number of edges cut by the partition. Our algorithm has an approximation ratio of $O(\sqrt{\log n \log k})$ for general graphs, and an $O(1)$ approximation for graphs with excluded minors. This is an improvement upon the $O(\log n)$ algorithm of Krauthgamer, Naor, Schwartz and Talwar (2014). Our approximation ratio matches the best known ratio for the Minimum (Uniform) k-Partitioning problem.

We extend our results to the case of "unrelated weights" and to the case of "unrelated d-dimensional weights". In the former case, different vertices may have different weights and the weight of a vertex may depend on the set P_i the vertex is assigned to. In the latter case, each vertex u has a d-dimensional weight $r(u,i) = (r_1(u,i), \dots, r_d(u,i))$ if u is assigned to P_i. Each set P_i has a d-dimensional capacity $c(i) = (c_1(i), \dots, c_d(i))$. The goal is to find a partition such that $\sum_{u \in P_i} r(u,i) \leq c(i)$ coordinate-wise.

1 Introduction

We study the Minimum Nonuniform Partitioning problem, which was recently proposed by Krauthgamer, Naor, Schwartz and Talwar (2014). We are given a graph $G = (V, E)$, parameter k and k numbers (capacities) $\rho_1, \dots, \rho_k$. Our goal is to partition the graph G into k pieces (bins) $P_1, \dots, P_k$ satisfying capacity constraints $|P_i| \leq \rho_i n$ so as to minimize the number of cut edges. The problem is a generalization of the Minimum k-Partitioning problem studied by Krauthgamer, Naor, and Schwartz (2009), in which all bins have equal capacity $\rho_i = 1/k$.

The problem has many applications (see Krauthgamer et al. 2014). Consider an example in cloud computing: Imagine that we need to distribute n computational tasks – vertices of the graph – among k machines, each with capacity $\rho_i n$. Different tasks communicate with each other. The amount of communication between tasks u and v equals the weight of the edges between the corresponding vertices u and v. Our goal is to distribute tasks among k machines subject to capacity constraints so as to minimize the total amount of communication between machines.[1]

[⋆] Supported by NSF CAREER award CCF-1150062 and NSF award IIS-1302662.

[1] In this example, we need to solve a variant of the problem with edge weights.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 812–822, 2014.

The problem is quite challenging. Krauthgamer et al. (2014) note that many existing techniques do not work for this problem. Particularly, it is not clear how to solve this problem on tree graphs[2] and consequently how to use Räcke's (2008) tree decomposition technique. Krauthgamer et al. (2014) give an $O(\log n)$ bi-criteria approximation algorithm for the problem: the algorithm finds a partition $P_1, \dots, P_k$ such that $|P_i| \leq O(\rho_i n)$ for every i and the number of cut edges is $O(\log n \, OPT)$. The algorithm first solves a configuration linear program and then uses a new sophisticated method to round the fractional solution.

In this paper, we present a rather simple SDP based $O(\sqrt{\log n \log k})$ bi-criteria approximation algorithm for the problem. We note that our approximation guarantee matches that of the algorithm of Krauthgamer, Naor, and Schwartz (2009) for the the Minimum k-Partitioning problem (which is a special case of Minimum Nonuniform Partitioning, see above). Our algorithm uses a technique of "orthogonal separators" developed by Chlamtac, Makarychev, and Makarychev (2006) and later used by Bansal, Feige, Krauthgamer, Makarychev, Nagarajan, Naor, and Schwartz (2011) for the Small Set Expansion problem. Using orthogonal separators, it is relatively easy to get a distribution over partitions $\{P_1, \dots, P_k\}$ such that $\mathbb{E}[|P_i|] \leq O(\rho_i n)$ for all i and the expected number of cut edges is $O(\sqrt{\log n \log(1/\rho_{min})} \, OPT)$ where $\rho_{min} = \min_i \rho_i$. The problem is that for some i, P_i may be much larger than its expected size. The algorithm of Krauthgamer et al. (2014) solves a similar problem by first simplifying the instance and then grouping parts P_i into "mega-buckets". We propose a simpler fix: Roughly speaking, if a set P_i contains too many vertices, we remove some of these vertices and re-partition the removed vertices into k pieces again. Thus we ensure that all capacity constraints are (approximately) satisfied. It turns out that every vertex gets removed a constant number of times in expectation. Hence, the re-partitioning step increases the number of cut edges only by a constant factor. Another problem is that $1/\rho_{min}$ may be much larger than k. To deal with this problem, we transform the SDP solution (eliminating "short" vectors) and redefine thresholds ρ_i so that $1/\rho_{min}$ becomes $O(k)$.

Our technique is quite robust and allows us to solve more general versions of the problem, *Nonuniform Graph Partitioning with unrelated weights* and *Nonuniform Graph Partitioning with unrelated d-dimensional weights.*

Minimum Nonuniform Graph Partitioning with unrelated weights captures the variant of the problem where we assign vertices (tasks/jobs) to unrelated machines and the weight of a vertex (the size of the task/job) depends on the machine it is assigned to.

Definition 1.1 (Minimum Nonuniform Graph Partitioning with Unrelated Weights). *We are given a graph $G = (V, E)$ on n vertices and a natural number $k \geq 2$. Additionally, we are given k normalized measures $\mu_1, \dots, \mu_k$ on V (satisfying $\mu_i(V) = 1$) and k numbers $\rho_1, \dots, \rho_k \in (0, 1)$. Our goal is to partition the graph into k pieces (bins) $P_1, \dots, P_k$ such that $\mu_i(P_i) \leq \rho_i$ so as to minimize the number of cut edges. Some pieces P_i may be empty.*

We will only consider instances of Minimum Nonuniform Graph Partitioning that have a feasible solution. We give an $O_\varepsilon(\sqrt{\log n \log \min(1/\rho_{min}, k)})$ bi-criteria approximation algorithm for the problem.

[2] Our algorithm gives a constant factor bi-criteria approximation for trees.

Theorem 1.1. *For every $\varepsilon > 0$, there exists a randomized polynomial-time algorithm that given an instance of Minimum Nonuniform Graph Partitioning with unrelated weights finds a partition $P_1, \dots, P_k$ satisfying $\mu_i(P_i) \leq 5(1+\varepsilon)\rho_i$. The expected cost of the solution is at most $D \times OPT$, where OPT is the optimal value, $D = O_\varepsilon(\sqrt{\log n \log \min(1/\rho_{min}, k)})$ and $\rho_{min} = \min_i \rho_i$. For graphs with excluded minors $D = O_\varepsilon(1)$.*

Nonuniform Graph Partitioning with unrelated d-dimensional weights further generalizes the problem. In this variant of the problem, we assume that we have d resources (e.g. CPU speed, random access memory, disk space, network). Each piece P_i has $c_j(i)$ units of resource $j \in \{1, \dots, d\}$, and each vertex u requires $r_j(u,i)$ units of resource $j \in \{1, \dots, d\}$ when it is assigned to piece P_i. We need to partition the graph so that capacity constraints for all resources are satisfied. The d-dimensional version of Minimum (uniform) k-Partitioning was previously studied by Amir et al. (2014). In their problem, all $\rho_i = 1/k$ are the same, and r_j's do not depend on i.

Definition 1.2 (Minimum Nonuniform Graph Partitioning with Unrelated d-Dimensional Weights). *We are given a graph $G = (V, E)$ on n vertices. Additionally, we are given non-negative numbers $c_j(i)$ and $r_j(u,i)$ for $i \in \{1, \dots, k\}$, $j \in \{1, \dots, d\}$, $u \in V$. Our goal is to find a partition of V into $P_1, \dots, P_k$ subject to capacity constraints $\sum_{u \in V} r_j(u,i) \leq c_j(i)$ for every i and j so as to minimize the number of cut edges.*

We present a bi-criteria approximation algorithm for this problem.

Theorem 1.2. *For every $\varepsilon > 0$, there exists a randomized polynomial-time algorithm that given an instance of Minimum Nonuniform Graph Partitioning with unrelated d-dimensional weights finds a partition $P_1, \dots, P_k$ satisfying*

$$\sum_{v \in V} r_j(v,i) \leq 5d(1+\varepsilon)c_j(i) \quad \textit{for every } i \textit{ and } j.$$

The expected cost of the solution is at most $D \times OPT$, where OPT is the optimal value, $D = O_\varepsilon(\sqrt{\log n \log k})$. For graphs with excluded minors $D = O_\varepsilon(1)$.

We note that this result is a simple corollary of Theorem 1.1 we let $\mu'_i(u) = \max_j(r_j(u,i)/c_j(i))$ and then apply our result to measures $\mu_i(u) = \mu'_i(u)/\mu'_i(V)$ (we describe the details in the full version of this paper (Makarychev and Makarychev, 2014, Appendix C)).

We remark that our algorithms work if edges in the graph have arbitrary positive weights. However, for simplicity of exposition, we describe the algorithms for the setting where all edge weights are equal to one. To deal with arbitrary edge weights, we only need to change the SDP objective function.

Our paper strengthens the result of Krauthgamer et al. (2014) in two ways. First, it improves the approximation factor from $O(\log n)$ to $O(\sqrt{\log n \log k})$. Second, it studies considerably more general variants of the problem, Minimum Nonuniform Partitioning with unrelated weights and Minimum Nonuniform Partitioning with unrelated d-dimensional weights. We believe that these variants are very natural. Indeed, one of the main motivations for the Minimum Nonuniform Partitioning problem is its applications to scheduling and load balancing: in these applications, the goal is to assign tasks to machines so as to minimize the total amount of communication between different machines, subject to capacity constraints. The constraints that we study in the paper are very general and analogous to those that are often considered in the scheduling literature. We note that the method developed in Krauthgamer et al. (2014) does not handle these more general variants of the problem.

2 Algorithm

SDP Relaxation. Our relaxation for the problem is based on the SDP relaxation for the Small Set Expansion (SSE) problem of Bansal et al. (2011). We write the SSE relaxation for every cluster P_i and then add consistency constraints similar to constraints used in Unique Games. For every vertex u and index $i \in \{1, \ldots, k\}$, we introduce a vector $\bar{u}_i$. In the integral solution, this vector is simply the indicator variable for the event "$u \in P_i$". It is easy to see that in the integral case, the number of cut edges equals (1). Indeed, if u and v lie in the same P_j, then $\bar{u}_i = \bar{v}_i$ for all i; if u lies in $P_{j'}$ and v lies in $P_{j''}$ (for $j' \neq j''$) then $\|\bar{u}_i - \bar{v}_i\|^2 = 1$ for $i \in \{j', j''\}$ and $\|\bar{u}_i - \bar{v}_i\|^2 = 0$ for $i \notin \{j', j''\}$. The SDP objective is to minimize (1).

We add constraint (2) saying that $\mu_i(P_i) \leq \rho_i$. We further add spreading constraints (4) from Bansal et al. (2011) (see also Louis and Makarychev (2014)). The spreading constraints above are satisfied in the integral solution: If $u \notin P_i$, then $\bar{u}_i = 0$ and both sides of the inequality equal 0. If $u \in P_i$, then the left hand side equals $\mu_i(P_i)$, and the right hand side equals ρ_i.

We write standard ℓ_2^2-triangle inequalities (6) and (7). Finally, we add consistency constraints. Every vertex u must be assigned to one and only one P_i, hence constraint (5) is satisfied. We obtain the following SDP relaxation.

SDP Relaxation

$$\min \frac{1}{2} \sum_{i=1}^{k} \sum_{(u,v)\in E} \|\bar{u}_i - \bar{v}_i\|^2 \tag{1}$$

subject to

$$\sum_{u\in V} \|\bar{u}_i\|^2 \mu_i(u) \leq \rho_i \qquad \text{for all } i \in [k] \tag{2}$$

$$\sum_{v\in V} \langle \bar{u}_i, \bar{v}_i \rangle \mu_i(v) \leq \|\bar{u}_i\|^2 \rho_i \tag{3}$$

$$\text{for all } u \in V,\ i \in [k] \tag{4}$$

$$\sum_{i=1}^{k} \|\bar{u}_i\|^2 = 1 \qquad \text{for all } u \in V \tag{5}$$

$$\|\bar{u}_i - \bar{v}_i\|^2 + \|\bar{v}_i - \bar{w}_i\|^2 \geq \|\bar{u}_i - \bar{w}_i\|^2 \qquad \text{for all } u, v, w \in V,\ i \in [k] \tag{6}$$

$$0 \leq \langle \bar{u}_i, \bar{v}_i \rangle \leq \|\bar{u}_i\|^2 \qquad \text{for all } u, v \in V,\ i \in [k] \tag{7}$$

Small Set Expansion and Orthogonal Separators. Our algorithm uses a technique called "orthogonal separators". The notion of orthogonal separators was introduced in Chlamtac, Makarychev, and Makarychev (2006), where it was used in an algorithm for Unique Games. Later, Bansal et al. (2011) showed that the following holds. If the SDP solution satisfies constraints (3), (4), (6), and (7), then for every $\varepsilon \in (0,1)$, $\delta \in (0,1)$, and $i \in [k]$, there exist a distortion $D_i = O_\varepsilon(\sqrt{\log n \log(1/(\delta\rho_i))})$, and a probability distribution over subsets of V such that for a random set $S_i \subset V$ ("orthogonal separator") distributed according to this distribution, we have for $\alpha = 1/n$,

- $\mu_i(S_i) \leq (1+\varepsilon)\rho_i$ (always);
- For all u, $\Pr(u \in S_i) \in [(1-\delta)\alpha\|\bar{u}_i\|^2, \alpha\|\bar{u}_i\|^2]$;
- For all $(u,v) \in E$, $\Pr(u \in S_i, v \notin S_i) \leq \alpha D_i \cdot \|\bar{u}_i - \bar{v}_i\|^2$.

We let $D = \max_i D_i$. This statement was proved in Bansal et al. (2011) implicitly; for completeness we prove it in the full version of this paper (Makarychev and Makarychev, 2014, Theorem A.1). For graphs with excluded minors and bounded genus graphs, $D = O_\varepsilon(1)$.

Algorithm. Let us examine a somewhat naïve algorithm for the problem inspired by the algorithm of Bansal et al. (2011) for Small Set Expansion. We shall maintain the set of active (yet unassigned) vertices $A(t)$. Initially, all vertices are active, i.e. $A(0) = V$. At every step t, we pick a random index $i \in \{1,\dots,k\}$ and sample an orthogonal separator $S_i(t)$ as described above. We assign all active vertices from $S_i(t)$ to the bin number i:

$$P_i(t+1) = P_i(t) \cup (S_i(t) \cap A(t)),$$

and mark all newly assigned vertices as inactive i.e., we let $A(t+1) = A(t) \setminus S_i(t)$. We stop when the set of active vertices $A(t)$ is empty. We output the partition $\mathcal{P} = \{P_1(T),\dots,P_k(T)\}$, where T is the index of the last iteration.

We can show that the number of edges cut by the algorithm is at most $O(D \times OPT)$, where D is the distortion of orthogonal separators. Furthermore, the expected weight of each P_i is $O(\rho_i)$. However, weights of some pieces may significantly deviate from the expectation and may be much larger than ρ_i. So we need to alter the algorithm to guarantee that all sizes are bounded by $O(\rho_i)$ simultaneously. We face a problem similar to the one Krauthgamer, Naor, Schwartz and Talwar (2014) had to solve in their paper. Their solution is rather complex and does not seem to work in the weighted case. Here, we propose a very simple fix for the naïve algorithm we presented above. We shall store vertices in every bin in layers. When we add new vertices to a bin at some iteration, we put them in a new layer on top of already stored vertices. Now, if the weight of the bin number i is greater than $5(1+\varepsilon)\rho_i$, we remove bottom layers from this bin so that its weight is at most $5(1+\varepsilon)\rho_i$. Then we mark the removed vertices as active and jump to the next iteration. It is clear that this algorithm always returns a solution satisfying $\mu_i(P_i) \leq 5(1+\varepsilon)\rho_i$ for all i. But now we need to prove that the algorithm terminates, and that the expected number of cut edges is still bounded by $O(D \times OPT)$.

Before proceeding to the analysis, we describe the algorithm in detail.

Algorithm for Nonuniform Partitioning with Unrelated Weights

Input: a graph $G = (V, E)$ on n vertices; a positive integer $k \leq n$; a sequence of numbers $\rho_1, \dots, \rho_k \in (0, 1)$ (with $\rho_1 + \cdots + \rho_k \geq 1$); weights $\mu_i : V \to \mathbb{R}^+$ (with $\mu_i(V) = 1$).

Output: a partitioning of vertices into disjoint sets $P_1, \dots, P_k$ such that $\mu_i(P_i) \leq 5(1 + \varepsilon)\rho_i$.

- The algorithm maintains a partitioning of V into a set of active vertices $A(t)$ and k sets $P_1(t), \dots P_k(t)$, which we call bins. For every inactive vertex $u \notin A(t)$, we remember its depth in the bin it belongs to. We denote the depth by $\mathrm{depth}_u(t)$. If $u \in A(t)$, then we let $\mathrm{depth}_u(t) = \perp$.
- Initially, set $A(0) = V$; and $P_i(0) = \varnothing$, $\mathrm{depth}_u(t) = \perp$ for all i; $t = 0$.
- **while** $A(t) \neq \varnothing$
  1. Pick an index $i \in \{1, \dots, k\}$ uniformly at random.
  2. Sample an orthogonal separator $S_i(t) \subset V$ with $\delta = \varepsilon/4$ as described in Section 2.
  3. Store all active vertices from the set $S_i(t)$ in the bin number i. If $\mu_i(P_i(t) \cup (S_i(t) \cap A(t))) \leq 5(1 + \varepsilon)\rho_i$, then simply add these vertices to $P_i(t + 1)$:
     $$P_i(t + 1) = P_i(t) \cup (S_i(t) \cap A(t)).$$
     Otherwise, find the largest depth d such that $\mu_i(P_i(t+1)) \leq 5(1+\varepsilon)\rho_i$, where
     $$P_i(t + 1) = \{u \in P_i(t) : \mathrm{depth}_u(t) \leq d\} \cup (S_i(t) \cap A(t)).$$
     In other words, add to the bin number i vertices from $S_i(t) \cap A(t)$ and remove vertices from the bottom layers so that the weight of the bin is at most $5(1 + \varepsilon)\rho_i$.
  4. If we put at least one new vertex in the bin i at the current iteration, that is, if $A(t) \cap S_i(t) \neq \varnothing$, then set the depth of all newly stored vertices to 1; increase the depth of all other vertices in the bin i by 1.
  5. Update the set of active vertices: let $A(t+1) = V \setminus \bigcup_j P_j(t+1)$ and $\mathrm{depth}_u(t+1) = \perp$ for $u \in A(t + 1)$. Let $t = t + 1$.
- Set $T = t$ and return the partitioning $P_1(T), \dots, P_k(T)$.

Note that Step 3 is well defined. We can always find an index d such that $\mu_i(P_i(t+1)) \leq 5(1 + \varepsilon)\rho_i$, because for $d = 0$, we have $P_i(t + 1) = S_i(t) \cap A(t)$ and thus

$$\mu(P_i(t + 1)) = \mu_i(S_i(t) \cap A(t)) \leq \mu_i(S_i(t)) \leq (1 + \varepsilon)\rho_i < 5(1 + \varepsilon)\rho_i,$$

by the first property of orthogonal separators.

Analysis. We will first prove Theorem 2.1 that states that the algorithm has approximation factor $D = O_\varepsilon(\sqrt{\log n \log(1/\rho_{min})})$ on arbitrary graphs, and $D = O_\varepsilon(1)$ on graphs excluding a minor. Then we will show how to obtain $D = O_\varepsilon(\sqrt{\log n \log k})$ approximation on arbitrary graphs (see Appendix B in the full version of this paper (Makarychev and Makarychev, 2014)). To this end, we will transform the SDP solution and redefine measures μ_i and capacities ρ_i so that $\rho_{min} \geq \delta/k$, then apply Theorem 2.1. The new SDP solution will satisfy all SDP constraints except possibly for constraint (5); it will however satisfy a relaxed constraint

$$\sum_{i=1}^{k} \|\bar{u}_i\|^2 \in [1-\delta, 1] \qquad \text{for all } u \in V. \tag{5$'$}$$

Thus in Theorem 2.1, we will assume only that the solution satisfies the SDP relaxation with constraint (5) replaced by constraint (5′).

Theorem 2.1. *The algorithm returns a partitioning $P_1(T), \dots, P_k(T)$ satisfying $\mu_i(P_i) \leq 5(1+\varepsilon)\rho_i$. The expected number of iterations of the algorithm is at most $\mathbb{E}[T] \leq 4n^2k + 1$ and the expected number of cut edges is at most $O(D \times SDP) = O(D \times OPT)$, where $D = O_\varepsilon(\sqrt{\log n \log(1/\rho_{min})})$ is the distortion of orthogonal separators; $\rho_{min} = \min_i \rho_i$. If the graph has an excluded minor, then $D = O_\varepsilon(1)$ (the constant depends on the excluded minor).*

We assume only that the SDP solution given to the algorithm satisfies the SDP relaxation with constraint (5) replaced by constraint (5′).

As we mentioned earlier, the algorithm always returns a valid partitioning. We need to verify that the algorithm terminates in expected polynomial time, and that it produces cuts of cost at most $O(D \times OPT)$ (see also Remark C.1 in the full version of this paper (Makarychev and Makarychev, 2014)).

The state of the algorithm at iteration t is determined by the sets $A(t)$, $P_1(t), \dots, P_k(t)$ and the depths of the elements. We denote the state by $\mathcal{C}(t) = \{A(t), P_1(t), \dots, P_k(t), \operatorname{depth}(t)\}$. Observe that the probability that the algorithm is in the state $\mathcal{C}^*$ at iteration $(t+1)$ is determined only by the state of the algorithm at iteration t. It does not depend on t (given $\mathcal{C}(t)$). So the states of the algorithm form a Markov random chain. The number of possible states is finite (since the depth of every vertex is bounded by n). To simplify the notation, we assume that for $t \geq T$, $\mathcal{C}(t) = \mathcal{C}(T)$. This is consistent with the definition of the algorithm — if we did not stop the algorithm at time T, it would simply idle, since $A(t) = \varnothing$, and thus $S_i(t) \cap A(t) = \varnothing$ for $t \geq T$.

We are interested in the probability that an inactive vertex u which lies *in the top layer* of one of the bins (i.e., $u \notin A(t)$ and $\operatorname{depth}_u(t) = 1$) is removed from that bin within m iterations. We let

$$f(m, u, \mathcal{C}^*) = \Pr(\exists t \in [t_0, t_0 + m] \text{ s.t. } u \in A(t) \mid \mathcal{C}(t_0) = \mathcal{C}^*, \operatorname{depth}_u(t_0) = 1).$$

That is, $f(m, u, \mathcal{C}^*)$ is the probability that u is removed from the bin i at one of the iterations $t \in [t_0, t_0 + m]$ given that at iteration t_0 the state of the algorithm is $\mathcal{C}^*$ and u is in the top layer of the bin i. Note that the probability above does not depend on t_0 and thus $f(m, u, \mathcal{C}^*)$ is well defined. We let

$$f(m) = \max_{u \in V} \max_{\mathcal{C}^*} f(m, u, \mathcal{C}^*).$$

Our fist lemma gives a bound on the expected number of steps on which a vertex u is active in terms of $f(m)$.

Lemma 2.1. *For every possible state of the algorithm $\mathcal{C}^*$, every vertex u, and natural number t_0,*

$$\sum_{t=t_0}^{t_0+m} \Pr(u \in A(t) \mid \mathcal{C}(t_0) = \mathcal{C}^*) \leq \frac{k}{(1-2\delta)\alpha(1-f(m-1))}. \tag{8}$$

Proof. The left hand side of inequality (8) equals expected number (conditioned on $\mathcal{C}(t_0) = \mathcal{C}^*$) of iterations t in the interval $[t_0, t_0+m]$ at which u is active i.e., $u \in A(t)$. Our goal is to upper bound this quantity.

Initially, at time t_0, u is active or inactive. At every time t when u is active, u is thrown in one of the bins P_i with probability at least (here, we use that the SDP solution satisfies constraint (5′))

$$\frac{1}{k}\sum_{i=1}^{k}(1-\delta)\alpha\|\bar{u}_i\|^2 \geq \frac{(1-2\delta)\alpha}{k}.$$

So the expected number of iterations passed since u becomes active till u is stored in one of the bins and thus becomes inactive is at most $k/((1-2\delta)\alpha)$.

Suppose that u is stored in a bin i at iteration t, then $u \in P_i(t+1)$ and $\mathrm{depth}_u(t+1) = 1$. Thus, the probability that u is reactivated till iteration t_0+m i.e., the probability that for some $\tau \in [(t+1), t_0+m] \subset [(t+1), (t+1)+(m-1)]$, $u \in A(\tau)$ is at most $f(m-1)$. Consequently, the expected number of iterations $t \in [t_0, t_0+m]$ at which u is active is bounded by

$$\frac{k \cdot 1}{(1-2\delta)\alpha} + \frac{k \cdot f(m)}{(1-2\delta)\alpha} + \frac{k \cdot f^2(m)}{(1-2\delta)\alpha} + \cdots = \frac{k}{(1-2\delta)\alpha(1-f(m))}.$$

We now show that $f(m) \leq 1/2$ for all m.

Lemma 2.2. *For all natural m, $f(m) \leq 1/2$.*

Proof. We prove this lemma by induction on m. For $m = 0$, the statement is trivial as $f(0) = 0$.

Consider an arbitrary state $\mathcal{C}^*$, bin i^*, vertex u, and iteration t_0. Suppose that $\mathcal{C}(t_0) = \mathcal{C}^*$, $u \in P_{i^*}(t_0)$ and $\mathrm{depth}_u(t_0) = 1$ i.e., u lies in the top layer in the bin i^*. We need to estimate the probability that u is removed from the bin i^* till iteration t_0+m. The vertex u is removed from the bin i^* if and only if at some iteration $t \in \{t_0, \ldots, t_0+m-1\}$, u is "pushed away" from the bin by new vertices (see Step 2 of the algorithm). This happens only if the weight of vertices added to the bin i^* at iterations $\{t_0, \ldots, t_0+m-1\}$ plus the weight of vertices in the first layer of the bin at iteration t_0 exceeds $5(1+\varepsilon)\rho_i$. Since the weight of vertices in the first layer is at most $(1+\varepsilon)\rho_i$, the weight of vertices added to the bin i^* at iterations $\{t_0, \ldots, t_0+m-1\}$ must be greater than $4(1+\varepsilon)\rho_{i^*}$.

We compute the expected weight of vertices thrown in the bin i^* at iterations $t \in \{t_0, \ldots, t_0+m-1\}$. Let us introduce some notation: $M = \{t_0, \ldots, t_0+m-1\}$;

$i(t)$ is the index i chosen by the algorithm at the iteration t. Let X_{M,i^*} be the weight of vertices thrown in the bin i^* at iterations $t \in M$. Then,

$$\mathbb{E}\big[X_{M,i^*} \mid \mathcal{C}(t_0) = \mathcal{C}^*\big] = \mathbb{E}\Big[\sum_{\substack{t \in M \\ s.t.\ i(t)=i^*}} \mu_{i^*}\big(S_{i^*}(t) \cap A(t)\big) \mid \mathcal{C}(t_0) = \mathcal{C}^*\Big] \tag{9}$$

$$= \sum_{t \in M} \sum_{v \in V} \Pr\big(i(t) = i^* \text{ and } v \in S_{i^*}(t) \cap A(t) \ \mid\ \mathcal{C}(t_0) = \mathcal{C}^*\big)\mu_{i^*}(v).$$

The event "$i(t) = i^*$ and $v \in S_{i^*}(t)$" is independent from the event "$v \in A(t)$ and $\mathcal{C}(t_0) = \mathcal{C}^*$". Thus,

$$\Pr\big(i(t) = i^* \text{ and } v \in S_{i^*}(t) \cap A(t) \ \mid\ \mathcal{C}(t_0) = \mathcal{C}^*\big)$$
$$= \Pr\big(i(t) = i^* \text{ and } v \in S_{i^*}(t)\big) \cdot \Pr\big(v \in A(t) \ \mid\ \mathcal{C}(t_0) = \mathcal{C}^*\big).$$

Since $i(t)$ is chosen uniformly at random in $\{1, \dots, k\}$, we have $\Pr(i(t) = i^*) = 1/k$. Then, by property 2 of orthogonal separators, $\Pr(v \in S_{i^*}(t) \mid i(t) = i^*) \le \alpha\|\bar{v}_{i^*}\|^2$. We get

$$\Pr\big(i(t) = i^* \text{ and } v \in S_{i^*}(t) \cap A(t) \ \mid\ \mathcal{C}(t_0) = \mathcal{C}^*\big) \le$$
$$\le \frac{\alpha\|\bar{v}_{i^*}\|^2}{k} \cdot \Pr\big(v \in A(t) \ \mid\ \mathcal{C}(t_0) = \mathcal{C}^*\big).$$

We now plug this expression in (9) and use Lemma 2.1,

$$\begin{aligned}\mathbb{E}[X_{M,i^*} \mid \mathcal{C}(t_0) = \mathcal{C}^*] &\le \sum_{v \in V} \frac{\alpha\|\bar{v}_{i^*}\|^2\mu_{i^*}(v)}{k} \cdot \sum_{t \in M} \Pr\big(v \in A(t) \ \mid\ \mathcal{C}(t_0) = \mathcal{C}^*\big)\\ &\le \sum_{v \in V} \frac{\alpha\|\bar{v}_{i^*}\|^2\mu_{i^*}(v)}{k} \cdot \frac{k}{(1-2\delta)\alpha(1-f(m-1))}\\ &= \sum_{v \in V} \frac{\|\bar{v}_{i^*}\|^2\mu_{i^*}(v)}{(1-2\delta)(1-f(m-1))}.\end{aligned}$$

Finally, observe that $1 - f(m-1) \ge 1/2$ by the inductive hypothesis, and $\sum_{v \in V}\|\bar{v}_{i^*}\|^2\mu_{i^*}(v) \le \rho_{i^*}$ by the SDP constraint (2). Hence, $\mathbb{E}[X_{M,i^*} \mid \mathcal{C}(t_0) = \mathcal{C}^*] \le 2\rho_{i^*}/(1-2\delta)$. By Markov's inequality,

$$\Pr\big(X_{M,i^*} \ge 4(1+\varepsilon)\rho_{i^*}\big) \le \frac{2\rho_{i^*}}{4(1-2\delta)(1+\varepsilon)\rho_{i^*}} \le \frac{1}{2},$$

since $\delta = \varepsilon/4$. This concludes the proof.

As an immediate corollary of Lemmas 2.1 and 2.2, we get that for all $u \in V$,

$$\sum_{t=0}^{\infty} \Pr(u \in A(t)) = \lim_{m \to \infty} \sum_{t=0}^{m} \Pr(u \in A(t)) \le \frac{2k}{(1-2\delta)\alpha} \le \frac{4k}{\alpha}. \tag{10}$$

Proof (Proof of Theorem 2.1). We now prove Theorem 2.1. We first bound the expected running time. At every iteration of the algorithm $t < T$, the set $A(t)$ is not empty. Hence, using (10), we get

$$\mathbb{E}[T] \le \mathbb{E}\Big[\sum_{t=0}^{\infty} |A(t)|\Big] + 1 = \sum_{v \in V} \sum_{t=0}^{\infty} \Pr(v \in A(t)) + 1 \le n \cdot \frac{4k}{\alpha} + 1 = 4kn^2 + 1.$$

We now upper bound the expected size of the cut. For every edge $(u,v) \in E$ we estimate the probability that (u,v) is cut. Suppose that (u,v) is cut. Then, u and v belong to distinct sets $P_i(T)$. Consider the iteration t at which u and v are separated the first time. A priori, there are two possible cases:

1. At iteration t, u and v are active, but only one of the vertices u or v is added to some set $P_i(t+1)$; the other vertex remains in the set $A(t+1)$.
2. At iteration t, u and v are in some set $P_i(t)$, but only one of the vertices u or v is removed from the set $P_i(t+1)$.

It is easy to see that, in fact, the second case is not possible, since if u and v were never separated before iteration t, then u and v must have the same depth (i.e., $\mathrm{depth}_u(t) = \mathrm{depth}_v(t)$) and thus u and v may be removed from the bin i only together.

Consider the first case, and assume that $u \in P_{i(t)}(t+1)$ and $v \in A(t+1)$. Here, as in the proof of Lemma 2.2, we denote the index i chosen at iteration t by $i(t)$. Since $u \in P_{i(t)}(t+1)$ and $v \in A(t+1)$, we have $u \in S_{i(t)}(t)$ and $v \notin S_{i(t)}(t)$. Write

$$\begin{aligned}
&\Pr(u, v \in A(t);\, u \in S_{i(t)}(t);\ v \notin S_{i(t)}(t)) = \\
&\qquad = \Pr(u, v \in A(t)) \cdot \Pr(u \in S_{i(t)}(t);\ v \notin S_{i(t)}(t)) \\
&\qquad = \Pr(u, v \in A(t)) \cdot \sum_{i=1}^{k} \frac{\Pr(u \in S_i(t);\ v \notin S_i(t) \mid i(t) = i)}{k}.
\end{aligned}$$

We replace $\Pr(u, v \in A(t))$ with $\Pr(u \in A(t)) \ge \Pr(u, v \in A(t))$, and then use the inequality $\Pr(u \in S_i(t);\ v \notin S_i(t)) \le \alpha D\ \|\bar{u}_i - \bar{v}_i\|^2$, which follows from the third property of orthogonal separators. We get

$$\begin{aligned}
&\Pr(u, v \in A(t);\ u \in S_{i(t)}(t);\ v \notin S_{i(t)}(t)) \le \\
&\qquad\qquad \le \Pr(u \in A(t)) \times \Big(\frac{1}{k}\sum_{i=1}^{k} \alpha D\ \|\bar{u}_i - \bar{v}_i\|^2\Big).
\end{aligned}$$

Thus, the probability that u and v are separated at iteration t is upper bounded by $\Big(\Pr(u \in A(t)) + \Pr(v \in A(t))\Big) \times \Big(\frac{1}{k}\sum_{i=1}^{k} \alpha D\ \|\bar{u}_i - \bar{v}_i\|^2\Big)$. The probability that the edge (u,v) is cut (at some iteration) is at most

$$\begin{aligned}
&\Big(\sum_{t=0}^{\infty} \Pr(u \in A(t)) + \Pr(v \in A(t))\Big) \times \Big(\frac{1}{k}\sum_{i=1}^{k} \alpha D\ \|\bar{u}_i - \bar{v}_i\|^2\Big) \le \\
&\qquad\qquad \le \frac{8k}{\alpha}\Big(\frac{1}{k}\sum_{i=1}^{k} \alpha D\ \|\bar{u}_i - \bar{v}_i\|^2\Big) = 8\sum_{i=1}^{k} D\ \|\bar{u}_i - \bar{v}_i\|^2.
\end{aligned}$$

To bound the first term on the left hand side we used inequality (10). We get the desired bound on the expected number of cut edges:

$$\sum_{(u,v)\in E} \Pr((u,v) \text{ is cut}) \leq 8 \sum_{(u,v)\in E} \sum_{i=1}^{k} D \left\|\bar{u}_i - \bar{v}_i\right\|^2 = 16D \cdot SDP,$$

where SDP is the SDP value.

References

Räcke, H.: Optimal hierarchical decompositions for congestion minimization in networks. In: STOC 2008 (2008)

Bansal, N., Feige, U., Krauthgamer, R., Makarychev, K., Nagarajan, V., Naor, J., Schwartz, R.: Min-max Graph Partitioning and Small Set Expansion. In: FOCS 2011 (2011)

Chlamtac, E., Makarychev, K., Makarychev, Y.: How to Play Unique Games Using Embeddings. In: FOCS 2006 (2006)

Krauthgamer, R., Naor, J., Schwartz, R.: Partitioning graphs into balanced components. In: SODA 2009 (2009)

Krauthgamer, R., Naor, J., Schwartz, R., Talwar, K.: Non-Uniform Graph Partitioning. In: SODA 2014 (2014)

Louis, A., Makarychev, K.: Approximation Algorithm for Sparsest k-Partitioning. In: SODA 2014 (2014)

Makarychev, K., Makarychev, Y.: Nonuniform Graph Partitioning with Unrelated Weights. arXiv:1401.0699 [cs.DS] (2014)

Räcke, H.: Optimal hierarchical decompositions for congestion minimization in networks. In: STOC 2008 (2008)

Precedence-Constrained Scheduling of Malleable Jobs with Preemption*

Konstantin Makarychev[1] and Debmalya Panigrahi[2,**]

[1] Microsoft Research, Redmond, WA
komakary@microsoft.com
[2] Duke University, Durham, NC
debmalya@cs.duke.edu

Abstract. Scheduling jobs with precedence constraints on a set of identical machines to minimize the total processing time (makespan) is a fundamental problem in combinatorial optimization. In practical settings such as cloud computing, jobs are often *malleable*, i.e., can be processed on multiple machines simultaneously. The instantaneous processing rate of a job is a non-decreasing function of the number of machines assigned to it (we call it the processing function). Previous research has focused on practically relevant concave processing functions, which obey the law of diminishing utility and generalize the classical (non-malleable) problem. Our main result is a $(2 + \epsilon)$-approximation algorithm for concave processing functions (for any $\epsilon > 0$), which is the best possible under complexity theoretic assumptions. The approximation ratio improves to $(1 + \epsilon)$ for the interesting and practically relevant special case of power functions, i.e., $p_j(z) = c_j \cdot z^\gamma$.

1 Introduction

In the *precedence-constrained scheduling* problem (we call it the PS problem), the goal is to schedule a set of jobs with precedence constraints on a set of identical machines so as to minimize the overall time for processing them (called the *makespan*). One of the first results in approximation algorithms was a 2-approximation for this problem due to Graham in 1966 [13]. On the negative side, this problem was shown to be NP-hard to approximate to a ratio better than 4/3 by Lenstra and Rinnooy Kan in 1978 [19]. In spite of substantial effort, the gap between these two bounds remained open for three decades. Recently, Svensson [27] has provided strong evidence that improving Graham's result might in fact be impossible by showing that it is tight under certain complexity theoretic assumptions.

A natural generalization of the PS problem considered in the literature is that of *malleable* jobs, i.e., jobs that can be processed simultaneously on multiple machines. This is particularly relevant in practice for domains such as cloud computing, operating

* Part of this work was done when both authors were at Microsoft Research, Redmond, WA. For details omitted from this extended abstract, the reader is referred to the full version of the paper [21].

** Supported in part by a Duke University startup grant.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 823–834, 2014.

systems, high performance computing, project management, etc. where a fixed set of resources must be distributed among precedence-constrained tasks to complete them by the earliest possible time. At any given time, the processing rate of a job is a function of the number of machines assigned to it (we call it the *processing function*). The goal is to produce a schedule of minimum makespan.

Formally, the input comprises a *directed acyclic graph* (DAG) $G = (J, E)$, where each vertex $j \in J$ represents a job j and has a given size $s_j > 0$. The arcs in E represent the *precedence constraints* on the jobs, i.e., if $(j_1, j_2) \in E$, then job j_1 has to be completed before job j_2 can be processed. Let m denote the number of identical machines on which these jobs have to be scheduled. We are also given processing functions for the jobs $p_j : \{0, 1, 2, \ldots, m\} \to \mathbb{R}_0^+$ that map the number of machines assigned to the rate at which the job is processed. (Clearly, $p_j(0) = 0$ for all processing functions.)

The output of the algorithm is a schedule A, which is represented by a continuum of functions $A_t(j)$ over time $t > 0$. $A_t(j)$ represents the number of machines allocated to job j at time t. The schedule must satisfy:

- *Capacity constraints*: For any time $t \in (0, \infty)$, the number of allocated machines at time t is at most m, i.e., $\sum_{j \in J} A_t(j) \leq m$.
- *Precedence constraints*: For any arc $(j_1, j_2) \in E$ and any time $t \in (0, \infty)$, if $A_t(j_2) > 0$, then the job j_1 must be finished by time t, i.e. $\int_0^t p_{j_1}(A_{t'}(j_1))dt' \geq s_{j_1}$. Let the set of jobs that can be processed at a given time (i.e., all their predecessors in G have been completely processed) be called the set of *available* jobs. Then, the precedence constraints enforce that the schedule picks a subset of available jobs to process at any given time.

The *makespan* (or length) of the schedule is defined as the time when all jobs finish processing, i.e., $\ell(A) = \sup\{t : \sum_{j \in J} A_t(j) > 0\}$. The objective of the algorithm is to minimize the makespan of the schedule. We call this the *generalized precedence-constrained scheduling* or GPS problem.

Preemption. It is important to note that we allow *preemption*, i.e., at any point of time, the remaining volume of an available job can be scheduled on any number of machines independent of the history of where it was processed earlier. Therefore, our schedule is defined simply by the number of machines allocated to a job at any given time, and not by the identities of the machines themselves. This is a departure from the bulk of the existing literature in precedence-constrained scheduling with malleable jobs, where preemption is typically disallowed. However, our motivation for allowing preemption comes from the fact that it is allowed in many application domains (such as scheduling in cloud computing) and has been widely considered in the broader scheduling literature.

Processing Functions. Following [17, 22], we consider processing functions that are (1) *non-decreasing* (assigning more machines does not decrease the rate of processing) and (2) *concave*[1] (the processing rate obeys the law of diminishing marginal utility because of greater overhead in coordination, communication costs, etc. between the

[1] We note that it is optimal to process an arbitrary available job on all machines simultaneously if all processing functions are *convex*.

machines processing a job).[2] Note that concave processing functions generalize the classical PS problem ($p_j(z) = 1$ iff $z \geq 1$ for all jobs j).

Integrality of Machines. The existing literature is divided between allowing fractional allocation of machines to jobs (e.g. [22,23]) or enforcing integrality of machine allocations (e.g., [16,17,20]). Accordingly, the processing functions are defined on the entire interval $[0, m]$ or on the discrete values $\{0, 1, \ldots, m\}$. Since we allow job preemption, a fractional assignment of machines to jobs can be realized by a round robin schedule even if there is no inherent support in the application for jobs to share a machine. Therefore, if the processing function is defined only on an integral domain, we extend it to the continuous domain by linear interpolation between adjacent points. In the rest of the paper, we will assume that the processing functions $p_j(.)$ are defined on the continuous domain $[0, m]$ and fractional schedules are valid.

Our Results. Our main result is a $(2+\epsilon)$-approximation algorithm for the GPS problem for concave processing functions. Note that this matches the best known bounds for the PS problem.

Theorem 1.1. *For any $\epsilon > 0$, there is a deterministic algorithm* GPSALGO *for the* GPS *problem that has an approximation factor of $(2 + \epsilon)$ for concave processing functions.*

We note that if preemption is disallowed, then the best approximation ratio known is 3.29 due to Jansen and Zhang [17].

In practice, a particularly relevant set of processing functions are *power functions* (for examples of their practical importance, see, e.g., [22]), i.e., $p_j(z) = c_j \cdot z^\gamma$ for $c_j > 0$. We show that our algorithm is in fact *optimal* for this special case. (Note that (1) power functions do not generalize the PS problem and (2) while the multiplier c_j can depend on the job, the exponent γ in the power functions has to be universal for our analysis.)

Theorem 1.2. *For any $\epsilon > 0$,* GPSALGO *has an approximation factor of $(1 + \epsilon)$ if the processing functions are power functions.*

Our Techniques. It would be natural to try to extend the greedy approach of Graham's algorithm for the PS problem to our problem. The basic scheduling rule of Graham's algorithm is the following: if there is an idle machine and an available job that is currently not being processed, then schedule this job on the machine. Note that this algorithm is *online* in the sense that it can operate on an instance where a job is revealed only after it becomes available. We categorically refute the possibility of extending this greedy approach to the GPS problem by giving a polynomial lower bound on the approximation factor obtained by *any* online algorithm for the GPS problem.

Theorem 1.3. *No online algorithm for the* GPS *problem can have a sub-polynomial competitive ratio, even if all the processing functions are a fixed power function.*

[2] Other classes of processing functions have also been considered in the literature (see related work), but monotonicity and concavity are two basic qualitative features of processing functions in most applications.

Instead, we employ an LP rounding approach (following the work of Chudak and Shmoys [5] and Skutella [26]). In designing the LP relaxation, we introduce a variable x_{ja} denoting the duration for which job j is processed simultaneously by a machines. Then, the processing time for job j is $X_j = \sum_a x_{ja}$. The goal is then to minimize the makespan T subject to the following constraints: (1) the total processing time of jobs on any chain (maximal directed path in the precedence graph) $\sum_{j \in C} X_j$ is a lower bound on the makespan T; (2) the total number of machine-hours for all jobs $\sum_j \sum_a a \cdot x_{ja}$ is a lower bound on mT; and (3) the total processing volume for any single job $\sum_a p_j(a) \cdot x_{ja}$ is at least its size s_j.

First, we solve our LP to obtain optimal values of x_{ja}'s. Next, we structure this solution by showing that $x_{ja} = 0$ for all *except one* value of a for each job j in an optimal solution w.l.o.g. Let us call this value b_j^*. We now create a feasible schedule by using the following simple rule: at any given time, we distribute the available jobs among all the m machines in proportion to their values of b_j^*. The key property that we use in the analysis is the following: (1) if there are too few available jobs (quantified by the sum of b_j^*'s of available jobs being less than m), then the non-decreasing property of the processing function ensures that for every chain, at least one job is being processed faster than in the LP solution, and (2) if there are too many available jobs (quantified by the sum of b_j^*'s of available jobs being greater than m), then the concavity of the processing function ensures that remaining overall ratio of the job volumes and machine-hours is decreasing at a faster rate than in the optimal LP solution. These two observations lead to the conclusion that the makespan of the schedule is at most twice the LP objective.

For the class of power functions, i.e., $p_j(z) = c_j \cdot z^\gamma$ (we only consider $\gamma \in [0, 1]$ since otherwise, the function is convex for which we have already shown that there is a simple optimal algorithm) our LP is exact (up to a factor of $(1 + \epsilon)$ for any $\epsilon > 0$). The main insight is that the special structure of power functions allows us to employ simple linear algebraic inequalities to design a function that trades off the two cases above. More precisely, we show that the gains/losses made by the algorithm over the optimal LP solution for the processing time of chains are exactly compensated by the losses/gains made by it over the LP solution for the overall number of machine-hours in the two situations described above.

Related Work. The precedence-constrained scheduling problem with malleable jobs has a long history in approximation algorithms. Du and Leung [6] showed that the problem is NP-hard even for a small number of machines and gave optimal algorithms if the precedence graph has special structure. Turek *et al* [28] considered the problem of scheduling malleable tasks in the absence of precedence constraints and obtained approximation algorithms for both the preemptive and non-preemptive situations. In the presence of precedence constraints, several families of processing functions have been considered. Our model was originally suggested by Prasanna and Musicus [22–24] and subsequently used by Jansen and Zhang [17], who obtained as approximation factor of 3.29 for the non-preemptive version of our problem. In some papers, the concavity requirement is replaced by a weaker constraint that the size of the jobs increases as more machines are assigned to it [16, 20]. For this model, the best known approximation factor is 4.73 [16]. A third (more general) model that has been considered is that of arbitrary speed up curves. Here, the processing rate not only depends on the number of assigned machines and the job being processed, but also on the stage of processing of a job [8,9]. Most of the literature in this model is geared toward minimizing the flow-time (rather than the makespan) (see e.g. [2,10,11]), including in the presence of precedence constraints [25]. For a detailed survey on scheduling parallelizable jobs, the reader is referred to [7].

Since the work of Graham, both upper bounds (particularly, the trailing $o(1)$ factor in the approximation ratio) (see, e.g., [12,18]) and lower bounds [19,27] for the PS problem have been extensively studied. Moreover, multiple variants of this problem have been considered. This includes optimizing for other metrics such as completion time (see, e.g., [1] and references contained therein), handling machines with non-identical speeds [3,5], dealing with online input (e.g., [15]), etc. For a more detailed history of precedence constrained scheduling, the reader is referred to the surveys of Graham *et al* [14] and Chen *et al* [4].

2 Linear Program

In this section, we give a linear programming relaxation for the problem. In the discrete case, when the optimal solution allocates only an integral number of machines to each jobs, we let $A = \{1, \ldots, m\}$. In the continuous case, when the number of machines can be any real number from $[0, m]$, We pick an $\varepsilon > 0$, and let $A = \{(1-\varepsilon)^k \in [0, m] : k \in \mathbb{Z}\}$. Now for every job $j \in J$ and every value $a \in A$, we introduce a variable x_{ja}. In the intended solution corresponding to the optimal solution of GPS, x_{ja} is equal to the amount of time at which the number of machines used by the job j is between $(1-\varepsilon)a$ and a (in the discrete case, $\varepsilon = 0$). We let T be the makespan of the schedule. Our goal is to minimize T. We write two constraints on T that are satisfied in the optimal solution.

To write the first constraint, we consider an arbitrary chain of jobs C. All jobs $j \in C$ must be processed sequentially one after another. It takes at least $\sum_{a \in A} x_{ja}$ amount of time to finish job j. Thus, for every chain C,

$$T \geq \sum_{j \in C} \sum_{a \in A} x_{ja}. \tag{1}$$

To write the second constraint, we count the number of machine hours used by the optimal solution. On one hand, every job j uses at least $\sum_{a \in A}(1-\varepsilon)ax_{ja}$ machine hours. So the total number of machine hours is lower bounded by $\sum_{j \in J} \sum_{a \in A}(1-\varepsilon)ax_{ja}$. On the other hand, the number of machine hours is upper bounded by mT. So we have

$$mT \geq \sum_{j \in J} \sum_{a \in A}(1-\varepsilon)ax_{ja}. \tag{2}$$

To simplify notation, we let $\tilde{T} = T/(1-\varepsilon)$. We finally add constraint (5) saying that every job j is completed in the optimal solution. We obtain the following LP relaxation.

$$\textbf{minimize } \tilde{T} \quad \textbf{subject to}$$

$$\sum_{j\in C}\sum_{a\in A} x_{ja} \le \tilde{T} \qquad \text{for every chain } C \tag{3}$$

$$\sum_{j\in J}\sum_{a\in A} a\, x_{ja} \le \tilde{T} m \tag{4}$$

$$\sum_{a\in A} x_{ja}\, p_j(a) \ge s_j \qquad \text{for every job } j \in J \tag{5}$$

$$x_{ja} \ge 0 \qquad \text{for all } j \in J,\ a \in A \tag{6}$$

Since the LP solution corresponding to the optimal solution satisfies all the constraints of the linear program, we have $LP \le OPT$, where LP is the cost of the optimal LP solution, and OPT is the cost of the optimal combinatorial solution.

This linear program has infinitely many variables and exponentially many constraints, but using standard methods we can solve this linear program up to any precision $(1+\varepsilon')$ in polynomial-time. For details, the reader is referred to the full version of the paper [21].

3 Simplified LP Solution

It turns out, that every solution to the LP (3–6) can be converted to another simpler solution in which for every job j one and only one x_{ja} is nonzero. Suppose x^*_{ja} is the optimal solution to the LP (3–6), define y^*_j's and b^*_j's as follows:

$$y^*_j = \sum_{a\in A} x^*_{ja}; \quad b^*_j = \frac{1}{y^*_j}\sum_{a\in A} x^*_{ja} a. \tag{7}$$

Claim 1 *Variables y^*_j and b^*_j satisfy the following constraints (similar to (3–5)):*

$$\sum_{j\in C} y^*_j \le \tilde{T} \qquad \textit{for every chain } C \tag{3$'$}$$

$$\sum_{j\in J} b^*_j y^*_j \le \tilde{T} m \tag{4$'$}$$

$$y^*_j\, p_j(b^*_j) \ge s_j \qquad \textit{for every job } j \in J \tag{5$'$}$$

Proof. For every chain C, we have $\sum_{j\in C} y^*_j = \sum_{j\in C}\sum_{a\in A} x^*_{ja} \le \tilde{T}$. Then,

$$\sum_{j\in J} b^*_j y^*_j = \sum_{j\in J} y^*_j \cdot \frac{1}{y^*_j}\sum_{a\in A} x^*_{ja} = \sum_{j\in J}\sum_{a\in A} x^*_{ja} \le \tilde{T} m.$$

Finally, for every j, we have $y^*_j\, p_j(b^*_j) = y^*_j\, p_j\Big(\frac{\sum_{a\in A} x^*_{ja} a}{\sum_{a\in A} x^*_{ja}}\Big)$. Let $\lambda_{ja} = x^*_{ja} / \sum_{a\in A} x^*_{ja}$. Then, $\sum_{a\in A}\lambda_{ja} = 1$ for every j. From concavity of the function $p_j(\cdot)$, we have

$$y^*_j\, p_j(b^*_j) = y^*_j\, p_j\Big(\sum_{a\in A}\lambda_{ja} a\Big) \ge y^*_j \sum_{a\in A}\lambda_{ja} p_j(a) = \sum_{a\in A} x^*_{ja} p_j(a) \ge s_j.$$

We can further assume that all constraints (5′) are tight i.e., for every j, we have $y_j^* \, p_j(b_j^*) = s_j$. Indeed, if (5′) is not tight for some j, then we can decrease y_j^* by letting $y_j^* = s_j / p_j(b_j^*)$.

4 Algorithm

We now describe the approximation algorithm. We first solve the LP relaxation and obtain a solution x_{ja}^*. Using Claim 1, we convert this solution to the solution (y_j^*, b_j^*) of the simplified LP (3′-5′). We assume that all constraints (5′) are tight (see above). Then we start the "rounding" procedure.

We schedule jobs iteratively. In every iteration, we schedule the next batch of jobs in the interval $[t, t + \varDelta t]$ and then advance time from t to $t + \varDelta t$. Thus, at the beginning of every iteration, we already have a schedule for the time interval $[0, t]$. For every job j, we keep the remaining size of j in the variable $s_j^*(t)$. Initially, $s_j^*(0) = s_j$. We also update the LP solution: we maintain variables $y_j^*(t)$ that indicate the time required by the remaining portion of job j if b_j^* machines are allotted to it. In other words, we maintain the invariant:

$$y_j^*(t) \cdot p_j(b_j^*) = s_j^*(t). \tag{8}$$

Initially, $y_j^*(0) = y_j^*$. Hence, for $t = 0$, this invariant holds.

To schedule the next batch of jobs, we find all unfinished jobs that can be scheduled now without violating precedence constraints. We call these *available* jobs. We denote the set of all available jobs at time t by $\varLambda(t)$. For every available job $j \in \varLambda(t)$ we compute

$$m_j^*(t) = m \cdot \frac{b_j^*}{\sum_{j \in \varLambda(t)} b_j^*}.$$

We allocate $m_j^*(t)$ machines to job j for the time interval of length

$$\varDelta t = \min_{j \in \varLambda(t)} \frac{s_j^*(t)}{p_j(m_j^*(t))}.$$

Observe that the total number of machines we allocate is m. For all $j \in \varLambda(j)$, we update $s_j^*(t + \varDelta t)$ and $y_j^*(t + \varDelta t)$:

$$s_j^*(t + \varDelta) = s_j^*(t) - p_j(m_j^*(t))\,\varDelta t \tag{9}$$

$$y_j^*(t + \varDelta t) = y_j^*(t) - \frac{p_j(m_j^*(t))}{p_j(b_j^*)}\,\varDelta t. \tag{10}$$

Note that this maintains invariant (8). We set $t = t + \varDelta t$ and proceed to the next iteration. The algorithm terminates when $s_j^*(t) = 0$ for all j.

5 Analysis

We now analyze the algorithm. First, observe that the algorithm correctly maintains the remaining sizes $s_j^*(t)$: at time t, the remaining size of the job j is indeed $s_j^*(t)$.

Note that all $s_j^*(t)$ remain nonnegative (that is how we pick Δt). Moreover, at the end of every iteration one of the available jobs, specifically, the job j' for which $\Delta t = s_{j'}^*(t)/p_{j'}(m_{j'}^*(t))$ (again see the definition of Δt), is completed, i.e., $s_{j'}^*(t+\Delta t) = 0$. So the number of iterations of the algorithm is at most n, and the running time of the algorithm is polynomial in n. Also, note that all $y_j^*(t)$ are nonnegative by (8).

We now need to upper bound the makespan of the schedule produced by the algorithm. We prove the following standard lemma.

Lemma 5.1. *There exists a chain of jobs C^* such that at every point of time t one and only one job from C^* is scheduled by the algorithm.*

Proof. Consider the job j that finished last in the schedule generated by the algorithm. We add this job to our chain. This job was not scheduled earlier because it depends on some other job j' that finished just before j started. We add j' to our schedule as well. We then pick the job j' depends on, and so on. We continue this process until we encounter a job that does not depend on any other job. This job started at time $t = 0$. Thus, the jobs in the constructed chain cover the time line from the beginning to the end of the schedule.

We now show that for every t, the following inequality holds,

$$\sum_{j\in C^*} y_j^*(t) + \frac{1}{m}\sum_{j\in J} b_j^* y_j^*(t) \le 2\tilde{T} - t. \tag{11}$$

Note that for $t = 0$, the inequality follows from (3′) and (4′). This inequality implies that the makespan is at most $2\tilde{T} \le 2(1+\varepsilon)T$, since all $y_j^*(t)$ are nonnegative and thus the left hand side of (11) is nonnegative.

Lemma 5.2. *Inequality (11) holds in the beginning and end of every iteration.*

Proof. We assume that (11) holds at time t at the beginning of some iteration and prove that (11) holds at time $t+\Delta t$ at the end of this iteration. In an iteration, the RHS of (11) decreases by Δt. Our goal is to show that one of the following happens in any iteration:

- **Condition (a):** $\sum_{j\in C^*} y_j^*(t)$ (the first term in the LHS of (11)) decreases by at least Δt, or
- **Condition (b):** $\frac{1}{m}\sum_{j\in J} b_j^* y_j^*(t)$ (the second term in the LHS of (11)) decrease by at least Δt.

Since both the terms in the LHS of (11) are non-increasing, the lemma follows.

By Lemma 5.1, the algorithm schedules exactly one job in the chain C^* in the time interval $[t, t+\Delta t]$. We denote this job by j'. By equation (10), we have

$$y_{j'}^*(t+\Delta t) = y_{j'}^*(t) - \frac{p_{j'}(m_{j'}^*(t))}{p_{j'}(b_{j'}^*)}\,\Delta t = y_{j'}^*(t) - \frac{p_{j'}\left(\frac{m\, b_{j'}^*}{\sum_{j\in\Lambda(t)} b_j^*}\right)}{p_{j'}(b_{j'}^*)}\,\Delta t.$$

Denote $\alpha(t) = \sum_{j\in\Lambda(t)} b_j^*$. Rewrite the expression above as follows:

$$y_{j'}^*(t+\Delta t) = y_{j'}^*(t) - \frac{p_{j'}\left(b_{j'}^* \cdot m/\alpha(t)\right)}{p_{j'}(b_{j'}^*)}\,\Delta t. \tag{12}$$

Case 1: $\alpha(t) \le m$: Since $p_{j'}(\cdot)$ is a non-decreasing function and $\alpha(t) \le m$, we have $p_{j'}\big(b^*_{j'} \cdot m/\alpha(t)\big) \ge p_{j'}(b^*_{j'})$. Using this fact in (12), we have

$$\sum_{j \in C^*} y^*_j(t + \varDelta t) \le \sum_{j \in C^*} y^*_j(t) - \varDelta t.$$

Therefore, condition (a) holds in this case.
Case 2: $\alpha(t) \ge m$: We estimate the second term in the LHS of (11). Using (10), we have

$$\begin{aligned}\frac{1}{m}\sum_{j \in \varLambda(t)} b^*_j y^*_j(t + \varDelta t) &= \frac{1}{m}\sum_{j \in \varLambda(t)} b^*_j\Big(y^*_j(t) - \frac{p_j\big(b^*_j\, m/\alpha(t)\big)}{p_j(b^*_j)}\,\varDelta t\Big)\\ &= \frac{1}{m}\sum_{j \in \varLambda(t)} b^*_j y^*_j(t) - \frac{1}{m}\sum_{j \in \varLambda(t)} b^*_j\,\frac{p_j\big(b^*_j\, m/\alpha(t)\big)}{p_j(b^*_j)}\,\varDelta t.\end{aligned}$$

Since $\alpha(t) \ge m$, we have $p_j\big(b^*_j\, m/\alpha(t)\big) \ge (m/\alpha(t)) \cdot p_j(b^*_j)$, since $p_j(\cdot)$ is a concave function with $p_j(0) = 0$. Thus,

$$\begin{aligned}&\frac{1}{m}\sum_{j \in \varLambda(t)} b^*_j y^*_j \cdot (t + \varDelta t) \le \frac{1}{m}\sum_{j \in \varLambda(t)} b^*_j y^*_j(t) - \frac{1}{m}\sum_{j \in \varLambda(t)} b^*_j\,\frac{m p_j(b^*_j)}{\alpha(t) p_j(b^*_j)}\,\varDelta t\\ &= \frac{1}{m}\sum_{j \in \varLambda(t)} b^*_j y^*_j(t) - \sum_{j \in \varLambda(t)} \frac{b^*_j}{\alpha(t)}\,\varDelta t = \frac{1}{m}\sum_{j \in \varLambda(t)} b^*_j y^*_j(t) - \varDelta t,\end{aligned}$$

where the last equation follows from the definition of $\alpha(t)$. Therefore, condition (b) holds in this case.

Combining the two cases, no matter whether $\alpha(t) \le m$ or $\alpha(t) \ge m$,

$$\sum_{j \in C^*} y^*_j(t + \varDelta t) + \frac{1}{m}\sum_{j \in J} b^*_j y^*_j(t + \varDelta t) \le \sum_{j \in C^*} y^*_j(t) + \frac{1}{m}\sum_{j \in J} b^*_j y^*_j(t) - \varDelta t.$$

This completes the proof.

6 Analysis for Power Functions

We now analyze the algorithm for power functions, i.e., $p_j(z) = c_j \cdot z^\gamma$ for some $\gamma \le 1$ and constants $c_j > 0$. Let $\delta = 1 - \gamma$. We now show that for every t, the following inequality holds:

$$\left(\sum_{j \in C^*} y^*_j(t)\right)^{\delta} \cdot \left(\frac{1}{m}\sum_{j \in J} b^*_j y^*_j(t)\right)^{\gamma} \le \tilde{T} - t. \tag{13}$$

Note that for $t = 0$, the inequality follows from (3′) and (4′). Inequality (13) implies that the makespan is at most $\tilde{T}$: all $y^*_j(t)$ are nonnegative, hence the left hand side of inequality (13) is also nonnegative, consequently $t \le \tilde{T}$. Our main technical tool will be the following fact, which is an easy consequence of Hölder's inequality.

Fact 1. *Suppose* $X, Y \geq 0$ *and* $\gamma, \delta \in [0,1]$ *such that* $\gamma + \delta = 1$. *For any* $\Delta X \in [0, X]$, $\Delta Y \in [0, Y]$, *define* $\Delta\left(X^{\delta} \cdot Y^{\gamma}\right) = X^{\delta} \cdot Y^{\gamma} - (X - \Delta X)^{\delta} \cdot (Y - \Delta Y)^{\gamma}$. *Then,*

$$\Delta\left(X^{\delta} \cdot Y^{\gamma}\right) \geq (\Delta X)^{\delta} \cdot (\Delta Y)^{\gamma}.$$

Proof. Define vectors $\mathbf{f} = ((X - \Delta X)^{\delta}, (\Delta X)^{\delta})$ and $\mathbf{g} = ((Y - \Delta Y)^{\gamma}, (\Delta Y)^{\gamma})$. Then, by Hölder's inequality, we have

$$\langle \mathbf{f}, \mathbf{g} \rangle \leq \|\mathbf{f}\|_{1/\delta} \|\mathbf{g}\|_{1/\gamma} \Rightarrow \quad (X - \Delta X)^{\delta} \cdot (Y - \Delta Y)^{\gamma} + (\Delta X)^{\delta} \cdot (\Delta Y)^{\gamma} \leq X^{\delta} \cdot Y^{\gamma}.$$

The lemma follows by rearranging terms.

Using this fact, we inductively prove that inequality (13) holds throughout the rounding algorithm.

Lemma 6.1. *Inequality (13) holds at the beginning and end of every iteration.*

Proof. We assume that (11) holds at time t at the beginning of some iteration and prove that (11) holds at time $t + \Delta t$ at the end of this iteration. By Lemma 5.1, the algorithm schedules exactly one job in the chain C^* in the time interval $[t, t + \Delta t]$. We denote this job by j'. We have

$$y^*_{j'}(t + \Delta t) = y^*_{j'}(t) - \frac{p_{j'}(m^*_{j'}(t))}{p_{j'}(b^*_{j'})} \Delta t = y^*_{j'}(t) - \frac{p_{j'}\left(\frac{m\, b^*_{j'}}{\sum_{j \in \Lambda(t)} b^*_j}\right)}{p_{j'}(b^*_{j'})} \Delta t.$$

Denote $\alpha(t) = \sum_{j \in \Lambda(t)} b^*_j$. Rewrite the expression above as follows:

$$y^*_{j'}(t + \Delta t) = y^*_{j'}(t) - \frac{p_{j'}\Big(b^*_{j'} \cdot m/\alpha(t)\Big)}{p_{j'}(b^*_{j'})} \Delta t = y^*_{j'}(t) - \left(\frac{m}{\alpha(t)}\right)^{\gamma} \Delta t.$$

We estimate the second term in (13).

$$\begin{aligned}
&\frac{1}{m} \sum_{j \in \Lambda(t)} b^*_j y^*_j \cdot (t + \Delta t) = \frac{1}{m} \sum_{j \in \Lambda(t)} b^*_j \Big(y^*_j(t) - \frac{p_j\big(b^*_j\, m/\alpha(t)\big)}{p_j(b^*_j)} \Delta t\Big) \\
&= \frac{1}{m} \sum_{j \in \Lambda(t)} b^*_j y^*_j(t) - \frac{1}{m} \sum_{j \in \Lambda(t)} b^*_j \frac{p_j\big(b^*_j\, m/\alpha(t)\big)}{p_j(b^*_j)} \Delta t = \frac{1}{m} \sum_{j \in \Lambda(t)} b^*_j y^*_j(t) - \left(\frac{\alpha(t)}{m}\right)^{\delta} \Delta t.
\end{aligned}$$

Using Fact 1, we have

$$\begin{aligned}
&\Delta\Big(\Big(\sum_{j \in C^*} y^*_j(t)\Big)^{\delta} \cdot \Big(\frac{1}{m} \sum_{j \in J} b^*_j y^*_j(t)\Big)^{\gamma}\Big) \geq \Big(\Delta\Big(\sum_{j \in C^*} y^*_j(t)\Big)\Big)^{\delta} \cdot \Big(\Delta\Big(\frac{1}{m} \sum_{j \in J} b^*_j y^*_j(t)\Big)\Big)^{\gamma} \\
&= \quad \Big(\Big(\frac{m}{\alpha(t)}\Big)^{\gamma} \Delta t\Big)^{\delta} \cdot \Big(\Big(\frac{\alpha(t)}{m}\Big)^{\delta} \Delta t\Big)^{\gamma} = \Delta t.
\end{aligned}$$

Note that when we move from time t to time $t + \Delta t$, the right hand side of (11) decreases by Δt. This completes the proof.

7 Lower Bound for Online Algorithms

In this section, we show that the GPS problem has a polynomial lower bound in the online setting. Specifically, we show that even if $p_j(z) = \sqrt{z}$ for all jobs j, the competitive ratio of any online algorithm is at least $\Omega(n^{1/4})$, where n is the number of jobs. Note that in this case our offline algorithm gives an almost exact solution ($(1 + \varepsilon)$ approximation for arbitrary $\varepsilon > 0$). In the online model, a job is given to the algorithm only once all the jobs it depends on are finished. So this result shows that any approximation algorithm should use the information about the future schedule and cannot make the decision solely based on the set of currently available jobs.

We describe the strategy for the adversary. The adversary works in phases. In phase i, she presents l independent jobs $u_{i1}, \ldots, u_{il}$ to the algorithm. These jobs depend on the single job in the previous phase that has finished last. That is, if u_{ij_i} is the job that finished last in the phase i, then in the next phase $(i + 1)$, all jobs $u_{(i+1)s}$ depend on this job u_{ij_i}. We assume that the number of machines is $m = 1$.

We lower bound the makespan of the schedule produced by the online algorithm. To finish all l jobs given to the algorithm in one phase, we need to spend time at least $\sqrt{l}$. (The optimal way to allocate machines is to assign $1/l$ machines to each job.) Thus, the total length of the schedule is at least $k\sqrt{l}$.

Now consider the following solution. Initially, all jobs in the chain $v_{1j_1}, v_{2j_2}, \ldots, v_{kj_k}$ are scheduled sequentially (assigning 1 machine to each job). Once all jobs v_{ij_i} are finished, the remaining $kl - k$ jobs are scheduled in parallel by assigning $1/(kl - l)$ machines to every job. The length of the schedule equals $k + \sqrt{kl - k}$. The lower bound now follows by setting $k = l$ (note that $n = kl$). This lower bound can be extended to randomized algorithms using standard techniques, which we omit for brevity.

We note that this lower bound is almost tight for $p_j(z) = \sqrt{z}$: any algorithm that does not idle (i.e., which always allocates all available machines) has competitive ratio at most $\sqrt{n}$, since the maximum possible rate of processing all jobs is $\sqrt{n}$ (when we process all n jobs in parallel) and the minimum possible rate is 1 (when we allocate all machines to a single job).

References

1. Bansal, N., Khot, S.: Optimal long code test with one free bit. In: FOCS, pp. 453–462 (2009)
2. Chan, H.-L., Edmonds, J., Pruhs, K.: Speed scaling of processes with arbitrary speedup curves on a multiprocessor. Theory Comput. Syst. 49(4), 817–833 (2011)
3. Chekuri, C., Bender, M.A.: An efficient approximation algorithm for minimizing makespan on uniformly related machines. J. Algorithms 41(2), 212–224 (2001)
4. Chen, B., Potts, C.N., Woeginger, G.J.: A review of machine scheduling: Complexity, algorithms and approximability. Handbook of Combinatorial Optimization 3, 21–169 (1998)
5. Chudak, F.A., Shmoys, D.B.: Approximation algorithms for precedence-constrained scheduling problems on parallel machines that run at different speeds. J. Algorithms 30(2), 323–343 (1999)
6. Du, J., Leung, J.Y.-T.: Scheduling tree-structured tasks on two processors to minimize schedule length. SIAM J. Discrete Math. 2(2), 176–196 (1989)
7. Dutot, P.-F., Mounie, G., Trystram, D.: Scheduling parallel tasks approximation algorithms. In: Handbook of Scheduling: Algorithms, Models, and Performance Analysis (2004)

8. Edmonds, J.: Scheduling in the dark. Theor. Comput. Sci. 235(1), 109–141 (2000)
9. Edmonds, J., Chinn, D.D., Brecht, T., Deng, X.: Non-clairvoyant multiprocessor scheduling of jobs with changing execution characteristics. J. Scheduling 6(3), 231–250 (2003)
10. Edmonds, J., Pruhs, K.: Scalably scheduling processes with arbitrary speedup curves. ACM Transactions on Algorithms 8(3), 28 (2012)
11. Fox, K., Im, S., Moseley, B.: Energy efficient scheduling of parallelizable jobs. In: SODA, pp. 948–957 (2013)
12. Gangal, D., Ranade, A.G.: Precedence constrained scheduling in (2 - 7/(3p+1)) optimal. J. Comput. Syst. Sci. 74(7), 1139–1146 (2008)
13. Graham, R.L.: Bounds for certain multiprocessing anomalies. Siam Journal on Applied Mathematics (1966)
14. Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics 5(2), 287–326 (1979)
15. Huo, Y., Leung, J.Y.-T.: Online scheduling of precedence constrained tasks. SIAM J. Comput. 34(3), 743–762 (2005)
16. Jansen, K., Zhang, H.: An approximation algorithm for scheduling malleable tasks under general precedence constraints. ACM Transactions on Algorithms 2(3), 416–434 (2006)
17. Jansen, K., Zhang, H.: Scheduling malleable tasks with precedence constraints. J. Comput. Syst. Sci. 78(1), 245–259 (2012)
18. Lam, S., Sethi, R.: Worst case analysis of two scheduling algorithms. SIAM J. Comput. 6(3), 518–536 (1977)
19. Lenstra, J.K., Kan, R.A.H.G.: Complexity of Scheduling under Precedence Constraints. Operations Research 26(1), 22–35 (1978)
20. Lepère, R., Trystram, D., Woeginger, G.J.: Approximation algorithms for scheduling malleable tasks under precedence constraints. Int. J. Found. Comput. Sci. 13(4), 613–627 (2002)
21. Makarychev, K., Panigrahi, D.: Precedence-constrained scheduling of malleable jobs with preemption. CoRR, abs/1404.6850 (2014)
22. Srinivasa Prasanna, G.N., Musicus, B.R.: Generalised multiprocessor scheduling using optimal control. In: SPAA, pp. 216–228 (1991)
23. Srinivasa Prasanna, G.N., Musicus, B.R.: Generalized multiprocessor scheduling for directed acyclic graphs. In: SC, pp. 237–246 (1994)
24. Srinivasa Prasanna, G.N., Musicus, B.R.: The optimal control approach to generalized multiprocessor scheduling. Algorithmica 15(1), 17–49 (1996)
25. Robert, J., Schabanel, N.: Non-clairvoyant scheduling with precedence constraints. In: SODA, pp. 491–500 (2008)
26. Skutella, M.: Approximation algorithms for the discrete time-cost tradeoff problem. In: SODA, pp. 501–508 (1997)
27. Svensson, O.: Hardness of precedence constrained scheduling on identical machines. SIAM J. Comput. 40(5), 1258–1274 (2011)
28. Turek, J., Wolf, J.L., Yu, P.S.: Approximate algorithms scheduling parallelizable tasks. In: SPAA, pp. 323–332 (1992)

Unbounded Entanglement Can Be Needed to Achieve the Optimal Success Probability

Laura Mančinska[1] and Thomas Vidick[2]

[1] Centre for Quantum Technologies, National University of Singapore
[2] Simons Institute, University of California at Berkeley

Abstract. Quantum entanglement is known to provide a strong advantage in many two-party distributed tasks. We investigate the question of how much entanglement is needed to reach optimal performance. For the first time we show that there exists a purely classical scenario for which no finite amount of entanglement suffices. To this end we introduce a simple two-party nonlocal game H, inspired by a paradox of Lucien Hardy. In our game each player has only two possible questions and can provide answers in a countable set. We exhibit a sequence of strategies which use entangled states in increasing dimension d and succeed with probability $1 - O(d^{-c})$ for some $c \geq 0.13$. On the other hand, we show that any strategy using an entangled state of local dimension d has success probability at most $1 - \Omega(d^{-2})$. In addition, we show that any strategy restricted to producing answers in a set of cardinality at most d has success probability at most $1 - \Omega(d^{-2})$.

Keywords: nonlocal game, value of the game, entanglement, dimension witness.

1 Introduction

Entanglement plays a key role in quantum information processing. The almost unnaturally strong correlations it implies were initially seen as a weird, if not undesirable [10], feature of quantum mechanics. Yet more recently entanglement is increasingly being thought of as a distributed resource that allows cooperating parties to accomplish otherwise impossible tasks, such as unconditionally secure cryptography [11], randomness certification [8,25] and expansion [29,9], or classical communication with improved efficiency [2]. All these tasks are purely classical scenarios in which the use of shared entanglement provides a strong advantage. It is thus natural to ask how much of this new resource is needed in order to achieve optimal performance. The question arises in areas such as quantum Shannon theory [31], communication complexity [6], nonlocality [4], and many others. Perhaps surprisingly, very few general results are known. Concrete examples have been used to establish lower bounds on the amount of entanglement required. Yet the problem of proving upper bounds is a recurrent sticking point and few such bounds are known; two-player XOR games provide a rare exception [7].

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 835–846, 2014.

This unfortunate state of affairs has led to the question of whether such bounds exist in principle. The question can be abstractly formulated as follows:

> *Given a classical distributed task $\mathcal{T}$ of complexity $\mathcal{C}$, is it the case that $\mathcal{T}$ can be completed with vanishing error probability using entangled states of local dimension $d(\mathcal{C})$, where $d(\mathcal{C}) \in \mathbb{Z}^+$ is a finite bound depending on $\mathcal{C}$ only?* $(\star)$

The key aspect of this question is whether any fixed task $\mathcal{T}$ can be associated a *finite* complexity in terms of its dependence on the use of entanglement. If this cannot be done, we may consider a less demanding finite-precision variant of the above question. That is, we could require to achieve $\mathcal{T}$ *approximately*, within some finite precision ε, and allow $d = d(\mathcal{C}, \varepsilon)$ to depend on ε as well. (For now we are purposefully leaving terms such as complexity, precision, etc., loosely defined; they will be made more concrete in the coming paragraphs.)

We study both questions in the context of nonlocal games. In such games, two separated parties are provided with questions x and y respectively, chosen according to a pre-specified distribution $\pi(x, y)$. Without communicating they must respectively provide answers a and b. The task $\mathcal{T}$ is to maximize the probability that their answers satisfy a pre-determined criterion $V(a, b|x, y) = 1$. The entangled value, $\omega^*(G)$, of such a game $G = (V, \pi)$ is the largest success probability achievable using finite-dimensional entangled states (see Section 2 for precise definitions). It is known that for any $d \in \mathbb{Z}^+$ there exists a nonlocal game for which any strategy attaining the optimal success probability requires measurements on a shared state with local dimension at least d [1,30,3]. Thus the upper bound $d(\mathcal{C})$ on the entanglement needed cannot be a universal constant, *i.e.*, its value must depend on some measure of complexity of the game.

Our results. We introduce a nonlocal game H for which the answer to question $(\star)$ is negative. While $\omega^*(H) = 1$, we show that this success probability can only be achieved in the limit of strategies using entangled states of increasing dimension. More precisely, the game H is such that for any $\varepsilon > 0$,

- H can be won with probability $1 - \varepsilon$ using a shared state of local dimension $O(1/\varepsilon^{7.3\ldots})$;
- any strategy that wins H with probability $1-\varepsilon$ uses a state of local dimension $\Omega(1/\sqrt{\varepsilon})$.

See Theorem 1 for a precise statement. The game H is inspired by Hardy's paradox [15,16]; it has two questions per party and countably infinite answer sets (see Section 3.1 for a brief review of Hardy's paradox, and Section 3.2 for a complete description of the game). We also show that in order to win H with probability $1 - \varepsilon$ the quantum players must use a strategy that assigns positive probability to at least $\Omega(1/\sqrt{\varepsilon})$ distinct answers per party (see Theorem 2).

Our result has some bearing on the computational complexity of computing the entangled value $\omega^*(G)$ of a general nonlocal game G. If the input size is defined to be the total number of questions and answers in G, then $\omega^*(G)$ is known to be NP-hard to compute [17]. The problem is not known to be in NP however.

In fact, no non-trivial upper bounds on its complexity are currently known—it is even unknown whether it can be decided if $\omega^*(G) = 1$. The only known decidability result is for a related parameter $\omega^*_{FV}(G)$, the field-theoretic value of G. Here the tensor-product condition on Alice's and Bob's strategy is relaxed to a commutativity requirement (due to this relaxation, $\omega^*(G) \leq \omega^*_{FV}(G)$ for any game G). The parameter ω^*_{FV} is computable provided that the optimal commuting strategy is finite-dimensional [30,23]. In such a case $\omega^*(G) = \omega^*_{FV}(G)$ since any finite-dimensional commuting strategy can be used to construct a tensor-product strategy with the same success probability. Equality in the general case would follow from a positive resolution of Tsirelson's conjecture (see *e.g.* [12] for a formulation and discussion of the conjecture).

Our results concerning the game H show that no a priori upper bound on the dimension of optimal strategies exists for games with finite question and countable answer sets. Hence, in this case the "naïve" algorithm that computes $\omega^*(G)$ by performing an exhaustive search over all possible strategies in increasing dimension (and to within increasing precision) will keep finding strategies with increasing success probability. Of course, in practice one will be content with a good approximation to $\omega^*(G)$. In this case one can interpret our result as placing a lower bound, depending on the desired approximation, on the dimension in which the search must be performed. The game H can thus be used as a "dimension witness", *for any dimension*: strategies achieving success probability at least $\omega^*(H) - \varepsilon$ must necessarily use entanglement of local dimension $d = \Omega(1/\sqrt{\varepsilon})$. Examples of such constructions are already known [5,27,3], but they all require game with increasing numbers of questions in order to witness increasing dimensions. In particular, Slofstra [27] provides an n-question two-answer XOR game G_n for which achieving $\omega^*(G) - \varepsilon$ requires dimension $\min(2^{\Omega(\sqrt{n})}, \Omega(\varepsilon^{-1/12}))$. Briët et al. [3] provide an $(1/\varepsilon)^{\Theta(1/\varepsilon)}$-question two-answer XOR game G_ε for which the dependence on ε is $\Omega(1/\varepsilon)$. To avoid having infinitely many possible answers, we can limit our game to answers of length $\ell = \Theta(\log 1/\varepsilon)$. This gives a game H_ℓ with two questions and poly$(1/\varepsilon)$ answers. To win H_ℓ with probability $\omega^*(H_\ell) - \varepsilon$, a state of dimension at least $\Omega(1/\sqrt{\varepsilon})$ is needed (see Corollary 1).

Related Work. Prior to our work two examples of nonlocal games were known for which the optimal success probability can only be approached in the limit of infinite-dimensional shared entanglement. Leung et al. [21] consider a game with quantum questions and answers, which has inspired Regev and Vidick [26] to produce a game with quantum questions but classical answers. The latter game has two possible 3 × 3-dimensional states as questions, and two possible answers per player. Our game provides the first example with classical questions and answers, thus resolving a question first formally asked in [23]. The same question was also asked [30], but specifically for games with finite question and answer sets. Although there is some numerical evidence that infinite-dimensional entanglement may be needed in that case as well [24], the question remains tantalizingly open. Interestingly all of these examples, including the one presented in this paper, have nearly-optimal strategies that seem to crucially rely on embezzlement [18,28].

Organization of the Paper. The remainder of the paper is organized as follows. In Section 2 we formally define the classical and entangled values of a nonlocal game. In Section 3 we review Hardy's paradox, introduce the game H, and show that the classical value of H is 3/4 while its entangled value equals 1. In Section 4 we establish our main result by showing that restricting the dimension of shared entanglement bounds the achievable success probabilities away from one (see Theorem 1). We also show that restricting the number of different answers has the same effect (see Theorem 2). We conclude in Section 5.

2 Preliminaries

In this section we explain the terminology used to discuss one-round two-player nonlocal games. A reader familiar with nonlocal games is encouraged to proceed directly to the next section.

A two-party nonlocal game G consists of a probability distribution π over a set of the form $\mathcal{I}_A \times \mathcal{I}_B$, called the set of questions; sets of answers $\mathcal{O}_A$ and $\mathcal{O}_B$, and a verification function $V : \mathcal{O}_A \times \mathcal{O}_B \times \mathcal{I}_A \times \mathcal{I}_B \to \{0, 1\}$. We only consider the case where the sets $\mathcal{I}_A, \mathcal{I}_B$ are countable. With probability $\pi(x, y)$ the referee sends the two players, traditionally called Alice and Bob, questions $x \in \mathcal{I}_A$ and $y \in \mathcal{I}_B$, respectively. Without communicating, the players must produce answers $a \in \mathcal{O}_A$ and $b \in \mathcal{O}_B$, respectively. They win if $V(a, b, x, y) = V(a, b|x, y) = 1$ and lose otherwise. Classical players can use shared randomness to enhance their strategy while quantum players can use shared entanglement. The goal of the players is to maximize their probability of winning. In case of classical players, we call this probability the *(classical) value* of game G, denoted $\omega(G)$. In the quantum case we call it the *entangled value* of G, denoted $\omega^*(G)$. Since quantum players are at least as powerful as classical ones, $\omega(G) \leq \omega^*(G)$ for any game G.

2.1 Classical Strategies and Value

Any classical strategy for a game G can be specified using a pair functions

$$\alpha : \mathcal{I}_A \times R \to \mathcal{O}_A \quad \text{and} \quad \beta : \mathcal{I}_B \times R \to \mathcal{O}_B, \tag{1}$$

where R consists of the possible values of shared randomness (which, without loss of generality, also includes any private randomness). We define $\alpha(x, r)$ to be Alice's answer on question x given that the shared randomness takes value r. We define $\beta(y, r)$ similarly. If the shared randomness is distributed according to $\tau : R \to [0, 1]$, then the classical value of the game is

$$\omega(G) := \sup_{\alpha,\beta} \omega(G|\alpha, \beta) := \sup_{\alpha,\beta} \sum_r \tau(r) \sum_{x,y} \pi(x, y) V\Big(\alpha_r(x), \beta_r(y)|x, y\Big), \tag{2}$$

where $\alpha_r(x) := \alpha(x, r)$ and $\beta_r(y) := \beta(y, r)$. Note that for any $r \in R$ the pair (α_r, β_r) specifies a deterministic strategy. Since $\omega(G|\alpha, \beta)$ is a convex combination of $\omega(G|\alpha_r, \beta_r)$, there exists some $r \in R$ for which $\omega(G|\alpha_r, \beta_r) \geq \omega(G|\alpha, \beta)$.

Therefore, it suffices to consider only deterministic strategies (α, β), *i.e.*,

$$\omega(G) = \sup_{\substack{\alpha:\mathcal{I}_A\to\mathcal{O}_A \\ \beta:\mathcal{I}_B\to\mathcal{O}_B}} \sum_{x,y} \pi(x,y) V\big(\alpha(x), \beta(y)|x,y\big). \tag{3}$$

In the full version of this paper [22] we show that if a game either has finitely many questions, or finitely many answers, then the classical value can always be achieved exactly, *i.e.*, the supremum in (3) can be replaced with a maximum. We also give an example showing that this not always the case if both question and answer sets are infinite.

2.2 Quantum Strategies and Value

A quantum strategy is specified by a finite-dimensional shared state $|\psi\rangle \in \mathbb{C}^{d_A} \otimes \mathbb{C}^{d_B}$, a POVM $\mathcal{A}^x = \{A^x_a : a \in \mathcal{O}_A\}$ for every question $x \in \mathcal{I}_A$ to Alice, and a POVM $\mathcal{B}^y = \{B^y_b : b \in \mathcal{O}_B\}$ for every question $y \in \mathcal{I}_B$ to Bob. Upon receiving questions x and y, the parties measure their systems with POVMs $\mathcal{A}^x$, $\mathcal{B}^y$ respectively and answer with their respective measurement outcomes a and b. The quantum value of the game is given by

$$\omega^*(G) = \sup \sum_{x,y} \pi(x,y) V(a,b|x,y) \langle\psi| \big(A^x_a \otimes B^y_b\big) |\psi\rangle, \tag{4}$$

where the supremum is taken over all finite-dimensional shared states $|\psi\rangle$ and sets of POVMs $\{\mathcal{A}^x\}_x$ and $\{\mathcal{B}^y\}_y$.

3 Hardy's Game

We introduce a nonlocal game that we call Hardy's game, as it is based on Hardy's paradox [15,16]. We first describe the paradox, then the game we derive from it, and end this section by exhibiting a sequence of good strategies for quantum players of this game.

3.1 Hardy's Paradox

Hardy's paradox is a two-party experimental setup whose outcomes, as predicted by quantum mechanics, cannot be reproduced by any local hidden variables theory. The setup is noteworthy for being minimal in a sense that it involves only two qubits, on which both of the parties perform two measurements with two outcomes each. Our description of the setup roughly follows the explanations given in [20].

Suppose Alice and Bob share a two-qubit state

$$|\psi\rangle := \frac{\sin(\theta)|v'_0, v'_0\rangle - \cos(\theta)\big(|v'_0, v'_1\rangle + |v'_1, v'_0\rangle\big)}{\sqrt{1+\cos^2(\theta)}}, \tag{5}$$

where $\mathcal{S}' = \{|v_0'\rangle, |v_1'\rangle\} \subseteq \mathbb{R}^2$ is any orthonormal basis and $0 < \theta < \frac{\pi}{2}$. Let $\mathcal{S} = \{|v_0\rangle, |v_1\rangle\} \subseteq \mathbb{R}^2$ be the basis obtained by rotating $\mathcal{S}'$ by angle 2θ in the xz plane of the Bloch sphere, *i.e.*,

$$|v_0\rangle = \cos(\theta)|v_0'\rangle + \sin(\theta)|v_1'\rangle, \tag{6}$$

$$|v_1\rangle = -\sin(\theta)|v_0'\rangle + \cos(\theta)|v_1'\rangle. \tag{7}$$

If Alice and Bob measure the state $|\psi\rangle$ using the bases $\mathcal{S}$ or $\mathcal{S}'$ the following conditions are satisfied (see also Figure 1):

(i) if both parties measure in $\mathcal{S}$, the outcome pair $(0,0)$ occurs with probability $p_\theta > 0$ (see Equation (8) for an exact formula);
(ii) if one of the parties measures in $\mathcal{S}$ and the other in $\mathcal{S}'$, the outcome pair $(0,0)$ never occurs;
(iii) if both parties measure in $\mathcal{S}'$, the outcome pair $(1,1)$ never occurs.

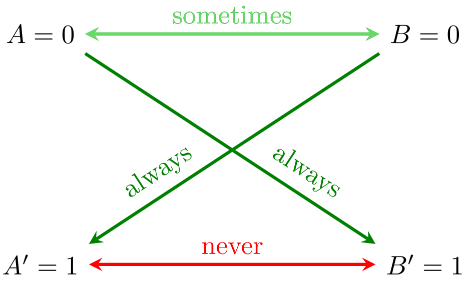


Fig. 1. (Color online.) Summary of correlations between the outcomes of Alice and Bob's measurements in Hardy's setup. For instance, the arrow $A = 0 \rightarrow B' = 1$ marked "always" means that, if Alice measures her system in basis $\mathcal{S}$ (which she does when her input is A), and Bob measures his system in basis $\mathcal{S}'$ (which he does when his input is B'), then whenever Alice obtains the outcome 0 Bob always obtains the outcome 1.

To see that condition (i) is satisfied, we note that

$$p_\theta := |\langle v_0, v_0|\psi\rangle|^2 = \frac{\cos^4(\theta)\sin^2(\theta)}{1+\cos^2(\theta)} > 0 \tag{8}$$

as $0 < \theta < \pi$. To verify the other two conditions one can check that $\langle v_0, v_0'|\psi\rangle = 0$, $\langle v_0', v_0|\psi\rangle = 0$, and $\langle v_1', v_1'|\psi\rangle = 0$.

Any two-qubit state that is neither product nor maximally entangled can be expressed in the form of Equation (5) for appropriately chosen basis $\mathcal{S}'$ and angle θ [16,14]. Therefore, almost any two-qubit state can be used to perform an experiment whose outcomes will satisfy conditions (i)–(iii). Different values of θ

will result in different p_θ; one can verify that the maximum p_θ is $\frac{5\sqrt{5}-11}{2} \approx 0.09$ and is attained at $\theta = \arccos\left(\left(\frac{\sqrt{5}-1}{2}\right)^{1/2}\right) < \frac{\pi}{4}$.

Local hidden variables theories cannot reproduce the predictions (i)–(iii) made by quantum mechanics. Any such theory assigns definite outcomes (depending only on the hidden variables) for each of the four measurements: A, A', B and B'. Condition (i) guarantees that for some setting of the hidden variables the outcomes associated with A and B will both be 0. According to Condition (ii) this implies that for the same setting of hidden variables the outcomes associated with B' and A' must both be 1. This, however, contradicts Condition (iii).

3.2 A Nonlocal Game from Hardy's Paradox

A different construction that is often used to disprove the existence of non-contextual hidden variables theories are the so-called Kochen-Specker sets [13,19]. These sets can easily be turned into a nonlocal game with entangled value 1 and classical value bounded away from 1 [7]. Hardy's paradox in itself does not immediately yield such a game; indeed it is not immediately clear how to express condition (i) in the context of a nonlocal game. To accommodate with that condition we propose the following nonlocal game that we call *Hardy's game* and denote as H.

In game H there are two questions per player: Alice's question set is $\mathcal{I}_A = \{A, A'\}$, Bob's question set is $\mathcal{I}_B = \{B, B'\}$, and the questions are uniformly distributed, *i.e.*, $\pi(X,Y) = \frac{1}{4}$ for all $X \in \mathcal{I}_A$, $Y \in \mathcal{I}_B$. The possible answers for both parties are binary strings of arbitrary but finite length, *i.e.*, $\mathcal{O}_A, \mathcal{O}_B = \{0,1\}^*$. The verification function $V : \mathcal{O}_A \times \mathcal{O}_B \times \mathcal{I}_A \times \mathcal{I}_B \to \{0,1\}$ is defined by $V(a,b|X,Y) = 1$ if and only if all of the following conditions are satisfied.

1. The answer strings a and b have the same length, *i.e.*, $|a| = |b|$.
2. If $X = A$ and $Y = B$, then $a_i = b_i = 0$ for some position $i \in [n]$, where $n := |a| = |b|$.
3. For each position $i \in [n]$:
   (a) if $(X,Y) = (A,B')$ or $(X,Y) = (A',B)$, then $a_i = 1$ or $b_i = 1$;
   (b) if $(X,Y) = (A',B')$, then $a_i = 0$ or $b_i = 0$.

In the above, Condition 3 requires that each pair of answer bits (a_i, b_i) satisfies the last two conditions in Hardy's paradox. Condition 2 requires that for some i, the pair (a_i, b_i) satisfies the first condition in Hardy's paradox with certainty.

3.3 The Classical and Entangled Values

In this section we determine both the classical value and the entangled value of Hardy's game. We refer to the full version of this paper [22] for the proof of the following lemma.

Lemma 1. *The classical value of Hardy's game is* $\omega(H) = 3/4$.

Lemma 2. *The entangled value of Hardy's game is* $\omega^*(H) = 1$*. Furthermore, for any* $\varepsilon > 0$ *there exists a quantum strategy that succeeds with probability at least* $1 - \varepsilon$ *using only answers of length* $\ell = \Theta(\log(1/\varepsilon))$ *and entangled state of local dimension* $d = \Theta(\varepsilon^{1/\log c}) = \Theta(1/\varepsilon^{7.35\ldots})$*, where* $c = \frac{13-5\sqrt{5}}{2}$*.*

Proof. For each $n \in \mathbb{N}$ consider a strategy $\mathfrak{S}_n$ in which players share n copies of the state $|\psi\rangle$ defined in Equation (5)), for some value of θ. To produce an n-bit answer string they measure each of the n copies in basis $\mathcal{S} = \{|v_0\rangle, |v_1\rangle\}$ upon receiving an unprimed question and in basis $\mathcal{S}'$ (see Equation (7)) upon receiving a primed question. To analyze the success probability of strategy $\mathfrak{S}_n$ recall the three conditions from Section 3.2. Since both players answer strings of length n, Condition 1 is always satisfied. As discussed in Section 3.1, the outcomes obtained by measuring $|\psi\rangle$ in basis $\mathcal{S}$ and $\mathcal{S}'$ satisfy the conditions from Hardy's paradox. This implies that Alice's and Bob's answer bits (a_i, b_i) always satisfy Condition 3. Finally, note that upon question (A, B) we have $a_i = b_i = 0$ with probability p_θ (see Equation (8)). Hence, Condition 2 is satisfied with probability $1 - (1 - p_\theta)^n$. If $\theta = \arccos\left(\left(\frac{5\sqrt{5}-1}{2}\right)^{1/2}\right)$, then $\mathfrak{S}_n$ errs with probability $\frac{1}{4}\left(\frac{13-5\sqrt{5}}{2}\right)^n$. Let $c = \frac{13-5\sqrt{5}}{2} \approx 0.91 < 1$, fix any $\varepsilon > 0$ and consider $n = \left\lceil \frac{\log \varepsilon}{\log c} \right\rceil = \Theta(\log(1/\varepsilon))$. Then $\mathfrak{S}_n$ errs with probability at most ε, answers strings of length $\Theta(\log(1/\varepsilon))$ and uses entanglement of local dimension $d = 2^n = \Theta(\varepsilon^{1/\log c})$.

4 Finite Strategies Do Not Achieve the Entangled Value

In this section we present our main result, that the entangled value $\omega^*(H) = 1$ of Hardy's game cannot be attained by any finite-dimensional strategy. We also show that the same holds of strategies with bounded answer length (irrespective of the dimension of the entangled state they are based on).

4.1 Strategies with Bounded Entanglement

Theorem 1. *Let* $\varepsilon > 0$*, and let* d_A, d_B *be integers. Consider a strategy that wins Hardy's game with probability at least* $1 - \varepsilon$ *and uses an entangled state of local dimensions* d_A, d_B*. Then* $\min(d_A, d_B) \geq 1/(24\sqrt{\varepsilon})$*. As a consequence, Hardy's game cannot be won with probability one using finite-dimensional entanglement.*

We first give a key lemma with the following informal interpretation. Consider any pair of answers (s, t) associated with Alice's questions A and A' respectively. Then we can find a question $Y \in \{B, B'\}$ for Bob such that for any $u \in \{0,1\}^*$ the answer pair (s, u) is rejected upon question (A, Y) or the answer pair (t, u) is rejected upon question (A', Y). To state the lemma, first define the following sets:

$$\begin{aligned}
S_0 &:= \left\{(s,t) \in (\{0,1\}^*)^2 : |s| \neq |t|\right\},\\
S_1 &:= \left\{(s,t) \in (\{0,1\}^*)^2 : |s| = |t|,\ \exists i, s_i = 0 \wedge t_i = 1\right\},\\
S_2 &:= \left\{(s,t) \in (\{0,1\}^*)^2 : |s| = |t|,\ \forall j\, (s_j = 0 \implies t_j = 0)\right\}.
\end{aligned}$$

Then $(\{0,1\}^*)^2$ is the disjoint union of S_0, S_1 and S_2. In addition, for any $s, t \in \{0,1\}^*$ let

$$\mathcal{U}'_s := \{v \in \{0,1\}^* : (s,v) \text{ is an invalid pair of answers on questions } (A, B')\}, \tag{9}$$

and let $\mathcal{P}'_t$ be the set of answers v for Bob such that (t, v) is rejected upon question (A', B'). Similarly define $\mathcal{U}_s$ and $\mathcal{P}_t$ as the sets of Bob's answers corresponding to rejected answer pairs for questions (A, B) and (A', B) respectively.[1]

Lemma 3. *For any $(s,t) \in S_0$ it holds that $\mathcal{U}_s \cup \mathcal{P}_t = \mathcal{U}'_s \cup \mathcal{P}'_t = \{0,1\}^*$. For any $(s,t) \in S_1$ it holds that $\mathcal{U}'_s \cup \mathcal{P}'_t = \{0,1\}^*$. For any $(s,t) \in S_2$ it holds that $\mathcal{U}_s \cup \mathcal{P}_t = \{0,1\}^*$.*

Proof. Fix $(s,t) \in (\{0,1\}^*)^2$. Note that each of the three sets S_0, S_1 and S_2 is invariant under joint permutation of the coordinates of s and t, so we may assume without loss of generality that $s = 0^n 1^m$ for some n, m. To simplify notation, we use "$\star$" to denote any element from $\{0,1\}$. For example, we write $t = 0^2\star^2$ to mean $t \in \{0000, 0001, 0010, 0011\}$. We split our analysis into three different cases.

Suppose $(s,t) \in S_0$, so that $t = \star^k$ for some $k \neq n+m$. To succeed upon question (A', B) and Alice's answer $t = \star^k$, Bob must answer a string u that has length k. On the other hand, to succeed upon (A, B) and Alice's answer $s = 0^n 1^m$, Bob must answer a string u of length $n+m$. For any possible answer u of length ℓ, either $\ell \neq n+m$ or $\ell \neq k$. The same reasoning applies if Bob's question is B instead of B'.

Next, suppose $(s,t) \in S_1$, so that $t = r\star^m$ for some $r \in \{0,1\}^n \setminus \{0\}^n$. To succeed upon question (A', B') and Alice's answer $t = r\star^m$, Bob must answer a string v that has zeros in the positions corresponding to the ones in the string r. Since $r \neq 0^n$, we have $v \neq 1^n\star^m$. On the other hand, to succeed upon question (A, B') and Alice's answer $s = 0^n 1^m$, Bob must answer a string $v = 1^n\star^m$. Hence for any answer v of Bob's, either the answer pair (s, v) is rejected upon question (A, B') or the answer pair (t, v) is rejected upon question (A', B'). Thus we have $\mathcal{U}'_s \cup \mathcal{P}'_t = \{0,1\}^*$.

Finally, suppose $(s,t) \in S_2$, so that $t = 0^n\star^m$. To succeed upon question (A', B) and Alice's answer $t = 0^n\star^m$, Bob must answer some $u = 1^n\star^m$. On the other hand, to succeed upon question (A, B) and Alice's answer $s = 0^n 1^m$, Bob must answer a string u that contains a zero in the first n positions (so $u \neq 1^n\star^m$). Hence for any answer u of Bob's, either the answer pair (s, u) is rejected upon question (A, B) or the answer pair (t, u) is a rejected upon question (A', B). Thus we have $\mathcal{U}_s \cup \mathcal{P}_t = \{0,1\}^*$.

Based on Lemma 3 one can already give a short proof for the fact that no finite strategy can achieve success probability $\omega^*(H) = 1$ in Hardy's game. Proving the dimension estimate stated in Theorem 1 requires a more quantitative argument that can be found in the full version of this paper [22].

[1] The letters $\mathcal{P}$ and $\mathcal{U}$ stand for the primed and unprimed questions for Alice.

4.2 Strategies with Bounded Number of Answers

We now show that in order to succeed with higher and higher probability, quantum players must use longer and longer answer strings. The proof of the following theorem can be found in the full version of this paper [22].

Theorem 2. *Let $\varepsilon > 0$, and let L be an integer. Consider a quantum strategy that wins Hardy's game with probability at least $1-\varepsilon$ and for which either Alice's or Bob's answers always fall within a set of cardinality L. Then $L \geq 1/(16\sqrt{\varepsilon})$.*

We can use Hardy's game to certify the dimension of an entangled state. According to Theorem 1, if Alice and Bob succeed at Hardy's game with high probability, then they must possess an entangled state of high local dimension. In practice it is not reasonable to test entanglement via a scheme involving infinitely many outcomes. Therefore, we consider games H_ℓ that are obtained from Hardy's game by limiting the parties to answer strings of length at most $\ell \in \mathbb{N}$. For an appropriately chosen $\ell = \Theta(\log 1/\varepsilon)$, the game H_ℓ has two questions and poly$(1/\varepsilon)$ answers per player; moreover Lemma 2 shows that $\omega^*(H_\ell) \geq 1 - \varepsilon$. Now, as a direct consequence of Theorem 1 we obtain the following corollary.

Corollary 1. *Let $\varepsilon > 0$ and let $\ell = \Theta(\log 1/\varepsilon)$. Any strategy that wins H_ℓ with probability at least $\omega^*(H_\ell) - \varepsilon$ must use a state of dimension at least $\Omega(1/\sqrt{\varepsilon})$.*

5 Discussion and Open Problems

By exhibiting Hardy's game we have shown that there exist two-party distributed tasks with classical questions and answers for which no finite amount of entanglement allows the parties to perform optimally. We have also exhibited an infinite sequence of strategies, using entanglement of increasing dimension, which obtain success probabilities that tend to the optimal $\omega^*(H) = 1$. Each strategy in our sequence produces answers of fixed length, increasing with the dimension of the entangled state. We have showed that to some extent this is necessary: strategies producing answers with bounded length succeed with probabilities that are bounded away from one.

As long as one allows POVMs with an infinite number of outcomes there is a meaningful limit to our sequence of strategies for Hardy's game. The resulting infinite-dimensional strategy uses an entangled state of the form $\bigotimes_{i=1}^{\infty} |\psi\rangle$, where $|\psi\rangle$ is a fixed two-qubit entangled state, and POVM elements take the form $M_a^x = \bigotimes_{i=1}^{\infty} M_{a_i}^x$. Here the dimension of the limiting strategy is uncountably infinite. We leave as an open question the possibility that there could exist a countably-infinite dimensional strategy with value 1 in Hardy's game.

In general there are at least four possible scenarios for the achievability of the entangled value of a nonlocal game:

1. $\omega^*(G)$ can be achieved using some finite-dimensional shared state;
2. a countably-infinite dimensional shared state $|\psi\rangle = \sum_{i,j=1}^{\infty} c_{ij} |i,j\rangle$ is needed to achieve $\omega^*(G)$;

3. an uncountably-infinite dimensional shared state is needed to achieve $\omega^*(G)$;
4. there is no single strategy that succeeds with probability $\omega^*(G)$.

We provide an example of case (3). There are no known examples for cases (2) and (4) and it would be interesting to know whether such games exist.

The most pressing question left open by our work is whether a finite-dimensional shared state is always sufficient to achieve $\omega^*(G)$ for games with finite question and answer sets. As already mentioned in the introduction, there are numerical results which suggest that this is not always the case [24]. It is also interesting to investigate what kind of trade-offs may be required in terms of the dimension of strategies which approach the optimum. In particular, are there any general upper bounds on the dimension d sufficient to achieve value $\omega^*(G) - \varepsilon$?

Acknowledgments. The first author thanks the anonymous referees of QIP 2014 for pointing out a mistake in an earlier proof of the non-quantitative part of Theorem 1, Aleksandrs Belovs, Richard Cleve, David Roberson, Stephanie Wehner, Harry Buhrman and his group at CWI for helpful discussions. Both authors are supported by the Ministry of Education, Singapore under the Tier 3 grant MOE2012-T3-1-009. Thomas Vidick is also supported by the Newton Institute, Cambridge. Parts of this work was completed during a workshop at the Institute for Mathematical Sciences, Singapore.

References

1. Bar-Yossef, Z., Jayram, T.S., Kerenidis, I.: Exponential separation of quantum and classical one-way communication complexity. SIAM J. Comput. 38(1), 366–384 (2008)
2. Bennett, C.H., Wiesner, S.J.: Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states. Phys. Rev. Lett. 69, 2881–2884 (1992)
3. Briët, J., Buhrman, H., Toner, B.: A generalized Grothendieck inequality and nonlocal correlations that require high entanglement. Commun. Math. Phys. 305(3), 827–843 (2011)
4. Brunner, N., Cavalcanti, D., Pironio, S., Scarani, V., Wehner, S.: Bell nonlocality. To appear in Rev. Mod. Phys. (2013)
5. Brunner, N., Pironio, S., Acin, A., Gisin, N., Méthot, A.A., Scarani, V.: Testing the dimension of hilbert spaces. Phys. Rev. Lett. 100, 210503 (2008)
6. Buhrman, H., Cleve, R., Massar, S., de Wolf, R.: Nonlocality and communication complexity. Rev. Mod. Phys. 82, 665–698 (2010)
7. Cleve, R., Hoyer, P., Toner, B., Watrous, J.: Consequences and limits of nonlocal strategies. In: Proceedings of the 19th IEEE Annual Conference on Computational Complexity, pp. 236–249 (2004)
8. Colbeck, R.: Quantum And Relativistic Protocols For Secure Multi-Party Computation. PhD thesis, Trinity College, University of Cambridge (2006)
9. Coudron, M., Yuen, H.: Infinite randomness expansion and amplification with a constant number of devices (2013)
10. Einstein, A., Podolsky, B., Rosen, N.: Can quantum-mechanical description of physical reality be considered complete? Phys. Rev. 47, 777–780 (1935)

11. Ekert, A.K.: Quantum cryptography based on Bell's theorem. Phys. Rev. Lett. 67(6), 661–663 (1991)
12. Fritz, T.: Tsirelson's problem and Kirchberg's conjecture. Rev. Math. Phys. 24(5), 1250012 (2012)
13. Gleason, A.M.: Measures on the closed subspaces of a Hilbert space. J. Math. Mech. 6, 885–893 (1957)
14. Goldstein, S.: Nonlocality without inequalities for almost all entangled states for two particles. Phys. Rev. Lett. 72, 1951–1951 (1994)
15. Hardy, L.: Quantum mechanics, local realistic theories, and Lorentz-invariant realistic theories. Phys. Rev. Lett. 68, 2981–2984 (1992)
16. Hardy, L.: Nonlocality for two particles without inequalities for almost all entangled states. Phys. Rev. Lett. 71, 1665–1668 (1993)
17. Ito, T., Kobayashi, H., Matsumoto, K.: Oracularization and two-prover one-round interactive proofs against nonlocal strategies. In: Proceedings of the 24th IEEE Conference on Computational Complexity, pp. 217–228 (2009)
18. Jonathan, D., Plenio, M.B.: Entanglement-assisted local manipulation of pure quantum states. Phys. Rev. Lett. 83, 3566–3569 (1999)
19. Kochen, S., Specker, E.P.: The problem of hidden variables in quantum mechanics. J. Math. Mech. 17, 59–87 (1967)
20. Laloe, F.: Do we really understand quantum mechanics? Strange correlations, paradoxes, and theorems. Am. J. Phys. 69, 655 (2001)
21. Leung, D., Toner, B., Watrous, J.: Coherent state exchange in multi-prover quantum interactive proof systems. Chicago Journal of Theoretical Computer Science 2013(11) (2013)
22. Mančinska, L., Vidick, T.: Unbounded entanglement can be needed to achieve the optimal success probability. arXiv:1402.4145
23. Navascués, M., Pironio, S., Acín, A.: A convergent hierarchy of semidefinite programs characterizing the set of quantum correlations. New J. Phys. 10(7), 073013 (2008)
24. Pál, K.F., Vértesi, T.: Maximal violation of a bipartite three-setting, two-outcome Bell inequality using infinite-dimensional quantum systems. Phys. Rev. A 82, 022116 (2010)
25. Pironio, S., Acín, A., Massar, S., de La Giroday, A.B., Matsukevich, D.N., Maunz, P., Olmschenk, S., Hayes, D., Luo, L., Manning, T.A., Monroe, C.: Random numbers certified by Bell's theorem. Nature 464(7291), 10 (2010)
26. Regev, O., Vidick, T.: Quantum XOR Games (2012)
27. Slofstra, W.: Lower bounds on the entanglement needed to play XOR non-local games. J. Math. Phys. 52(10) (2011)
28. van Dam, W., Hayden, P.: Universal entanglement transformations without communication. Phys. Rev. A 67, 060302 (2003)
29. Vazirani, U., Vidick, T.: Certifiable quantum dice: or, true random number generation secure against quantum adversaries. In: Proceedings of the 44th Symposium on Theory of Computing, STOC 2012, pp. 61–76 (2012)
30. Wehner, S., Christandl, M., Doherty, A.C.: Lower bound on the dimension of a quantum system given measured data. Phys. Rev. A 78, 062112 (2008)
31. Wilde, M.M.: From classical to quantum Shannon theory (2011)

QCSP on Semicomplete Digraphs

Petar Dapić[1,*], Petar Marković[1,*], and Barnaby Martin[2,**]

[1] Departman za matematiku i informatiku, University of Novi Sad, Serbia
[2] School of Science and Technology, Middlesex University,
The Burroughs, Hendon, London NW4 4BT, U.K.

Abstract. We study the (non-uniform) quantified constraint satisfaction problem QCSP(H) as H ranges over semicomplete digraphs. We obtain a complexity-theoretic trichotomy: QCSP(H) is either in P, is NP-complete or is Pspace-complete. The largest part of our work is the algebraic classification of precisely which semicompletes enjoy only essentially unary polymorphisms, which is combinatorially interesting in its own right.

1 Introduction

The *quantified constraint satisfaction problem* QCSP(B), for a fixed *template* (structure) B, is a popular generalisation of the *constraint satisfaction problem* CSP(B). In the latter, one asks if a primitive positive sentence (the existential quantification of a conjunction of atoms) Φ is true on B, while in the former this sentence may be positive Horn (where universal quantification is also permitted). Much of the theoretical research into CSPs is in respect of a large complexity classification project – it is conjectured that CSP(B) is always either in P or NP-complete [11]. This *dichotomy* conjecture remains unsettled, although dichotomy is now known on substantial classes (e.g. structures of size ≤ 3 [19,6] and smooth digraphs [12,2]). Various methods, combinatorial (graph-theoretic), logical and universal-algebraic have been brought to bear on this classification project, with many remarkable consequences. A conjectured delineation for the dichotomy was given in the algebraic language in [7].

Complexity classifications for QCSPs appear to be harder than for CSPs. Indeed, a classification for QCSPs will give a fortiori a classification for CSPs (if $B \uplus K_1$ is the disjoint union of B with an isolated element, then QCSP($B \uplus K_1$) and CSP(B) are polynomially equivalent). Just as CSP(B) is always in NP, so QCSP(B) is always in Pspace. However, no overarching polychotomy has been conjectured for the complexities of QCSP(B), as B ranges over finite structures, but the only known complexities are P, NP-complete and Pspace-complete. It seems plausible that these complexities are the only ones that can be so obtained (for more on this see [9]).

[*] The first and second authors were supported by the grant no. 174018 of the Ministry of Education and Science of Serbia.

[**] The third author was supported by EPSRC grant EP/L005654/1.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 847–858, 2014.

In this paper we study the complexity of QCSP(H), where H is a semicomplete digraph, i.e. an irreflexive graph so that for each distinct vertices x_i and x_j at least one of $x_i x_j$ or $x_j x_i$ (and possibly both) is in $E(H)$. We prove that each such problem is either in P, is NP-complete or is Pspace-complete. In some respects, our paper is a companion to the classifications for partially reflexive forests [16] and partially reflexive cycles [14], however our work here differs in two important ways. Firstly, this classification is a complete trichotomy instead of a partial classification between P and NP-hard. Secondly, this classification uses the algebraic method to derive hardness results, whereas in [16,14] surjective polymorphisms appear only for tractability. Indeed, we believe our use of the algebraic method here is the most complex so far for any QCSP trichotomy complexity classification. The first published QCSP trichotomy appeared in (the preprints of) [5] and used relatively straightforward application of the algebraic method pioneered in the same paper. Subsequently, a combinatorial QCSP trichotomy appeared, essentially for irreflexive pseudoforests, in [17]. The task to unite [17,16,14], with the spirit of [10], to a QCSP trichotomy for partially reflexive pseudoforests, remains open-ended and ambitious. Two other notable trichotomies have appeared in the QCSP literature in the form of [3] and [4], though both are slightly unorthodox. The former deals with a variant of the QCSP, which allows for relativisation of the universal quantifier, and the latter deals with infinite equality languages.

Our work follows in the spirit of the CSP dichotomy for semicomplete digraphs given long ago in [1]. What we uncover is that the semicompletes with at most one cycle, whose CSPs are in P as per [1], beget QCSPs which remain in P. However, of the semicompletes with more than one cycle, whose CSPs are NP-complete, some produce QCSPs of maximal complexity while others remain no more than NP-complete. Our classification is as follows.

Theorem 1. *Let H be a semicomplete digraph.*

- *If H contains at most one cycle then QCSP(H) is in P, else*
- *H contains a source and a sink and QCSP(H) is NP-complete, else*
- *QCSP(H) is Pspace-complete.*

The tractability results, membership for both P and NP, are relatively straightforward and date back to the last author's 2006 Ph.D. [15]. The natural conjecture was made (not in print) for the trichotomy but repeated efforts to settle it combinatorially failed. The present work arose from a discussion in Dagstuhl about two conjectures involving an algebraic approach, which had always been deemed appropriate as semicomplete digraphs are cores for which all polymorphisms are surjective. The first of these conjectures sought to deal with a large subclass of the semicompletes conjectured to be Pspace-complete, those with neither source nor sink (termed smooth). If it could be proved that all polymorphisms of smooth semicompletes with multiple (i.e. more than one) cycles are essentially unary, then it would be known from [5] that the corresponding QCSP is Pspace-complete. The largest part of this paper is in proving this result. The remaining cases are where there is more than only one cycle and no source (dually resp., sink) but there is a sink (dually resp., source). Suppose then, w.l.o.g, that H^{+m} is built from a smooth semicomplete with multiple cycles H by iteratively adding m sinks. Suppose K_n is the irreflexive n-clique and let K_n^{+m} be the same graph with m sinks iteratively added. The second Dagstuhl conjecture held that, just as the polymorphisms $\mathrm{Pol}(H)$ should be contained in $\mathrm{Pol}(K_n)$, i.e. be only essentially unary, perhaps $\mathrm{Pol}(H^{+m})$ should be contained in $\mathrm{Pol}(K_n^{+m})$, and that would be enough to prove Pspace-completeness for the corresponding QCSP. This conjecture turned out to be false, but some substitute digraphs for K_n in this position were found and so the complexity result follows nonetheless.

As previously stated, the bulk of our work is in proving all smooth semicomplete digraphs with multiple cycles have only essentially unary polymorphisms. It is easy to see this is not true of any of the other semicompletes, for each of which a simple ternary essential polymorphism (i.e. one that is not essentially unary) may be given. Thus, we in fact give another, algebraic, classification.

Theorem 2. *Let H be a semicomplete digraph. If H is smooth and not itself a cycle, then H admits only essentially unary polymorphisms; otherwise H has an essential polymorphism.*

This may be seen as the first part of a larger research program, beginning with semicomplete digraphs, which may continue eventually to larger classes. For example, it is known precisely which smooth digraphs have a weak near unanimity polymorphism [2] and which digraphs enjoy Mal'cev [8].

This paper is organised as follows. After the preliminaries we deal with upper bounds and essential polymorphisms in Section 3. We then deal with the central topic of those semicompletes which have only essentially unary polymorphisms in Section 4. Finally, we deal with the remaining cases of source-without-sink and sink-without-source in Section 5. For reasons of space most proofs are omitted.

2 Preliminaries

Let $[n] := \{1, \ldots, n\}$. All graphs in what follows are directed, that is just a binary relation on a set. We denote *digraphs* by G, H, etc. and their vertex and edge sets by $V(.)$ and $E(.)$ (or $\rightarrow$, $\Rightarrow$; where $\leftrightarrow$, $\Leftrightarrow$ indicates double edge), respectively, where we might omit the (.) if this is clear. We switch rather freely between postfix notations, such as $xy \in E$, and infix notations such as $x \rightarrow y$. If $v \in H$, then $v^+ := \{x \in V(H) : vx \in E(H)\}$ and $v^- := \{x \in V(H) : xv \in E(H)\}$.

A digraph H is *semicomplete* if it is irreflexive (loopless) and for any two vertices i and j, at least one of ij and ji is an edge of H. If H never has both ij and ji, then it is furthermore a *tournament*. For technical reasons we deny the trivial tournament with a single vertex and no edges. The equivalence relation of strong connectedness is defined in the usual way and its equivalence classes will be called strong components. If the strong component has one element, it is trivial, otherwise nontrivial. We start by noting that, just like in the case of tournaments, in semicomplete graphs the strong components can be linearly ordered, so that there is an edge out of every vertex in a smaller strong component into every vertex of a larger strong component (but never an edge going the other way, obviously).

The problems CSP(H) and QCSP(H) each take as input a sentence $\varPhi$, and ask whether this sentence is true on H. For the former, the sentence involves the existential quantification of a conjunction of atoms – *primitive positive* (pp) logic. For the latter, the sentence involves the arbitrary quantification of a conjunction of atoms – *positive Horn* (pH) logic. It is well-known, for finite H, that CSP(H) and QCSP(H) are in NP and Pspace, respectively.

The *direct product* $G \times H$ of two digraphs G and H has vertex set $\{(x,y) : x \in V(G), y \in V(H)\}$ and edge set $\{(x,u)(y,v) : x,y \in V(G), u,v \in V(H), xy \in E(G), uv \in E(H)\}$. Direct products are (up to isomorphism) associative and commutative. The kth power G^k of a graph G is $G \times \ldots \times G$ (k times). A homomorphism from a graph G to a graph H is a function $h : G \to H$ such that, if $xy \in E(G)$, then $h(x)h(y) \in E(H)$. A *k-ary polymorphism* of a graph H is a homomorphism from H^k to H. A polymorphism f is *idempotent* when, for all x, $f(x,\ldots,x) = x$. An operation $f : H^k$ to H is termed *essentially unary* if there is a unary operation g and co-ordinate i so that $f(x_1,\ldots,x_k) = g(x_i)$. If f is not essentially unary then we describe f as *essential.*

A digraph is a *core* if all of its endomorphisms are automorphisms. All finite semicomplete digraphs are cores, for which all polymorphisms are surjective. For cores it is well-known the constants are pp-definable up to automorphism. That is, if H^c is H with all constants named, and H is a core, then CSP(H) and CSP(H^c) are poly time equivalent; and the same applies to the QCSP. A similar argument may be given in the algebraic language and the implication is that we may as well assume all the polymorphisms of a semicomplete digraph H are idempotent (because this is true for H^c which is actually the structure we will be working on).

The now-celebrated *algebraic approach* to CSP rests on one half of a Galois correspondence, where it is observed that the relations that are invariant under (preserved by) the polymorphisms of H are precisely the relations that are pp-definable in H. For QCSP, we obtain a similar characterisation substituting surjective polymorphisms for polymorphisms and pH for pp. The consequence of this is that if the polymorphisms (resp., surjective polymorphisms) of H are contained as a subset of those of H', then there is a poly time reduction from CSP(H') to CSP(H) (resp., QCSP(H') to QCSP(H)); that is, the polymorphisms control the complexity.

If $\varPhi$ is an input for QCSP(H) with quantifier-free part φ, then with this we associate the digraph D_φ whose vertices are variables of φ and edges are given by the atoms in φ. If $\varPhi$ is existential, i.e. also an input to CSP(H), then the relationship between $\varPhi$ and $D_\varPhi$ is that of canonical query to canonical database [13].

In a digraph, a *source* (resp., *sink*) is a vertex with in-degree (resp. out-degree) 0. A digraph with no sources or sinks is called *smooth.* In a semicomplete graph, a source s (resp., sink t) satisfies, for all $x \neq s$ (resp., $x \neq t$), $xs \notin E(H)$ and $sx \in E(H)$ (resp., $tx \notin E(H)$ and $xt \in E(H)$). A digraph may have multiple sources or sinks, but a semicomplete may have at most one of each. If H is a digraph, then let H^{+j} be H with, iteratively, j sinks added (i.e. each time we add a sink we make it forward-adjacent to each existing vertex). Let us label these added sinks, in order, $t_1, \ldots, t_j$ (thus t_j is the unique sink of H^{+j}). Similarly, let $\mathcal{H}^{-j}$ be $\mathcal{H}$ with j sources added. When the j is omitted it is presumed to be 1.

We mention some special semicomplete graphs that will appear in the paper. K_n is the irreflexive complete graph (clique) on vertex set $[n]$. For $i \neq j \in [n]$, K_n has both edges ij and ji. DC_3 is the directed 3-cycle. Let T_n be the transitive tournament on $[n]$ with the natural order $<$ corresponding to the edge relation (i.e. $ij \in E(T_n)$ iff $i < j$).

3 Complexity Upper Bounds and Essential Polymorphisms

The main results of this section date back to the third author's Ph.D. [15] (available from his website) and are presented there combinatorially and in much fuller detail. The first is very straightforward.

Proposition 1. *Let H be a digraph with both a source s and a sink t, then QCSP(H) is in NP.*

Proof. Let Φ be an input to QCSP(H) with quantifier-free part φ. Suppose φ has an atom v_iv_j so that Φ quantifies v_i universally, then Φ is a no-instance since φ will never be satisfied when v_i is evaluated as t. Dually, we may assume φ has no atom v_iv_j so that Φ quantifies v_j universally; and we find that Φ can not contain universally quantified variables involved in atoms of φ. Thus, we may ignore universally quantified variables and evaluate Φ as an input to CSP(H) in NP.

We now turn our attention to the poly time cases.

Proposition 2. *For all $n \geq 1$, QCSP(T_n) is in P.*

Proof. The ternary median function $f(x, y, z) = \mathrm{med}(x, y, z)$ is a polymorphism of T_n which is a majority operation. The tractability of QCSP(T_n) follows from [5].

It is well-known that QCSP(K_2) and QCSP(DC_3) admit a majority polymorphism and are therefore in P (see [5]). We are now interested in the semicomplete graphs K_2^{+j}, K_2^{-j}, DC_3^{+j} and DC_3^{-j} (for $j > 0$). Proof of the following appears in the appendix.

Proposition 3. *For $j \geq 0$, each of QCSP(K_2^{+j}), QCSP(K_2^{-j}), $QCSP(DC_3^{+j})$ and QCSP(DC_3^{-j}) are in P.*

We now deal with the semicompletes that admit essential polymorphisms.

Proposition 4. *If H is a semicomplete digraph with at most one cycle or a source or a sink, then H admits an essential polymorphism.*

Proof. It was noted in the proof of Proposition 2 that the transitive tournaments admit a median polymorphism. Afterwards it was noted further that K_2 and DC_3 admit majority polymorphisms (and indeed the median may be used here).

Let H be a semicomplete digraph and recall H^+ to be the same digraph with a sink t added, to which all other vertices have a forward edge. Then H has the polymorphism $f(x, y, z) = x$, unless ($y = t$ or $z = t$) in which case $f(x, y, z) = t$. It follows that semicompletes with sink admit an essential polymorphism. The result for semicompletes with a source is symmetric and the result follows.

4 Semicompletes with Essentially Unary Polymorphisms

Theorem 3. *Let H be a smooth semicomplete digraph with precisely two strong components. Then all idempotent polymorphisms of H are projections.*

Theorem 4. *Let H be a smooth semicomplete digraph with two non-trivial strong components. Then all idempotent polymorphisms of H are projections.*

Theorem 5. *Let H be a smooth semicomplete digraph with more than two strong components. Then all idempotent polymorphisms of H are projections.*

We sum these up in the following corollary.

Corollary 1. *Let H be a smooth semicomplete digraph that is not strongly connected. Then all idempotent polymorphisms of H are projections.*

4.1 The Strongly Connected Case

Definition 1. *A subset $L \subset V$ is nice if the induced subgraph on L is strongly connected and all idempotent polymorphisms of G restrict to L as projections.*

Lemma 1. *Let L be a nice subset of V and let v be a vertex such that $v^+ \cap L \neq \emptyset \neq v^- \cap L$. Then $L \cup \{v\}$ is nice.*

Lemma 2. *Let $L = \{a, b\}$ be compatible with (i. e. closed under) the idempotent polymorphisms of G and let $a \to b \to a$. If $v \in V \setminus L$ is such that $v^+ \cap L \neq \emptyset \neq v^- \cap L$ and f is an n-ary idempotent polymorphism of G, then there exists i, $1 \leq i \leq n$, such that on the subset $\{a, b, v\}$ the restriction of f is equal to the ith projection.*

A *congruence* of a tournament $(V, \to)$ is an equivalence relation ρ on V such that for all (x_1, x_2), $(y_1, y_2) \in \rho$ such that $(x_1, y_1) \notin \rho$, $x_1 \to y_1$ iff $x_2 \to y_2$. If ρ is a congruence of the tournament $T = (V, \to)$, then the *factor tournament* T/ρ is the tournament $(V/\rho, \Rightarrow)$, where $a/\rho \Rightarrow b/\rho$ iff $a/\rho \neq b/\rho$ and $a \to b$.

We also introduce the interval notation for a digraph $G = (\{a_1, a_2, \ldots, a_n\}, \to)$ with the fixed Hamiltonian cycle $a_1 \to a_2 \to \ldots \to a_n \to a_1$: $[a_i, a_j]$ is the set of all vertices that are traversed by shortest path starting at a_i, ending at a_j and which uses only the directed edges of the Hamiltonian cycle. For instance, $[a_2, a_1] = \{a_1, a_2, \ldots, a_n\}$, while $[a_1, a_2] = \{a_1, a_2\}$. We also define $[a_i, a_j) := [a_i, a_j] \setminus \{a_j\}$, $(a_i, a_j] := [a_i, a_j] \setminus \{a_i\}$ and $(a_i, a_j) := [a_i, a_j] \setminus \{a_i, a_j\}$.

Definition 2. *Let $T = (\{a_1, \ldots, a_n\}, \to)$ be a strongly connected tournament with the fixed Hamiltonian cycle $C = a_1 \to a_2 \to \ldots \to a_n \to a_1$, where $n \geq 3$. T is* locally transitive *with respect to the cycle C iff there exists a function $\varphi_T : \{1, \ldots, n\} \to \{1, \ldots, n\}$ such that:*

1. *$\varphi_T(i) \notin \{i-1, i\}$ and $\varphi_T(1) \notin \{1, n\}$,*
2. *$a_i^+ = (a_i, a_{\varphi_T(i)}]$ and*
3. *$a_{\varphi_T(i+1)} \in [a_{\varphi_T(i)}, a_i)$ and $a_{\varphi_T(1)} \in [a_{\varphi_T(n)}, a_n)$.*

In particular, since the locally transitive tournament T is semicomplete, we get that $a_{\varphi_T(i)+1} \to a_i$ and from the definition above follows that

$$(4) \qquad a_i \to a_{i+1}, \qquad (a_{i+1} \to a_{\varphi_T(i)} \text{ or } a_{i+1} = a_{\varphi_T(i)}) \quad \text{and} \quad a_i^+ \backslash \{a_{i+1}\} \subseteq a_{i+1}^+$$

(where the addition here is modulo n, so $n + 1 = 1$). Note also that local transitivity depends on the fixed Hamiltonian cycle C. It is easy to construct five-element Hamiltonian tournament which is locally transitive with respect to one of its Hamiltonian cycles, but not with respect to another.

We will use the easier notation for a locally transitive tournament T when the vertex set is $\{1, 2, \ldots, n\}$, where we will understand, unless otherwise stated, that the fixed Hamiltonian cycle is $1 \to 2 \to \ldots \to n \to 1$, and $a_i = i$, so we will have $(\varphi_T(i) + 1) \to i$ instead of $a_{\varphi_T(i)+1} \to a_i$ et cetera.

Definition 3. *A locally transitive tournament $T = (\{1, \ldots, n\}, \to)$ is* regular *iff $n = 2k + 1$ for some positive integer k and for all $1 \leq i < j \leq 2k + 1$, $i \to j$ iff $j - i \leq k+1$ (otherwise $j \to i$). In other words, in the unique (up to isomorphism) regular locally transitive tournament with $2k+1$ vertices, $\varphi_T(i) = i+k$ if $i \leq k+1$, and $\varphi_T(i) = i - k - 1$ if $i > k + 1$.*

Lemma 3. *Let $T = (\{1, \ldots, n\}, \to)$ be a locally transitive tournament such that φ_T is a permutation of $\{1, \ldots, n\}$. Then T is regular.*

Definition 4. *The semicomplete graph $G_T = (V, E)$ will be called a P-graph parametrized by the locally transitive tournament $T = (\{1, \ldots, n\}, \to)$ if there exists a partition ρ of the vertex set V into nonempty subsets $A_1, \ldots, A_n$ such that for all $i \neq j$ and all $a \in A_i$ and $b \in A_j$, $ab \in E$ iff $i \to j$ in T.*

Theorem 6. *Every idempotent polymorphism f of a P-graph G_T parametrized by the locally transitive tournament T is a projection, except when G_T is the 3-cycle.*

Lemma 4. *Let $G = (V, \to)$ be a strongly connected semicomplete graph which contains at least one 2-cycle. Then for each 2-cycle $a \leftrightarrow b$ in G, the set $\{a, b\}$ is closed with respect to all idempotent polymorphisms of G and each binary idempotent polymorphism of G restricted to $\{a, b\}$ is a projection.*

Definition 5. *Let $G = (V, \to)$ be a strongly connected semicomplete graph. We say that L* splits *G if $\emptyset \neq L \subsetneq V$ is a subset with the following properties:*

1. *$\{L, L^+, L^-\}$ is a partition of V and*
2. *for any 2-cycle $a \leftrightarrow b$ in G, $\{a, b\}$ is contained in one of L, L^- or L^+.*

Lemma 5. *Let $G = (V, \to)$ be a strongly connected semicomplete graph which is not a cycle. Let L_0 be either a 2-cycle or a nice subset of V. Then either all idempotent polymorphisms of G are projections, or there exists a subset $L \subset V$ such that L splits G, $L_0 \subset L$ and either the induced subgraph on L is a 2-cycle, or L is nice.*

Lemma 6. *Let $G = (V, \to)$ be a strongly connected semicomplete graph which is not a P-graph and let L split G. Then there exist vertices $a_0, a_1, b_0 \in V$ such that $a_1 \leftarrow a_0 \to b_0 \to a_1$ and that either*

1. *$b_0 \in L^-$ and a_0, a_1 are in the same strong component, or two consecutive strong components, of the induced subgraph on L^+, or*
2. *$b_0 \in L^+$ and a_0, a_1 are in the same strong component, or two consecutive strong components, of the induced subgraph on L^-.*

Lemma 7. *If a strongly connected tournament $G = (V, \to)$ is not a P-graph and for all $v \in V$, all strong components of the induced subgraphs on v^+ and on v^- are of sizes 1 or 3, then there is a 3-cycle $a \to b \to c \to a$ in G such that all idempotent polymorphisms of G restrict to $\{a, b, c\}$ as projections.*

Theorem 7. *A strongly connected semicomplete digraph which is not a cycle has all its idempotent polymorphisms being projections.*

Proof. We prove it by an induction on $|V| = n$. By Theorem 6, if G is a P-graph, we are done, so we assume that G is not a P-graph. For $n = 2$ the only semicomplete digraph must be a cycle. If $n = 3$ and G is not a cycle, then there is a 2-cycle $a \leftrightarrow b$ in G, and the third vertex c must satisfy either $a \to c \to b$ or $b \to c \to a$ (possibly even both!), so by Lemma 4 and Lemma 2 all idempotent polymorphisms are projections. Also, if $n = 4$, then G is a P-graph parametrized by the 3-cycle if G is the only 4-element strongly connected tournament or in the case when $V = \{a, b, c, d\}$ has exactly one 2-cycle $a \leftrightarrow b$, $c \in \{a, b\}^+$ and $d \in \{a, b\}^-$. Otherwise, from Lemmas 4, 2 and 1 follows that all idempotent polymorphisms of G are projections.

Now assume that $n > 4$ and that the Theorem holds in all strongly connected semicomplete graphs with fewer than n vertices. If there exists a 2-cycle $a \leftrightarrow b$, then we set $L_0 = \{a, b\}$. Otherwise, G is a tournament, and if there exists any vertex $v \in V$ and a strong component L_0 of the induced subgraph on v^- or on v^+ such that $|L_0| > 3$, then L_0 is clearly pp-definable with constants in G, so L_0 must be nice by the inductive assumption. Finally, if G is a tournament and for all $v \in V$ all strong components of the induced subgraphs on v^- and on v^+ have at most three elements, then by Lemma 7 follows that there is a three element subset L_0 which is nice.

Let L be a maximal nice subset of V such that $L_0 \subset L$. If $L \neq V$, then by Lemma 5, L splits G. Now from Lemma 6 follows that either a strong component L' of the induced subgraph on a_0^+ contains $L \cup \{a_1, b_0\}$ (if (1) of Lemma 6 holds), or that a strong component L' of the induced subgraph on a_1^- contains $L \cup \{a_0, b_0\}$ (if (2) of Lemma 6 holds). Either way, L' is pp-definable with constants in G, $L \subsetneq L' \subsetneq V$ and the induced subgraph on L' is strongly connected, so by the inductive assumption L' is nice. This contradicts the assumed maximality of L. So, the only alternative is $L = V$, but then the Theorem holds by niceness of L.

Our main complexity result now follows from [5].

Corollary 2. *If H is a smooth semicomplete digraph with more than one cycle, then QCSP(H) is Pspace-complete.*

5 Remaining Semicomplete Digraphs

Recall $\mathcal{T}_n$ to be the transitive tournament on $[n]$ with the natural order $<$ corresponding to the edge relation. Let $\overline{\mathcal{T}_n}$ be $\mathcal{T}_n$ with the extant edge $E(1, n)$ augmented by $E(n, 1)$, i.e. this becomes a double-edge. Let $\mathcal{K}_{2\to 2}$ be the semicomplete graph built from disjoint copies $\mathcal{H}_1$ and $\mathcal{H}_2$ of $\mathcal{K}_2$ with all edges added from $\mathcal{H}_1$ to $\mathcal{H}_2$. More generally, let $\mathcal{K}_{2\to 1^k \to 2}$ be the semicomplete graph built from disjoint copies $\mathcal{H}_1$ and $\mathcal{H}_2$ of $\mathcal{K}_2$ with a transitive tournament T_k inbetween.

5.1 Some Pspace-hardness Results

Proposition 5. *For each $k > 0$, QCSP($K_{2\to 2}$) and QCSP($K^+_{2\to 2}$) are Pspace-complete.*

Corollary 3. *Let $G = (V, \to)$ be a finite digraph without loops. Let G contain either*

1. *a copy of $K_{2\to 2}$ such that $a \leftrightarrow b \to c \leftrightarrow d$ such that any automorphism of this copy extends by the identity map to an automorphism of G and moreover, $a^+ \cup b^+ = V$, or*
2. *a copy of K_3, $a \leftrightarrow b \leftrightarrow c \leftrightarrow a$ such that any permutation of $\{a, b, c\}$ extends by the identity map to an automorphism of G and moreover $a^+ \cup b^+ = a^+ \cup c^+ = b^+ \cup c^+ = V$,*

then QCSP(G) is Pspace-complete.

Proposition 6. *For $n \geq 3$, both QCSP($\overline{T}_n$) and QCSP($\overline{T}^+_n$) are Pspace-complete.*

Proposition 7. *For any digraph H, QCSP(H^+) and QCSP$_{[\exists/H]}$(H^+) are equivalent.*

For H a subset of the domain of the structure H', let QCSP$_{[\exists/H]}(H')$ be the variant of QCSP(H') in which the existential variables are restricted to being chosen from H.

Proposition 8. *Let H be a digraph. For each $j > 1$ there exists a polytime reduction from QCSP$_{[\exists/H]}$(H^+) to QCSP(H^{+j}).*

Corollary 4. *For any digraph H and each $j > 1$, QCSP(H^+) reduces to QCSP(H^{+j}).*

Corollary 5. *For each $j > 0$, QCSP($\overline{T}_n^{+j}$) and QCSP($K_{2\to 2}^{+j}$) are both Pspace-complete.*

5.2 The Algebraic Part

Definition 6. *Let $G = (V, \to)$ be a directed graph. We define the relation $\preceq_G$ on V by $x \preceq_G y$ iff $x^- \subseteq y^-$.*

Proposition 9. *Assume that G is semicomplete. Then $\preceq_G$ is a partial order, $\preceq_G$ has the largest element t iff t is a sink, and dually for least elements and sources.*

Lemma 8. *Let $G = (V, \to)$ be a semicomplete graph without sources, but with the sink t. Let $f : V^m \to V$ be any idempotent mapping such that its restriction to $V \setminus \{t\}$ is the first projection. f is a polymorphism of G iff for all $b_1, b_2, \dots, b_m \in V$, $b_1 \preceq_G f(b_1, b_2, \dots, b_m)$.*

Definition 7. *Let $G = (V, E)$ be a digraph. We define the partition of the vertex set V into V^G_{min}, V^G_{max}, V^G_{both} and V^G_{none} so that all vertices in V^G_{max} are minimal, but not maximal, in the order $\preceq_G$, all vertices in V^G_{min} are maximal, but not minimal, in the order $\preceq_G$, all vertices in V^G_{both} are both minimal and maximal in the order $\preceq_G$, while vertices in V^G_{none} are neither minimal nor maximal in the order $\preceq_G$. When the digraph G is understood, we will omit the superscript ^{G}.*

Definition 8. *Let $G = (V, E)$ be a digraph. We define the digraph $S(G) = (V, \to)$ by:*

1. *For all $x, y \in V_{max} \cup V_{both}$, $x \not\leftrightarrow y$,*
2. *For all $x, y \in V_{min}$, $x \leftrightarrow y$,*
3. *For all $x, y \in V_{none}$, $x \to y$ iff $E(x, y)$.*
4. *For all $x \in V_{min}$ and $y \in V_{none} \cup V_{max}$, $x \to y$, but $\neg y \to x$,*
5. *For all $x \in V_{none}$ and $y \in V_{max}$, $x \to y$, but $\neg y \to x$,*
6. *For all $x \in V_{both}$ and $y \in V_{none} \cup V_{min}$, $x \to y$, but $\neg y \to x$.*

Proposition 10. *$V^{S(G)}_{min} = V^G_{min}$, $V^{S(G)}_{max} = V^G_{max}$, $V^{S(G)}_{both} = V^G_{both}$ and $V^{S(G)}_{none} = V^G_{none}$. Consequently, $S(S(G)) = S(G)$.*

Proposition 11. *A permutation α of the vertex set V of the digraph $G = (V, \to)$ (more generally, universe A of a finite relational structure) is an automorphism iff it is structure-preserving.*

Lemma 9. *The following statements hold for any digraph G:*

1. *$Aut(G) \subseteq Aut(V, \preceq_G)$,*
2. *$Aut(G) \subseteq Aut(S(G))$,*
3. *$\preceq_G \subseteq \preceq_{S(G)}$ and*
4. *If G is smooth and semicomplete, then so is $S(G)$.*
5. *If G is not a cycle and semicomplete, then neither is $S(G)$.*

Corollary 6. *Let $G = (V, E)$ be a smooth semicomplete digraph which is not a cycle. Then $Pol(G^+) \subseteq Pol(S(G)^+)$.*

Definition 9. *Let $G = (V, E)$ be a digraph. We define the digraph $L(G)$ on the set V in the following way:*

1. *For all $x \in V_{both} \cup V_{min}$ and $y \in V_{none} \cup V_{max}$, $x \to y$, but $\neg y \to x$,*
2. *For all $x \in V_{none}$ and $y \in V_{max}$, $x \to y$, but $\neg y \to x$,*
3. *For all $x, y \in V_{min} \cup V_{both}$, $x \leftrightarrow y$,*
4. *For all $x, y \in V_{none}$, $x \to y$ iff $E(x, y)$,*
5. *For all $x, y \in V_{max}$, $x \leftrightarrow y$.*

The next Lemma follows directly from Definition 9.

Lemma 10. *Let G be a digraph. Either $V = V^G_{both} = V^{L(G)}_{both}$, or $V^{L(G)}_{min} = V^G_{both} \cup V^G_{min}$, $V^{L(G)}_{none} = V^G_{none}$, $V^{L(G)}_{max} = V^G_{max}$ and $V^{L(G)}_{both} = \emptyset$.*

Corollary 7. *Let $G = (V, E)$ be a smooth semicomplete digraph which is not a cycle. Then $Pol(S(G)^+) \subseteq Pol(L(G)^+)$.*

Theorem 8. *Let $G = (V, E)$ be a smooth semicomplete digraph which is not a cycle. Then QCSP(G^{+j}) is Pspace complete for all $j > 0$.*

Corollary 8. *If H is semicomplete with more than one cycle and either: 1.) a sink but no source, or 2.) a source but no sink, then QCSP(H) is Pspace-complete.*

Proof. Case 1 is taken care of by Theorem 8 and Case 2 is symmetric.

6 Conclusion

We can now piece together proofs of our central theorems.

Proof (of Theorem 1). The cases in P follow from Propositions 2 and 3. The NP upper bound follows from Proposition 1 and the NP lower bound follows from [1]. All (finite-domain) QCSPs are in Pspace so, finally, the Pspace-hard cases follow from Corollaries 2 and 8.

Proof (of Theorem 2). From Proposition 4 and Theorem 7.

References

1. Bang-Jensen, J., Hell, P., MacGillivray, G.: The complexity of colouring by semicomplete digraphs. SIAM J. Discrete Math. 1(3), 281–298 (1988)
2. Barto, L., Kozik, M., Niven, T.: The CSP dichotomy holds for digraphs with no sources and no sinks (a positive answer to a conjecture of Bang-Jensen and Hell). SIAM Journal on Computing 38(5), 1782–1802 (2009)

3. Bodirsky, M., Chen, H.: Relatively quantified constraint satisfaction. Constraints 14(1), 3–15 (2009)
4. Bodirsky, M., Chen, H.: Quantified equality constraints. SIAM J. Comput. 39(8), 3682–3699 (2010)
5. Börner, F., Bulatov, A.A., Chen, H., Jeavons, P., Krokhin, A.A.: The complexity of constraint satisfaction games and qcsp. Inf. Comput. 207(9), 923–944 (2009); Technical report "Quantified Constraints and Surjective Polymorphisms" was 2002 and Conference Version "Quantified Constraints: Algorithms and Complexity" was CSL 2003 (2003)
6. Bulatov, A.: A dichotomy theorem for constraint satisfaction problems on a 3-element set. J. ACM 53(1), 66–120 (2006)
7. Bulatov, A., Krokhin, A., Jeavons, P.G.: Classifying the complexity of constraints using finite algebras. SIAM Journal on Computing 34, 720–742 (2005)
8. Carvalho, C., Egri, L., Jackson, M., Niven, T.: On maltsev digraphs. In: Kulikov, A., Vereshchagin, N. (eds.) CSR 2011. LNCS, vol. 6651, pp. 181–194. Springer, Heidelberg (2011)
9. Chen, H.: Meditations on quantified constraint satisfaction. In: Constable, R.L., Silva, A. (eds.) Kozen Festschrift. LNCS, vol. 7230, pp. 35–49. Springer, Heidelberg (2012)
10. Feder, T., Hell, P., Jonsson, P., Krokhin, A.A., Nordh, G.: Retractions to pseudoforests. SIAM J. Discrete Math. 24(1), 101–112 (2010)
11. Feder, T., Vardi, M.: The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. SIAM Journal on Computing 28, 57–104 (1999)
12. Hell, P., Nešetřil, J.: On the complexity of H-coloring. Journal of Combinatorial Theory, Series B 48, 92–110 (1990)
13. Kolaitis, P.G., Vardi, M.Y.: A logical approach to constraint satisfaction. In: Creignou, N., Kolaitis, P.G., Vollmer, H. (eds.) Complexity of Constraints. LNCS, vol. 5250, pp. 125–155. Springer, Heidelberg (2008)
14. Madelaine, F., Martin, B.: Qcsp on partially reflexive cycles - the wavy line of tractability. In: Bulatov, A.A., Shur, A.M. (eds.) CSR 2013. LNCS, vol. 7913, pp. 322–333. Springer, Heidelberg (2013)
15. Martin, B.: Logic, Computation and Constraint Satisfaction. PhD thesis, University of Leicester (2006)
16. Martin, B.: QCSP on partially reflexive forests. In: Lee, J. (ed.) CP 2011. LNCS, vol. 6876, pp. 546–560. Springer, Heidelberg (2011)
17. Martin, B., Madelaine, F.: Towards a trichotomy for quantified H-coloring. In: Beckmann, A., Berger, U., Löwe, B., Tucker, J.V. (eds.) CiE 2006. LNCS, vol. 3988, pp. 342–352. Springer, Heidelberg (2006)
18. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley (1994)
19. Schaefer, T.J.: The complexity of satisfiability problems. In: Proceedings of STOC 1978, pp. 216–226 (1978)

Fast Pseudorandomness for Independence and Load Balancing*
(Extended Abstract)

Raghu Meka[1], Omer Reingold[1], Guy N. Rothblum[1], and Ron D. Rothblum[2]

[1] Microsoft Research
{meka,omer.reingold}@microsoft.com, rothblum@alum.mit.edu
[2] Weizmann Institute of Science
ron.rothblum@weizmann.ac.il

Abstract. We provide new constructions of several fundamental pseudorandom objects. Loosely speaking, these constructions obtain exponential improvements in efficiency compared to previous constructions with comparable randomness complexity. Our measure of efficiency is the number of word operations, as captured by the well-established unit-cost word RAM model. Our main results are the following:

1. A family of $(1/n)$-almost $\log n$-wise independent Boolean hash functions with $O(\log n)$ description length (or seed length) and $O(\log\log n)$ operations per evaluation.
   Prior constructions with similar seed lengths required $\Theta(\log n)$ operations.
2. ε-biased sequences for $\varepsilon = 1/\mathrm{poly}(n)$ with seed length $O(\log n \log\log n)$ and $O((\log\log n)^2)$ operations (to evaluate an output bit or a block of up to $\log n$ consecutive bits).
   Prior constructions achieve $O(\log n)$ seed length, but require $\Theta(\log n)$ operations. This construction implies pseudorandom generators with similar efficiency that fool classes such as low-degree polynomials and read-once CNFs.
3. Hash functions for placing n balls in n bins such that with all but probability $1/n$ the maximal load is $O(\log n/\log\log n)$ (which is optimal), with seed-length $O(\log n \log\log n)$ and $O((\log\log n)^2)$ operations per evaluation.
   The previously known construction with similar seed length required $\Theta(\log n \log\log n)$ operations. Indeed, our construction is an efficient instantiation of that construction, due to Celis, Reingold, Segev and Wieder (FOCS 2011).

These constructions are all simultaneously within $\log\log n$ factors of the optimal seed length, and within $(\log\log n)^2$ factors of the optimal computational efficiency.

1 Introduction

Randomness is a valuable resource in the theory and the practice of computing. Designs of randomized data structures, algorithms, and protocols often assume

* The full version is available on the authors' homepages.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 859–870, 2014.

access to a truly random object, such as a random function or a sequence of random bits. In many settings, however, these random objects must be replaced with a succinct and efficiently computable explicit construction that mimics some of their properties. For example, a pseudorandom bit generator, or a small family of hash functions.

In this work, we are interested in fast and efficient constructions of pseudorandom objects. Motivated by applications to data structures and algorithms, we aim for efficiency in the *(unit-cost) word RAM model.* This model measures complexity in terms of *word operations*, and is a (perhaps *the*) central model in the theory and practice of data structure and algorithm design. Applications aside, the word-RAM complexity of pseudorandom objects is a foundational question, and it has been explored in an important body of work (e.g. [NN93, Sie04, CRSW13, Tho13]). We focus on three fundamental pseudorandom objects: almost-independent hash functions, small-bias generators, and load balancing hash functions. Loosely speaking, we obtain exponential efficiency improvements compared to the fastest known constructions for each of these objects, while maintaining (up to $\log\log$ factors) the best seed length. We proceed with an overview of our contributions and their relationship to prior works. A more detailed discussion of the RAM model follows. We note that these results have already found applications in subsequent recent works of Reingold, Rothblum and Weider [RRW14] and of Reingold and Vardi [RV14].

Limited Independence. Families of hash functions with limited independence are central objects in the study of hashing and derandomization, and have many applications to algorithms and data structures. Limited independence suffices in many cases where fully random hash functions are used, and functions with limited independence have the advantage of a much more succinct and space-efficient representation. A hash family is said to be ε-almost k-wise independent if for any k fixed inputs, their (joint) output distribution is ε-close to uniform (in statistical distance, where the probability is over the choice of a function from the family). The use of limited independence in computer science begins with the seminal work of Carter and Wegman [CW79, WC81]. See also the survey of Luby and Wigderson [LW05].

A particularly interesting parameter choice is that of $(1/\text{poly}(n))$-almost $\log n$-wise independent Boolean hash functions, which lend themselves to showing concentration bounds with polynomially small errors. For this setting of parameters, known constructions achieve optimal seed length of $O(\log n)$ bits [NN93], but computing these functions is expensive, and requires $O(\log n)$ word operations (where the word length is $O(\log n)$). Alternatively, Siegel [Sie04] and Thorup [Tho13] propose functions with $n^{\Omega(1)}$-independence that can be evaluated in a *constant* number of word operations, but use storage or key length $n^{\Omega(1)}$ (for functions with constant evaluation time, larger key length or storage is essential, even for logarithmic output length and independence, by Siegel's lower bound [Sie04]). Our first result is a new construction of *Boolean* hash functions with logarithmic seed length/storage and fast evaluation.

Construction 1 (Fast Almost Independent Hash Family). *We construct a family of* $(1/\text{poly}(n))$*-almost* $\log n$*-wise independent functions from* $\{0,1\}^w$ *to* $\{0,1\}$*, with a* $O(\log n)$*-bit seed, which can be evaluated in* $O(\log\log n)$ *word operations.*

Moreover, for any $m \in [\log n]$*, we construct a* $(1/\text{poly}(n))$*-almost* $((\log n)/m)$*-wise independent family of hash functions from* $\{0,1\}^w$ *to* $\{0,1\}^m$*, with a* $O(\log n)$*-bit seed, which can be evaluated in* $O(\log\log n)$ *operations.*

This construction follows from a new fast small-bias generator described below. It achieves an exponential efficiency improvement over past work, while still having optimal seed length / storage (up to constant factors). Moreover, it also allows fast computation of m-bit outputs: as the output size increases to $m > 1$ bits, the independence guarantee decreases proportionally to $(\log n)/m$.

It is interesting to compare our upper bound with the lower bound of Siegel [Sie04] for computing k-wise independent hash functions. Siegel's seminal *cell-probe* lower bound shows that when hashing from a universe of size n to a range of size 2^m with m-bit long *cells*, to get k-wise independent hash functions that can be evaluated in time $t < k$ (t here is the number of cell probes), one needs storage or key length $\Omega(k \cdot n^{1/t})$ (this is the length of the hash key that allows computation in t cell probes). Adapted to the RAM model, Siegel's lower bound says that to get $\log n$-wise independent hash functions with $\log n$ output length, even using $\log^\gamma n$ time, requires large $\Omega(2^{\log^{1-\gamma} n})$ storage. In contrast, we focus on *Boolean* functions, and obtain both logarithmic seed length/storage and $O(\log\log n)$ time.

Small-Bias Generators. A pseudorandom generator maps a short random seed into a longer output that is indistinguishable from the uniform distribution to a certain class of tests (or distinguishers). A small-bias generator is one that "fools" linear tests. Namely, G is an ε-biased generator if for every non-zero test vector $\mathbf{t}$, it holds that:

$$\Pr_{seed}[\langle \mathbf{t}, G(seed)\rangle = 0] \in [1/2 - \varepsilon, 1/2 + \varepsilon] \tag{1.1}$$

(the inner product is taken over $\mathbb{GF}(2)$). The notion of ε-biased generators was introduced by the seminal work of [NN93], who also gave the first constructions of such generators. Since then, a rich sequence of works provided alternative constructions and related notions [AGHP92, AIK$^+$90, RSW93], [AMN98, EGL$^+$98, AM95, BATS13]. Small-bias generators have found numerous applications throughout theoretical computer science, from derandomization [NN93], to learning theory [AM95], to efficient low-degree tests and short PCPs [BFLS91, FGL$^+$96, BSGH$^+$06]. In coding theoretic terms, they are equivalent to linear error correcting codes over $\mathbb{GF}(2)$ where all codewords have relative Hamming weight between $(1-\varepsilon)/2$ and $(1+\varepsilon)/2$ [NN93, AGHP92]. They are also an important tool in finding explicit constructions, e.g. for graphs [Nao92, AR94, MW04], two-source extractors [Raz05], and psuedorandom generators that fool low degree polynomials [BV10, Lov09, Vio09] or Read-Once CNFs [GMR$^+$12], and many more.

One especially important motivation for the study of small-bias generators, is their application to the construction of almost k-wise independent hash functions. For this application, [NN93] introduced the relaxed notion of (k, ε)-biased generators, which are only required to fool linear tests of weight at most k (i.e. vectors with at most k non-zero coordinates). Any (k, ε)-biased generator immediately implies an $(\varepsilon \cdot 2^{k/2})$-almost k-wise independent Boolean hash family. Thus, to construct good almost k-wise independent hash functions, we need a (k, ε)-biased generator with bias smaller than $2^{-k/2}$. Indeed, Construction 1 above is obtained by building a new fast $(\log n, \mathrm{poly}(1/n))$-biased generator, whose output bits (and blocks) can be computed in few word operations.

Known constructions of ε-biased generators achieve asymptotically optimal seed length of $O(\log(n/\varepsilon))$ up to constant factors, [NN93, AGHP92]. Moreover, [NN93] gave a generator with constant bias, where each of the output bits could be computed using $O(1)$ word operations. However, for smaller biases, e.g. bias $\mathrm{poly}(1/n)$ (which is necessary for obtaining almost $\log n$-wise independent hash functions), known construction are not nearly so efficient. In particular, they require $O(\log n)$ word operations for computing each output bit. This is true even for the relaxed requirement of $(\log n, \varepsilon)$-bias.

Construction 2 (ε-Biased Generator). *We construct a $(1/\mathrm{poly}(n))$-biased generator G with seed length $O(\log n \log\log n)$, where each bit of G's output can be computed in $O((\log\log n)^2)$ word operations.*

Construction 2 gives an exponential efficiency improvement over prior constructions, at the cost of a $\log\log n$ blowup in the seed length. This is our most technically elaborate construction. It utilizes almost $\log n$-wise independent hash families (see above), and load balancing hash families (see below).

Significant attention has been devoted to the circuit or arithmetic complexity of computing ε-biased generators, yielding generators computable in circuit classes such as NC^0, $\mathrm{AC}^0[\oplus]$, or low-degree polynomials [CM01, GV04, AIK06, MST06, HV06, Hea08, Shp09]. These works, like ours, contend with the fundamental question of efficiently computing pseudorandom objects. Our work, however, focuses on complexity in terms of running time in the word-RAM model, a very different model (which is arguably more relevant for many applications and prevalent architectures). Indeed, as noted above, the original work of Naor and Naor [NN93] also considered the word-RAM model (see above). Another difference is in showing that word RAM running-time measure allows for efficient constructions of a rich variety of pseudorandom objects. This is not the case if we restrict locality or algebraic degree: for example, no generator of constant degree or locality can have super-polynomial stretch [MST06].

Given the ubiquitous nature of ε-biased generators and almost k-wise independent hash functions, we get similar improvements for several constructions (see the full version for details):

- Efficient Pseudorandom Generators (PRGs) for low-degree polynomials over $\mathbb{GF}(2)$: The works of [BV10, Lov09, Vio09] show that sum of small-bias generators fool low-degree polynomials over $\mathbb{GF}(2)$. As a result we get very efficient PRGs for this well-studied class of test functions.

- Efficient PRGs for Read-Once CNFs: The work of [GMR+12] uses small-bias generators to fool read-once CNFs and hence we get very efficient PRGs for read-once CNFs.

Load Balancing. A fundamental fact in the analysis of randomized algorithms is that if n items are hashed into n bins, using a truly random hash function, then with high probability each bin contains at most $O(\log n / \log \log n)$ items. It is natural to try and construct succinct explicit functions that share this important property. It is known that $O(\log n / \log \log n)$-wise independent hash families (with $\log n$ output bits) have this property, but these require a large $O(\log^2 n / \log \log n)$-bit seed, and evaluation takes $O(\log n / \log \log n)$ word operations. Until recently no constructions with smaller seed length were known (variants of this question were posed in [ADM+99, PPR07]). In recent work, Celis *et al.* [CRSW13] construct a family with seed length $O(\log n \log \log n)$, obtaining the first improvement over the seed length of a generic $\log n$-wise independent hash family. That construction did not improve the evaluation time, but they also present a second construction, which supports evaluation in $\sqrt{\log n}$ operations. While the seed length of this second construction increased to $\log^{3/2} n$, it was the first construction with a sub-polynomial seed that beat the evaluation time of $O(\log n)$. Indeed, in light of Siegel's lower bound [Sie04], this second construction gives a separation between the number of operations needed to compute load balancing and $\log n$-wise independence.

In this work we obtain an exponential efficiency improvement (even with respect to the second construction of [CRSW13]), while maintaining the best known seed length (that of their first construction).

Construction 3 (Load-Balancing Hash Family). *We construct a family of load-balancing hash functions with seed length $O(\log n \log \log n)$, which can be evaluated in $O((\log \log n)^2)$ word operations.*

In fact, this is an instantiation of the [CRSW13] construction, using the fast almost-independent hash functions of Construction 1. The efficient block computation property of Construction 1 is essential for this instantiation.

The Unit-Cost Word RAM Model. Throughout this work, we consider the *unit-cost Word RAM model* in which the elements are taken from a universe of size u, and each element can be stored in a single word of length $w = O(\log u)$ bits. Throughout this work we take the universe size to be poly(n) and the word length to be $O(\log n)$ (the standard setting of parameters).

The unit cost RAM model has been the subject of much research, and is considered the standard model for analyzing the efficiency of data structures and hashing schemes; see, e.g., [DP08, Hag98, HMP01, Mil99, PP08] and the references therein. In this model, it is assumed that a certain set of operations on words come at unit cost (an abstraction for the set of instructions supported by a CPU). This set of supported operations is an important part of the model, both from a practical and from a foundational point of view. Indeed, for a given algorithm, its running time may vary greatly depending on the set of unit-cost operations. We aim for a "minimal" model, and assume that the following (standard) operations can be executed in constant time on w-bit operands: bitwise Boolean operations, parity, left and right bit shifts by an arbitrary number of positions. We also assume constant-time addition, subtraction and multiplication over finite fields (we refer to these as *field operations* throughout). Our main results are stated assuming support for field operations over $\mathbb{GF}(2^w)$. Our constructions can also be instantiated over prime fields $\mathbb{GF}(p)$ (using addition and multiplication only), at the cost of an additional $O(\log\log n)$-overhead. See the full version for details on results over $\mathbb{GF}(p)$.

We elaborate on these two variants of supported field operations. Addition and subtraction over any finite field, and multiplication over a prime field $\mathbb{GF}(p)$, are all standard operations. Multiplication over $\mathbb{GF}(2^w)$, which requires taking the polynomial product modulo an irreducible polynomial, is used in a wide range of applications (and implementations) including error correcting codes [RS60, LRPP09, RCL$^+$13] and cryptography [oSN01, Dwo07]. Notably, modern processors provide (partial) support for multiplication over binary fields [GK12].

We avoid more complex (and non-standard) field operations such as division, inversion, or exponentiation, which are considered to be expensive operations (see, e.g., [Ram96], [And96] for two works working in the unit-cost Word RAM model where special care is taken to avoid division). We note, however, that if we allow such powerful unit-cost operations, then simpler and more efficient solutions are known. For example, if we allow unit-cost *exponentiation*, then the "powering construction" of [AGHP92] is a poly$(1/n)$-biased generator that can be evaluated in $O(1)$ time. A construction obtaining a poly$(1/n)$-biased generator with $O(\log n)$ seed length, using $O(1)$ divisions, was communicated to us by Zuckerman [Zuc].

1.1 Overview of Constructions and Techniques

Our constructions combine both algebraic as well as combinatorial techniques from the works of Naor and Naor [NN93] and Alon *et al.* [AGHP92], in addition to employing the load-balancing hash function construction of to Celis *et al.* [CRSW13]. We start by constructing (almost) $O(\log n)$-wise independent Boolean hash functions, which can be used to construct hash functions for load balancing. We then combine the two constructions ($O(\log n)$-wise independence and load balancing) to construct poly$(1/n)$-biased sequences. Thus, the $O(\log n)$-wise independent family is used both directly, and as part of the load balancing construction. In other words, our construction can be viewed as going from ε-bias (against logarithmically sparse tests) to load balancing, and back again to (full-fledged) ε-bias.

Looking further "under the hood", the starting point of our construction of $O(\log n)$-wise independence is one of the ε-bias constructions from [AGHP92]. In this respect, our construction can be viewed as transforming a (not efficient enough) ε-biased generator to a more efficient $O(\log n)$-wise independent hash family, and back again to a (still efficient) ε-biased generator. (See the full version for an elaboration on this perspective.)

We proceed with a more detailed overview of constructions and techniques. We begin with our construction of $O(\log n)$-wise independence directly. As mentioned above, (almost) $\log n$-wise independence follows from fooling (logarithmically) sparse linear tests.

(k, ε)-Biased Generator. We wish to construct a generator G which fools k-sparse linear tests in the sense of satisfying Eq. (1.1) for vectors $\mathbf{t}$ which have at most k non-zero coordinates. For $m = O(k + \log(n/\varepsilon))$, the generator G stretches a $2m$-bit seed[1] to nm bits (alternatively, to n blocks of m output bits), where each m-bit output block can be computed very efficiently. The construction uses the field $\mathbb{GF}(2^m)$.[2] The generator's seed consists of two field elements $\beta, \gamma \in \mathbb{GF}(2^m)$, and it outputs n field elements, or nm bits in total. We index these output elements using an arbitrary subset $A \subseteq \mathbb{GF}(2^m)$ of size n. For $\alpha \in A$, the α-th output element is:

$$(G(\beta, \gamma))_\alpha \stackrel{\text{def}}{=} \gamma \cdot \sum_{i=0}^{k-1} (\alpha\beta)^i, \tag{1.2}$$

and we treat each of these field elements as an m-bit block.

Fast Bit and Block Computation. The main advantage of this construction is that the α-th block can be computed very efficiently, in $O(\log k)$ word operations (and so can each individual output bit). The efficient computation procedure uses the equality:

$$\begin{aligned}(G(\beta, \gamma))_\alpha &= \gamma \cdot \sum_{i=0}^{k-1} (\alpha\beta)^i = \gamma \cdot (1 + (\alpha\beta)) \cdot \sum_{i=0}^{\frac{k}{2}-1} ((\alpha\beta)^2)^i \\ &= \gamma \cdot (1+(\alpha\beta)) \cdot (1 + (\alpha\beta)^2) \cdot \sum_{i=0}^{\frac{k}{4}-1} ((\alpha\beta)^4)^i = \ldots = \gamma \cdot \prod_{j=0}^{\log k - 1} (1+(\alpha\beta)^{2^j})\end{aligned}$$

(w.l.o.g. we take k to be a power of 2)[3]. This final product can be computed using $O(\log k)$ operations on m-bit words: $2 \log k + 1$ multiplications and $\log k$ additions.

[1] If either $k = \Omega(\log n)$ or n is polynomial in $1/\varepsilon$ then the seed-length is optimal up to a constant factor. In general, though, the optimal dependence of the seed on n is additive $\log \log n$. See the full version for details.

[2] The construction can be revised, using a transformation due to Rao [Rao07], to use instead the field $\mathbb{GF}(p)$ for some prime p. This incurs a $\log \log n$ overhead in both efficiency and seed length. See the full version for more details. As this generator is the main building block we use, all other constructions in the paper can similarly be made to work with operations over $\mathbb{GF}(p)$.

[3] We can also write $G(\beta, \gamma) = \gamma \cdot ((\alpha\beta)^{k+1} - 1))/(\alpha\beta - 1)$ and compute it using $O(\log k)$ multiplications and one division. Here we try to avoid using divisions.

One parameter range of particular interest is a $(\log n, \text{poly}(1/n))$-biased generator. For this setting of parameters, each output block is of length $O(\log n)$, and can be computed using $O(\log \log n)$ word operations (on $(\log n)$-bit words). As discussed above, this generator immediately yields a $\text{poly}(1/n)$-almost $\log n$-wise independent Boolean hash family, where the hash functions can be evaluated in $O(\log \log n)$ time.

Proof of Pseudorandomness. We would like to prove that G is a (k, ε)-biased generator. Namely that G produces a distribution over nm-bit strings that fools k-sparse linear tests over $\mathbb{GF}(2)$. Towards this, we first show that when viewing the output of G as n field elements of $\mathbb{GF}(2^m)$, it fools any k-sparse linear test over $\mathbb{GF}(2^m)$. We then prove that *every* (k, ε)-biased generator for tests over $\mathbb{GF}(2^m)$ remains (k, ε)-biased when we interpret its output as bits. This is proven by showing that a k-sparse linear test over $\mathbb{GF}(2)$ can be "simulated" by a k-sparse linear test over $\mathbb{GF}(2^m)$.

The direct proof that G is a (k, ε)-biased generator is fairly simple. Still we describe the way we obtained the construction, as we hope that this approach may find further applications. Our starting point is an ε-biased generator due to [AGHP92] (slightly modified to generate ε-biased sequences over $\mathbb{GF}(2^m)$ rather than over $\mathbb{GF}(2)$). By itself, the generator is not efficient enough (as most output elements require $\Omega(\log n)$ operations to evaluate). To improve construction's efficiency, we use a reduction due to [NN93] from ε-bias to (k, ε)-bias. This reduction is used in [NN93] to *reduce the seed length.*[4] Our goal is quite different: we will use the reduction to *improve efficiency*, obtaining (k, ε)-biased sequences that require $\log k$ operations (rather than $\log n$). A priori, there is no reason to expect the reduction to have this consequence, but a careful instantiation of the reduction does work.

Almost k-Wise Independence and Load Balancing. The $(\log n, \text{poly}(1/n))$-biased generator obtained above directly implies a $\text{poly}(1/n)$-almost $\log n$-wise independent Boolean hash family, where functions in the family can be evaluated in $O(\log \log n)$ word operations. Moreover, because of the efficient block computation property, for every $t \in \{1, 2, 3, \ldots \log n\}$, we also get a $\text{poly}(1/n)$-almost $(\log n/t)$-wise independent hash family from $[n]$ to $\{0,1\}^t$, where the functions can be evaluated in $O(\log \log n)$ word operations (we emphasize that this is the cost to obtain *all* of the output bits simultaneously). For every such t, a hash function h is described by two $GF(2^m)$ elements β and γ and $h_{\beta,\gamma}(i)$ is defined to be the t-long suffix of $G(\beta, \gamma))_{\alpha_i}$, where α_i is the i'th field element in A.

Such almost k-wise independent functions are used in [CRSW13] to construct load-balancing hash functions. The [CRSW13] construction uses $\log \log n$ such hash functions (each of these hash functions is evaluated once). The w-th function, for $w \in [\log \log n]$, is taken from a family that is (almost) $O(\log n/2^w)$-wise independent with 2^w-bit outputs. We instantiate their construction with our new

[4] More specifically, the dependence on n in the seed of (k, ε)-biased sequences is better than the dependence for ε-bias sequences.

hash families, improving the evaluation time from $\tilde{O}(\log n)$ to $O((\log\log n)^2)$ word operations. The seed length remains $O(\log n \cdot \log\log n)$.

(Full-Fledged) ε-Biased Generator. Having used a $(\log n, \mathrm{poly}(1/n))$-biased generator to obtain fast load-balancing, we now use fast load-balancing (together with the $(\log n, \mathrm{poly}(1/n))$-biased generator again) to obtain a fast full-fledged ε-biased generator (for *all* tests, i.e. without any sparsity restriction). This part of our construction is inspired by the combinatorial techniques of [NN93]. Special care is needed to simultaneously preserve both the small error as well as the efficient computation. To handle this we introduce a stronger notion of load balancing we call "granular" load balancing, which may be of interest elsewhere. See the full version for details.

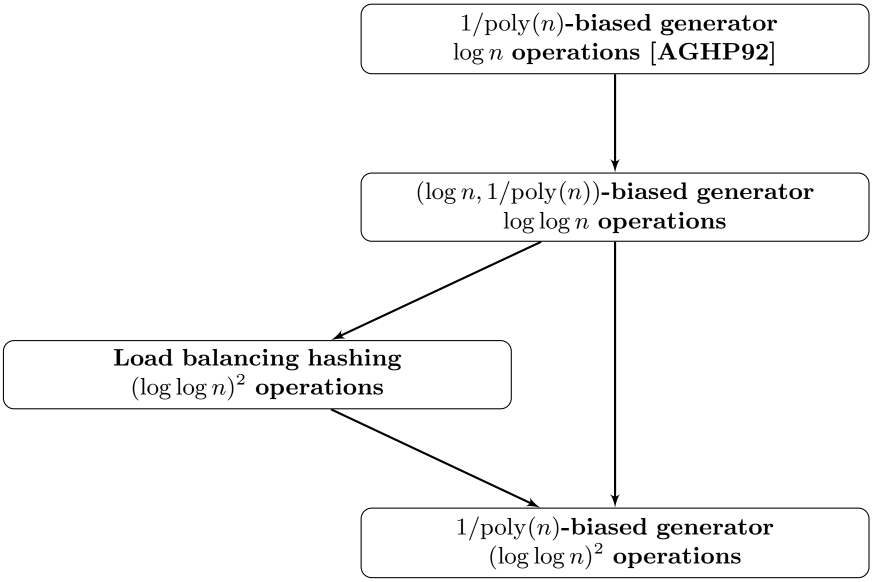

$1/\mathrm{poly}(n)$-**biased generator**
$\log n$ **operations [AGHP92]**
$(\log n, 1/\mathrm{poly}(n))$-**biased generator**
$\log\log n$ **operations**
Load balancing hashing
$(\log\log n)^2$ **operations**
$1/\mathrm{poly}(n)$-**biased generator**
$(\log\log n)^2$ **operations**

Fig. 1. Roadmap for the construction of the fast ε-biased generator

Acknowledgments. We would like to thank Udi Wieder and Moni Naor for useful discussions.

References

ADM$^+$99. Alon, N., Dietzfelbinger, M., Miltersen, P.B., Petrank, E., Tardos, G.: Linear hash functions. J. ACM 46(5), 667–683 (1999)

AGHP92. Alon, N., Goldreich, O., Håstad, J., Peralta, R.: Simple construction of almost k-wise independent random variables. Random Struct. Algorithms 3(3), 289–304 (1992)

AIK$^+$90. Ajtai, M., Iwaniec, H., Komlós, J., Pintz, J., Szemerédi, E.: Construction of a thin set with small Fourier coefficients. Bulletin of the London Mathematical Society 22, 583–590 (1990)

AIK06. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in NC^0. SIAM J. Comput. 36(4), 845–888 (2006)

AM95. Alon, N., Mansour, Y.: Epsilon-discrepancy sets and their application for interpolation of sparse polynomials. Inf. Process. Lett. 54(6), 337–342 (1995)

AMN98. Azar, Y., Motwani, R., Naor, J.: Approximating probability distributions using small sample spaces. Combinatorica 18(2), 151–171 (1998)

And96. Andersson, A.: Faster deterministic sorting and searching in linear space. In: FOCS, pp. 135–144 (1996)

AR94. Alon, N., Roichman, Y.: Random Cayley graphs and expanders. Random Struct. Algorithms 5(2), 271–285 (1994)

BATS13. Ben-Aroya, A., Ta-Shma, A.: Constructing small-bias sets from algebraic-geometric codes. Theory of Computing 9, 253–272 (2013)

BFLS91. Babai, L., Fortnow, L., Levin, L.A., Szegedy, M.: Checking computations in polylogarithmic time. In: STOC, pp. 21–31 (1991)

BSGH$^+$06. Ben-Sasson, E., Goldreich, O., Harsha, P., Sudan, M., Vadhan, S.P.: Robust PCPs of proximity, shorter PCPs, and applications to coding. SIAM J. Comput. 36(4), 889–974 (2006)

BV10. Bogdanov, A., Viola, E.: Pseudorandom bits for polynomials. SIAM J. Comput. 39(6), 2464–2486 (2010)

CM01. Cryan, M., Miltersen, P.B.: On pseudorandom generators in NC. In: Sgall, J., Pultr, A., Kolman, P. (eds.) MFCS 2001. LNCS, vol. 2136, pp. 272–284. Springer, Heidelberg (2001)

CRSW13. Elisa Celis, L.: Omer Reingold, Gil Segev, and Udi Wieder. Balls and bins: Smaller hash families and faster evaluation. SIAM J. Comput. 42(3), 1030–1050 (2013)

CW79. Carter, L., Wegman, M.N.: Universal classes of hash functions. J. Comput. Syst. Sci. 18(2), 143–154 (1979)

DP08. Dietzfelbinger, M., Pagh, R.: Succinct data structures for retrieval and approximate membership (extended abstract). In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 385–396. Springer, Heidelberg (2008)

Dwo07. Dworkin, M.: Recommendations for Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. NIST Special Publication 800-38D (2007)

EGL$^+$98. Even, G., Goldreich, O., Luby, M., Nisan, N., Velickovic, B.: Efficient approximation of product distributions. Random Struct. Algorithms 13(1), 1–16 (1998)

FGL$^+$96. Feige, U., Goldwasser, S., Lovász, L., Safra, S., Szegedy, M.: Interactive proofs and the hardness of approximating cliques. J. ACM 43(2), 268–292 (1996)

GK12. Gueron, S., Kounavis, M.E.: Intel® carry-less multiplication instruction and its usage for computing the GCM mode, rev 2.01 (2012)

GMR$^+$12. Gopalan, P., Meka, R., Reingold, O., Trevisan, L., Vadhan, S.P.: Better pseudorandom generators from milder pseudorandom restrictions. In: FOCS, pp. 120–129 (2012)

GV04. Gutfreund, D., Viola, E.: Fooling parity tests with parity gates. In: Jansen, K., Khanna, S., Rolim, J.D.P., Ron, D. (eds.) RANDOM and APPROX 2004. LNCS, vol. 3122, pp. 381–392. Springer, Heidelberg (2004)

Hag98. Hagerup, T.: Sorting and searching on the word RAM. In: Meinel, C., Morvan, M. (eds.) STACS 1998. LNCS, vol. 1373, pp. 366–398. Springer, Heidelberg (1998)

Hea08. Healy, A.: Randomness-efficient sampling within nc^1. Computational Complexity 17(1), 3–37 (2008)

HMP01. Hagerup, T., Miltersen, P.B., Pagh, R.: Deterministic dictionaries. J. Algorithms 41(1), 69–85 (2001)

HV06. Healy, A., Viola, E.: Constant-depth circuits for arithmetic in finite fields of characteristic two. In: Durand, B., Thomas, W. (eds.) STACS 2006. LNCS, vol. 3884, pp. 672–683. Springer, Heidelberg (2006)

Lov09. Lovett, S.: Unconditional pseudorandom generators for low degree polynomials. Theory of Computing 5(1), 69–82 (2009)

LRPP09. Lacan, J., Roca, V., Peltotalo, J., Peltotalo, S.: Reed-Solomon Forward Error Correction (FEC) Schemes. RFC 5510 (Proposed Standard) (April 2009)

LW05. Luby, M., Wigderson, A.: Pairwise independence and derandomization. Foundations and Trends in Theoretical Computer Science 1(4) (2005)

Mil99. Miltersen, P.B.: Cell probe complexity - a survey. In: 19th Conference on the Foundations of Software Technology and Theoretical Computer Science (FSTTCS), 1999. Advances in Data Structures Workshop (1999)

MST06. Mossel, E., Shpilka, A., Trevisan, L.: On epsilon-biased generators in NC^0. Random Struct. Algorithms 29(1), 56–81 (2006)

MW04. Meshulam, R., Wigderson, A.: Expanders in group algebras. Combinatorica 24(4), 659–680 (2004)

Nao92. Naor, M.: Constructing Ramsey graphs from small probability spaces. IMB Research Report RJ(8810) (1992)

NN93. Naor, J., Naor, M.: Small-bias probability spaces: Efficient constructions and applications. SIAM J. Comput. 22(4), 838–856 (1993)

oSN01. National Institute of Standards and Technology (NIST). Federal information processing standards publication (FIPS 197). Advanced Encryption Standard (AES) (2001)

PP08. Pagh, A., Pagh, R.: Uniform hashing in constant time and optimal space. SIAM J. Comput. 38(1), 85–96 (2008)

PPR07. Pagh, A., Pagh, R., Ruzic, M.: Linear probing with constant independence. In: STOC, pp. 318–327 (2007)

Ram96. Raman, R.: Priority queues: Small, monotone and trans-dichotomous. In: Díaz, J. (ed.) ESA 1996. LNCS, vol. 1136, pp. 121–137. Springer, Heidelberg (1996)

Rao07. Rao, A.: An exposition of Bourgain's 2-source extractor. Electronic Colloquium on Computational Complexity (ECCC) 14(034) (2007)

Raz05. Raz, R.: Extractors with weak random seeds. In: STOC, pp. 11–20 (2005)

RCL$^+$13. Roca, V., Cunche, M., Lacan, J., Bouabdallah, A., Matsuzono, K.: Simple Reed-Solomon Forward Error Correction (FEC) Scheme for FECFRAME. RFC 6865 (Proposed Standard) (February 2013)

RRW14. Reingold, O., Rothblum, R.D., Wieder, U.: Pseudorandom graphs in data structures (manuscript, 2014)

RS60. Reed, I.S., Solomon, G.: Polynomial codes over certain finite fields. Journal of the Society for Industrial and Applied Mathematics 8(2), 300–304 (1960)

RSW93. Razborov, A.A., Szemerédi, E., Wigderson, A.: Constructing small sets that are uniform in arithmetic progressions. Combinatorics, Probability & Computing 2, 513–518 (1993)

RV14. Reingold, O., Vardi, S.: Tighter bounds for local computations (manuscript, 2014)

Shp09. Shpilka, A.: Constructions of low-degree and error-correcting epsilon-biased generators. Computational Complexity 18(4), 495–525 (2009)

Sie04. Siegel, A.: On universal classes of extremely random constant-time hash functions. SIAM J. Comput. 33(3), 505–543 (2004)

Tho13. Thorup, M.: Simple tabulation, fast expanders, double tabulation, and high independence. In: FOCS (2013)

Vio09. Viola, E.: The sum of d small-bias generators fools polynomials of degree d. Computational Complexity 18(2), 209–217 (2009)

WC81. Wegman, M.N., Carter, L.: New hash functions and their use in authentication and set equality. J. Comput. Syst. Sci. 22(3), 265–279 (1981)

Zuc. Zuckerman, D.: Personal Communication

Determining Majority in Networks with Local Interactions and Very Small Local Memory

George B. Mertzios[1,⋆], Sotiris E. Nikoletseas[2,⋆⋆],
Christoforos L. Raptopoulos[2,3,⋆⋆⋆], and Paul G. Spirakis[2,4,†]

[1] School of Engineering and Computing Sciences, Durham University, UK
[2] Computer Technology Institute (CTI) and University of Patras, Greece
[3] Computer Science Department, University of Geneva, Switzerland
[4] Department of Computer Science, University of Liverpool, UK
george.mertzios@durham.ac.uk, nikole@cti.gr,
raptopox@ceid.upatras.gr, p.spirakis@liverpool.ac.uk

Abstract. We study here the problem of determining the majority type in an arbitrary connected network, each vertex of which has initially two possible types (states). The vertices may have a few additional possible states and can interact in pairs only if they share an edge. Any (population) protocol is required to stabilize in the initial majority, i.e. its output function must interpret the local state of each vertex so that each vertex outputs the initial majority type. We first provide a protocol with 4 states per vertex that *always* computes the initial majority value, under any fair scheduler. Under the uniform probabilistic scheduler of pairwise interactions, we prove that our protocol stabilizes in expected polynomial time for any network and is quite fast on the clique. As we prove, this protocol is optimal, in the sense that there does not exist any population protocol that always computes majority with fewer than 4 states per vertex. However this does not rule out the existence of a protocol with 3 states per vertex that is correct with high probability (whp). To this end, we examine an elegant and very natural majority protocol with 3 states per vertex, introduced in [2] where its performance has been analyzed for the clique graph. In particular, it determines the correct initial majority type in the clique very fast and whp under the uniform probabilistic scheduler. We study the performance of this protocol in arbitrary networks. We prove that, when the two initial states are put uniformly at random on the vertices, the protocol of [2] converges to the initial majority with probability higher than the probability of converging to the initial minority. In contrast, we present an infinite family of graphs, on which the protocol of [2] can fail, i.e. it can converge to the initial minority type whp, even when the difference between the initial majority and the initial minority is $n - \Theta(\ln n)$. We also present another infinite family of graphs in which the protocol of [2] takes an expected

[⋆] Partially supported by the EPSRC Grant EP/K022660/1.
[⋆⋆] Partially supported by the MULTIPLEX project – 317532.
[⋆⋆⋆] Partially supported by the SHARPEN project – PE6 (1081).
[†] Partially supported by the MULTIPLEX project – 317532, the EU ERC Project ALGAME and the EEE/CS School of the University of Liverpool.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 871–882, 2014.

exponential time to converge. These two negative results build upon a very positive result concerning the robustness of the protocol of [2] on the clique, namely that if the initial minority is at most $\frac{n}{7}$, the protocol fails with exponentially small probability. Surprisingly, the resistance of the clique to failure causes the failure in general graphs. Our techniques use new domination and coupling arguments for suitably defined processes whose dynamics capture the antagonism between the states involved.

1 Introduction

One of the most natural computational problems in many physical systems is to compute the *majority*, i.e. to determine accurately which type of an element of the system appears more frequently. For instance, the majority problem is encountered in various settings such as in voting [8, 10], in epidemiology and interacting particles systems [13], in diagnosis of multiprocessor systems [17], in social networks [14,16] etc. In distributed computing, the majority problem is an important and natural special case of the central problem of reaching *consencus* within a system [6,12], where a number of processes have to agree on any single data value (e.g. leader election [7]). In all these physical systems, some pairs of elements may interact with each other while other pairs may not be able to interact directly. This structure of the possible pairwise interactions between elements of the system can be modeled by a network (i.e. graph), where elements and possible interactions are represented by vertices and edges, respectively.

In order to solve the majority computation problem in a network, we first need to make some assumptions on the underlying model of computation. Much research has been done under the assumption that there exists a central authority, as well as unlimited available memory and full information about the whole network (see e.g. [5, 18]). However, in many real systems we do not have (or we do not wish to have) such a powerful computational model. The weaker the considered model of computation is (e.g. lack of central authority, partial or no information about the system, lack of memory etc.), the more challenging the majority computation becomes.

One of the ways to study distributed systems where agents may interact in pairs and each individual agent is extremely limited (in fact, being equipped only with a finite number of possible states) is by using *population protocols* [1, 3]. Then the complex behavior of the system emerges from the rules governing the possible pairwise interactions of the agents. Population protocols have been defined by analogy to population processes [11] in probability theory and have already been used in various fields, such as in statistical physics, genetics, epidemiology, chemistry and biology [4].

In particular, population protocols are *scalable*, i.e. they work independently of the size n of the underlying network (called the *interaction graph*) and the value of n is not even known to the protocol. Furthermore they are *anonymous*, i.e. there is only one transition function which is common to all entities/agents: the result of an interaction of an agent u at state q_u with an agent v at state q_v is the same regardless of the identity of u and v. The transition function of a population protocol only specifies the result of every possible interaction, without specifying *which* pairs of agents interact or *when* they are chosen to interact. Usually it is assumed that interactions between agents happen under some kind of a *fairness* condition. For a survey we refer to [3].

In this direction, a very natural and simple population protocol for the majority problem on the clique (i.e. the complete graph), where initially every vertex has one of two possible types (states), has been introduced and analyzed in [2]. In particular, the protocol of [2] assigns only 3 possible states to every agent (i.e. there is a 3×3 transition table capturing all possible interactions) and the interactions between agents are dictated by a *probabilistic scheduler* (i.e. all pairs have the same probability to interact at any step). Every vertex has an identity v, but it is unaware of the identity of any other vertex, as well as of its own identity. Although the underlying interaction graph in [2] is assumed to be a clique, the authors distinguish in their protocol the agents u and v participating in an interaction into an "initiator" and a "responder" of the interaction (when agents u and v interact, each of them becomes initiator or responder with equal probability). Their main result is that, if initially the difference between the initial majority from the initial minority in the complete graph with n vertices is $\omega(\sqrt{n}\log n)$, their protocol converges to the correct initial majority value in $O(n \log n)$ time with high probability.

Most works on population and majority dynamics so far considered only two entity types (e.g. the voter model [8], the Moran process [15]). The analysis of population dynamics with more than two types is challenging. As an example we refer to the model of [2], in which, although agents can have initially one of only two types (red and green), the protocol itself allows every agent to be in one among three different states (red, green and blank) at every subsequent time point. Even though this model is quite simple, it is very hard to be analyzed. Computing the majority with as few states as possible in the more general case, where the interaction graph has an arbitrary structure (as opposed to the complete graph that has been mainly considered so far) remained an open problem.

1.1 Our Contribution

In this paper we study the majority problem in an *arbitrary* underlying interaction graph G, where initially every vertex has two possible states (red and green). We consider here the weakest and simplest possible model of computation. In particular, we assume the existence of no central authority and we allow every vertex of G to have only a (small) constant number of available types (or states). Although every vertex of G has a unique identity, no vertex is aware of its own identity or the identity of any other vertex. Furthermore, although only two adjacent vertices can interact, vertices of G do not even know to which other vertices they are adjacent.

First, we focus on the problem of *always* computing the correct majority value in an *arbitrary* (directed or undirected) interaction graph G, regardless of how large the initial difference between the majority and the minority is. In particular, assuming that the interacting pairs of vertices are chosen by an arbitrary fair scheduler, we derive matching lower and upper bounds on the number of available states, for which there exists a population protocol that always computes the correct majority value. For the lower bound, we prove that there does not exist any population protocol that achieves this with at most 3 different states per vertex. On the other hand, for the matching upper bound we provide a population protocol with 4 states per vertex, which always computes the correct majority value, even if initially the difference between majority and minority is 1. To the best of our knowledge, this is the first 4-state population protocol that correctly computes the majority value in a two type population on an arbitrary interaction graph. In particular, the 4-state majority protocol proposed in [3] only works when the interaction graph is complete. Furthermore we provide polynomial upper bounds on the expected time needed by our new protocol to converge, and we show that in certain cases the running time is $O(n \log n)$, i.e. the same as for the fast protocol of [2].

Second, we provide a detailed analysis of the 3-state protocol of [2] on an arbitrary interaction graph G. Our first result in this direction is that, when the two initial types (red and green) are distributed on the vertices of an arbitrary graph G uniformly at random, the protocol of [2] will converge to the initial majority with higher probability than to the initial minority. The proof of this relies on a well known result in extremal combinatorics (in particular, on Hall's marriage Theorem). Furthermore we present an infinite family of graphs $\{G_n\}_{n\in\mathbb{N}}$ on which the protocol of [2] can fail (i.e. it can converge to the initial minority) with high probability, even when the difference between the initial majority and the initial minority is as large as $n - \Theta(\ln n)$. Then we present another infinite family of graphs $\{G'_n\}_{n\in\mathbb{N}}$ on which the protocol of [2] can take an exponential expected number of steps to converge. In particular, this rules out the possibility to use a Markov chain Monte-Carlo approach to approximate the probability that the protocol of [2] converges to the correct majority value.

In order to prove our results on the classes $\{G_n\}_{n\in\mathbb{N}}$ and $\{G'_n\}_{n\in\mathbb{N}}$, we first proved the intermediate result that for any $\varepsilon > 0$, if the minority has size at most $(\frac{1}{7} - \varepsilon)n$ in the complete graph with n vertices, then the protocol of [2] converges to the initial minority with exponentially small probability. The latter result shows that, although the performance of the protocol of [2] can drop significantly when the interaction graph G is not the complete graph, it is quite robust when G is the complete graph. Our proof concerning the robustness of the protocol of [2] in the complete graph is novel and uses a non-trivial coupling argument which can be of independent interest.

2 The Model and Notation

A *population protocol* consists of a finite set Q of states/types[1], a finite set of input symbols X, an input function $\iota : X \to Q$, a finite set of output symbols Y, an output function $\gamma : Q \to Y$, and a joint transition function $\delta : Q \times Q \to Q \times Q$.

[1] In the original formulation of population protocols these are called *states*, but we chose to also use the term *type* to avoid confusion with the states in a Markov chain.

If, for any pair of states $q_1, q_2 \in Q$, $\delta(q_a, q_b) = (q'_a, q'_b)$ implies that $\delta(q_b, q_a) = (q'_b, q'_a)$, then the population protocol is called *symmetric*. A population protocol is executed by a fixed finite *population* of agents with types in Q. We assume that each agent has an identity $v \in V$, but agents are oblivious to their own identity and to identities of agents they interact with.

Initially, each agent is assigned a type according to an *input* $x : V \to X$ that maps agent identities to input symbols. In the general population protocol model, agents are identified with the vertices of an *interaction graph*, whose edges indicate the possible agent interactions that may take place. Here, an interaction graph is a simple connected graph G (i.e. without loops or multiple edges), which can be directed or undirected. A function $C : V \to Q$ is called a *configuration*. We will say that the population protocol reaches configuration C at time t if, for every agent v in the population, the state of v at time t is $C(v)$.

In the original model, agents do not send messages or share memory; instead, an interaction between two agents updates both of their types according to a joint transition function (which can be also represented by a table). Interactions between agents are planned by a scheduler under a general "fairness" condition; the actual mechanism for choosing which agents interact is abstracted away. The fairness condition states that for any two configurations C, C', if C occurs infinitely often and C' is reachable from C, then C' also occurs infinitely often.

In this paper, we consider a special case of a fair scheduler, namely the *probabilistic scheduler*, which is defined on directed graphs as follows. During each execution step, a directed edge $(v, u) \in E$ is chosen uniformly at random from E, where v (i.e. the tail of (v, u)) is called the *initiator* and u (i.e. the head of (v, u)) is called the *responder* of the interaction. Then, agents v and u update their types jointly according to δ. In particular, if v is of type q_v and u is of type q_u, the type of v (respectively u) becomes q'_v (respectively q'_u), where $(q'_v, q'_u) = \delta(q_v, q_u)$. The types of all other agents remain unchanged. The probabilistic scheduler is defined on undirected graphs similarly, by replacing every undirected edge $\{v, u\}$ by the two directed edges (v, u) and (u, v). That is, in an undirected graph G, the probabilistic scheduler selects first an undirected edge uniformly at random and then it selects equiprobably one of its endpoint as the initiator. Note that a symmetric protocol does not distinguish between initiators and responders. Thus, if the protocol is symmetric, the probabilistic scheduler on undirected graphs just chooses at each execution step one undirected edge uniformly at random and lets its endpoints interact according to the transition function.

Given the probabilistic scheduler, a population protocol *computes* a (possibly partial) function $g : X^V \to Y$ with error probability at most ϵ, if for all $x \in g^{-1}(Y)$, the population eventually reaches a configuration C that satisfies the following properties with probability at least $1 - \epsilon$: (a) all agents agree on the correct output, i.e. $g(x) = \gamma(C(v))$ for all $v \in V$ and (b) this holds also for every configuration reachable from C. A population protocol *stably computes* a (possibly partial) function $g : X^V \to Y$ if, for *every* fair scheduler, the population eventually reaches a configuration C that satisfies both properties (a) and (b).

Observation 1. *If a symmetric population protocol stably computes a function on an undirected interaction graph G, then it also stably computes the same function on a directed interaction graph G' that comes from G by assigning to every edge of G one or two directions.*

2.1 Majority with High Probability on the Clique

Angluin et al. [2] proposed a population protocol for computing majority with high probability (whp) in the case where the interaction graph is a clique. Their protocol uses just 3 types $Q = \{b, g, r\}$. For convenience, we will sometimes refer to these types as the *blank, green* and *red* type respectively. The joint transition function δ is given by:

$$\delta(x, y) = \begin{cases} (x, x), & \text{if } x = y \text{ or } y = b \\ (x, b), & \text{if } (x, y) \in \{(g, r), (r, g)\}. \end{cases} \tag{1}$$

One of the main results in [2] is that if the underlying interaction graph is a clique K_n (i.e. the complete graph on n vertices) and interactions are planned according to the above probabilistic scheduler, then with high probability $1 - o(1)$, the above 3-type majority protocol converges to the initial majority value if the initial difference between the majority and the minority is $\omega(\sqrt{n} \log n)$.

2.2 Representation

Using the probabilistic scheduler to plan agent interactions has the advantage that we can describe evolution by using a discrete time Markov chain $\mathcal{M}$. For general interaction graphs, the state space $\mathcal{S}$ of $\mathcal{M}$ can have up to $|V|^{|Q|}$ states, namely one for each configuration $C : V \to Q$. Specifically for the model of [2], we denote by W_t (respectively R_t and G_t) the set of agents in state b (respectively r and g) at time t. Note that if the interaction graph has a high degree of symmetry, then $\mathcal{S}$ can be reduced significantly. One such example is the clique K_n, in which case we can describe a state of $\mathcal{M}$ by the tuple $(|R_t|, |G_t|)$ (where we have also used the fact that $|W_t| = n - |G_t| - |R_t|$).

3 At Least 4 Types Are Needed for Majority

Definition 1 (rank). *For any population protocol P, denote by $Q(P)$ the set of types used by P. Let $\mathcal{P}_g$ be a class of population protocols that stably compute the function g. The* rank *of $\mathcal{P}_g$ is*

$$R(\mathcal{P}_g) \stackrel{def}{=} \min_{P \in \mathcal{P}_g} |Q(P)|. \tag{2}$$

In this section, we prove that 3 states are not sufficient to stably compute the majority function in any 2-type population and for any interaction graph.

Theorem 1. *Let $\mathcal{P}_{majority}$ be the class of population protocols that stably compute the majority function in any 2-type population of agents and for any interaction graph. Then $R(\mathcal{P}_{majority}) > 3$.*

4 Computing Majority in Arbitrary Interaction Graphs

In this section we introduce a symmetric population protocol with 4 states (called the *ambassador protocol*) which, given an *arbitrary* undirected graph $G = (V, E)$ as the underlying interaction graph of the population, *stably computes* the majority of the types of the vertices of G (even if the majority differs only by one from the minority). Assuming that the input symbols are g (for green) and r (for red), the set of states in the ambassador protocol is $Q = \{(g, 0), (g, 1), (r, 0), (r, 1)\}$. The input function ι is such that $\iota(g) = (g, 1)$ and $\iota(r) = (r, 1)$. The output function γ is such that $\gamma((g, i)) = g$ and $\gamma((r, i)) = r$, where $i \in \{0, 1\}$. Finally, for simplicity of the presentation, we present the transition function δ in the form of a table in Figure 1.

$u \setminus v$	$(g,0)$	$(g,1)$	$(r,0)$	$(r,1)$
$(g,0)$	$-$	$((g,1),(g,0))$	$-$	$((r,1),(r,0))$
$(g,1)$	$((g,0),(g,1))$	$-$	$((g,0),(g,1))$	$((g,0),(r,0))$
$(r,0)$	$-$	$((g,1),(g,0))$	$-$	$((r,1),(r,0))$
$(r,1)$	$((r,0),(r,1))$	$((r,0),(g,0))$	$((r,0),(r,1))$	$-$

Fig. 1. The transition matrix of the ambassador model

The transition matrix of Figure 1 can be interpreted as follows. Every row and every column is labeled by a state of Q. The cell that belongs to the row labeled with state $q_1 \in Q$ and to the column labeled with state $q_2 \in Q$ has either the symbol "$-$" or an ordered pair of states $(q'_1, q'_2) \in Q \times Q$. In the cases where this cell has the ordered pair (q'_1, q'_2), then $\delta(q_1, q_2) = (q'_1, q'_2)$, otherwise $\delta(q_1, q_2) = (q_1, q_2)$. Note that the ambassador protocol is defined on undirected interaction graphs, i.e. in every interaction there is no initiator or responder. That is, whenever $\delta(q_1, q_2) = (q'_1, q'_2)$ then $\delta(q_2, q_1) = (q'_2, q'_1)$.

The main idea of the ambassador protocol can be described as follows. The first component of the state of a vertex u denotes the color of u. That is, whenever u is at state (q, i) (resp. (r, i)), where $i \in \{0, 1\}$, this is interpreted as "u is currently colored green (resp. red)". Furthermore, if the second component of the state of u is 1 (resp. 0), this is interpreted as "u has an ambassador" (resp. "u has no ambassador"). Whenever two vertices u and v interact during the execution of the protocol, a "battle" takes place between the colors of u and v, where the "ambassadors" of u and v (whenever they exist) try to spread their own color to the other vertex, in the following sense:

- Assume that u and v have different colors. If both u and v have an ambassador, then they both keep their own colors in the next step, but both their ambassadors disappear. If u has an ambassador and v has no ambassador, then v takes the color of u in the next step and the ambassador now moves from u to v. If both u and v have no ambassador then their state remains the same at the next step.
- Assume that both u and v have the same color; then they both maintain their color in the next step. If they both have (or if they both do not have)

an ambassador, then their state remains the same at the next step. If one of them (say u) has an ambassador and the other one (say v) does not have one, then the ambassador moves from u to v.

The correctness of the ambassador protocol under the assumption of an *arbitrary* fair scheduler and its efficiency under the probabilistic scheduler is proved in the next two theorems.

Theorem 2. *Given any (un)directed graph, if there exists initially a majority, then the* 4*-state ambassador protocol stably computes the initial majority value.*

Theorem 3. *Let G be an arbitrary connected undirected interaction graph with n vertices. Assuming the probabilistic scheduler, if initially there are k red vertices and $\ell \neq k$ green vertices, then the expected time until the* 4*-state ambassador protocol converges is $O(n^6)$. If, additionally, the interaction graph is the complete graph K_n, then the expected time until the protocol converges is $O\left(\frac{\ln n}{|k-\ell|}n^2\right)$.*

From Theorem 3, we can see that the running time of our 4-state protocol in the case where the difference between the majority and the minority is $\Theta(n)$ is $O(n \ln n)$, which is comparable to the running time of the fast 3-state protocol of [2].

5 The Model of Angluin et al. on Arbitrary Graphs

In this section, we provide a detailed analysis of the 3-state protocol of [2] on arbitrary interaction graphs G. In particular, in Subsection 5.1, we present our result concerning the random initial placement of individuals on the vertices of the interaction graph. In Subsection 5.2, we prove our auxiliary result that, when the minority is sufficiently small, the probability that the protocol of Angluin et al. fails in computing the majority value is exponentially small. Although this result shows the robustness of the protocol of [2] in the clique, we use it as an intermediate step in proving in Subsection 5.3 that there exists a family of graphs in which the protocol can fail with high probability. Finally, in Subsection 5.4, we prove the existence of a family of graphs in which the protocol of [2] can take an exponential expected number of steps to reach consensus.

5.1 Random Initial Placement

In the next theorem we provide a sufficient condition under which the majority protocol described in [2] correctly identifies the initial majority with probability at least $\frac{1}{2}$. This result is in wide contrast to the negative result of Subsection 5.3 (cf. Theorem 6), in which we highlight a case where the majority protocol of [2] fails with high probability. The proof of the next theorem is based on Hall's marriage Theorem (cf. chapter 5, [9]):

Theorem 4. *For any strongly connected directed graph G, if the initial assignment of individuals to the vertices of G is random, then the majority protocol described in [2] correctly identifies the initial majority with probability at least $\frac{1}{2}$.*

5.2 Clique

By the discussion in Subsection 2.2, the state space of the Markov chain $\mathcal{M}$ describing the evolution of the protocol at time t contains tuples of the form $(|R_t|, |G_t|)$, where R_t (resp. G_t) is the set of vertices of type r (resp. g) at time t. In particular, in this subsection we are interested in upper bounding $\Pr\{\text{absorption at } (n, 0)|\text{initially at } (\epsilon n, n - \epsilon n)\}$. The core of our proof lies in the definition of two discrete time processes $\mathcal{W}$ and $\mathcal{C}$ that "filter" the information from the original Markov chain $\mathcal{M}$.

Definition 2 (The Blank Process $\mathcal{W}$). *This process keeps track of the number of blank vertices over time, i.e.* $\mathcal{W}(t) \stackrel{def}{=} \langle \#\ \textit{vertices of type } b \textit{ at time } t\rangle$.

For convenience, we will use the following notation to describe transitions of $\mathcal{M}$: We will write $g \to r$ to describe a transition of the form $(x, y) \to (x-1, y)$, for some $x, y \in \{1, 2, \dots, n\}$. More specifically, $g \to r$ is used to describe a transition where a directed edge (v, u) is chosen by the scheduler, v is of type g, and u is of type r. Similarly, we will use $r \to g$ for transitions of the form $(x, y) \to (x, y-1)$, $g \to b$ for transitions of the form $(x, y) \to (x, y+1)$ and $r \to b$ for transitions of the form $(x, y) \to (x+1, y)$. We note that the state of $\mathcal{M}$ at any time t can be fully described by the initial state and by a sequence of transitions among $\{g \to r, r \to g, g \to b, r \to b\}$.

Definition 3 (The Contest Process $\mathcal{C}$). *Transitions of $\mathcal{M}$ are paired recursively, starting from time 0 as follows: Every transition that increases the number of blanks is paired with the earliest subsequent transition that decreases the number of blanks and is not paired yet.* [2] *For an arbitrary time t, we denote by $\tau(t)$ (or just τ for short) the number of pairs until time t. The Contest Process $\mathcal{C}$ is defined over the time scale τ, where $\mathcal{C}(0) = |R_0|$ and for $\tau = 1, 2, \dots,$*

$$\mathcal{C}(\tau) = \begin{cases} \mathcal{C}(\tau-1)+1, & \textit{if the } \tau\textit{-th pair is } (r \to g, r \to b), \\ \mathcal{C}(\tau-1)-1, & \textit{if the } \tau\textit{-th pair is } (g \to r, g \to b) \textit{ and} \\ \mathcal{C}(\tau-1), & \textit{otherwise.} \end{cases} \tag{3}$$

Notice that the processes $\mathcal{W}$ and $\mathcal{C}$ are dependent. As a matter of fact, $\mathcal{C}$ is not even defined using the same time scale as $\mathcal{W}$ and $\mathcal{M}$ (to indicate this, we have used the convention that t is the time variable for processes $\mathcal{M}, \mathcal{W}$, while τ is the time variable for process $\mathcal{C}$). However, observe that if we initially begin with no blanks (i.e. $|R_0| + |G_0| = n$ hence $\mathcal{W}(0) = 0$), then whenever $\mathcal{W}$ decreases its value, we have a transition step of $\mathcal{C}$.

Lemma 1 (Relating $\mathcal{C}$ and $\mathcal{M}$). *For any $T \in \mathbb{N}$, denote by $\mathcal{C}_{|T}$ the value of $\mathcal{C}$ given only states $\mathcal{M}(t), t = 0, 1, \dots, T$ (i.e. given the history of $\mathcal{M}$ up to time T). Then, $\mathcal{C}_{|T} \geq |R_T|$ for any $T \in \mathbb{N}$. Furthermore, if $\mathcal{C}_{|T} = 0$, then all vertices are of type g.*

[2] We assume that the pairing concerns only transitions that change the state of $\mathcal{M}$. In particular, transitions of the form $b \to r, b \to g, b \to b, g \to g$ and $r \to r$ are ignored in this pairing as irrelevant.

Lemmas 2 and 3 below concern the domination of processes $\mathcal{W}$ and $\mathcal{C}$ by appropriate birth-death processes and are used in the proof of Theorem 5.

Lemma 2 (Domination of $\mathcal{W}$). *Let $\alpha, \beta, \kappa \in \{1, \ldots, n-1\}$, with $\alpha < \beta$. Let also $\mathcal{B}_\mathcal{W}$ be a birth-death process, which has state space $\mathcal{S}_{\mathcal{B}_\mathcal{W}} = \{S_0, \ldots, S_n\}$, with S_n an absorbing state and transition probability matrix P, with $P(S_i, S_{i+1}) = 1$ for all $i \in \{0, \ldots, \alpha\} \cup \{\beta, \ldots, n-1\}$, $\frac{P(S_i,S_{i+1})}{P(S_i,S_{i-1})} = \frac{2\kappa}{\alpha}$ for all $i \in \{\alpha+1, \ldots, \beta-1\}$ and $P(S_i, S_i) = \Pr(\mathcal{W}(t) = i|\mathcal{W}(t-1) = i)$, for all $t \geq 1$ and for all $i \in \{\alpha+1, \ldots, \beta-1\}$. Then, given that the vertices of type r are at most κ, the process $\mathcal{W}$ is stochastically dominated by $\mathcal{B}_\mathcal{W}$ in the following sense: $\Pr(\mathcal{W}(t) > x|\mathcal{W}(0) = 0) \leq \Pr(\mathcal{B}_\mathcal{W}(t) \in \cup_{y>x} S_y|\mathcal{B}_\mathcal{W}(0) = S_0)$, for any time t and $x \in \{0, \ldots, n\}$.*

Lemma 3 (Domination of $\mathcal{C}$). *Let β, κ be positive integers, with $\beta+\kappa < n$. Let also $\mathcal{B}_\mathcal{C}$ be a birth-death process, which has state space $\mathcal{S}_{\mathcal{B}_\mathcal{C}} = \{T_0, \ldots, T_n\}$, with T_0, T_n absorbing states and transition probability matrix Q, with $Q(T_i, T_{i+1}) = 1$ for all $i \in \{\kappa, \ldots, n-1\}$, $\frac{Q(T_i,T_{i+1})}{Q(T_i,T_{i-1})} = \frac{\kappa}{n-\beta-\kappa}$ for all $i \in \{1, \ldots, \kappa-1\}$ and $Q(T_i, T_i) = \Pr(\mathcal{C}(\tau) = i|\mathcal{C}(\tau-1) = i)$, for all $\tau \geq 1$ and for all $i \in \{1, \ldots, \kappa-1\}$. Then, given that the vertices of type b are at most β, the process $\mathcal{C}$ is stochastically dominated by $\mathcal{B}_\mathcal{C}$ in the following sense: $\Pr(\mathcal{C}(\tau) > x|\mathcal{C}(0) = |R_0|) \leq \Pr(\mathcal{B}_\mathcal{C}(\tau) \in \cup_{y>x} T_y|\mathcal{B}_\mathcal{C}(0) = |R_0|)$, for any τ and $x \in \{0, \ldots, n\}$.*

The main Theorem of this subsection is stated below.

Theorem 5. *Let $\epsilon < \frac{1}{7}$. For large enough n, starting from ϵn agents of type r and $(1-\epsilon)n$ agents of type g on the clique K_n, the probability that the clique eventually contains only agents of type r is at most $e^{-\Theta(n)}$.*

We note here that the upper bound on ϵ in the statement of Theorem 5 is only used to facilitate exposition of our arguments in the proof. We claim that this upper bound can be increased further by using the same proof ideas, but that we cannot reach arbitrarily close to $\frac{1}{2}$. However, we conjecture that the constant ϵ in Theorem 5 can be as as large as $\frac{1}{2} - \epsilon'$, where $\epsilon' > 0$ is a small constant. We also conjecture that starting from a single agent of type r on the clique K_n, the probability that the clique eventually contains only agents of type r is at least $e^{-\Theta(n)}$.

5.3 Minority Domination

Consider the lollipop graph, which consists of a complete graph K_{n_1} on n_1 vertices, among which vertex v is connected to the leftmost vertex u of a line graph L_{n_2} on n_2 vertices, with $n_1 + n_2 = n$. In this subsection we prove that the protocol of [2] may fail with high probability in the lollipop graph.

Theorem 6. *Consider a lollipop graph on n vertices, which consists of a complete graph K_{n_1} on $n_1 \geq 100 \ln n$ vertices, among which vertex v is connected to the leftmost vertex u of a line graph L_{n_2} on $n_2 = n - n_1$ vertices. Suppose that initially vertex v and all vertices in L_{n_2} are of type r, while all vertices in $K_{n_1} \backslash \{v\}$ are of type g. Then with high probability, eventually only vertices of type g will remain in the graph.*

The proof builds upon the proof of Theorem 5 and also uses the following fact: given a line graph L_m, in which the leftmost vertex is of type g and all other $m-1$ vertices are of type r, the probability that the protocol of [2] eventually reaches a configuration in which all vertices are of type g is $\frac{1}{2(m-1)}$. For the proof of Theorem 6, we define stochastic processes $\mathcal{W}'$ and $\mathcal{C}'$ as the processes $\mathcal{W}$ and $\mathcal{C}$ from Subsection 5.2, but concerning the clique K_{n_1}. These processes take into account only transitions involving either edges that belong to the clique or the directed edge (u, v). We note that Lemma 1 continues to hold for the modified process $\mathcal{C}'$ with only one modification (because of the existence of the edge $\{v, u\}$) concerning its second part: T must satisfy $\mathcal{C}'_{|T} = 0$ and $\mathcal{C}'_{|T-1} > 0$ in order to be able to deduce that K_{n_1} has only vertices of type g.

5.4 Expected Exponential Time to Absorption

In this subsection we prove that the protocol of [2] can take an expected exponential time of steps to reach consensus in the case where the underlying graph consists of two disjoint cliques on n_1 and n_2 vertices each with a single edge between them (see Figure 2(a)). The main part of the proof is dedicated in proving an upper bound on the expected number of steps needed for the protocol to reach consensus in the case of a directed graph H, consisting of a single clique K_{n_1} and a vertex v outside the clique, which is connected to a vertex u of the clique with a directed edge (v, u) (see Figure 2(b)). In particular, similarly to Subsection 5.3, we define stochastic processes $\mathcal{W}''$ and $\mathcal{C}''$ as the processes $\mathcal{W}$ and $\mathcal{C}$ from Subsection 5.2, but concerning the clique K_{n_1}. We then prove suitably modified versions of Lemmas 2 and 3 and we apply results on birth-death processes.

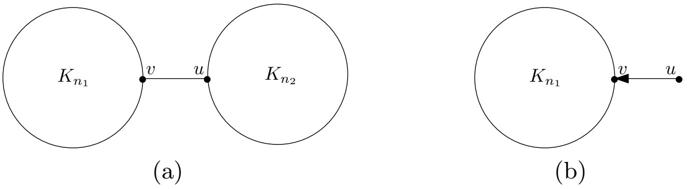


Fig. 2. (a) An interaction graph G with 2 disjoint K_{n_1} and K_{n_2}, connected with a single edge $\{u, v\}$. (b) The graph H consisting of a clique K_{n_1} with an arc (u, v).

Theorem 7. *Let G be an interaction graph on n vertices, which consists of 2 disjoint cliques on n_1 and n_2 vertices each and a single edge between them (see Figure 2(a)). Suppose that initially, all vertices in the K_{n_1} clique are of type g, while all vertices in the K_{n_2} clique are of type r. Then the majority protocol of [2] needs an expected exponential number of steps to reach consensus.*

References

1. Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M.J., Peralta, R.: Computation in networks of passively mobile finite-state sensors. Distributed Computing 18, 235–253 (2006)
2. Angluin, D., Aspnes, J., Eisenstat, D.: A simple population protocol for fast robust approximate majority. Distributed Computing 21(2), 87–102 (2008)
3. Aspnes, J., Ruppert, E.: An introduction to population protocols. In: Garbinato, B., Miranda, H., Rodrigues, L. (eds.) Middleware for Network Eccentric and Mobile Applications, pp. 97–120. Springer (2009)
4. Cook, M., Soloveichik, D., Winfree, E., Bruck, J.: Programmability of chemical reaction networks. In: Condon, A., Harel, D., Kok, J.N., Salomaa, A., Winfree, E. (eds.) Algor. Bioprocesses. Nat. Comp. Series, pp. 543–584. Springer (2009)
5. De Marco, G., Pelc, A.: Randomized algorithms for determining the majority on graphs. Combinatorics, Probability and Computing 15(6), 823–834 (2006)
6. DeGroot, M.H.: Reaching a consensus. Journal of the American Statistical Association 69(345), 118–121 (1974)
7. Fischer, M., Jiang, H.: Self-stabilizing leader election in networks of finite-state anonymous agents. In: Shvartsman, M.M.A.A. (ed.) OPODIS 2006. LNCS, vol. 4305, pp. 395–409. Springer, Heidelberg (2006)
8. Holley, R.A., Liggett, T.M.: Ergodic theorems for weakly interacting infinite systems and the voter model. The Annals of Probability 3, 643–663 (1975)
9. Jukna, S.: Extremal Combinatorics with Applications to Computer Science. Springer (2011)
10. Kearns, M., Tan, J.: Biased voting and the democratic primary problem. In: Papadimitriou, C., Zhang, S. (eds.) WINE 2008. LNCS, vol. 5385, pp. 639–652. Springer, Heidelberg (2008)
11. Kurtz, T.G.: Approximation of Population Processes. Society for Industrial and Applied Mathematics (1987)
12. Lamport, L., Shostak, R., Pease, M.: The byzantine generals problem. ACM Trans. Program. Lang. Syst. 4(3), 382–401 (1982)
13. Liggett, T.M.: Interacting Particle Systems. Springer (2004)
14. Mizrachi, A.: Majority vote and monopolies in social networks. Master's thesis, Department of Communication Systems Engineering, Faculty of Engineering, Ben-Gurion University of the Negev (2013)
15. Moran, P.A.P.: Random processes in genetics. Mathematical Proceedings of the Cambridge Philosophical Society 54(1), 60–71 (1958)
16. Mossel, E., Neeman, J., Tamuz, O.: Majority dynamics and aggregation of information in social networks. Autonomous Agents and Multi-Agent Systems 28(3), 408–429 (2014)
17. Preparata, F.P., Metze, G., Chien, R.T.: On the connection assignment problem of diagnosable systems. IEEE Trans. on Electr. Computers 16, 848–854 (1967)
18. Saks, M., Werman, M.: On computing majority by comparisons. Combinatorica 11(4), 383–387 (1991)

Lower Bounds for Oblivious Subspace Embeddings

Jelani Nelson[1] and Huy L. Nguyễn[2]

[1] Harvard University, Cambridge MA 02138, USA
minilek@seas.harvard.edu
[2] Princeton University, Princeton NJ 08540, USA
hlnguyen@princeton.edu

Abstract. An *oblivious subspace embedding (OSE)* for some $\varepsilon, \delta \in (0, 1/3)$ and $d \leq m \leq n$ is a distribution $\mathcal{D}$ over $\mathbb{R}^{m \times n}$ such that

$$\mathop{\mathbb{P}}_{\Pi \sim \mathcal{D}} (\forall x \in W,\ (1-\varepsilon)\|x\|_2 \leq \|\Pi x\|_2 \leq (1+\varepsilon)\|x\|_2) \geq 1 - \delta$$

for any linear subspace $W \subset \mathbb{R}^n$ of dimension d. We prove any OSE with $\delta < 1/3$ has $m = \Omega((d + \log(1/\delta))/\varepsilon^2)$, which is optimal. Furthermore, if every Π in the support of $\mathcal{D}$ is sparse, having at most s non-zero entries per column, we show tradeoff lower bounds between m and s.

1 Introduction

A *subspace embedding* for some $\varepsilon > 0$ and linear subspace W is a matrix Π s.t.

$$\forall x \in W,\ (1-\varepsilon)\|x\|_2 \leq \|\Pi x\|_2 \leq (1+\varepsilon)\|x\|_2.$$

An *oblivious subspace embedding (OSE)* for some $\varepsilon, \delta \in (0, 1/3)$ and $d \leq m \leq n$ is a distribution $\mathcal{D}$ over $\mathbb{R}^{m \times n}$ s.t. for any d-dimensional subspace $W \subset \mathbb{R}^n$

$$\mathop{\mathbb{P}}_{\Pi \sim \mathcal{D}} (\forall x \in W,\ (1-\varepsilon)\|x\|_2 \leq \|\Pi x\|_2 \leq (1+\varepsilon)\|x\|_2) \geq 1 - \delta. \tag{1}$$

That is, for any linear subspace $W \subset \mathbb{R}^n$ of bounded dimension, a random Π drawn according to $\mathcal{D}$ is a subspace embedding for W with good probability.

OSE's were first introduced in [17] and have since been used to provide fast approximate randomized algorithms for numerical linear algebra problems such as least squares regression [5,12,14,17], low rank approximation [4,5,14,17], minimum margin hyperplane and minimum enclosing ball [16], and approximating leverage scores [11]. For example, consider the least squares regression problem: given $A \in \mathbb{R}^{n \times d}, b \in \mathbb{R}^n$, compute

$$x^* = \mathrm{argmin}_{x \in \mathbb{R}^d} \|Ax - b\|_2.$$

The optimal solution x^* is such that Ax^* is the projection of b onto the column span of A. Thus by computing the singular value decomposition (SVD) $A =$

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 883–894, 2014.

$U\Sigma V^T$ where $U \in \mathbb{R}^{n\times r}, V \in \mathbb{R}^{d\times r}$ have orthonormal columns and $\Sigma \in \mathbb{R}^{r\times r}$ is a diagonal matrix containing the non-zero singular values of A (here r is the rank of A), we can set $x^* = V\Sigma^{-1}U^Tb$ so that $Ax^* = UU^Tb$ as desired. Given that the SVD can be approximated in time $\tilde{O}(nd^{\omega-1})$[1] [7, Section 6.2] where $\omega < 2.373\ldots$ is the exponent of square matrix multiplication [20], we can solve the least squares regression problem in this time bound. We note [7] only discusses computing the SVD of $d \times d$ square matrices in time $\tilde{O}(d^\omega)$. However, this can be extended to $n \times d$ rectangular matrices $A = U\Sigma V^T$ in the stated time by (1) computing the QR decomposition $A = QR$ in time $O(nd^{\omega-1})$ [7, Section 4.1], then (2) computing the SVD of the square matrix R in time $\tilde{O}(d^\omega)$.

A simple argument [5, Theorem 13] then shows that if one instead computes

$$\tilde{x} = \text{argmin}_{x\in\mathbb{R}^d} \|\Pi Ax - \Pi b\|_2$$

for some subspace embedding Π for the $(d+1)$-dimensional subspace spanned b and the columns of A, then $\|A\tilde{x} - b\|_2 \le (1 + O(\varepsilon))\|Ax^* - b\|_2$, i.e. $\tilde{x}$ serves as a near-optimal solution to the original regression problem. The running time then becomes $\tilde{O}(md^{\omega-1})$, which can be a large savings for $m \ll n$, plus the time to compute ΠA and Πb and the time to find Π.

It is known that a random gaussian matrix with $m = O((d + \log(1/\delta))/\varepsilon^2)$ is an OSE (see for example the net argument in [5]). While this leads to small m, and furthermore Π is oblivious to A, b so that its computation is "for free", the time to compute ΠA is $\tilde{O}(mnd^{\omega-2})$, which is worse than solving the original least squares problem. Sarlós constructed an OSE $\mathcal{D}$, based on the fast Johnson-Lindenstrauss transform of Ailon and Chazelle [1], with the properties that (1) $m = \tilde{O}(d/\varepsilon^2)$, and (2) for any vector $y \in \mathbb{R}^n$ and Π in the support of $\mathcal{D}$, Πy can be computed in time $O(n\log n)$. This implies an approximate least squares regression algorithm running in time $O(nd\log n) + \tilde{O}(d^\omega/\varepsilon^2)$.

A recent line of work sought to improve the $O(nd\log n)$ term above to a quantity that depends only on the sparsity of the matrix A as opposed to its ambient dimension. The works [5,12,14] give an OSE with $m = O(d^2/\varepsilon^2)$ where every Π in the support of the OSE has only $s = 1$ non-zero entry per column. The work [14] also showed how to achieve $m = O(d^{1+\gamma}/\varepsilon^2), s = \text{poly}(1/\gamma)/\varepsilon$ for any constant $\gamma > 0$. Using these OSE's together with other optimizations (for details see [5, Theorems 13, 19]), these works imply approximate regression algorithms running in time $O(\text{nnz}(A) + (d^3 \log d)/\varepsilon^2)$ (the $s = 1$ case), or $O_\gamma(\text{nnz}(A) + d^{\omega+\gamma}/\varepsilon^2)$ or $O_\gamma((\text{nnz}(A) + d^2)\log(1/\varepsilon) + d^{\omega+\gamma})$ (the case of larger s).

As seen above we now have several upper bounds, though our understanding of lower bounds for the OSE problem is lacking. Any subspace embedding, and thus any OSE, must have $m \ge d$ since otherwise some non-zero vector in the subspace will be in the kernel of Π and thus not have its norm preserved. Furthermore, it quite readily follows from the works [10,13] that any OSE must have $m = \Omega(\min\{n, \log(d/\delta)/\varepsilon^2\})$ (see Corollary 1). Thus the best known lower bound to date is $m = \Omega(\min\{n, d+\varepsilon^{-2}\log(d/\delta)\})$, while the best upper bound is $m = O(\min\{n, (d+\log(1/\delta))/\varepsilon^2\})$ (the OSE supported only on the $n\times n$ identity

[1] We say $g = \tilde{O}(f)$ when $g = O(f \cdot \text{polylog}(f))$.

matrix is indeed an OSE with $\varepsilon = \delta = 0$). If one is willing to introduce the very slight restriction that $n = \Omega(d \log(nd)/\varepsilon)$ (the $\log(nd)$ term ideally should not appear on the right hand side), then a streaming regression lower bound of [4] implies the improved lower bound $m = \Omega(d/\varepsilon)$. We remark that although many problems (regression [17], low-rank approximation [17], approximating leverage scores [11], and k-means clustering [3]) can make use of OSE's with distortion $1+\varepsilon_0$ for some constant ε_0 to achieve $(1+\varepsilon)$-approximation to the final problem, this is not always true. For example, in [2] Andoni and Nguyễn give a one-pass streaming algorithm for ℓ_1 heavy hitters of the eigenvalues of a real symmetric matrix updated in the turnstile streaming model. That is, $\varepsilon, \phi \in (0, 1/2)$ are given up front, and $A \in \mathbb{R}^{n\times n}$ receives updates of the form "add/subtract value v to entry (i, j) of A". At the end of a sequence of updates, we would like to output the eigenvalues of A with multiplicative error $1+\varepsilon$ and additive error $\varepsilon^2 \phi S_1^{1/\phi}(A) + |\lambda_{1/\phi}|$, where λ_k is the kth largest eigenvalue of A in magnitude, and $S_1^k(A)$ is the sum of magnitudes of all but the top k eigenvalues. Their solution is to maintain $\Pi A \Pi^T$ in the stream, with $\Pi \in \mathbb{R}^{m\times n}$ satisfying two properties: (1) being an ε-subspace embedding for a particular subspace of dimension $1/\phi$, and (2) being an $O(1)$-subspace embedding for at most n different m/q-dimensional subspaces simultaneously (which can be achieved by setting $\delta < 1/n$ and performing a union bound) for any desired $q \le m$; the space complexity though increases polynomially with q. Thus it is important to understand the required dependence on ε since it affects the setting of m for (1), which then also affects (2), which directly translates into the final space complexity.

Our contribution I: We show for any $\varepsilon, \delta \in (0, \frac{1}{3})$, any OSE with distortion $1+\varepsilon$ and error probability δ has $m = \Omega(\min\{n, (d+\log(1/\delta))/\varepsilon^2\})$, which is optimal.

We also make progress in understanding the tradeoff between m and s. The work [15] observed via a simple reduction to nonuniform balls and bins that any OSE with $s = 1$ must have $m = \Omega(d^2)$. Also recall the upper bound of [14] of $m = O(d^{1+\gamma}/\varepsilon^2), s = \mathrm{poly}(1/\gamma)/\varepsilon$ for any constant $\gamma > 0$.

Our contribution II: We show that for δ a fixed constant and $n > 100d^2$, any OSE with $m = o(\varepsilon^2 d^2)$ must have $s = \Omega(1/\varepsilon)$. Thus a phase transition exists between sparsity $s = 1$ and super-constant sparsity somewhere around m being d^2. We also show that for $m < d^{1+\gamma}$ and $\gamma \in ((10 \log\log d)/(\alpha \log d), \alpha/4)$ and $2/(\varepsilon\gamma) < d^{1-\alpha}$, for any constant $\alpha > 0$, it must hold that $s = \Omega(\alpha/(\varepsilon\gamma))$. Thus the $s = \mathrm{poly}(1/\gamma)/\varepsilon$ dependence of [14] is correct (although our lower bound requires $m < d^{1+\gamma}$ as opposed to $m < d^{1+\gamma}/\varepsilon^2$).

Note as mentioned above for eigenvalue heavy hitters, (2) requires $\Pi \in \mathbb{R}^{m\times n}$ to be a subspace embedding for dimension m/q, and the space increases polynomially with q. Thus if Π has at most $1/\gamma$ non-zero entries per column, then q must be $d^{\Omega(\gamma)}$, which translates into $d^{\Omega(\gamma)}$ multiplicatively increased space complexity for that algorithm.

Our proof in the first contribution follows Yao's minimax principle combined with concentration arguments and Cauchy's interlacing theorem. Our proof in the second contribution uses a bound for nonuniform balls and bins and the simple fact that for *any* distribution over unit vectors, two i.i.d. samples are not negatively correlated in expectation.

1.1 Notation

We often abbreviate "orthonormal" as o.n. We let $O^{n\times d}$ denote the set of $n \times d$ real matrices with o.n. columns. For a linear subspace $W \subseteq \mathbb{R}^n$, we let $\mathbf{proj}_W : \mathbb{R}^n \to W$ denote the projection operator onto W. That is, if the columns of U form an o.n. basis for W, then $\mathbf{proj}_W x = UU^T x$. In the case that A is a matrix, we let $\mathbf{proj}_A$ denote the projection onto the column span of A. Throughout this document, unless otherwise specified all norms $\|\cdot\|$ are $\ell_2 \to \ell_2$ operator norms in the case of matrix argument, and ℓ_2 norms for vector arguments. The norm $\|A\|_F$ denotes Frobenius norm, i.e. $(\sum_{i,j} A_{i,j}^2)^{1/2}$. For a matrix A, $\kappa(A)$ denotes the condition number of A, i.e. the ratio of the largest to smallest singular value. We use $[n]$ for integer n to denote $\{1,\ldots,n\}$. We use $A \lesssim B$ to denote $A \le CB$ for some absolute constant C, and similarly for $A \gtrsim B$.

2 Dimension Lower Bound

Let $U \in O^{n\times d}$ be such that the columns of U form an o.n. basis for a d-dimensional linear subspace W. Then the condition in Eq. (1) is equivalent to all singular values of $\varPi U$ lying in the interval $[1-\varepsilon, 1+\varepsilon]$. Thus for any such U an OSE has $\kappa(\varPi U) \le 1+\varepsilon$ with probability $1-\delta$ over the randomness of $\varPi$. Thus $\mathcal{D}$ being an OSE implies the condition

$$\forall U \in O^{n\times d} \underset{\varPi\sim\mathcal{D}}{\mathbb{P}} (\kappa(\varPi U) > 1+\varepsilon) < \delta \tag{2}$$

We now show a lower bound for m in any distribution $\mathcal{D}$ satisfying Eq. (2) with $\delta < 1/3$. Our proof will use a couple lemmas. The first is quite similar to the Johnson-Lindenstrauss lemma itself. Without the appearance of the matrix D, it would follow from the analyses in [6,9] using Gaussian symmetry. The proof uses the Hanson-Wright inequality [8] and is omitted in this version.

Lemma 1. *Let u be a unit vector drawn at random from S^{n-1}, and let $E \subset \mathbb{R}^n$ be an m-dimensional linear subspace for some $1 \le m \le n$. Let $D \in \mathbb{R}^{n\times n}$ be a diagonal matrix with smallest singular value σ_{min} and largest singular value σ_{max}. Then for any $0 < \varepsilon < 1$*

$$\underset{u}{\mathbb{P}} \left(\|\mathbf{proj}_E D u\|^2 \notin (\tilde{\sigma}^2 \pm \varepsilon\sigma_{max}^2) \cdot \frac{m}{n} \right) \lesssim e^{-\Omega(\varepsilon^2 m)}$$

for some $\sigma_{min} \le \tilde{\sigma} \le \sigma_{max}$.

The next lemma follows from Cauchy's interlacing theorem [18, Exercise 1.3.14]:

Lemma 2. *Suppose $A \in \mathbb{R}^{n\times m}, A' \in \mathbb{R}^{(n+1)\times m}$ such that $n+1 \le m$ and the first n rows of A, A' agree. Then the singular values of A, A' interlace. That is, if the singular values of A are $\sigma_1, \dots, \sigma_n$ and those of A' are $\beta_1, \dots, \beta_{n+1}$,*

$$\beta_1 \le \sigma_1 \le \beta_2 \le \sigma_2 \le \dots \le \beta_n \le \sigma_n \le \beta_{n+1}.$$

Lastly, we need the following corollary of [10, Theorem 9]. A proof is omitted in this version. A similar conclusion can be obtained using [13], but requiring the assumption that $d < n^{1-\gamma}$ for some constant $\gamma > 0$.

Corollary 1. *Any OSE $\mathcal{D}$ over $\mathbb{R}^{m\times n}$ must have $m = \Omega(\min\{n, \varepsilon^{-2}\log(d/\delta)\})$.*

Now we prove the main theorem of this section.

Theorem 1. *Let $\mathcal{D}$ be any OSE with $\varepsilon, \delta < 1/3$. Then $m = \Omega(\min\{n, d/\varepsilon^2\})$.*

Proof. We assume $d/\varepsilon^2 \le cn$ for some constant $c > 0$. Our proof uses Yao's minimax principle. Thus we must construct a distribution $\mathcal{U}_{hard}$ such that

$$\mathop{\mathbb{P}}_{U\sim\mathcal{U}_{hard}}(\kappa(\Pi_0 U) > 1+\varepsilon) < \delta. \tag{3}$$

cannot hold for any $\Pi_0 \in \mathbb{R}^{m\times n}$ which does not satisfy $m = \Omega(d/\varepsilon^2)$. The particular $\mathcal{U}_{hard}$ we choose is as follows: we let the d columns of U be independently drawn uniform random vectors from the sphere, post-processed using Gram-Schmidt to be orthonormal. That is, the columns of U are an o.n. basis for a random d-dimensional linear subspace of $\mathbb{R}^n$.

Let $\Pi_0 = LDW^T$ be the singular value decomposition (SVD) of Π_0, i.e. $L \in O^{m\times n}, W \in O^{n\times n}$, and D is $n\times n$ with $D_{i,i} \ge 0$ for all $1 \le i \le m$, and all other entries of D are 0. Note that $W^T U$ is distributed identically as U, which is identically distributed as $W'U$ where W' is an $n\times n$ block diagonal matrix with two blocks. The upper-left block of W' is a random rotation $M \in O^{m\times m}$ according to Haar measure. The bottom-right block of W' is the $(n-m)\times(n-m)$ identity matrix. Thus it is equivalent to analyze the singular values of the matrix $LDW'U$. Also note that left multiplication by L does not alter singular values, and the singular values of $DW'U$ and $D'MA^TU$ are identical, where A is the $n\times m$ matrix whose columns are $e_1, \dots, e_m$. Also D' is an $m\times m$ diagonal matrix with $D'_{i,i} = D_{i,i}$. Thus we wish to show that if m is sufficiently small, then

$$\mathop{\mathbb{P}}_{M\sim O^{m\times m}, U\sim\mathcal{U}_{hard}}\left(\kappa(D'MA^TU) > 1+\varepsilon\right) > \frac{1}{3} \tag{4}$$

Henceforth in this proof we assume for the sake of contradiction that $m \le c\cdot\min\{d/\varepsilon^2, n\}$ for some small positive constant $c > 0$. Also note that we may assume by Corollary 1 that $m = \Omega(\min\{n, \varepsilon^{-2}\log(d/\delta)\})$.

Assume that with probability strictly larger than 2/3 over the choice of U, we can find unit vectors z_1, z_2 so that $\|A^TUz_1\|/\|A^TUz_2\| > 1+\varepsilon$. Now suppose we have such z_1, z_2. Define $y_1 = A^TUz_1/\|A^TUz_1\|, y_2 = A^TUz_2/\|A^TUz_2\|$. Then a random $M \in O^{m\times m}$ has the same distribution as $M'T$, where M' is i.i.d. as M, and T can be any distribution over $O^{m\times m}$, so we write $M = M'T$. T may even depend on U, since $M'U$ will then still be independent of U and a random rotation (according to Haar measure). Let T be the $m \times m$ identity matrix with probability 1/2, and R_{y_1,y_2} with probability 1/2 where R_{y_1,y_2} is the reflection across the bisector of y_1, y_2 in the plane containing these two vectors, so that $R_{y_1,y_2}y_1 = y_2, R_{y_1,y_2}y_2 = y_1$. Now note that for any fixed choice of M' it must be the case that $\|D'M'y_1\| \geq \|D'M'y_2\|$ or $\|D'M'y_2\| \geq \|D'M'y_1\|$. Thus $\|D'M'Ty_1\| \geq \|D'M'Ty_2\|$ occurs with probability 1/2 over T, and the reverse inequality occurs with probability 1/2. Thus for this fixed U for which we found such z_1, z_2, over the randomness of M',T we have $\kappa(D'MA^TU) \geq \|D'MA^TUz_1\|/\|D'MA^TUz_2\|$ is greater than $1+\varepsilon$ with probability at least 1/2. Since such z_1, z_2 exist with probability larger than 2/3 over choice of U, we have established Eq. (4). It just remains to establish the existence of such z_1, z_2.

Let the columns of U be $u^1, \ldots, u^d$, and define $\tilde{u}^i = A^Tu^i$, $\tilde{U} = A^TU$. Let U_{-d} be the $n\times(d-1)$ matrix whose columns are $u^1, \ldots, u^{d-1}$, and let $\tilde{U}_{-d} = A^TU_{-d}$. Write $A = A^{\|} + A^{\perp}$, where the columns of $A^{\|}$ are the projections of the columns of A onto the column span of U_{-d}, i.e. $A^{\|} = U_{-d}U_{-d}^TA$. Then

$$\|A^{\|}\|_F^2 = \|U_{-d}U_{-d}^TA\|_F^2 = \|\tilde{U}_{-d}\|_F^2 = \sum_{i=1}^{d-1}\sum_{r=1}^{m}(u_r^i)^2. \tag{5}$$

By Lemma 1 with $D = I$ and $E = \mathrm{span}(e_1, \ldots, e_m)$, followed by a union bound over the $d-1$ columns of U_{-d}, the right hand side of Eq. (5) is between $(1-C_1\varepsilon)(d-1)m/n$ and $(1+C_1\varepsilon)(d-1)m/n$ with probability at least $1-C(d-1)\cdot e^{-C'C_1\varepsilon^2 m}$ over the choice of U. This is $1-d^{-\Omega(1)}$ for $C_1 > 0$ sufficiently large since $m = \Omega(\varepsilon^{-2}\log d)$. Now, if $\kappa(\tilde{U}) > 1+\varepsilon$ then z_1, z_2 with the desired properties exist. Suppose for the sake of contradiction that both $\kappa(\tilde{U}) \leq 1+\varepsilon$ and $(1-C_1\varepsilon)(d-1)m/n \leq \|\tilde{U}_{-d}\|_F^2 \leq (1+C_1\varepsilon)(d-1)m/n$. Since the squared Frobenius norm is the sum of squared singular values, and since $\kappa(\tilde{U}_{-d}) \leq \kappa(\tilde{U})$ due to Lemma 2, all the singular values of $\tilde{U}_{-d}$, and hence $A^{\|}$, are between $(1-C_2\varepsilon)\sqrt{m/n}$ and $(1+C_2\varepsilon)\sqrt{m/n}$. Then by the Pythagorean theorem the singular values of $A^{\perp}$ are in the interval $[\sqrt{1-(1+C_2\varepsilon)^2m/n}, \sqrt{1-(1-C_2\varepsilon)^2m/n}] \subseteq [1-(1+C_3\varepsilon)m/n, 1-(1-C_3\varepsilon)m/n]$.

Since the singular values of $\tilde{U}$ and $\tilde{U}^T$ are the same, it suffices to show $\kappa(\tilde{U}^T) > 1+\varepsilon$. For this we exhibit two unit vectors x_1, x_2 with $\|\tilde{U}^Tx_1\|/\|\tilde{U}^Tx_2\| > 1+\varepsilon$. Let $B \in O^{m\times d-1}$ have columns forming an o.n. basis for the column span of AA^TU_{-d}. Since B has o.n. columns and u^d is orthogonal to colspan(U_{-d}),

$$\|\mathbf{proj}_{\tilde{U}_{-d}}\tilde{u}^d\| = \|BB^TA^Tu^d\| = \|B^TA^Tu^d\| = \|B^T(A^{\perp})^Tu^d\|.$$

Let $(A^{\perp})^T = C\Lambda E^T$ be the SVD, where $C \in \mathbb{R}^{m\times m}, \Lambda \in \mathbb{R}^{m\times m}, E \in \mathbb{R}^{n\times m}$. As usual C, E have o.n. columns, and Λ is diagonal with all entries in $[1-(1+C_3\varepsilon)m/n, 1-(1-C_3\varepsilon)m/n]$. Condition on U_{-d}. The columns of E form an o.n. basis for the column space of $A^{\perp}$, which is some m-dimensional subspace of the $(n-d+1)$-dimensional orthogonal complement of the column space of

U_{-d}. Meanwhile u^d is a uniformly random unit vector drawn from this orthogonal complement, and thus $\|E^T u^d\|^2 \in [(1-C_4\varepsilon)^2 m/(n-d+1), (1+C_4\varepsilon)^2 m/(n-d+1)] \subset [(1-C_5\varepsilon)m/n, (1+C_5\varepsilon)m/n]$ with probability $1-d^{-\Omega(1)}$ by Lemma 1 and the fact that $d \le \varepsilon n$ and $m = \Omega(\varepsilon^{-2}\log d)$. Note also that $\|\Lambda E^T u^d\| = \|\tilde{u}^d\| = (1 \pm C_6\varepsilon)\sqrt{m/n}$ with probability $1-d^{-\Omega(1)}$ since Λ has bounded singular values.

Also note $E^T u/\|E^T u\|$ is uniformly random in S^{m-1}, and also $B^T C$ has orthonormal rows since $B^T C C^T B = B^T B = I$, and thus again by Lemma 1 with E being the row space of $B^T C$ and $D = \Lambda$, we have $\|B^T C \Lambda E^T u\| = \Theta(\|E^T u\| \cdot \sqrt{d/m}) = \Theta(\sqrt{d/n})$ with probability $1 - e^{-\Omega(d)}$.

We first note that by Lemma 2 and our assumption on the singular values of $\tilde{U}_{-d}$, $\tilde{U}^T$ has smallest singular value at most $(1 + C_2\varepsilon)\sqrt{m/n}$. We then set x_2 to be a unit vector such that $\|\tilde{U}^T x_2\| \le (1 + C_2\varepsilon)\sqrt{m/n}$.

It just remains to construct x_1 so that $\|\tilde{U}^T x_1\| > (1 + \varepsilon)(1 + C_2\varepsilon)\sqrt{m/n}$. To construct x_1 we split into two cases:

Case 1 ($m \le cd/\varepsilon$): When $n = \Omega(d \log(nd)/\varepsilon)$, this case is ruled out by a regression lower bound of [4]. However, since we make no such assumption on n, we provide our own different analysis. This is omitted in this version.

Case 2 ($cd/\varepsilon \le m \le cd/\varepsilon^2$): We show we can choose x_1 of unit norm so that $\|\tilde{U}^T x_1\|^2 \ge (1 - C_8\varepsilon)m/n + C_9\sqrt{md}/n$ for some constants C_8, C_9. The details are omitted in this version. Now note that for $m < cd/\varepsilon^2$, the right hand side is at least $(1 + 10(C_2 + 1)\varepsilon)^2 m/n$ and thus $\|\tilde{U}^T x_1\| \ge (1 + 10(C_2 + 1)\varepsilon)\sqrt{m/n}$. ∎

Remark 1. There is an alternate way to obtain the above result in a similar fashion. Consider the distribution of $n \times d$ matrices U with i.i.d. $N(0, 1/n)$ entries. Let $V\Sigma W^T = U$ be the SVD of U. Let $\mathcal{U}_{hard}$ be the distribution of the aforementioned matrix V. By Bai-Yin theorem, with high probability, the singular values of U are bounded by $1 \pm O(\varepsilon)$. Therefore, if $\kappa(\Pi V) < 1 + \varepsilon$ then $\kappa(\Pi U) < 1 + O(\varepsilon)$ with high probability. Thus, for contradiction, assume that $\mathbb{P}_{V \sim \mathcal{U}_{hard}}(\kappa(\Pi V) > 1 + \varepsilon) < \delta$. Then $\mathbb{P}(\kappa(\Pi U) > 1 + O(\varepsilon)) < 2\delta$. Proceed through the above proof verbatim with the new matrix U to the step where we need to show the existence of z_1, z_2 such that $\|A^T U z_1\|/\|A^T U z_2\| > 1 + \Omega(\varepsilon)$ with probability at least 2/3. Note that $A^T U$ is an $m \times d$ matrix with i.i.d. Gaussian entries. Thus, existence of z_1, z_2 follows from showing that if $m = O(d/\varepsilon^2)$, then the condition number of an $m \times d$ random Gaussian matrix is at least $1 + \Omega(\varepsilon)$ with probability at least 2/3. This step in principle can be established via the Marchenko-Pastur law. We choose to include the above elementary self-contained proof instead of this similar approach.

3 Sparsity Lower Bound

In this section, we consider the trade-off between m, the number of columns of the embedding matrix Π, and s, the number of non-zeroes per column of Π. In this section, we only consider the case $n \ge 100d^2$. By Yao's minimax principle, we only need to argue about the performance of a fixed matrix Π over a distribution over U. Let the distribution of the columns of U be d i.i.d. random standard basis vectors in $\mathbb{R}^n$. With probability at least 99/100, the columns of U are distinct and form a valid orthonormal basis for a d dimensional subspace of $\mathbb{R}^n$. If Π succeeds on this distribution of U conditioned on the fact that the columns of U are orthonormal with probability at least 99/100, then it succeeds in the original distribution with probability at least 98/100. In section 3.1, we show a lower bound on s in terms of ε, whenever the number of columns m is much smaller than $\varepsilon^2 d^2$. In section 3.2, we show a lower bound on s in terms of m, for a fixed $\varepsilon = 1/2$. Finally, in section 3.3, we show a lower bound on s in terms of both ε and m, when they are both sufficiently small.

3.1 Lower Bound in Terms of ε

Theorem 2. *If $n \geq 100d^2$ and $m \leq \varepsilon^2 d(d-1)/32$, then $s = \Omega(1/\varepsilon)$.*

First we need a few simple lemmas.

Lemma 3. *Let $\mathcal{P}$ be a distribution over finite dimensional vectors of norm at most 1 and u and v be independent samples from $\mathcal{P}$. Then $\mathbb{E}\langle u, v\rangle \geq 0$.*

Proof. Let $\{u_i\}_i$ and $\{v_i\}_i$ be the coordinates of u and v. We have $\mathbb{E}\langle u, v\rangle = \sum_i \mathbb{E}\, u_i v_i = \sum_i (\mathbb{E}\, u_i)(\mathbb{E}\, v_i) = \sum_i (\mathbb{E}\, u_i)^2 \geq 0$. ■

Lemma 4. *Let X be a random variable bounded by 1 and $\mathbb{E}\, X \geq 0$. Then for any $0 < \delta < 1$, we have $\mathbb{P}(X \leq -\delta) \leq 1/(1+\delta)$.*

Proof. We prove the contrapositive. If $\mathbb{P}(X \leq -\delta) > 1/(1+\delta)$, then $\mathbb{E}\, X \leq -\delta\, \mathbb{P}(X \leq -\delta) + \mathbb{P}(X > -\delta) < -\delta/(1+\delta) + 1 - 1/(1+\delta) = 0$. ■

Proof (of Theorem 2). Let u_i be the i column of ΠU, r_i and z_i be the index and the value of the coordinate of the maximum absolute value of u_i, and v_i be u_i with the coordinate at position r_i removed. Let p_{2j-1}(respectively, p_{2j}) be the fractions columns of Π whose entry of maximum absolute value is on row j and is positive (respectively, negative). Let $C_{i,j}$ be the indicator variable indicating whether $r_i = r_j$ and z_i and z_j are of the same sign. Let $E = \mathbb{E}\, C_{1,2} = \sum_{i=1}^{2m} p_i^2$. Let $C = \sum_{i<j\leq d} C_{i,j}$. We have

$$\mathbb{E}\, C = \frac{d(d-1)}{2}\sum_{i=1}^{2m} p_i^2 \geq \frac{d(d-1)}{4m} \geq 8\varepsilon^{-2}$$

If i_1, i_2, i_3, i_4 are distinct then C_{i_1,i_2}, C_{i_3,i_4} are independent. If the pairs (i_1, i_2) and (i_3, i_4) share one index then $\mathbb{P}(C_{i_1,i_2} = 1 \wedge C_{i_3,i_4} = 1) = \sum_i p_i^3$ and $\mathbb{P}(C_{i_1,i_2} = 1 \wedge C_{i_3,i_4} = 0) = \sum_i p_i^2(1-p_i)$. Thus for this case,

$$\begin{aligned}\mathbb{E}(C_{i_1,i_2} - E])(C_{i_3,i_4} - E]) = (1 - 2\sum_i p_i^2 + \sum_i p_i^3)E^2 \\ - 2(1-E)E\sum_i p_i^2(1-p_i) + (1-E)^2\sum_i p_i^3\end{aligned}$$

$$\begin{aligned}&= E^2 - 2E^3 + E^2 \sum_i p_i^3 - (2E - 2E^2)(E - \sum_i p_i^3)\\ &\quad + (1 - 2E + E^2) \sum_i p_i^3\\ &= \sum_i p_i^3 - E^2 \le \left(\sum_i p_i^2\right)^{3/2}\end{aligned}$$

The last inequality uses that the ℓ_3 norm is at most the ℓ_2 norm. We have

$$\begin{aligned}\mathrm{Var}[C] &= \frac{d(d-1)}{2}\mathrm{Var}[C_{1,2}] + d(d-1)(d-2)\,\mathbb{E}(C_{i_1,i_2} - \mathbb{E}\,C_{i_1,i_2})(C_{i_1,i_3} - \mathbb{E}\,C_{i_1,i_3})\\ &\le \mathbb{E}\,C + (2\,\mathbb{E}\,C)^{3/2} \le 4(\mathbb{E}\,C)^{3/2}\end{aligned}$$

Above we used $\mathrm{Var}[C_{1,2}] \le \mathbb{E}\,C_{1,2}$ and $d(d-1)(d-2) \le (d(d-1))^{3/2}$. Therefore,

$$\mathbb{P}(C \le (\mathbb{E}\,C)/2) \le \frac{4\,\mathrm{Var}[C]}{(\mathbb{E}\,C)^2} \le O\left(\sqrt{\frac{m}{d(d-1)}}\right).$$

Thus, with probability at least $1 - O(\varepsilon)$, we have $C \ge 4\varepsilon^{-2}$. We now argue that there exist $1/\varepsilon$ pairwise-disjoint pairs (a_i, b_i) such that $r_{a_i} = r_{b_i}$ and z_{a_i} and z_{b_i} are of the same sign. Indeed, let d_{2j-1} (respectively, d_{2j}) be the number of u_i's with $r_i = j$ and z_i being positive (respectively, negative). Wlog, assume that $d_1, \ldots, d_t$ are all the d_i's that are at least 2. We can always get at least $\sum_{i=1}^t (d_i - 1)/2$ disjoint pairs. We have

$$\sum_{i=1}^t (d_i - 1)/2 \ge \frac{1}{2}\left(\sum_{i=1}^t d_i(d_i - 1)/2\right)^{1/2} = \frac{C^{1/2}}{2} \ge \varepsilon^{-1}$$

For each pair (a_i, b_i), by Lemmas 3 and 4, $\mathbb{P}[\langle v_{a_i}, v_{b_i}\rangle \le -\varepsilon] \le \frac{1}{1+\varepsilon}$ and these events for different i's are independent so with probability at least $1 - (1 + \varepsilon)^{-1/\varepsilon} \ge 1 - e^{\varepsilon/2-1}$, there exists some i such that $\langle v_{a_i}, v_{b_i}\rangle > -\varepsilon$. For Π to be a subspace embedding for the column span of U, it must be the case, for all i, that $\|u_i\| = \|\Pi U e_i\| \ge 1 - \varepsilon$. We have $|z_i| \ge s^{-1/2}\|u_i\| \ge s^{-1/2}(1 - \varepsilon)\ \forall i$. Therefore, $\langle u_{a_i}, u_{b_i}\rangle \ge s^{-1}(1 - \varepsilon)^2 - \varepsilon$. We have

$$\begin{aligned}\left\|\Pi U\left(\frac{1}{\sqrt{2}}(e_{a_i} + e_{b_i})\right)\right\|^2 &= \frac{1}{2}\|u_{a_i}\|^2 + \frac{1}{2}\|u_{b_i}\|^2 + \langle u_{a_i}, u_{b_i}\rangle\\ &\ge (1 - \varepsilon)^2(1 + s^{-1}) - \varepsilon\end{aligned}$$

However, $\|\Pi U\| \le 1 + \varepsilon$ so $s \ge (1 - \varepsilon)^2/(5\varepsilon)$. ■

3.2 Lower Bound in Terms of m

Theorem 3. *For $n \ge 100d^2$, $\frac{20\log\log d}{\log d} < \gamma < 1/12$ and $\varepsilon = 1/2$, if $m \le d^{1+\gamma}$, then $s = \Omega(1/\gamma)$.*

Proof. We first prove a standard bound for a certain balls and bins problem. The proof is included for completeness.

Lemma 5. *Let α be a constant in $(0,1)$. Consider the problem of throwing d balls independently and uniformly at random at $m \leq d^{1+\gamma}$ bins with $\frac{10 \log \log d}{\alpha \log d} < \gamma < 1/12$. With probability at least $99/100$, at least $d^{1-\alpha}/2$ bins have load at least $\alpha/(2\gamma)$.*

Proof. Let X_i be the indicator r.v. for bin i having $t = \alpha/(2\gamma)$ balls, and $X \overset{\text{def}}{=} \sum_i X_i$. Then

$$\mathbb{E}\, X_1 = \binom{d}{t} m^{-t}(1-1/m)^{d-t} \geq \left(\frac{d}{tm}\right)^t e^{-1} \geq d^{-\alpha}$$

Thus, $\mathbb{E}\, X \geq d^{1-\alpha}$. Because X_i's are negatively correlated,

$$\mathrm{Var}[X] \leq \sum_i \mathrm{Var}[X_i] = n(\mathbb{E}\, X_1 - (\mathbb{E}\, X_1)^2) \leq \mathbb{E}\, X.$$

By Chebyshev's inequality, $\mathbb{P}[X \leq d^{1-\alpha}/2] \leq \frac{4\,\mathrm{Var}[X]}{(\mathbb{E}\, X)^2} \leq 4d^{\alpha-1}$. Thus, w.p. $1-4d^{\alpha-1}$, there exist $d^{1-\alpha}/2$ bins with at least $\alpha/(2\gamma)$ balls. ■

Next we prove a slightly weaker bound for the non-uniform version.

Lemma 6. *Consider the problem of throwing d balls independently at $m \leq d^{1+\gamma}$ bins. In each throw, bin i receives the ball with probability p_i. With probability at least $99/100$, there exist $d^{1-\alpha}/2$ disjoint groups of balls of size $\alpha/(4\gamma)$ each such that all balls in the same group land in the same bin.*

Proof. The following procedure is inspired by the alias method, a constant time algorithm for sampling from a given discrete distribution (see e.g. [19]). We define a set of m virtual bins with equal probabilities of receiving a ball as follows. The following invariant is maintained: in the ith step, there are $m-i+1$ values $p_1, \ldots, p_{m-i+1}$ satisfying $\sum_j p_j = (m-i+1)/m$. In the ith step, we create the ith virtual bin as follows. Pick the smallest p_j and the largest p_k. Notice that $p_j \leq 1/m \leq p_k$. Form a new virtual bin from p_j and $1/m - p_j$ probability mass from p_k. Remove p_j from the collection and replace p_k with $p_k + p_j - 1/m$.

By Lemma 5, there exist $d^{1-\alpha}/2$ virtual bins receiving at least $\alpha/(2\gamma)$ balls. Since each virtual bin receives probability mass from at most 2 bins, there exist $d^{1-\alpha}/2$ groups of balls of size at least $\alpha/(4\gamma)$ such that all balls in the same group land in the same bin. ■

Finally we use the above bound for balls and bins to prove the lower bound. Let p_i be the fraction of columns of Π whose coordinate of largest absolute value is on row i. By Lemma 6, there exist a row i and $\alpha/(4\gamma)$ columns of ΠU such that the coordinates of maximum absolute value of those columns all lie on row i. Π is a subspace embedding for the column span of U only if $\|\Pi U e_j\| \in [1/2, 3/2]\ \forall j$. The columns of ΠU are s sparse so for any column of ΠU, the largest absolute value of its coordinates is at least $s^{-1/2}/2$. Therefore, $\|e_i^T \Pi U\|^2 \geq \alpha/(16\gamma s)$. Because $\|\Pi U\| \leq 3/2$, it must be the case that $s = \Omega(\alpha/\gamma)$. ■

3.3 Combining Both Types of Lower Bounds

Theorem 4. *For $n \geq 100d^2$, $m < d^{1+\gamma}$, $\alpha \in (0,1)$, $\frac{10 \log\log d}{\alpha \log d} < \gamma < \alpha/4$, $0 < \varepsilon < 1/2$, and $2/(\varepsilon\gamma) < d^{1-\alpha}$, we must have $s = \Omega(\alpha/(\varepsilon\gamma))$.*

Proof. Let u_i be the i column of ΠU, r_i and z_i be the index and the value of the coordinate of the maximum absolute value of u_i, and v_i be u_i with the coordinate at position r_i removed. Fix $t = \alpha/(4\gamma)$. Let p_{2i-1} (respectively, p_{2i}) be the fractions of columns of Π whose largest entry is on row i and positive (respectively, negative). By Lemma 6, there exist $d^{1-\alpha}/2$ disjoint groups of t columns of ΠU such that the columns in the same group have the entries with maximum absolute values on the same row. Consider one such group $G = \{u_{i_1}, \dots, u_{i_t}\}$. By Lemma 3 and linearity of expectation, $\mathbb{E} \sum_{u_i,u_j \in G, i\neq j} \langle v_i, v_j\rangle \geq 0$. Furthermore, $\sum_{u_i,u_j \in G, i\neq j} \langle v_i, v_j\rangle \leq t(t-1)$. Thus, by Lemma 4, $\mathbb{P}(\sum_{u_i,u_j \in G, i\neq j} \langle v_i, v_j\rangle \leq -t(t-1)(\varepsilon\gamma)) \leq \frac{1}{1+\varepsilon\gamma}$. This event happens independently for different groups, so with probability at least $1-(1+\varepsilon\gamma)^{-1/(\varepsilon\gamma)} \geq 1-e^{\varepsilon\gamma/2-1}$, there exists a group G such that

$$\sum_{u_i,u_j \in G, i\neq j} \langle v_i, v_j\rangle > -t(t-1)(\varepsilon\gamma)$$

The matrix Π is a subspace embedding for the column span of U only if for all i, we have $\|u_i\| = |\Pi U e_i\| \geq (1-\varepsilon)$. We have $|z_i| \geq s^{-1/2}\|u_i\| \geq s^{-1/2}(1-\varepsilon)$. Thus, $\sum_{u_i,u_j \in G, i\neq j} \langle u_i, u_j\rangle \geq t(t-1)((1-\varepsilon)^2 s^{-1} - \varepsilon\gamma)$. We have

$$\begin{aligned}\left\|\Pi U\left(\frac{1}{\sqrt{t}}\left(\sum_{i:u_i\in G} e_i\right)\right)\right\|^2 &\geq (1-\varepsilon)^2 + \frac{2}{t}\binom{t}{2}((1-\varepsilon)^2 s^{-1} - \varepsilon\gamma)\\ &\geq (1-\varepsilon)^2(1+(t-1)s^{-1}) - \alpha\varepsilon/4\end{aligned}$$

Because $\|\Pi U\| \leq 1+\varepsilon$, we must have $s \geq \frac{(\alpha/\gamma-4)(1-\varepsilon)^2}{(16+\alpha)\varepsilon}$. ■

Acknowledgments. J.N. did this work while a member at the Institute for Advanced Study, supported by NSF grants CCF-0832797 and DMS-1128155. HLN is supported by an IBM Ph.D. Fellowship and a Siebel Scholarship. We thank anonymous reviewers for many helpful comments.

References

1. Ailon, N., Chazelle, B.: The fast Johnson–Lindenstrauss transform and approximate nearest neighbors. SIAM J. Comput. 39(1), 302–322 (2009)
2. Andoni, A., Nguyễn, H. L.: Eigenvalues of a matrix in the streaming model. In: Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1729–1737 (2013)
3. Boutsidis, C., Zouzias, A., Mahoney, M.W., Drineas, P.: Stochastic dimensionality reduction for k-means clustering. CoRR, abs/1110.2897 (2011)

4. Clarkson, K.L., Woodruff, D.P.: Numerical linear algebra in the streaming model. In: Proceedings of the 41st ACM Symposium on Theory of Computing (STOC), pp. 205–214 (2009)
5. Clarkson, K.L., Woodruff, D.P.: Low rank approximation and regression in input sparsity time. In: Proceedings of the 45th ACM Symposium on Theory of Computing (STOC) (2013)
6. Dasgupta, S., Gupta, A.: An elementary proof of a theorem of Johnson and Lindenstrauss. Random Struct. Algorithms 22(1), 60–65 (2003)
7. Demmel, J., Dumitriu, I., Holtz, O.: Fast linear algebra is stable. Numer. Math. 108(1), 59–91 (2007)
8. Hanson, D.L., Wright, F.T.: A bound on tail probabilities for quadratic forms in independent random variables. Ann. Math. Statist. 42(3), 1079–1083 (1971)
9. Johnson, W.B., Lindenstrauss, J.: Extensions of Lipschitz mappings into a Hilbert space. Contemporary Mathematics 26, 189–206 (1984)
10. Kane, D.M., Meka, R., Nelson, J.: Almost optimal explicit Johnson-Lindenstrauss transformations. In: Proceedings of the 15th International Workshop on Randomization and Computation (RANDOM), pp. 628–639 (2011)
11. Mahoney, M.W., Drineas, P., Magdon-Ismail, M., Woodruff, D.P.: Fast approximation of matrix coherence and statistical leverage. In: Proceedings of the 29th International Conference on Machine Learning, ICML (2012)
12. Meng, X., Mahoney, M.W.: Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In: Proceedings of the 45th ACM Symposium on Theory of Computing (STOC) (2013)
13. Molinaro, M., Woodruff, D.P., Yaroslavtsev, G.: Beating the direct sum theorem in communication complexity with implications for sketching. In: Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1738–1756 (2013)
14. Nelson, J., Nguyễn, H.L.: OSNAP: Faster numerical linear algebra algorithms via sparser subspace embeddings. In: Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS) (2013)
15. Nelson, J., Nguyễn, H.L.: Sparsity lower bounds for dimensionality-reducing maps. In: Proceedings of the 45th ACM Symposium on Theory of Computing (STOC) (2013)
16. Paul, S., Boutsidis, C., Magdon-Ismail, M., Drineas, P.: Random projections for support vector machines. In: AISTATS (2013)
17. Sarlós, T.: Improved approximation algorithms for large matrices via random projections. In: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 143–152 (2006)
18. Tao, T.: Topics in random matrix theory. Graduate Studies in Mathematics, vol. 132. American Mathematical Society (2012)
19. Vose, M.D.: A linear algorithm for generating random numbers with a given distribution. IEEE Transactions on Software Engineering 17(9), 972–975 (1991)
20. Williams, V.V.: Multiplying matrices faster than Coppersmith-Winograd. In: Proceedings of the 44th ACM Symposium on Theory of Computing (STOC), pp. 887–898 (2012)

On Input Indistinguishable Proof Systems

Rafail Ostrovsky[1], Giuseppe Persiano[2,⋆], and Ivan Visconti[2,⋆]

[1] University of California at Los Angeles
rafail@cs.ucla.edu
[2] Università di Salerno
giuper@gmail.com, visconti@unisa.it

Abstract. We study *Input Indistinguishable Computation* (IIC), a security notion proposed by Micali, Pass, and Rosen in [14] and recently considered also by Garg, Goyal, Jain and Sahai in [9]. IIC aims at generalizing the notion of a *Witness Indistinguishable (WI)* proof system to general two-party functionalities and in its concurrent version (cIIC) also considers security against man-in-the-middle (MiM) attacks.

In this paper, we focus on the proof system functionality and compare IIC with two other security notions for proof systems: WI and Non-Malleability (NM). We address the following two questions.

1. Since IIC is a generalization of WI from proof systems to general 2PC, are all WI proofs also IIC secure?
2. Are cIIC proofs also NM?

We show, somewhat surprisingly, that *both* answers to the above questions are negative. Indeed, we show that there exists a WI proof system that is not IIC secure. We then show that a large class of WI proof systems, including the classical Blum's proof system for NP, are concurrently secure in the IIC sense. This answers the second question in the negative, since Blum's proofs are known to be malleable.

The consequence of our results is three-fold. 1) IIC is a too stringent notion and this leaves the possibility of security notions weaker than IIC with a satisfying level of security. 2) For important functionalities, such as the proof system functionality, classical constructions like Blum's protocol are cIIC secure. 3) cIIC security should be carefully evaluated when used as a security guarantee to model real-world concurrent attacks to protocols, as our results show that cIIC security does not guarantee non-malleability of proof systems. In contrast, standard simulation-based security [5,2] and concurrent non-malleable WI (a game-based security notion introduced by [15,16]) are secure against MiM attacks (the latter even in constant rounds).

1 Introduction

Proof systems were introduced in [12] and their security was defined using the *simulation paradigm* through the notion of *Zero Knowledge* (ZK). *Witness Indistinguishability* (WI[1]) introduced by Feige and Shamir [8] is instead a game-based

[⋆] Work partially done while visiting University of California at Los Angeles.

[1] We will use WI to mean both witness indistinguishability and indistinguishable.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 895–906, 2014.

security notion for proof systems requiring that the adversarial verifier be not able to distinguish which of two given witnesses has been used by the prover. WI is easily seen to be implied by ZK and, under plausible complexity assumptions, there exist WI proof systems that are not ZK [6].

It was later observed that ZK is not preserved if more sophisticated attacks are considered. Dwork et al. [7] initiated the study of security under *concurrent* composition. That is, the adversarial verifier can play multiple concurrent sessions keeping control over the scheduling of the messages. It is easy to see that ZK is not closed under concurrent composition whereas WI is.

In a man-in-the-middle (MiM) attack an adversary $\mathcal{A}$ acts as a verifier interacting with a honest prover and, at the same time, as a prover interacting with a honest verifier. Security against MiM attacks is called *non-malleability* [5]. Concurrency and MiM can be combined by considering an adversary playing as prover and verifier in multiple sessions. Formalizing security under such attacks in the simulation paradigm gives the notion of *concurrent non-malleable ZK* [2] (cNMZK) and guarantees non-transferability of proofs. The existence of a cNMZK protocol in the plain model with sub-logarithmic round complexity is a major open problem. Ostrovsky et al. [15,16] proposed a game-based security notion for proof systems, *concurrent non-malleable WI* (cNMWI), that implies security against concurrent MiM attacks. cNMWI guarantees non-transferability of proofs and is achievable in constant rounds, avoiding the complications of the simulation paradigm. Proofs (rather than arguments) have been achieved in [4].

Beyond Proof Systems: Concurrently-Secure 2PC. Security of two-part computation (2PC) has been traditionally formalized within the simulation paradigm. When concurrent attacks are considered though a series of impossibility results hinted at the fact that this notion might be too stringent. Indeed, Lindell [13] proved that concurrently secure 2PC can not be achieved for several interesting functionalities. This first impossibility result relied on the use of adaptive inputs through concurrent executions of protocols for some specific functionalities. The result has been then strengthened to the static input case by [2], and later broader impossibility results have been proved in [1,11].

Input-Indistinguishable Computation. Given the above limitations of simulation-based notions capturing security against concurrent MiM attacks, Micali et al. [14] proposed a game-based notion. Informally, the notion of *Input Indistinguishability Computation* (*IIC*, in short) tries to formalize the following security goal for 2PC: suppose there is more than one input for player P_1 that is consistent with the output obtained by player P_2; then, even a malicious P_2 cannot distinguish which of the possible consistent inputs has been actually used by P_1 in an execution. This is very similar in flavor to what WI requires from a proof system. The notion of IIC can be extended by considering the concurrent setting, where several sessions can be concurrently played and the adversary is allowed to play different roles in different sessions. The goal of cIIC is to guarantee that the output of the honest players in some sessions is not affected by the inputs used by honest players in other sessions, and this must hold for all inputs of the honest players that would produce the same outputs in those other sessions. The goal of the adversary is to play concurrently in different sessions to create correlations among their inputs/outputs.

Interestingly, in [14] it is first shown that IIC is not closed under concurrent composition. In particular, this is proved by showing a successful concurrent MiM attack on a protocol implementing the coin-flipping functionality. In the same paper [14], by building on top of some powerful non-malleable subprotocols, the authors showed a constant-round cIIC protocol for any two-party functionality. This is a major result since for the first time a meaningful security notion is shown to be feasible in the concurrent setting without relying on set-up assumptions.

The Recent Work of Garg et al. [9]. Very recently, Garg, Goyal, Jain and Sahai [9][2] gave a different notion of IIC with a simulation-based flavor, that we refer to as sIIC. Their notion also applies to randomized functionalities and is therefore more general. The proposed definition however is unsatisfactory if there exists a "splitting input" as discussed in [14]. Then [9] proposed another formulation referred to as exdIIC, that implies both IIC and sIIC. Of course the concepts of sIIC and exdIIC are naturally extended to the concurrent setting.

IIC: The Impact on Proof Systems. IIC is of major importance for the following two reasons: a) there are several popular computations as the Millionaire Problem, where the goal is simply to keep the input hidden among all other possible inputs that produce the same output, and IIC seems to be sufficient to achieve such a type of security; b) constant-round cIIC for any 2-party functionality has been achieved, while the same result under the standard simulation-based notion of secure computation is impossible to achieve, regardless of the round complexity. cIIC is therefore an appealing security notion to model security under concurrent composition in the plain model.

In this work we will focus on relations among IIC and two well-studied security notions for proof systems: WI and NM. The reason is two-fold. First, IIC has been proposed as a generalization of WI. Second, a MiM attacks to a protocol implementing the coin-flipping functionality proved that IIC is not closed under concurrent MiM attacks. Ignoring details of definitions one would expect at least one of the following two implications be true.

1. Since IIC is a generalization of WI to general two-party functionalities when the proof system functionality is taken into account then IIC and WI should coincide. In other words, every WI proof system should be IIC secure and any IIC-secure proof system should be WI.
2. Since a cIIC-secure protocol must defeat some forms of MiM attacks, then any cIIC-secure proof system should be non-malleable too.

1.1 Our Results

In this paper we analyze IIC-security in proof systems. Indeed IIC has been defined with the goal of lifting up the notion of WI from the proof system

[2] When referring to [9], we will actually refer to the full version available at [10].

functionality to general 2-party functionalities. Therefore any subtlety in IIC for the proof system functionality is potentially reflected on many other 2-party functionality (in particular to the functionalities that are similar to the proof system functionality). We embark on the task of finding answers to the above two questions that try to relate IIC to WI and non-malleable of proof systems.

We show that there exists a non-conversation-based[3] WI system that does not enjoy IIC. This is indeed surprising since the notion of IIC has been introduced to generalize the notion of WI to any other 2-party functionality, and thus one should expect that WI proofs of knowledge be IIC secure too. We then show that most of WI proofs of knowledge found in the literature are also secure in the IIC sense for the proof-system functionality even under concurrent composition. Specifically, this holds for *conversation-based* proofs, therefore contradicting the second claim. Indeed this class of WI proofs of knowledge contains several malleable protocols such as Blum's protocol.

We consider the notions of sIIC and exdIIC proposed in [9] and show that in contrast to IIC, every WI PoK is also sIIC. Of course since exdIIC implies IIC, there exists a (non-conversation-based) WI PoK that is not exdIIC secure. We will also prove that any conversation-based WI PoK is also exdIIC secure.

Consequences. In addition to the conceptual relevance of showing somewhat unexpected relations among security notions, our results have the following three consequences in applications of IIC.

First, IIC does not generalize WI but only a stronger form of it. The impact of this is that it is still possible to give a weaker definition of IIC that still captures the desired flavor, but that is easier to achieve. sIIC goes in this direction.

Second, there are important functionalities (e.g., the proof system functionality) such that classic constructions (e.g., Blum's protocol) are already cIIC secure. Therefore depending on the functionality in question, cIIC-security could come for free, without resorting to the general and inefficient (based on the use of expensive non-malleable subprotocols) constructions shown in previous work.

Third, when relying on cIIC for some 2-party functionalities, the actual meaning of cIIC for the given functionality should be carefully evaluated. Indeed while simulation-based secure 2PC provides strong enough guarantees, the security of cIIC can be unsatisfying. Such a decreased security *does not* depend on the fact that non-transferability of proofs requires simulation. Indeed, for the case of proof systems, it is still possible to obtain non-malleability (i.e., non-transferability of proofs) under an indistinguishability notion (e.g., NMWI [15]). We show that the formulation of cIIC does not give such a guarantee.

2 Definitions

A polynomial-time relation R is a relation for which it is possible to verify in time polynomial in $|x|$ whether $R(x, w) = 1$. We consider $\mathbb{NP}$-languages L and

[3] In a conversation-based proof the transcript identifies the common instance x and with overwhelming probability whether the verifier accepted or rejected.

denote by R_L the corresponding polynomial-time relation such that $x \in L$ if and only if there exists w such that $R_L(x, w) = 1$. We call such a w a *valid witness for* $x \in L$ and denote by $W_L(x)$ the set of valid witnesses for $x \in L$. We slightly abuse notation and, whenever L is clear from the context, we simply write $W(x)$ instead of $W_L(x)$. For sequences $X = (x_1, \cdots, x_m)$ and $W = (w_1, \cdots, w_m)$, by the writing "$W \in W(X)$" we mean that $w_i \in W(x_i)$ for $i = 1, \cdots, m$.

For a language L we will denote by L_n^m the set of sequences of m elements of L each of length at most n. A *negligible* function $\nu(k)$ is a function such that for any constant $c < 0$ and for all sufficiently large k, $\nu(k) < k^c$.

We stress that we will always refer to polynomial-time adversaries, therefore when we say proof systems or PoK, we actually refer to arguments.

We will use the standard definitions of proof system, WI and and the definition of IIC given in [14].

3 Input Indistinguishability vs WI

In this section we consider the notion of IIC [14] for the proof system functionality and compare it with the notion of a WI proof system [8]. While it is trivial to see from the definitions that any IIC proof is also WI, we show that the opposite implication does not hold.

We first show that a large class of WI proof systems (that includes all WI proof systems in the literature) also enjoys cIIC. However we will also show that this does not hold for all WI proof systems. The above large class consists of all WI proofs of knowledge that are *conversation-based*; that is, one can guess the output of the verifier (that is, whether the verifier accepts) by looking at the transcript of the protocol and, possibly, running in super-polynomial time. It is easy to see that all WI proof systems in the literature enjoy this property even in a very stringent sense, since the sole transcript is usually sufficient to efficiently guess whether the verifier accepts.

In this section, we denote the prover P by P_1 and the verifier V by P_2 in order to keep notation consistent with [14].

The Proof System Functionality $\mathcal{F}_{\mathcal{PK}}^L$. The input of (the prover) P_1 for functionality $\mathcal{F}_{\mathcal{PK}}^L$ for $\mathbb{NP}$ language L consists of a pair (x, y) whereas (the verifier) P_2 has in input only x. The output f_1 of P_1 in $\mathcal{F}_{\mathcal{PK}}^L((x, y), x)$ is defined as $f_1((x, y), x) = \perp$ (i.e., P_1 does not get any output); output f_2 of P_2 instead is defined as $f_2((x, y), x) = 1$ if y is a valid $\mathbb{NP}$ witness for $x \in L$; and $f_2((x, y), x) = 0$, otherwise.

Notice that $\mathcal{F}_{\mathcal{PK}}^L$ is defined with the two players having common input x. Whenever L is clear from the context, we will simply write $\mathcal{F}_{\mathcal{PK}}$.

Remark 1. One could think of using a different definition for the ideal functionality of a proof system where prover and verifier can have different statements as input. With such a different definition, then it happens that even the standard zero-knowledge PoK of Blum (e.g., sequential repetition of the classical 3-round protocol with a 1-bit challenge) would not be a secure instantiation (in the classical 2PC sense) of such a proof system functionality. The reason is that with such a functionality, the input statement x of the prover should remain private when playing with a V^* that runs on input a statement x' different from x. Instead in Blum's protocol the statement proven by the prover is not private at all. In general implementing such a functionality could require techniques/assumptions taken from general 2-party computation. Therefore we find such a definition of an ideal functionality less intuitive than the one that we use in the paper and that follows in spirit the formulation of [12].

Conversation-Based WI Proofs. We say that a WI proof is *conversation-based* if, given a transcript of the protocol it is possible to identify the common instance x and to compute with overwhelming probability the output of the honest verifier. We stress that no time bound is imposed on the decision procedure. All standard WI proofs (including Blum's protocol [3]) are in this category.

3.1 Conversation-Based WI ⇒ IIC

In this section we prove that all conversation-based WI proof systems with perfect completeness are also cIIC for the proof system functionality $\mathcal{F}_{\mathcal{PK}}$. Thus, following Definition 1 and 2 of [14], we exhibit, for any conversation-based WI proof, a first-party and second-party implicit input functions $\mathsf{IN}_1, \mathsf{IN}_2$ that fulfill all requirements of IIC.

Defining the Implicit-Input Functions for $\mathcal{F}_{\mathcal{PK}}$. We remind the reader that, according to the definition of IIC, implicit-input functions are not necessarily efficiently computable.

IN_1: Let $\mathsf{View}_1^*(\mathbf{e})$ be the full view of P_1^* of an execution $\mathbf{e}$ of (P_1^*, P_2) (this includes the private coins and the input of P_1^*). For each session i of $\mathbf{e}$, the output of IN_1 on input $\mathsf{View}_1^*(\mathbf{e})$ is defined as follows.

If $\mathtt{OUTPUT}_1^i(\mathbf{e}) = 1$ and the verifier P_2 accepted the proof (this can be decided because of the conversation-based property), then IN_1 outputs a pair consisting of the instance x that is obtained from $\mathsf{View}_1^*(\mathbf{e})$ (as it is the i-th common input), and the lexicographically first witness y for $x \in L$. Instead, IN_1 outputs $\perp$ for all sessions i in which $\mathtt{OUTPUT}_1^i(\mathbf{e}) = 0$ or the verifier P_2 did not accept the proof (again, this is can be decided using the conversation-based property).

IN_2: Let $\mathsf{View}_2^*(\mathbf{e})$ be the full view of P_2^* of an execution $\mathbf{e}$ of (P_1, P_2^*). For each session i of $\mathbf{e}$, if $\mathtt{OUTPUT}_2^i(\mathbf{e}) = 1$, IN_2 on input $\mathsf{View}_2^*(\mathbf{e})$ outputs the statement x that is obtained from $\mathsf{View}_2^*(\mathbf{e})$ since it is the i-th common input, and outputs $\perp$ otherwise.

First of all, notice that IN_1 and IN_2 are implicit functions (i.e., they both output $\perp$ in case of aborts).

Completeness. For any session i, Prob $\left[\ P_1(\mathsf{View}_1^i(\mathbf{e})) = f_1((x_i, y_i), x_i)\ \right] = 1$ and Prob $\left[\ P_2(\mathsf{View}_2^i(\mathbf{e})) = f_2((x_i, y_i), x_i)\ \right] = 1$. Indeed, for the former, notice that f_1 always outputs $\perp$, and honest prover P_1 never outputs a value different than $\perp$; for the latter, notice that P_2 outputs precisely a bit denoting accept or reject and this coincides with the output of f_2. The perfect completeness property of the WI proof is required to prover the IIC completeness.

Implicit Computation. Let P_2^* be the adversary. W Prob $\left[\ P_1(\mathsf{View}_1^i(\mathbf{e})) = \perp\ \right] = 1$ in session i, and also $f_1((x_i, y_i), x_i^*) = \perp$ where $x_i^* =$ is the i-th component of the output of $\mathsf{IN}_2(\mathsf{View}_2^*(\mathbf{e}))$. Here notice that regardless of the value of $\mathtt{OUTPUT}_1^i(\mathbf{e})$, both $P_1(\mathsf{View}_1^i(\mathbf{e})) = \perp$ and $f_1((x_i, y_i), x_i^*)$ are always equal to $\perp$.

Let P_1^* be the adversary. If $\mathtt{OUTPUT}_2^i$ is false then Prob $\left[\ P_2(\mathsf{View}_2^i(\mathbf{e})) = \perp\ \right] = 1$ since the output delivery message of the i session is not in the view of $\mathbf{e}$. In case $\mathtt{OUTPUT}_2^i$ is true, we have that the i-th component (x_i, y_i^*) of the output of $\mathsf{IN}_1(\mathsf{View}_1^*(\mathbf{e}))$, is a valid theorem-witness pair for the i-th component x_i of the input of P_2 in the i-th session, only if P_2 gives in output 1. Therefore the implicit function IN_1 always outputs a value that makes the evaluation of f_2 consistent with the output of P_2[4].

Input Indistinguishability and Independence. We next show that for any adversary P_2^* it holds that $\{\mathsf{EXPT}^{P_1,P_2^*}((\mathbf{x}, \mathbf{y}^1), (\mathbf{x}, \mathbf{y}^2), \mathbf{x}; 1^n)\}$ is indistinguishable from $\{\mathsf{EXPT}^{P_1,P_2^*}((\mathbf{x}, \mathbf{y}^2), (\mathbf{x}, \mathbf{y}^1), \mathbf{x}; 1^n)\}$.

Indeed, the input function IN_2 selects x_i from $\mathsf{View}_2^*(\mathbf{e})$ independently of the other inputs. Since those other inputs are the only differences between the two experiments we have that by the WI of the views, the outputs $(\mathbf{x}^*, \mathsf{View}_2^*(\mathbf{e}))$ of both experiments are computationally indistinguishable.

Let us now consider adversary P_1^*. In this case we have that, since the verifier has as input only $\mathbf{x}$, both experiments correspond to $\{\mathsf{EXPT}^{P_1^*,P_2}((\mathbf{x}, \mathbf{y}), \mathbf{x}, \mathbf{x}; 1^n)\}$. The input function IN_1 defined above selects the instance-witness pair (x_i, y_i) for the i-session from the view of the i-th execution independently of the witness that has been actually used (as long as the transcript is accepting), since IN_1 considers the first witness in lexicographic order. Therefore both experiments produce the same output $(\mathbf{y}^*, \mathsf{View}_1^*(\mathbf{e}))$.

From Fixed Roles to a General Adversarial Behavior. Notice that in the discussion above, we have considered a fixed-role adversay only; that is, an adversary that either plays the role of the prover in all sessions or it plays the role of verifier. Intuitively, we used the following two facts: 1) when the adversary is a verifier, it has as input only x and the output by EXPT is a tuple of pairs (y, View), one for each session, where y is actually independent of the witness used by the prover, and View is a witness indistinguishable transcript; 2) when the adversary is a prover, the honest verifier has no private input and thus the two experiments EXPT of the definition collapse in one experiment only, so that indistinguishability of the output is trivial.

[4] The fact that IN_1 uses the conversation-based property is critical. Indeed we will exploit this to show that there exists a WI proof system that does not enjoy IIC.

In general, the adversary could play a man-in-the-middle attack; that is, it could play the role of the prover in some sessions and the role of the verifier in other sessions. We next argue that the above analysis still works. Indeed, based on the two possible sequences of inputs of the honest provers, we have that:

1) in the sessions where the adversary played as verifier, the output of EXPT contains witnesses unrelated to the ones used by honest provers and views that do not allow one to distinguish which witness has been used; 2) in the sessions where the adversary played as prover, we will have statements and views that again can only vary according to the sequence of witnesses used by honest provers in other sessions, which in turn means that, by the witness indistinguishability of those proofs, these views are indistinguishable as well.

Thus we proved the following theorem.

Theorem 1. *Any conversation-based WI proof system for an* $\mathbb{NP}$ *language* L *is IIC (even under concurrent composition) for* $\mathcal{F}^L_{\mathcal{PK}}$.

3.2 WI $\not\Rightarrow$ IIC for $\mathcal{F}_{\mathcal{PK}}$

Here, we show that there exist WI proof systems that do not enjoy IIC.

Theorem 2. *There exists a WI proof system* Π *for* $\mathbb{NP}$ *language* L *that is not IIC for functionality* $\mathcal{F}^L_{\mathcal{PK}}$.

Proof. Consider the classical proof system pBLUM that consists of parallel executions of Blum's protocol for the $\mathbb{NP}$-complete language of the Hamiltonian graphs [3]. More precisely, in the first round of pBLUM with security parameter k and input graph G, the prover selects k random permutations $\pi_1, \ldots, \pi_k$, computes graphs $G_1, \ldots, G_k$ where $G_i = \pi_i(G)$ and sends the commitments of the adjacency matrices of the k graphs. The verifier picks random bits $b_1, \ldots, b_k$ and sends them to the verifier. Finally, for each i for which $b_i = 0$, the prover opens all the commitments of the adjacency matrix of G_i and sends π_i; instead, for each i for which $b_i = 1$, the prover opens the commitments of the adjacency matrix of G_i that correspond to edges in a Hamiltonian cycle. The verifier accepts if and only if all the k answers obtained are correct.

It is easy to see that pBLUM is a WI proof system with perfect completeness and negligible (in k) soundness error for the language of the Hamiltonian graphs. Also, pBLUM is conversation-based since the final decision of the verifier is solely based on the transcript.

To prove Theorem 2, we artificially modify pBLUM by requiring the honest verifier V to randomly select $j \in \{1, \ldots, k\}$ and to neglect the answer of the prover in the j-th parallel execution in deciding whether to accept or not. The resulting protocol, mBLUM, enjoys perfect completeness and negligible soundness error and is still WI. However, mBLUM is not conversation-based. Indeed, a malicious prover P^* could use a string s hardwired in its code to decide to play wrongly in exactly one of the k parallel executions while playing honestly in the remaining ones. Notice that the honest verifier for mBLUM will accept the proof of P^* with non-negligible probability $1/k$. Indeed, it will accept exactly when the randomly selected parallel execution of the protocol corresponds to the one specified by s. Therefore, by looking at the transcript one can not guess and be correct with overwhelming probability if the honest verifier accepted or not. Notice that this holds unconditionally, even when the private input and coins of the prover are known.

Formally, assume that the instance is x and the witness is y. We have that the Implicit Computation property does now hold when P_1^* is the above adversarial prover and P_2 is the above honest verifier. Indeed in the above execution it happens that $P_2(\mathsf{View}_2^1) = 1$ with probability $1/k$ and $P_2(\mathsf{View}_2^1) = 0$ with probability $1 - 1/k$. The probability is over (a subset of) the private coins of P_2 that are independent from the transcript. Therefore to satisfy the Implicit Computation property one should have an implicit function IN_1 that on input View_1^* guesses with overwhelming probability the output of P_2. However, View_1^* does not contain the private coins used by P_2 to discard one of the parallel executions of Blum's protocol, therefore any implicit function IN_1 fails with non-negligible probability[5].

The existence of a WI proof system that is not IIC proves that IIC is a generalization to general functionalities of some special forms of WI only.

4 Simulation-Based IIC: sIIC and exdIIC

Here we study some relations among WI, IIC and sIIC and exdIIC [10]. We stress that even though we focus on the proof-system functionality, it is expected that our results extend to several other functionalities.

We next briefly review the notions of sIIC and exdIIC and refer the reader to Definition 5 and Definition 7 of [10] for formal definitions.

Classical (simulation-based) concurrently-secure 2PC requires the existence of a simulator S for every real-world adversary A so that the views of the adversary in the real world and in the ideal world are indistinguishable. Roughly speaking, sIIC (called IIC in [10]) relaxes this requirement by allowing the simulator S to depend also on the pair of input vectors of the honest party and by only requiring that the distributions of the outputs of the two party to be indistinguishable. The definition of *extended Input Indistinguishable Computation* (exdIIC, for short) strengthens the notion of sIIC by requiring indistinguishability between the ideal and real world of the pair consisting of the output of the parties (so far, it is similar to sIIC) and the input of the adversary which for the real world is defined by means of an implicit function IN that extracts the input from the view. For further details on sIIC and exdIIC, see Definition 5 and Definition 7 of [10].

[5] Indeed even in case one can have a randomized implicit function IN_1 that with probability $1 - 1/k$ outputs $\perp$, it would work against the above P_1^*, but it would fail against another adversary P_1^{**} that just plays as honest prover and always convinces the verifier.

4.1 WI and IIC vs sIIC

Theorem 3. *Any WI PoK for an $\mathbb{NP}$ language L is sIIC for $\mathcal{F}^{L}_{\mathcal{PK}}$.*

Proof. First of all remember that in the definition of [9], the simulator can be different for different pairs of inputs obtained by the honest player.

When the real-world prover is honest, the simulator plays as verifier in the ideal world and simulates a prover against the malicious verifier. The simulator internally has two witnesses (sIIC allows a different simulator for each pair of inputs for the honest player) for the theorem corresponding to the input statement and picks one of them to be used with the malicious verifier, playing then the protocol of the honest prover.

When the real-world verifier is honest, the simulator plays as prover in the ideal world and simulates a verifier against the malicious prover. Since the honest player (i.e., the verifier) has no witness, the simulator will have no witness as well. However the PoK property guarantees that the simulator can extract a witness from the malicious prover and can then play it in the ideal world.

It is easy to see that if the output of the ideal-world experiment differs form the one of the real world, one can easily break the WI of the proof system. Indeed notice that for the sessions where the simulator is an ideal-world verifier, the only deviation with respect to the real world consists in the fact that the simulator might use a different witness. For the sessions where the simulator is a real-world verifier, the only deviation with respect to the real world consists in the fact that the simulator has to extract a witness from P^* in order to play it in the ideal world. The PoK property guarantees that this can be done.

Notice that rewinding the adversary in the concurrent setting is often dangerous and can blow up the running time of the simulator. Nevertheless, since here the simulator rewinds only the malicious prover, there is no issue with its running time. The reason is that rewinds are related to extractions and can therefore be done sequentially, applying the extractor to the final transcript. During each extraction, there are no rewinds related to other sessions.

4.2 WI and IIC vs exdIIC

With the purpose of having a definition that also captures the security goals of IIC, Garg et al. in [9] defined exdIIC and proved that it implies both IIC of [14] and sIIC. One might think that exdIIC is a strengthening of IIC that requires stronger security properties (indeed it is a simulation-based notion) and it could be possible that several protocols that are IIC are not exdIIC. We show that for the proof system functionality this is not the case as we prove that any conversation-based WI PoK is also secure in the exdIIC sense.

Theorem 4. *Any conversation-based WI PoK for $\mathbb{NP}$ language L is exdIIC for functionality $\mathcal{F}^{L}_{\mathcal{PK}}$.*

Proof. The definition of exdIIC considers the indistinguishability of ideal and real world experiments also including 1) in the distribution of the ideal-world experiment, the inputs sent by the adversary to the trusted party and 2) in the distribution of the real-world experiment, the outputs of implicit functions $\mathsf{IN}_1, \mathsf{IN}_2$ on inputs the views of the adversaries.

The proof given for the case of sIIC is not sufficient here because we need to define implicit functions first, and then we must make sure that the inputs sent by the simulator to the ideal functionality are consistent with the outputs of the implicit functions.

Since the honest verifier of a PoK runs on input just the statement of the theorems, the implicit function IN_2 can just output the list of theorems (accepting or not) belonging to the view received in input. Therefore the only problem is to define the implicit function IN_1 that receives as input the view of the adversarial prover. Our choice is to have IN_1 to output, for each theorem in the view received in input, the first valid witness in lexicographic order provided that the transcript of the session is accepting[6], otherwise it will output $\perp$.

In order to have that the inputs sent to the functionality by the ideal-world adversary be consistent with the output of IN_1 we consider a simulator that first runs internally as verifier against the real-world adversary and checks if it gets a convincing proof. If this is the case, the simulator send to the functionality the witness that it has hardwired in its code. Notice that since in the definition of [9] there is a simulator for any pair of inputs, we have that there always exists a simulator that contains hardwired in its code the first witness in lexicographic order corresponding to the theorem specified in the input of the prover.

The case of an ideal-world adversarial verifier is simpler. As soon as a proof starts the simulator sends the theorem to the ideal functionality, and if it gets as output 1, it runs internally the honest prover procedure using the witness that it has hardwired in its code. If instead it receives 0, it just sends and abort message to the real-world adversarial verifier (the same things is of course done by a prover when the theorem to be proved is different from the one expected by the verifier). As for the case of sIIC, this proposed simulation is indistinguishable by the WI of the underlying proof system. Therefore the theorem holds.

It is worthy to notice the point in which the above theorem would fail in case of non-conversation based WI proof systems. Indeed we have that IN_1 could output a witness even when the verifier, because of his private coins, does not accept that transcript. Therefore in the real-world experiment, the verifier would output 0 while the output of IN_1 would be a witness. Then in the ideal-world experiment the simulator would be required to send the same witness, which of course allows the honest verifier of the ideal world to obtain 1 as output. This would clearly make ideal and real worlds distinguishable.

Acknowledgments. Work supported in part by NSF grants 09165174, 1065276, 1118126 and 1136174, US-Israel BSF grant 2008411, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S.

[6] This restricts the statement of our theorem to conversation-based proofs only.

Office of Naval Research under Contract N00014 -11 -1-0392. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

1. Agrawal, S., Goyal, V., Jain, A., Prabhakaran, M., Sahai, A.: New impossibility results for concurrent composition and a non-interactive completeness theorem for secure computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 443–460. Springer, Heidelberg (2012)
2. Barak, B., Prabhakaran, M., Sahai, A.: Concurrent non-malleable zero knowledge. In: 47th FOCS. IEEE Computer Society Press (2006)
3. Blum, M.: How to Prove a Theorem So No One Else Can Claim It. In: Proceedings of the International Congress of Mathematicians, pp. 1444–1451 (1986)
4. Cao, Z., Visconti, I., Zhang, Z.: On constant-round concurrent non-malleable proof systems. Inf. Process. Lett. 111(18), 883–890 (2011)
5. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography. In: 23rd ACM STOC, pp. 542–552. ACM Press (1991)
6. Dwork, C., Naor, M.: ZAPs and their applications. In: 41st FOCS, pp. 283–293. IEEE Computer Society Press (2000)
7. Dwork, C., Naor, M., Sahai, A.: Concurrent zero-knowledge. In: 30th ACM STOC, pp. 409–418. ACM Press (1998)
8. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: 22nd ACM STOC, pp. 416–426. ACM Press (1990)
9. Garg, S., Goyal, V., Jain, A., Sahai, A.: Concurrently secure computation in constant rounds. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 99–116. Springer, Heidelberg (2012)
10. Garg, S., Goyal, V., Jain, A., Sahai, A.: Concurrently secure computation in constant rounds (full version) (2012), http://goo.gl/iPXSbe
11. Garg, S., Kumarasubramanian, A., Ostrovsky, R., Visconti, I.: Impossibility results for static input secure computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 424–442. Springer, Heidelberg (2012)
12. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal on Computing 18(1), 186–208 (1989)
13. Lindell, Y.: Lower Bounds for Concurrent Self Composition. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 203–222. Springer, Heidelberg (2004)
14. Micali, S., Pass, R., Rosen, A.: Input-indistinguishable computation. In: 47th FOCS, pp. 136–145. IEEE Computer Society Press (2006)
15. Ostrovsky, R., Persiano, G., Visconti, I.: Constant-round concurrent non-malleable zero knowledge in the bare public-key model. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 548–559. Springer, Heidelberg (2008)
16. Ostrovsky, R., Persiano, G., Visconti, I.: Concurrent non-malleable witness indistinguishability and its applications. Electronic Colloquium on Computational Complexity (ECCC) 13(95) (2006)

Secure Computation Using Leaky Tokens

Manoj Prabhakaran[1], Amit Sahai[2], and Akshay Wadia[2]

[1] UIUC
mmp@uiuc.edu
[2] UCLA
{sahai,awadia}@cs.ucla.edu

Abstract. Leakage-proof hardware tokens have been used to achieve a large number of cryptographic tasks recently. But in real life, due to various physical attacks, it is extremely difficult to construct hardware devices that are guaranteed to be leakage-proof. In this paper, we study the feasibility of general two-party computation using leaky hardware tokens.

Our main result is a completeness theorem that shows that *every* non-trivial leaky two-party functionality can be used for general secure computation. In fact, the protocol we construct is *non-interactive* and *unconditionally secure*. There are no restrictions on the leakage functions associated with the token, except that it does not render the tokens trivial, by revealing its entire secrets to the adversary.

1 Introduction

Hardware tokens have received considerable attention in recent years ([1,3,4,6, 7,9–11,14,17,18,20,22,24,29,33,34], to name a few). In earlier works, specific hardware devices were used to achieve specific cryptographic tasks. Later on, as the potential of hardware devices was better understood, general hardware tokens were used to achieve general tasks. For example, in [20], among other things, the authors show feasibility of an unconditional, non-interactive UC protocol for general functionalities, using stateful hardware tokens.

In all these works, the hardware-tokens used are considered to be *leakage-proof*: there is a function f associated with the token (programmed with an input from the party creating the token) such that a party who receives it can only access the input/output behaviour of $f(\cdot)$, and cannot learn anything else about the input already programmed into it.

Although theoretically clean, in practice it is difficult to construct hardware devices that are truly leakage-proof. In particular, once the hardware device is in the adversary's possession, it can be subjected to *physical* attacks, like measuring power consumption, that can reveal secret information. (See, for example, [5,28, 35] and references therein, for various physical attacks.) Given that hardware devices cannot be guaranteed to be leakage-proof in real life, the natural question that arises is *whether we can use such leaky tokens to achieve cryptographic tasks.*

In this paper, we study the feasibility of using such leaky tokens for constructing general secure two-party computation protocols. Our focus is on theoretical

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 907–918, 2014.

feasibility and not practical efficiency, and we consider information-theoretic security. We think of a leaky token as a hardware one-time implementation of a two-party functionality $f(\cdot, \cdot)$. The sender of the token chooses an input x and creates such a token. The receiver can query the token with input y, at which point the token outputs $f(x, y)$. To model leakage, the adversary is allowed to ask a (possibly randomized) "leakage query" $L \in \mathcal{L}$, where $\mathcal{L}$ is a class of allowable leakage functions. To this, the token responds with the leakage $L(x)$. Once the token is accessed (using a legitimate input or a leakage function), it loses all information about x.

We point out a few aspects of our leakage model. On the positive side, the leakage functions are unrestricted (computationally unbounded, randomized, communicating arbitrary amount of information), and we prove the *best possible result* in that, unless a leakage function that completely reveals the functionality of the token is available to the adversary (in which case the hardware could as well be replaced with plain messages), the tokens can be used for secure evaluation of any function. On the other hand, we model the leakage function as acting on a single token at a time. But we do allow the adversary to adaptively choose the leakage function for each token, based on the information gathered from previously accessed tokens. Also, our leakage is one-time. We leave it for future work to consider models of *multi-round leakage* and *joint leakage* which would allow the adversary to, among other things, leak correlated information from multiple tokens.

Our Results. We construct a *non-interactive, unconditionally secure* general two-party secure computation protocol in the leaky token hybrid model. Instead of using leaky tokens which implement specific functionalities, we prove the following *general completeness theorem:* every *non-trivial* leaky two-party functionality is complete for unconditionally secure two-party computation. A two-party functionality is called "trivial" if there is a leakage function in the allowable leakage class that exhausts all the entropy from a token that was initialized with a uniformly chosen sender's input. Otherwise, the functionality is called non-trivial.

Related Work. There has been a long line of work on leakage resilient primitives, for example [12, 23, 32] and several more recent works. We mention just a few of them which are more relevant to our setting. Hardware leakage has been theoretically modeled and studied in several works including [15, 21, 23]; these works consider a model with no protected parts (or tokens), but with significantly restricted leakage functions. The study of leakage resilience for interactive protocols was initiated in [13], which constructed secure multi-party computation protocols using leaky tokens, but relying on computational assumptions. The problem of tamperable tokens was raised in [8], who showed that certain token functions are naturally resistant to certain restricted classes of tampering, or can be encoded to become so; this is applicable only when the tampering or leakage function applied to the token is from a well-behaved class. Hardware tokens have also been used to construct One Time Programs (OTP). An OTP for a function f allows a party to evaluate f on a single input x chosen by the party dynamically. OTPs are typically implemented as a package containing some software along with some hardware tokens. OTPs were introduced in [18]. In [20], OTPs were used to construct an unconditionally secure non-interactive protocol for two-party computation.

On the question of completeness of functionalities, building on his earlier results [25, 26], in 2000, Kilian [27] presented an elegant protocol relying on Nash equilibrium, to show that any non-trivial asymmetric functionality is complete for security against malicious adversaries; later, [30,31] provide further completeness results. The protocols presented in these works are all interactive. A *non-interactive* completeness theorem for non-trivial functionalities was first shown in [2]. However, all off these results are in the non-leaky model. Our theorem can be seen as a generalization of the completeness of non-trivial asymmetric functionalities [2, 27] to the leaky setting.

1.1 Technical Overview

Our goal is to use leaky tokens to construct a non-interactive, unconditionally secure two-party protocol for general functionalities. In [20], the authors construct such a protocol in the One Time Memory (OTM) hybrid model, which was introduced in [18]. The OTM functionality is a two-party functionality which, on input a pair of bits (b_0, b_1) from the sender and a bit c from the receiver, returns b_c to the receiver. That is, an OTM behaves like an implementation of Oblivious Transfer (OT), but with the following crucial difference: in OT, after the receiver specifies its input bit and receives its output b_c, the sender gets an acknowledgment that the receiver has received its output. However, in OTM, no such acknowledgment is sent to the sender (for more on the consequences of this crucial difference, see [18] and Section 4.2 in [20]). In light of the construction given in [20], to achieve our goal, it is sufficient to realize the OTM functionality using leaky tokens.

We consider a token to be a one-time evaluable implementation of a deterministic, constant-sized function f of two variables, into which one variable has been programmed. A user can evaluate the token only once, on any input of its choice. But, we allow an adversarial user to adaptively specify leakage functions for each token (which can be randomized and computationally unbounded), in lieu of feeding it a valid input. We shall show that *any such token is complete for token-based non-interactive secure computation*, as long as none of the allowed leakage functions fully reveals the information programmed into the token. Here, completeness means that *any function* that takes inputs from two parties and provides output to only one of them (the receiver) can be securely implemented using a token-based non-interactive protocol, even if the receiver is malicious and can leak from the tokens, adaptively. (The sender could be malicious as well.)

Technically, the work most relevant to ours is a recent result of [2]. There, a similar result is shown when there is no leakage from the tokens; also, a *deterministic extraction* procedure given there turns out to be useful for us. But handling arbitrary leakage presents several new technical challenges.

As a simple example, consider the following kind of token. It implements 1-out-of-m string OTM: i.e., the receiver can obtain one of m strings that are programmed into the token. It also permits a simple leakage function, which allows a corrupt receiver to learn $m - 1$ out of the m strings. If $m = 2$ this function is the same as 1-out-of-2 OTM. But, for $m \geq 3$, can such a token be used to implement (non-leaky) 1-out-of-2 bit OTM?

It can in fact be argued that such a functionality, when implemented by a trusted third party, *cannot* be used to obtain OTM, even using interactive protocols! Suppose there is such an OTM protocol. Since the sender should not learn the receiver's input, the receiver will be able to run two simultaneous executions of this protocol, with two different inputs, while producing the same view for the sender; if the two executions have two different inputs to a 1-out-of-m OTM session, the receiver will use the leakage facility to learn both the inputs (since it is allowed to learn $m - 1 \geq 2$ inputs). This would enable the receiver to complete both executions and learn both inputs of the sender, contradicting the security of the protocol. *However*, the token model differs from the standard hybrid model in a crucial way. In the standard hybrid model, both the parties learn about when each session is initiated as well as when the outputs are delivered in each session. In contrast, in the token model, the sender does not learn at what point — and in particular, in what order — the various tokens are accessed by the receiver.

This leads us to the following simple protocol for OTM based on this token. The sender sends two tokens for 1-out-of-m string OTM. Each of the m strings in the first token, contains a random bit; in addition, the first of these m strings contains a pointer to (i.e., the index of) one of the m strings in the next token. Similarly, the second token contains m random bits, and in addition the first bit is bundled with a pointer to one of the m strings. The random bit in the first token that is pointed to by the second token is used to mask the sender's first input; similarly, the second input is masked by the random bit in the second token that is pointed to by the first token. The masked values are sent along with the tokens. An honest receiver who wants to access the first input would open the pointer stored in the second token first, and find out the position in the first token where the mask bit is. Note that the order in which the receiver opens the two tokens reveals its input, but this remains hidden from the sender.

Now, in this protocol, a malicious receiver needs to access (using a leakage function) one of the two tokens at first; in doing so, it can learn up to $m - 1$ strings in that token. But with probability $1/m$, the one string that it leaves out is the one that is pointed to by the other token. Thus, with probability at least $1/m$, the receiver learns only one of the two inputs of the sender. This uncertainty can be amplified by using the XOR of many such bits as the mask, to obtain a statistically secure bit OTM protocol.

The above class of functions and leakage are quite well-behaved. But in general, the leakage can be randomized, and comes from an infinite class of possible functions. Our main contribution is to show that, in spite of this, any token with non-trivializing leakage can be used to implement the above well-behaved class of functions and leakage. Our protocol is oblivious to the class of leakage functions (but does depend on the function f).[1]

The basic idea behind our protocol for implementing (leaky) 1-out-of-m OTM is similar to (but more sophisticated than) that of the 1-out-of-2 OT protocol in [2]. As in the OTM protocol there, roughly, our goal is to force the adversary to suffer a small amount of uncertainty about one of the inputs of the sender, and then amplify this uncertainty using many executions combined using a *deterministic extraction* strategy, which takes a sum of (small degree) products. (Standard randomized extraction is not applicable in this setting since the adversary can see the seed before it accesses the tokens, and hence the seed is not independent of the entropy source.) *But an additional challenge in our setting is that all of the leakage functions available to the adversary are not available to a honest receiver.* We need to translate this arbitrary gap between the power of the adversary and the honest receiver to that between the two in the leaky 1-out-of-m OTM functionality.

Another technical difference between the setting of [2] and ours is the following: when there is no leakage present (as in [2]), the non-triviality requirement on f implies that there are two specific inputs for the receiver which are "undominated" such that a secure protocol can be designed by ignoring the other inputs for f. That is, the honest receivers can be required to feed only one of these two inputs to f. In contrast, when leakage functions (even deterministic ones) are present, this is no more the case. This is exemplified by the leaky 1-out-of-m OTM itself: if the honest receiver is restricted to using only $m-1$ of the possible m inputs, then a leakage function would let a malicious receiver learn all the relevant secrets from the tokens, and then run the protocol with different inputs. Thus, in our case it is important that the honest receiver uses its entire domain of inputs.

Our solution involves a collection of m maps from the range of the function f to $\mathbb{Z}_p$ (for an appropriately chosen p), where m is the number of inputs to f that an honest receiver has. The tokens from the sender are grouped into "bundles" of $2m$ tokens each, two for each map. Within a bundle, the output from a token is meant to be mapped to $\mathbb{Z}_p$ using the map associated with that token, and then multiplied together. (The deterministic extraction in our case involves adding together the products from each bundle, modulo p.)

The non-triviality of the function given the leakage function family guarantees that for every admissible leakage function L, there is some y such that $H(f(X,y)|L(X))$ is lowerbounded by a constant. Suppose $\widehat{y}$ be the y in a bundle for which this uncertainty is accumulated the most. The maps above are chosen in such a way that a fraction of this uncertainty will be preserved in the product corresponding to $\widehat{y}$, while for every other y, at least with a

[1] For setting *concrete* parameters, one would need a concrete bound on the value of the entropy parameter measuring the non-triviality of the function in the presence of leakage.

constant probability, its corresponding product will become fixed to 0. Thus, even given the product corresponding to all other y's (which is fixed to 0 with some probability), there is some uncertainty in the product corresponding to $\widehat{y}$. Finally, for each y, the products from all the bundles are summed together to extract a mask corresponding to y. The above property of the maps ensures that, for some $\widehat{y}$, the extracted value is close to uniformly random, even conditioned on the extracted values corresponding to the other $m-1$ y's.

2 Preliminaries

Protocols and Security. We follow standard definitions of protocol execution and security as specified in [19] and [16]. We will use $\mathcal{F}_{\mathsf{OTM}}$ to denote the OTM functionality of [18, 20]. We will also use a generalization of OTM to the case of 1-out-of-m OTM, where the sender holds m inputs (instead of 2), out of which the receiver selects one. The reason we focus on the One Time Memory functionality is because of the following theorem:

Theorem 1 ([20],Theorem 13). *Let $f(x, y)$ be a non-reactive, sender-oblivious, polynomial-time computable two-party functionality. Then there exists an efficient, statistically UC-secure non-interactive protocol which realizes f in the $\mathcal{F}_{\mathsf{OTM}}$-hybrid model.*

Modeling Leakage. Let $f : \mathsf{X} \times \mathsf{Y} \to \mathsf{Z}$ be a constant-size deterministic function. Let $\mathcal{L}$ be a set of possibly randomized *leakage functions*. To model leakage, we define the *token functionality* $\mathcal{F}^{(f,\mathcal{L})}$. The functionality receives $x \in \mathsf{X}$ from the sender. An honest receiver sends $y \in \mathsf{Y}$ to $\mathcal{F}^{(f,\mathcal{L})}$ and obtains $f(x, y)$. A malicious receiver, on the other hand, may query $\mathcal{F}^{(f,\mathcal{L})}$ with a leakage function $L \in \mathcal{L}$ and obtain $L(x)$. If L is a randomized leakage function, $L(x)$ specifies a distribution, and the ideal functionality samples $w \leftarrow L(x)$ and returns it to the receiver. For conciseness of notation, for randomized leakage functions L, we will use $L(x)$ to also mean a sample from that distribution.

Functionality $\mathcal{F}^{(f,\mathcal{L})}$:

- On receiving the message $(\mathsf{input}, \mathsf{sid}, P_j, x)$ from party P_i, store the tuple $(P_i, P_j, \mathsf{sid}, x)$ and send $(\mathsf{received}, P_i, \mathsf{sid})$ to party P_j. Ignore all input messages from P_i with session id sid.
- **Honest Query.** On receiving $(\mathsf{output}, \mathsf{sid}, P_i, y)$ from party P_j, if no tuple of the form $(P_i, P_j, \mathsf{sid}, x)$ exists, do nothing. Else, send $(\mathsf{output}, \mathsf{sid}, P_i, f(x, y))$ to party P_j, and delete the tuple $(P_i, P_j, \mathsf{sid}, x)$.
- **Leaky Query.** On receiving $(\mathsf{leak}, \mathsf{sid}, P_i, L)$ from party P_j, if $L \notin \mathcal{L}$ or no tuple of the form $(P_i, P_j, \mathsf{sid}, x)$ exists, do nothing. Else, send $(\mathsf{leak}, \mathsf{sid}, P_i, L(x))$ to party P_j, and delete the tuple $(P_i, P_j, \mathsf{sid}, x)$.

It is clear that no security is possible if the adversary is allowed arbitrary leakage queries. For example, if the adversary is allowed to ask the identity map as its leakage query, then it learns x and renders the functionality 'trivial'. To avoid this, we restrict the adversary to use only those leakage queries that leave some uncertainty *in the output* corresponding to some Bob's input $y \in \mathsf{Y}$. This is formalized below.

Definition 1. *Let $f : \mathsf{X} \times \mathsf{Y} \to \mathsf{Z}$ be a constant-size deterministic function, and let $\mathcal{L}$ be a set of functions with domain X. Let X be the uniform distribution on X. Then, the token functionality $\mathcal{F}^{(f,\mathcal{L})}$ is called* non-trivial *if there exists a constant $c > 0$ such that*

$$\min_{L \in \mathcal{L}} \max_{y \in \mathsf{Y}} H(f(X, y) \mid L(X)) \geq c.$$

Amplifying Uncertainty. In our proofs, we will consider random variables that have constant uncertainty conditioned on the adversary's view. To amplify this uncertainty, we will take the sum of an appropriate number of such random variables. To this end, the following generalization of a lemma from [2] will be useful.

Lemma 1. *Let p be a fixed prime number. For any positive integer N, let $X_1, \cdots, X_N$ be N independent random variables over the alphabet $\mathbb{Z}_p$, such that for some constant $\epsilon > 0$, for all $i \in [N]$, for all $z \in \mathbb{Z}_p^N$, $\Pr[X_i = z] < 1 - \epsilon$. Then the statistical distance between the distribution of $\sum_{i=1}^{N} X_i$ (summation in $\mathbb{Z}_p$) and the uniform distribution over $\mathbb{Z}_p$ is negligible in N.*

3 Leaky-OTM from Leaky Tokens

In this section we show how to use any non-trivial leaky token to implement a functionality $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$ described below. Here, the parameter m is equal to the number of inputs for (honest) Bob to the token. We leave out from the notation of $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$ a parameter d specifying the length of the "messages" that Alice gives as input to $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$. d can be set to any constant in our following construction, by choosing the parameter $p \geq 2^d$.

Functionality $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$:

- **When Bob is honest:** function as a 1-out-of-m (string) OTM. i.e., accept m strings, each d bits long, $(x_1, \cdots, x_m)$ from Alice, and an index $i \in [m]$ from Bob. Output x_i to Bob.
- **When Bob is corrupt:** function as $(m-1)$-out-of-m OTM. Here, Alice's input is the same as above; Bob can input any set $S \subsetneq [m]$ and receive $\{x_j | j \in S\}$ as output.

Let $\mathcal{F}$ be any non-trivial leaky token functionality, with m inputs for (honest) Bob. The idea behind our protocol for $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$ is a generalization of an earlier protocol for OTM from non-trivial (non-leaky) tokens [2]. The main complication in allowing leakage is that honest Bob does not have access to all the inputs an adversarial Bob can have. In particular, unlike in [2], it is not true that we can find two "undominated" inputs for Bob. Indeed, restricting to any strict subset of the m inputs can render the function trivial in the presence of the leakage functions. This complicates our construction when $m > 2$. (When $m = 2$, one could use the protocol in [2], but our protocol does not become identical to that in [2] if we set $m = 2$.)

While the protocol in [2] considered tokens in bundles of two, we shall use bundles of size $2m$. Further, unlike in that protocol where the same set of maps were used in all tokens to map Bob's (input, output) pairs to elements in a field, we use m different sets of maps for the $2m$ tokens in the bundle, to map these pairs to $\mathbb{Z}_p$ (for a large enough prime constant p). We give the formal description of the protocol below, and defer the proof of security to the full version.

Set up: Let $\mathcal{F} = (f, \mathcal{L})$ be a non-trivial leaky functionality evaluating a function $f : \mathsf{X} \times \mathsf{Y} \to \mathbb{Z}_p$ for some prime p[2] and allowing a set of leakage functions $\mathcal{L}$ such that for all $L \in \mathcal{L}$, $\min_{y \in \mathsf{Y}} H(f(X, y)|L(X)) > c$ for some constant c, where X is uniformly distributed over X.

Alice can send tokens for $\mathcal{F}$ with an input $x \in \mathsf{X}$ of her choice. An honest Bob can evaluate the token once with $y \in \mathsf{Y}$ of his choice to obtain $f(x, y)$. Let $m = |Y|$ be the number of inputs for (honest) Bob. An adversarial Bob can evaluate $L(x)$ for a leakage function $L \in \mathcal{L}$ of his choice.

Output Maps: Define m maps $M_1, \cdots, M_m$, of the form $M_i : \mathsf{Y} \times \mathbb{Z}_p \to \mathbb{Z}_p$ as follows. Fix an arbitrary input $x^* \in \mathsf{X}$, and let $z_j^* = f(x^*, y_j)$. For $i = 1$ to m and $j = 1$ to m, define

$$M_i(y_j, z) = \begin{cases} z - z_j^* + 1 & \text{if } j = i, \\ z - z_j^* & \text{if } j \neq i \end{cases}$$

This ensures that for each i, j, $M_i(y_j, \cdot)$ is a permutation over $\mathbb{Z}_p$, and for each y_i, there is a map (namely M_i) such that $M_i(y_i, f(x^*, y_i)) = 1$ but $M_i(y_j, f(x^*, y_j)) = 0$ for all $j \neq i$.

Alice's program: Alice's input is m elements in $\mathbb{Z}_p$, $(s_1, \cdots, s_m)$.

```
1. Alice carries out the following computations:
   • For $\ell = 1$ to $\kappa$, for $i = 1$ to $m$ and $t \in \{1, 2\}$:
       • Pick $x_{\ell,i,t} \leftarrow \mathsf{X}$.
       • For $j = 1$ to $m$, let $R^j_{\ell,i,t} = M_i(y_j, f(x_{\ell,i,t}, y_j))$.
   • For $j = 1$ to $m$:
       • Let $\overline{R}^j_\ell = \Pi_{i=1}^m \Pi_{t=1}^2 R^j_{\ell,i,t}$.
       • let $r_j = \sum_{\ell=1}^{\kappa} \overline{R}^j_\ell$.
       • For $j = 1$ to $m$, let $z_j = s_j + r_j$.
2. Alice creates several $\mathcal{F}$-tokens with the following inputs:
   • Tokens labeled $(\ell, i, t)$ for $\ell \in [\kappa]$, $i \in [m]$, $t \in \{1, 2\}$, with inputs $x_{\ell,i,t}$.
```

[2] W.l.o.g, the output alphabet can be considered $\mathbb{Z}_p$ by choosing a prime p such that for each $y \in \mathsf{Y}$, $|\{f(x, y)|x \in \mathsf{X}\}| \leq p$.

- $m\lceil \log p \rceil$ more tokens to "communicate" the bits of $(z_1, \cdots, z_m)$: in a token to send a bit 0, use input $\hat{x}^0$, and in a token to send a bit 1, use input $\hat{x}^1$, where $\hat{x}^0$ and $\hat{x}^1$ are such that $f(\hat{x}^0, y^*) \neq f(\hat{x}^1, y^*)$ for some y^*.

Bob's program: Bob's input is an index $c \in [m]$.

1. Bob accesses the $\mathcal{F}$-tokens sent by Alice with the following inputs:
   - For the tokens labeled (ℓ, i, t) for $\ell \in [\kappa]$, $i \in [m]$, $t \in \{1, 2\}$, use input y_c to obtain an output $z_{\ell,i,t} = f(x_{\ell,i,t}, y_c)$.
   - Use input y^* in the remaining tokens to learn $(z_1, \cdots, z_m)$ (or just z_c).
2. For all (ℓ, i, t) let $R^c_{\ell,i,t} = M_i(y_c, z_{\ell,i,t})$; let $\overline{R}^c_\ell = \Pi^m_{i=1} \Pi^2_{t=1} R^c_{\ell,i,t}$.
3. Let $r_c = \sum^\kappa_{\ell=1} \overline{R}^c_\ell$. Then output $s_c = z_c - r_c$.

---

4 OTM from Leaky-OTM

Suppose Alice and Bob parties have access to parallel copies of $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$, for some constant m. If $m = 2$, $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$ is the same as OTM. In this section we show how to implement OTM non-interactively, using parallel copies of $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$ and a single message from Alice to Bob, even when $m \geq 3$. The protocol uses the deterministic extraction strategy of [2]. The protocol crucially relies on the fact that Alice does not learn when Bob accesses various copies of $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$ (which are implemented using tokens). Correspondingly, Alice never learns when Bob accesses the OTM that is being implemented.

Overview. Firstly, note that implementing OTM using $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$ tokens (for $m \geq 3$) is in fact impossible using a non-interactive protocol in which the order in which Bob accesses the token is *not adaptive*: Bob could mentally run two executions with two different values for his choice bits. Each execution would require the honest Bob to access at most one out of m positions in each $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$ instance. But an adversarial Bob can access $m - 1 \geq 2$ positions in each instance. Thus he can complete both executions successfully, and learn both inputs of Alice. (A similar argument can be used to rule out implementing OTM in $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$-hybrid is used where Alice learns when Bob accesses an $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$ instance, even if interaction is allowed, as long as Bob is computationally unbounded.)

So necessarily we shall need to rely on Bob being able to access the tokens adaptively, and in different order. This leads us to the following basic idea underlying our construction. Alice's inputs are hidden in two tokens, at random positions. In addition, the first token (in a fixed position) contains a pointer to the position in the second token where an input is hidden; similarly, the second token (in a fixed position) has a pointer to the position in the first token that holds the other input. To access the first input, Bob first opens the second token at the fixed position, to recover a pointer to the first token, and then accesses the first token to recover the input stored in that position from the first token.

Similarly, if Bob wants to access the second input, he should first open the first token, recover a pointer that tells him which position in the second token he should access, and then open the second token at that position.

A malicious Bob must first open one of the two tokens. When he opens one token, and recovers $m-1$ positions, there is a $1/m$ probability that the one position that he did not access is the one containing Alice's input in that token. This gives us a weak form of OTM, in which, with some probability, Bob learns at most one secret.

To amplify this to a full-fledged OTM using a non-interactive protocol, we use an deterministic extraction technique devloped in [2] (see the protocol in Section 3.1). A degree 2 function is used to combine the several individual secrets to form a random mask that is then used to mask the actual input of Alice. We give the formal description of the protocol below, and defer the proof of security to the full version.

Alice's program: Alice's input is two bits s_0, s_1.

1. Alice carries out the following computations:
   - For $i = 1$ to κ,
     - pick $x^0_i, x^1_i \leftarrow [m]$ and for each $j = 1$ to m, pick $b^0_{i,j}, b^1_{i,j} \leftarrow \{0,1\}$.
     - let $a^0_{i,1} = (x^0_i, b^0_{i,1})$ and $a^1_{i,1} = (x^1_i, b^1_{i,1})$; for $j = 2$ to m, let $a^0_{i,j} = (0, b^0_{i,j})$ and $a^1_{i,1} = (0, b^1_{i,j})$; for $j = 2$ to m.
     - let $R^0_i = b^0_{i,x^1_i}$ and $R^1_i = b^1_{i,x^0_i}$.
   - Let $r_0 = \sum_{i=1}^{\kappa} R^0_i$ and $r_1 = \sum_{i=1}^{\kappa} R^1_i$ (summation in $\mathbb{Z}_2$).
   - Let $z_0 = s_0 + r_0$ and $z_1 = s_1 + r_1$.
2. Alice invokes, in parallel, several copies of $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$ with the following inputs:
   - Sessions labeled $(i, \beta) \in [\kappa] \times \{0,1\}$ with input $(a^{\beta}_{i,1}, \ldots, a^{\beta}_{i,m})$.
   - Another session of $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$ to "communicate" the bits (z_0, z_1): Alice can use $(z_0, z_1, 0, \cdots, 0)$ as her input in this session. (An adversary may receive both these bits, if $m \geq 3$.)

Bob's program: Bob's input is a choice bit b.

1. Bob invokes the same copies of $\mathcal{F}$ as Alice with the following inputs:
   - For $i = 1$ to κ,
     - If $b = 0$, first access the session numbered $(i, 1)$ with input 1, and obtain x^1_i, and then access the session numbered $(i, 0)$ with input x^1_i to obtain the bit R^0_i.
     - If $b = 1$, first access session $(i, 0)$ with input 1, recover x^0_i and then access the session $(i, 1)$ with input x^0_i to recover R^1_i.
   - Recover z_b from the last session of $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$, using b as input.
2. After all sessions of $\widetilde{\mathcal{F}}^{(m)}_{\mathsf{OTM}}$ are completed, compute $r_b = \sum_{i=1}^{\kappa} R^b_i$, and $s_b = z_b - r_b$ (all operations in $\mathbb{Z}_2$). Output s_b.

References

1. Agrawal, S., Ananth, P., Goyal, V., Prabhakaran, M., Rosen, A.: Lower bounds in the hardware token model. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 663–687. Springer, Heidelberg (2014)
2. Agrawal, S., Goyal, V., Jain, A., Prabhakaran, M., Sahai, A.: New impossibility results for concurrent composition and a non-interactive completeness theorem for secure computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 443–460. Springer, Heidelberg (2012)
3. Bitansky, N., Canetti, R., Goldwasser, S., Halevi, S., Kalai, Y.T., Rothblum, G.N.: Program obfuscation with leaky hardware. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 722–739. Springer, Heidelberg (2011)
4. Brands, S.: Untraceable off-line cash in wallets with observers (extended abstract). In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 302–318. Springer, Heidelberg (1994)
5. Brumley, D., Boneh, D.: Remote timing attacks are practical. Computer Networks 48(5), 701–716 (2005)
6. Chandran, N., Goyal, V., Sahai, A.: New constructions for uc secure computation using tamper-proof hardware. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 545–562. Springer, Heidelberg (2008)
7. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
8. Choi, S.G., Kiayias, A., Malkin, T.: Bitr: built-in tamper resilience. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 740–758. Springer, Heidelberg (2011)
9. Cramer, R., Pedersen, T.P.: Improved privacy in wallets with observers (extended abstract). In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 329–343. Springer, Heidelberg (1994)
10. Damgård, I.B., Nielsen, J.B., Wichs, D.: Isolated proofs of knowledge and isolated zero knowledge. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 509–526. Springer, Heidelberg (2008)
11. Döttling, N., Kraschewski, D., Müller-Quade, J.: Unconditional and composable security using a single stateful tamper-proof hardware token. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 164–181. Springer, Heidelberg (2011)
12. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: IEEE 49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, pp. 293–302. IEEE (2008)
13. Garg, S., Jain, A., Sahai, A.: Leakage-resilient zero knowledge. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 297–315. Springer, Heidelberg (2011)
14. Gennaro, R., Lysyanskaya, A., Malkin, T., Micali, S., Rabin, T.: Algorithmic tamper-proof (atp) security: Theoretical foundations for security against hardware tampering. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 258–277. Springer, Heidelberg (2004)
15. Gennaro, R., Lysyanskaya, A., Malkin, T., Micali, S., Rabin, T.: Algorithmic tamper-proof (atp) security: Theoretical foundations for security against hardware tampering. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 258–277. Springer, Heidelberg (2004)
16. Goldreich, O.: The Foundations of Cryptography. Basic Techniques, vol. 1. Cambridge University Press (2001)

17. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious rams. J. ACM 43(3), 431–473 (1996)
18. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: One-time programs. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 39–56. Springer, Heidelberg (2008)
19. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. 18(1), 186–208 (1989)
20. Goyal, V., Ishai, Y., Sahai, A., Venkatesan, R., Wadia, A.: Founding cryptography on tamper-proof hardware tokens. IACR Cryptology ePrint Archive, 2010:153 (2010)
21. Ishai, Y., Prabhakaran, M., Sahai, A., Wagner, D.: Private circuits ii: Keeping secrets in tamperable circuits. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 308–327. Springer, Heidelberg (2006)
22. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003)
23. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003)
24. Katz, J.: Universally composable multi-party computation using tamper-proof hardware. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 115–128. Springer, Heidelberg (2007)
25. Kilian, J.: Founding cryptography on oblivious transfer. In: STOC, pp. 20–31 (1988)
26. Kilian, J.: A general completeness theorem for two-party games. In: STOC, pp. 553–560 (1991)
27. Kilian, J.: More general completeness theorems for secure two-party computation. In: Proceedings of the thirty-Second Annual ACM Symposium on Theory of Computing, pp. 316–324. ACM (2000)
28. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
29. Kolesnikov, V.: Truly efficient string oblivious transfer using resettable tamper-proof tokens. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 327–342. Springer, Heidelberg (2010)
30. Kraschewski, D., Maji, H.K., Prabhakaran, M., Sahai, A.: A full characterization of completeness for two-party randomized function evaluation. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 659–676. Springer, Heidelberg (2014)
31. Kraschewski, D., Müller-Quade, J.: Completeness theorems with constructive proofs for finite deterministic 2-party functions. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 364–381. Springer, Heidelberg (2011)
32. Micali, S., Reyzin, L.: Physically observable cryptography (extended abstract). In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004)
33. Moran, T., Naor, M.: Basing cryptographic protocols on tamper-evident seals. Theor. Comput. Sci. 411(10) (2010)
34. Moran, T., Segev, G.: David and goliath commitments: Uc computation for asymmetric parties using tamper-proof hardware. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 527–544. Springer, Heidelberg (2008)
35. Quisquater, J.-J., Samyde, D.: Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In: Attali, S., Jensen, T. (eds.) E-smart 2001. LNCS, vol. 2140, pp. 200–210. Springer, Heidelberg (2001)

An Improved Interactive Streaming Algorithm for the Distinct Elements Problem

Hartmut Klauck[1,⋆] and Ved Prakash[2]

[1] Centre for Quantum Technologies and Nanyang Technological University
hklauck@gmail.com
[2] Centre for Quantum Technologies and National University of Singapore
prakash18ved@gmail.com

Abstract. The exact computation of the number of distinct elements (frequency moment F_0) is a fundamental problem in the study of data streaming algorithms. We denote the length of the stream by n where each symbol is drawn from a universe of size m. While it is well known that the moments F_0, F_1, F_2 can be approximated by efficient streaming algorithms [1], it is easy to see that exact computation of F_0, F_2 requires space $\Omega(m)$. In previous work, Cormode et al. [9] therefore considered a model where the data stream is also processed by a powerful helper, who provides an interactive proof of the result. They gave such protocols with a polylogarithmic number of rounds of communication between helper and verifier for all functions in $\mathcal{NC}$. This number of rounds ($O(\log^2 m)$ in the case of F_0) can quickly make such protocols impractical.

Cormode et al. also gave a protocol with $\log m+1$ rounds for the exact computation of F_0 where the space complexity is $O\left(\log m \log n + \log^2 m\right)$ but the total communication $O\left(\sqrt{n} \log m\left(\log n + \log m\right)\right)$. They managed to give $\log m$ round protocols with polylog(m, n) complexity for many other interesting problems including F_2, Inner product and Range-sum, but computing F_0 exactly with polylogarithmic space and communication and $O(\log m)$ rounds remained open.

In this work, we give a streaming interactive protocol with $\log m$ rounds for exact computation of F_0 using $O\left(\log m\left(\log n + \log m \log\log m\right)\right)$ bits of space and the communication is $O\left(\log m\left(\log n + \log^3 m (\log\log m)^2\right)\right)$. The update time of the verifier per symbol received is $O(\log^2 m)$.

1 Introduction

In a seminal work [1], Alon, Matias and Szegedy studied the space complexity of both approximating the frequency moments of a data stream and computing them exactly. Streaming algorithms are usually designed to handle large data sets, and the algorithm should be able to process each data element with small

[⋆] This work is funded by the Singapore Ministry of Education (partly through the Academic Research Fund Tier 3 MOE2012-T3-1-009) and by the Singapore National Research Foundation.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 919–930, 2014.

time overhead, and should have small working space as well. For instance one of the striking results of Alon et al. is that the second frequency moment F_2 can be approximated up to constant factors arbitrarily close to 1 using only $O(\log n + \log m)$ space by a randomized algorithm, where m is the size of the universe and n is the stream length. The interested reader is referred to the survey by Muthukrishnan [21].

It is known that the frequency moments F_j for integer $j > 2$ are hard to even approximate by any streaming algorithm, i.e., any streaming algorithm giving a good approximation must have large space. Motivated by this and the paradigm of cloud computing, one can study a new model where a helper/prover is introduced. The hope is that while some problems require a lot of space to solve by an unassisted streaming algorithm, a helper who is not space restricted (and sees the stream in the same way as the verifier) might not only be able to compute the result, but be able to convince the client/verifier of the correctness of that result by providing an interactive proof, that can be verified by the client using small space only. In the past few years, there have been numerous papers [5–9] considering this idea.

Thus we have the following scenario: both the prover and client observe the stream. The client, who is unable to store the data, computes some sketch of the data within his space restrictions. The prover, having no space restriction, can store the entire data, compute the answer, and send it to the client. But the prover may not be honest, e.g. the prover may have financial incentives to not provide the correct answer. The client uses the sketch of the data to reject wrong claims with high probability. The prover can be thought of as an internet company which offers cloud computing services and operates huge data warehouses. The only formal restriction on the prover is that he cannot predict the future parts of the stream. From now on, we refer to the client as the verifier.

Besides providing upper bounds one can also show lower bounds on the model of prover assisted data streaming algorithms. Data streaming protocols can be simulated by Arthur-Merlin communication protocols, where Merlin is the prover and the data stream input is split across some players, who together constitute the verifier Arthur. Arthur-Merlin communication complexity was first introduced by Babai, Frankl and Simon [3] and was studied in greater detail by Klauck [15, 16]. These lower bounds have been used by Chakrabarti et al. [5] to give non-trivial lower bounds on approximating and computing exactly the k-th frequency moments for large enough k, in the setting where the proof provided is noninteractive, i.e., the prover provides an "annotation" to the data stream that is then verified without further interaction. Unfortunately analyzing model of interactive proofs with many rounds between prover and verifier in communication complexity seems to be out of reach for current techniques in communication complexity.

One of the fundamental problems in data streaming is to compute the number of distinct elements in a data stream, which is the zeroth frequency moment and is denoted by F_0. This problem has many application in areas such as query optimization, IP routing, and data mining [14]. By a simple reduction from the disjointness function [23], it is easy to get a lower bound of $\Omega(m)$ (assuming $m = \theta(n)$) on the streaming complexity of computing F_0 exactly without a prover. If we require exact computation of F_0 and sublinear verifier space, we have to look at the prover-verifier model.

By appealing to Klauck's [15] result on the MA complexity of disjointness, there is a lower bound on $hv = \Omega(m)$ to compute F_0 exactly in the online MA model as defined in [5], where h is the help cost and v is the space used by the streaming algorithm. Cormode et al. [9] gave interactive streaming protocols with $\log m$ rounds for various interesting problems like frequency moments, the Index function and computing Inner Products. They also gave a general purpose protocol that computes every function in $\mathcal{NC}$ with polylogarithmic space, communication and rounds. For the case of exactly computing F_0, the general purpose protocol uses $O(\log^2 m)$ rounds, which can very quickly become impractical. Hence the authors also describe a protocol using only $\log m$ rounds, where the help cost (i.e., communication) is not polylogarithmic in m and n. We improve their protocol so that both the communication h and the space v are polylogarithmic in m and n, while using only $\log m$ rounds of interaction.

1.1 Previous Work

Let m be the universe size and n be the length of the stream. Although we later state our complexity results in terms of m and n, *in this subsection, for simplicity, we assume m and n are roughly of the same order of magnitude*, i.e. $m = poly(n)$ following the previous works in [8, 9]. We note that the authors of [8, 9] stated the complexity of their protocols in terms of machine words, but in this work, all complexities are stated in bits. It is known that approximating F_0 up to a $(1 \pm \epsilon)$ multiplicative factor can be done in $O(\epsilon^{-2} + \log m)$ space using randomization, which is optimal as well [14].

Goldwasser, Kalai and Rothblum [11] proposed a delegation general purpose interactive protocol for log-space uniform $\mathcal{NC}$ circuits. Their protocol was presented formally in the streaming setting by Cormode, Mitzenmacher, and Thaler [8]. We state their results below for easy reference.

Fact 1. *[Theorem 3.1 from [8]]*
Let f be a function over an arbitrary field $\mathbf{F}$ that can be computed by a family of $O(\log S(n))$-space uniform arithmetic circuits(over $\mathbf{F}$) of fan-in 2, size $S(n)$ and depth $d(n)$. Then in the streaming model with a prover, there is a protocol which requires $O(d(n)\log S(n))$ rounds such that the verifier needs $O(\log S(n) \log |\mathbf{F}|)$ bits of space and the total communication between the prover and the verifier is $O(d(n)\log S(n)\log |\mathbf{F}|)$.

As a result of Fact 1, if we use the general purpose interactive protocol of [11] to compute F_0 exactly, it will require $\Omega(\log^2 m)$ rounds of interaction between the prover and verifier. Cormode, Mitzenmacher, and Thaler [8] gave an alternative interactive protocol for F_0 based on linearization, whereby the prover is more efficient in terms of running time. Their protocol requires $\log^2 m$ rounds where the verifier's space is $O(\log^2 m)$ bits and the total communication is $O(\log^3 m)$.

As far as we know, the only interactive protocol which uses $\log m$ rounds to compute F_0 is given in [9]. We note that the results stated in [9] assumed that $m = n$. Restating the complexity of the F_0 protocol in [9] in terms of m and n, the space of the verifier is $O\left(\log m \log n + \log^2 m\right)$ and the total communication is $O\left(\sqrt{n} \log m \left(\log n + \log m\right)\right)$ [18]. Compared to the other protocols (e.g. F_2 and Index) given in [9], the total communication is not polylogarithmic in m and n. We briefly explain why the communication blows up to $\tilde{O}(\sqrt{n})$ in Section 1.2.

Chakrabarti et al. [5, 7] studied the situation in which the prover provides a (lengthy) annotation/proof to the verifier after the data stream has ended. The verifier processes the annotation in a streaming fashion. This corresponds to randomized checking of noninteractive proofs (in the theory of interactive proofs, such systems are called Merlin Arthur games). For the exact computation of F_0 in this model, the help cost, h and the verifier's space, v are both $O(m^{2/3} \log m)$.

In other related work, Gur and Raz [12] gave a Arthur-Merlin-Arthur(AMA) streaming protocol for computing F_0 exactly with both h and v being $\widetilde{O}(\sqrt{m})$ (where $\widetilde{O}$ hides a polylog(m, n) factor). Klauck and Prakash [17] studied a restricted interactive model where the communication between the prover and verifier has to end once the stream is already seen. Very recently, Chakrabarti et al. [6] presented constant-rounds streaming interactive protocols with logarithmic complexity for several query problems, including the well studied INDEX problem.

1.2 Previous Results and Our Techniques

First, we briefly describe why the protocol of [9] for computing F_0 fails to have total communication polylogarithmic in m and n. It is easy to see that $F_0 = \sum_{i=1}^{m} h\left(f_i\right)$ where $f_i := |\{j \mid a_j = i\}|$ and $h : \mathbf{N} \to \{0, 1\}$ is given by $h(0) = 0$ and $h(x) = 1$ for $1 \leq x \leq n$. Since h depends on $n + 1$ points, the degree the polynomial $\tilde{h}$, obtained via interpolation, is at most n, where $\tilde{h}$ agree with h on $\{0, 1, \cdots, n\}$. If one was to naively apply the famous sum-check protocol of Lund et al. [20], the degree of the polynomial communicated at each round would be $O(n)$. This is even worse than the trivial protocol in which the prover either sends the frequency vector $f := (f_1, \cdots, f_m)$ or the sorted stream, with a cost of $\tilde{O}\left(\min(m, n)\right)$. Since the proof is just a sorted stream, its correctness can be checked by standard fingerprinting techniques as described in [17]. One obvious way to rectify this would be to reduce the degree of the polynomial to be communicated at each round. One way to reduce the degree of $\tilde{h}$ is to remove all heavy hitters from the stream, so that the degree of $\tilde{h}$ can be made small (because $h(x)$ may take any value for large x), which in turn means that the communication will be low. The heavy hitter protocol in [9] however uses a lot of communication just to identify all the heavy hitters, which causes the communication cost in their protocol to be high. In this work, we also reduce the degree of the polynomial to be communicated at each round. But instead of removing the heavy hitters, we write F_0 as a different formula. Such an approach was first used by Gur and Raz [12] to obtain an AMA-protocol for exact F_0. Here, the main technical point is to replace the OR function on n variables (which has high degree) with a approximating polynomial over a smaller finite field $\mathbf{F}_q$, so that this new polynomial has low degree. Such approximating polynomials were first constructed in [24,25] to prove circuit lower bounds. The degree of the approximating polynomial $p : \mathbf{F}_q^n \to \mathbf{F}_q$ depends on q. But choosing q to be small forces us to work inside the field $\mathbf{F}_q$, and the arithmetic will be correct modulo q. Hence, F_0 will be calculated modulo q. Note that we cannot choose $q > m$ as the approximating polynomial degree will be larger than m. By choosing the first $\log m$ primes, we can compute F_0 modulo these $\log m$ many primes with the help and verifier's cost being polylogarithmic in m and n (see Lemma 3). This does not increase the number of rounds because all these executions can be done in parallel. The exact value of F_0 can be reconstructed by the Chinese Remainder Theorem. As a result of decreasing the degree of the polynomial, our protocol no longer has perfect completeness. By parallel repetition, the probability that a honest prover succeeds can be made close to 1.

We now compare our results with previously known non-interactive and interactive protocols that compute F_0 exactly. For comparison purposes, we assume that $m = \theta(n)$. The results are collected in Table 1. We note that if we fix the number of rounds to $\log m$, our work improves the total communication from $O\left(\sqrt{m}\log^2 m\right)$ to $O\left(\log^4 m\left(\log\log m\right)^2\right)$, while only increasing the the verifier's space by a multiplicative factor of $\log\log m$. For practical purposes, the authors in [9] argue that the number of rounds in the general purpose construction of [11], which is $\Omega(\log^2 m)$, may be large enough to be offputting. All the other protocols Cormode et al. [9] devised only require $\log m$ rounds. In an article in Forbes [10] in 2013, it was reported that the National Security Agency's data center in Utah will reportedly be capable of storing a yottabyte of data. For a yottabyte-sized input, this corresponds to about 80 rounds of interaction if one uses a protocol with $\log m$ rounds. For a protocol with $\log^2 m$ rounds, more than 6000 rounds of interaction are needed.

Recently, Chakrabarti et al. [6] designed a streaming interactive protocol for the Index function with two messages[1] where both space and communication are $O(\log n \log\log n)$. Previous work gave a $\widetilde{O}(\sqrt{n})$ protocol in the this online MA model [5], whereas in [9], a $\log n$ round interactive protocol with $O(\log n \log\log n)$ space and communication is given. Since for the INDEX function, there is a two message protocol requiring only $O(\log n \log\log n)$ space and communication, one may ask whether a similar kind of protocol is possible for F_0 or other frequency moments. It is however easy to see that for $k \neq 1$, the k-th frequency moment, F_k is as hard as the Disjointness function. In any online communication protocol for the Disjointness function with 2 and 3 messages, there is a lower bound of $\Omega(n^{1/2})$ and $\Omega(n^{1/3})$ respectively [6]. Hence, it is not possible to compute F_0 exactly using only $1, 2$ or 3 messages with communication and space polylogarithmic in m and n. How about using a constant number of r messages, where

[1] The first message is from the verifier to the prover and this message depends on the stream and the verifier's private randomness. The second message is from the prover to the verifier, which depends on the stream and the message received from the verifier.

$r \geq 4$, to get communication and space polylogarithmic in m and n? It is believed that this is not possible: due to the recent results in [6], if Disjointness on n bits can be solved with a constant number of rounds and polylogarithmic complexity in the online one-way communication model, then the (ordinary) AM communication complexity of Disjointness will also be $polylog(n)$, which is unlikely, since Disjointness is the generic co-NP complete problem in communication complexity [3]. Hence, constant round protocols ($r \geq 4$) for $F_k (k \neq 1)$ with polylogarithmic complexity probably do not exist, but the current techniques in communication complexity (i.e., providing strong lower bound on the AM communication complexity of Disjointness) are not sufficient to prove this.

Table 1. Comparison of our protocol to previous protocols for computing the exact number of distinct elements in a data stream. The results are stated for the case where $m = \theta(n)$. The space and the total communication bounds are stated asymptotically.

Paper	Space	Total Communication	Number of Rounds
[5]	$m^{2/3} \log m$	$m^{2/3} \log m$	1
[8]	$\log^2 m$	$\log^3 m$	$\log^2 m$
[9]	$\log^2 m$	$\sqrt{m} \log^2 m$	$\log m$
This work	$\log^2 m \log\log m$	$\log^4 m\, (\log\log m)^2$	$\log m$

2 Preliminaries

In this section, we define the model of streaming computations with a helper/ prover and introduce basic notations from coding theory. The reader is referred to the full version [18] for a detailed discussion of the streaming model with a prover.

2.1 Data Streaming Model

The input is given as a data stream $\sigma = \langle a_1, \ldots, a_n \rangle$ of elements from a universe $\{1, \ldots, m\}$. The a_i are sometimes referred to as symbols. In our model we consider two parties, the prover, and the verifier. Both parties are able to access the data stream one element at a time, consecutively, and synchronously, i.e., no party can look into the future with respect to the other one.

Definition 1. *After the stream ends, both the prover $\mathcal{P}$ and verifier $\mathcal{V}$ exchange some messages between each other. We denote the output of $\mathcal{V}$ on input σ, given prover $\mathcal{P}$ and $\mathcal{V}$'s private randomness $\mathcal{R}$, by $out(\mathcal{V}, \mathcal{P}, \mathcal{R}, \sigma)$. During any phase of the interaction, $\mathcal{V}$ can output $\perp$ if $\mathcal{V}$ is not convinced that $\mathcal{P}$'s claim is valid. We say $\mathcal{P}$ is a valid prover if for all streams σ,*

$$Pr_R\, [out(\mathcal{V}, \mathcal{P}, \mathcal{R}, \sigma) = f(\sigma)] \geq 1 - \epsilon_c.$$

We say $\mathcal{V}$ is a valid verifier for f if there is at least one valid prover $\mathcal{P}$, and for all provers $\mathcal{P}'$ and all streams σ,

$$Pr_R\left[out(\mathcal{V},\mathcal{P},\mathcal{R},\sigma) \notin \{f(\sigma),\perp\}\right] \le \epsilon_s.$$

ϵ_c and ϵ_s are known as the completeness error and soundness error respectively. In this work, we take $\epsilon_c = \epsilon_s = \frac{1}{3}$. By standard boosting techniques, these probabilities can be made arbitrary close to 1 [2].

Definition 2. *We say there is a (h,v) streaming interactive protocol (SIP) with r rounds that computes f, if there is a valid verifier $\mathcal{V}$ for f such that:*

1. *$\mathcal{V}$ has only access to $O(v)$ bits of working memory.*
2. *There is a valid prover $\mathcal{P}$ for $\mathcal{V}$ such that $\mathcal{P}$ and $\mathcal{V}$ exchange at most $2r$ messages in total, and the sum of the length of all messages is $O(h)$ bits.*

2.2 Coding Theory

A q-ary linear code $\mathcal{C}$ of length n is a linear subspace of $\mathbf{F}_q^n$, where q is some prime power. If $\mathcal{C}$ has dimension k and minimum distance d, then we call it a $[n,k,d]_q$ code. Justesen codes [13] is a class of codes with constant alphabet size and they have constant relative distance and rate. It is known that Justesen codes are locally logspace constructible (see Lemma 3.3 of [19]). For more details of standard definitions in coding theory, the reader is referred to [26].

3 Our Results

For each $j \in [m]$ and $i \in [n]$, we define $\chi_i : [m] \to \{0,1\}$ such that $\chi_i(j) = 1 \Leftrightarrow a_i = j$. We can also interpret each $\chi_i : \{0,1\}^{\log m} \to \{0,1\}$ by associating each $j \in [m]$ with its binary expansion. It is easy to see that

$$F_0 = \sum_{j=1}^{m}\left(\bigvee_{i=1}^{n} \chi_i(j)\right) = \sum_{x_1\in\{0,1\}} \cdots \sum_{x_d\in\{0,1\}} \mathrm{OR}\left(\chi(x_1,\cdots,x_d)\right)$$

where $d = \log m$, $\chi : \{0,1\}^d \to \{0,1\}^n$ is

$$\chi(x_1,\cdots,x_d) := \left(\chi_1(x_1,\cdots,x_d),\cdots,\chi_n(x_1,\cdots,x_d)\right).$$

and $\mathrm{OR} : \{0,1\}^n \to \{0,1\}$ is the OR function on n variables.

Following the ideas of [20], we consider the low degree extension of χ_i over a larger field. Let q be a prime and λ an integer to be determined later. We extend the domain of χ_i from $\mathbf{F}_2^d$ to $\mathbf{F}_{q^\lambda}^d$. For $1 \le i \le n$ and $1 \le k \le d$, we denote the k^{th} bit of the binary expansion of a_i by $a_i^{(k)}$. The extension $\widetilde{\chi_i} : \mathbf{F}_{q^\lambda}^d \to \mathbf{F}_{q^\lambda}$ is given by

$$\widetilde{\chi_i}(x_1,\cdots,x_d) := \prod_{j=1}^{d}\left[\left(2a_i^{(j)} - 1\right)x_j + \left(1 - a_i^{(j)}\right)\right]. \tag{1}$$

Note that $\widetilde{\chi_i}(x_1, \cdots, x_d) = \chi_i(x_1, \cdots, x_d)$ for all $x \in \mathbf{F}_2^d$. Similarly, define $\widetilde{\chi} : \mathbf{F}_{q^\lambda}^d \to \mathbf{F}_{q^\lambda}^n$ in the natural way:

$$\widetilde{\chi}(x_1, \cdots, x_d) := (\widetilde{\chi_1}(x_1, \cdots, x_d), \cdots, \widetilde{\chi_n}(x_1, \cdots, x_d)) .$$

With this notation,

$$F_0 = \sum_{x_1 \in \{0,1\}} \cdots \sum_{x_d \in \{0,1\}} \mathrm{OR}\left(\widetilde{\chi}(x_1, \cdots, x_d)\right) . \tag{2}$$

Running the sum-check protocol naively to (2) would require the prover to send a degree n polynomial at each round. We replace the OR function in (2) with a low degree polynomial which approximates the OR function with high probability. This idea was first introduced in [22, 25] and was also used in [12] to obtain an AMA protocol for exact F_0.

Lemma 1. *Using $O(L \log n)$ bits of randomness, we can construct a polynomial $p : \mathbf{F}_q^n \to \mathbf{F}_q$ of individual degree at most $L(q-1)$, such that for every $x \in \{0,1\}^d$,*

$$Pr\left[p\left(\widetilde{\chi}(x_1, \cdots, x_d)\right) = \mathrm{OR}\left(\widetilde{\chi}(x_1, \cdots, x_d)\right)\right] \geq 1 - \frac{1}{6m \log m},$$

where L is the least integer such that

$$\left(\frac{2}{3}\right)^L \leq \frac{1}{6m \log m}. \tag{3}$$

Proof. Start with a $[\zeta n, n, \frac{1}{3}\zeta n]_q$-linear code $\mathcal{C}$, where $\zeta > 1$ is a constant to be chosen such that $\mathcal{C}$ exist. Let G be the generator matrix of $\mathcal{C}$. Choose uniformly at random $\alpha_1, \cdots, \alpha_L \in [\zeta n]$ where L is the least integer that satisfies (3) and define

$$p(x_1, \cdots, x_n) := 1 - \prod_{i=1}^{L} \left[1 - ((Gx)_{\alpha_i})^{q-1}\right] .$$

It is easy to see that the individual degree of p is at most $L(q-1)$. By properties of the code $\mathcal{C}$, for any $x \in \{0,1\}^n$,

$$\Pr_{\alpha_1, \cdots, \alpha_L}\left[p(x) \neq \bigvee_i x_i\right] \leq \left(\frac{2}{3}\right)^L \leq \frac{1}{6m \log m}.$$

□

Since $\mathbf{F}_{q^\lambda}$ can be viewed as a vector space over $\mathbf{F}_q$, we can view $p : \mathbf{F}_q^n \to \mathbf{F}_q$ as $\widetilde{p} : \mathbf{F}_{q^\lambda}^n \to \mathbf{F}_{q^\lambda}$, by applying p componentwise. By the union bound, the probability that

$$Pr\left[F_0 \pmod q = \sum_{x_1 \in \{0,1\}} \cdots \sum_{x_d \in \{0,1\}} \widetilde{p}\left(\widetilde{\chi}(x_1, \cdots, x_d)\right)\right] \geq 1 - \frac{1}{6 \log m}.$$

We first give a interactive protocol to compute $F_0 \pmod{q}$ with high probability. Let $q \leq 2\log m \log\log m + 2$ be a prime and λ be the smallest integer such that $q^{\lambda-1} \geq 6Ld\log m$. Before observing the stream, the prover and verifier agree on the code $\mathcal{C}$ as in Lemma 1. The verifier chooses $O(L\log n)$ random bits to define the polynomial $\widetilde{p}$ and sends this randomness to the prover. The verifier chooses randomly $r \in \mathbf{F}_{q^\lambda}^d$ and computes $\widetilde{p}\left(\widetilde{\chi}(r_1, \cdots, r_d)\right)$ in a streaming fashion. In the full version [18], we illustrate how the verifier can compute $\widetilde{p}\left(\widetilde{\chi}(r_1, \cdots, r_d)\right)$ given a one-pass over the stream without storing the whole input.

After the stream ends, the verification protocol proceeds in d rounds to compute $F_0 \pmod{q}$ with probability at least $1 - \frac{1}{6\log m}$. In the first round, the prover sends a polynomial $g_1(X_1)$ which is claimed to be

$$g_1(X_1) = \sum_{x_2 \in \{0,1\}} \cdots \sum_{x_d \in \{0,1\}} \widetilde{p}\left(\widetilde{\chi_1}\left(X_1, x_2, \cdots, x_d\right), \cdots, \widetilde{\chi_n}\left(X_1, x_2 \cdots, x_d\right)\right).$$

The polynomial $g_1(X_1)$ has degree $L(q-1)$ which can be described in $O(Lq\log q^\lambda)$ bits. The verifier need not store $g_1(X_1)$ but just need to compute $g_1(r_1)$, $g_1(0)$ and $g_1(1)$, which can be done in a streaming fashion. Note that if the prover is honest, then

$$F_0 \pmod{q} = g_1(0) + g_1(1). \tag{4}$$

In round $2 \leq j \leq d-1$, the verifier sends r_{j-1} to the prover who then sends the polynomial $g_j(X_j)$, which is claimed to be

$$g_j(X_j) = \sum_{x_{j+1} \in \{0,1\}} \cdots \sum_{x_d \in \{0,1\}} \widetilde{p}(\widetilde{\chi_1}\left(r_1, \cdots, r_{j-1}, X_j, x_{j+1}, \cdots, x_d\right), \cdots$$
$$\cdots, \widetilde{\chi_n}\left(r_1, \cdots, r_{j-1}, X_j, x_{j+1}, \cdots, x_d\right))$$

The verifier computes $g_j(r_j)$, $g_j(0)$ and $g_j(1)$ and proceeds to the next round only if the degree of g_j is at most $L(q-1)$ and

$$g_{j-1}(r_{j-1}) = g_j(0) + g_j(1).$$

In the final round, the verifier sends r_{d-1} to the prover who then sends the polynomial $g_d(X_d)$, which is claimed to be

$$g_d(X_d) = \widetilde{p}\left(\widetilde{\chi_1}\left(r_1, \cdots, r_{d-1}, X_d\right), \cdots, \widetilde{\chi_n}\left(r_1, \cdots, r_{d-1}, X_d\right)\right).$$

The verifier only accepts that (4) is computed correctly if g_d is of the correct degree, $g_{d-1}(r_{d-1}) = g_d(0) + g_d(1)$ and $g_d(r_d) = \widetilde{p}\left(\widetilde{\chi}(r_1, \cdots, r_d)\right)$.

Lemma 2. *In the case of the honest prover, the verifier will accept the wrong value of* $F_0 \pmod{q}$ *with probability at most* $\frac{1}{6\log m}$. *If*

$$\sum_{x_1 \in \{0,1\}} \cdots \sum_{x_d \in \{0,1\}} \widetilde{p}\left(\widetilde{\chi}(x_1, \cdots, x_d)\right)$$

correctly represents $F_0 \pmod q$ and if the prover cheats by sending some polynomial which does not need the requirements of the protocol, the verifier will accept with probability at most $\frac{L(q-1)d}{q^\lambda}$.

The proof of Lemma 2 can be found in the full version [18].

Analysis of Space and Communication. We now analyse the space needed by the verifier and the total communication between the prover and verifier over the $\log m$ rounds to verify $F_0 \pmod q$. First, let us look at the space complexity of the verifier. He needs to store $\alpha_1, \cdots, \alpha_L$ which will take $O(\log m \log n)$ bits of space. With $O(\log m \log\log m)$ bits of space, the verifier can compute $\widetilde{p}\,(\widetilde{\chi}(r_1, \cdots, r_d))$ when observing the stream. Note during the interaction with the prover after the stream ends, at each round $1 \le j \le d$, the verifier need not store the polynomial $g_j(X_j)$ but only need to evaluate g_j at a constant number of points. Hence, the space complexity of the verifier is $O\left(\log m\left[\log n + \log\log m\right]\right)$ bits.

We now bound the total communication between the prover and verifier. The verifier needs to communicate $\alpha_1, \cdots, \alpha_L$ and $r_1, \cdots, r_{d-1}$ to the prover, with cost $O(\log m \log n)$ and $O(\log m \log\log m)$ respectively. The prover, who needs to send $g_1(X_1), \cdots, g_d(X_d)$, uses $O(dLq \log q^\lambda) = O(q \cdot \log^2 m \log\log m)$ bits to communicate all these polynomials. Hence, the total communication is $O\left(\log m\left(\log n + q \log m \log\log m\right)\right)$ bits. We summarize our result below.

Lemma 3. *There exist an (h, v) SIP with $\log m$ rounds with $h = \log m\left(\log n + q\log m \log\log m\right)$ and $v = \log m\left(\log n + \log\log m\right)$ that computes $F_0 \pmod q$, where the completeness error is $\frac{1}{6\log m}$ and the soundness error $\frac{1}{3\log m}$ for any prime $q \le 2\log m \log\log m + 2$.*

Computing F_0 Exactly. Lemma 3 gives us an streaming interactive protocol to verify the correctness of $F_0 \pmod q$ with high probability for any prime $q \le 2\log m \log\log m + 2$. Now, we show how the prover can verify F_0 with high probability. Let $Q = \{q_1, \cdots, q_{\log m}\}$ be the first $\log m$ primes. Note that $q_{\log m} \le 2\log m \log\log m + 2$ for all $m \ge 2$ [4] and $\prod_{i=1}^{\log m} q_i > m$. The verifier computes $F_0 \pmod{q_i}$ for $i = 1, \cdots, \log m$. Note that this can be done in parallel and will cause the working space of the verifier and the total communication to increase, but the number of rounds is still $\log m$. By using the Chinese remainder theorem, the verifier can compute F_0 exactly given $F_0 \pmod{q_i}$ for $i = 1, \cdots, \log m$. By the union bound, the completeness and soundness error are $1/6$ and $1/3$ respectively.

In the preprocessing phase (even before seeing the data), the verifier and prover agree on a constant $\zeta > 0$ such that the linear code $\mathcal{C}_i := [\zeta n, n, \frac{1}{3}\zeta n]_{q_i}$ exists for all $1 \le i \le \log m$. Note that the same $\alpha_1, \cdots, \alpha_L$ can be used to define the polynomial $\widetilde{p_i} : \mathbf{F}_{q_i^\lambda}^n \to \mathbf{F}_{q_i^\lambda}$ for each $1 \le i \le \log m$. For each $1 \le i \le \log m$, the verifier needs to choose uniformly at random $r^{(i)} \in \mathbf{F}_{q_i^\lambda}^d$ and compute $\widetilde{p}\left(\widetilde{\chi}\left(r^{(i)}\right)\right)$. This can be done in space $O\left(\log^2 m \log\log m\right)$. Hence, the total space used by the verifier is $O(\log m\left(\log n + \log m \log\log m\right))$.

To bound the total communication, we need the following fact: Let p_n be the n^{th} prime, then it is known [4] that $\sum_{i=1}^{n} p_i = \Theta(n^2 \log n)$ for all $n \geq 2$. Hence, the total communication is $O\left(\log m \log n + \log^4 m (\log\log m)^2\right)$.

Running Time of the Verifier. First, we analyse the processing time of each symbol seen in the stream. We suppose it takes unit time to add and multiply two field elements from $\mathbf{F}_{q^\lambda}$. For each symbol a_k seen, the verifier needs to compute $\widetilde{\chi_k}\left(r^{(q)}\right)$ where $r^{(q)} \in \mathbf{F}_{q^\lambda}^{d}$ for each $q \in Q$. From (1), it is easy to see that the verifier needs $O(d) = O(\log m)$ time to compute $\widetilde{\chi_k}\left(r^{(q)}\right)$ for each $q \in Q$. Hence the total time taken by the verifier to process each symbol is $O(\log^2 m)$. We state our main theorem below.

Theorem 1. *There exist an* (h, v) *SIP with* $\log m$ *rounds with* $h = \log m \left(\log n + \log^3 m (\log\log m)^2\right)$ *and* $v = \log m \left(\log n + \log m \log\log m\right)$ *that computes* F_0 *exactly, where the completeness and soundness error are* $1/6$ *and* $1/3$ *respectively. The update time for the verifier per symbol received is* $O(\log^2 m)$*.*

References

1. Alon, N., Matias, Y., Szegedy, M.: The space complexity of approximating the frequency moments. Journal of Computer and System Sciences 58(1), 137–147 (1999); Earlier version in STOC 1996
2. Arora, S., Barak, B.: Complexity Theory: A Modern Approach. Cambridge University Press (2009)
3. Babai, L., Frankl, P., Simon, J.: Complexity classes in communication complexity theory. In: Proceedings of 27th IEEE FOCS, pp. 337–347 (1986)
4. Bach, E., Shallit, J.: Algorithmic number theory, volume 1: efficient algorithms. MIT Press, Cambridge (1996), `http://www.math.uwaterloo.ca/~shallit/ant.html`
5. Chakrabarti, A., Cormode, G., McGregor, A., Thaler, J.: Annotations in data streams. Electronic Colloquium on Computational Complexity (ECCC) 19, 22 (2012)
6. Chakrabarti, A., Cormode, G., McGregor, A., Thaler, J., Venkatasubramanian, S.: On interactivity in Arthur-Merlin communication and stream computation
7. Cormode, G., Mitzenmacher, M., Thaler, J.: Streaming Graph Computations with a Helpful Advisor. In: de Berg, M., Meyer, U. (eds.) ESA 2010, Part I. LNCS, vol. 6346, pp. 231–242. Springer, Heidelberg (2010)
8. Cormode, G., Mitzenmacher, M., Thaler, J.: Practical verified computation with streaming interactive proofs. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ITCS 2012, pp. 90–112. ACM, New York (2012)
9. Cormode, G., Thaler, J., Yi, K.: Verifying computations with streaming interactive proofs. Proc. VLDB Endow. 5(1), 25–36 (2011)
10. Foley, J.: As big data explodes, are you ready for yottabytes? (June 2013), `http://www.forbes.com/sites/oracle/2013/06/21/as-big-data-explodes-are-you-ready-for-yottabytes/`
11. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: interactive proofs for muggles. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, STOC 2008, pp. 113–122. ACM, New York (2008)

12. Gur, T., Raz, R.: Arthur-Merlin Streaming Complexity. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) ICALP 2013, Part I. LNCS, vol. 7965, pp. 528–539. Springer, Heidelberg (2013)
13. Justesen, J.: A class of constructive asymptotically good algebraic codes. IEEE Transactions on Information Theory 18, 652–656 (1972)
14. Kane, D.M., Nelson, J., Woodruff, D.P.: An optimal algorithm for the distinct elements problem. In: Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2010, pp. 41–52. ACM, New York (2010)
15. Klauck, H.: Rectangle size bounds and threshold covers in communication complexity. In: IEEE Conference on Computational Complexity, pp. 118–134. IEEE Computer Society (2003)
16. Klauck, H.: On Arthur Merlin games in communication complexity. CoRR, abs/1101.0523 (2011)
17. Klauck, H., Prakash, V.: Streaming computations with a loquacious prover. In: Proceedings of the 4th Conference on Innovations in Theoretical Computer Science, ITCS 2013, pp. 305–320. ACM, New York (2013)
18. Klauck, H., Prakash, V.: An improved interactive streaming algorithm for the distinct elements problem (2014), Paper available at http://arxiv.org/abs/1402.6800
19. Lachish, O., Newman, I., Shapira, A.: Space complexity vs. query complexity. Computational Complexity 17(1), 70–93 (2008)
20. Lund, C., Fortnow, L., Karloff, H., Nisan, N.: Algebraic methods for interactive proof systems. J. ACM 39(4), 859–868 (1992)
21. Muthukrishnan, S.: Data streams: Algorithms and applications. Foundations and Trends in Theoretical Computer Science 1(2) (2005)
22. Razborov, A.: Lower bounds on the size of bounded-depth networks over a complete basis with logical addition (russian). Matematicheskie Zametki 41(4), 333–338 (1987)
23. Razborov, A.: On the distributional complexity of disjointness. Theoretical Computer Science 106(2), 385–390 (1992)
24. Razborov, A.A., Epstein, M.A.: Lower bounds for the size of circuits of bounded depth in basis (1986)
25. Smolensky, R.: Algebraic methods in the theory of lower bounds for boolean circuit complexity. In: Proceedings of 19th ACM STOC, pp. 77–82 (1987)
26. van Lint, J.H.: Introduction to Coding Theory (Graduate Texts in Mathematics). 3rd rev. and exp. edn. Springer (1998)

A Faster Parameterized Algorithm for Treedepth*

Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil,
and Somnath Sikdar

Theoretical Computer Science, Department of Computer Science,
RWTH Aachen University, Aachen, Germany
{reidl,rossmani,fernando.sanchez,sikdar}@cs.rwth-aachen.de

Abstract. The width measure *treedepth*, also known as vertex ranking, centered coloring and elimination tree height, is a well-established notion which has recently seen a resurgence of interest. We present an algorithm which—given as input an n-vertex graph, a tree decomposition of width w, and an integer t—decides whether the input graph has treedepth at most t in time $2^{O(wt)} \cdot n$. We use this to construct further algorithms which do not require a tree decomposition as part of their input: A simple algorithm which decides treedepth in linear time for a fixed t, thus answering an open question posed by Ossona de Mendez and Nešetřil as to whether such an algorithm exists, a fast algorithm with running time $2^{O(t^2)} \cdot n$ and an algorithm for chordal graphs with running time $2^{O(t \log t)} \cdot n$.

Keywords: treedepth, vertex ranking, centered coloring, width measures, parameterized algorithms.

1 Introduction

The notion of treedepth has been introduced several times in the literature under different names. It seems that it was first formally studied by Pothen who used the term *minimum elimination tree* [19]; Katchalski et al. [12] who used the name *ordered colorings*; Bodlaender et al. in [1] under the term *vertex ranking*. More recently, Ossona de Mendez and Nešetřil brought the same concept to the limelight in the guise of *treedepth* in their book *Sparsity* [18].

There are several equivalent definitions of this term. One of the most intuitive characterizations of treedepth is via the degeneracy of the graph: a graph class has bounded treedepth if and only if the class is degenerate and there exists a constant l (that depends on the class) such that no graph from the class has an induced path of length at least l (Theorem 13.3 in [18]). That is, the condition that a graph has bounded treedepth imposes a slightly stronger restriction than just bounding the degeneracy of the graph. A particularly simple definition of treedepth is via the notion of vertex rankings. A *t-ranking* of a graph $G = (V, E)$

* Research funded by DFG-Project RO 927/13-1 "Pragmatic Parameterized Algorithms".

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 931–942, 2014.

is a vertex coloring $c\colon V \to \{1,\dots,t\}$ such that for any two vertices of the same color, any path connecting them has a vertex with a higher color. The minimum value of t for which such a coloring exists is the *treedepth* or the *vertex ranking number* of the graph. We denote the treedepth of a graph G by $\mathbf{td}(G)$. The vertex ranking number finds applications in sparse matrix factorization [9,13,15] and VLSI layout problems [14]. This notion also has important connections to the structure of sparse graphs. As Ossona de Mendez and Nešetřil show, a very general class of sparse graphs, the so-called graphs of bounded expansion, can be decomposed into pieces of bounded treedepth [17].

Formally, the Treedepth problem is to decide, given a graph G and an integer t, whether G has treedepth at most t. This decision problem is NP-complete even on co-bipartite graphs [1,19]. On trees, the problem can be decided in linear time [22]. Treedepth can be computed in polynomial time on permutation, circular permutation, interval, circular-arc, trapezoid graphs and cocomparability graphs of bounded dimension [6]. It is, however, NP-hard on chordal graphs [7]. The best-known approximation algorithm is due to Bodlaender et. al. [3] and has approximation ratio $O(\log^2 n)$, where n is the number of vertices in the graph. The best-known exact algorithm is due to Fomin, Giannopoulou and Pilipczuk [10] and runs in time $O^*(1.9602^n)$. For practical applications, several simple heuristics exist. One such heuristic is to find a balanced vertex separator, assign each vertex of the separator a distinct color and then recurse on the remaining components. This method shows that n-vertex planar graphs have a treedepth of $O(\sqrt{n})$. There are several good heuristics for obtaining balanced separators and some of the most practically useful ones rely on spectral techniques (see, for instance [20,23]).

Fixed-parameter tractability with the solution size as parameter follows directly from the fact that graphs of bounded treedepth are minor-closed. Then by the celebrated Graph Minors Theorem of Robertson and Seymour, graphs of treedepth t are characterized by a finite set of forbidden minors and whether a forbidden minor is present in an n-vertex graph can be tested in time $O(f(h)\cdot n^3)$ [21], where h is the number of vertices in the forbidden minor and f some recursive function. In their textbook *Sparsity* [18], Ossona de Mendez and Nešetřil point out that the property of having treedepth at most t is MSO-expressible for every fixed t and thus by Courcelle's theorem and the fact that graphs of bounded treedepth have bounded treewidth, the Treedepth problem can be decided in linear time. They propose the following problem: Is there a simple linear time algorithm to check $\mathbf{td}(G) \leqslant t$ for fixed t? Bodlaender et. al. in [1] provide a dynamic programming algorithm that takes as input a graph G and a tree decomposition of G of width w, and decides whether G has treedepth at most t in time[1] $2^{O(w^2 t)} \cdot n^2$. In this paper we present a linear time algorithm that decides whether $\mathbf{td}(G) \leqslant t$ in time $2^{O(wt)} \cdot n$, improving both the dependence on w and n. If indeed $\mathbf{td}(G) \leqslant t$, then the algorithm also constructs a treedepth

[1] The running time stated in [1] is *polynomial* time for a fixed t and w, but an analysis in these parameters is straightforward. From a personal communication [4], it seems that the running time can be improved to $2^{O(w^2 t)}n$.

decomposition within this time. That a better dynamic programming algorithm can be achieved using treedepth leads us to believe that representing the ranking of the vertices as a tree might be algorithmically helpful in other cases.

We can then easily extend this result to get a simple algorithm, which runs in time $2^{2^{O(t)}} \cdot n$, and, using a 5-approximation for treewidth by Bodlaender et. al. [2], an algorithm that runs in time $2^{O(t^2)} \cdot n$.

2 Preliminaries

We use standard graph-theoretic notation (see [8] for any undefined terminology, in particular tree decompositions). All our graphs are finite and simple. Given a graph G, we use $V(G)$ and $E(G)$ to denote its vertex and edge sets. For convenience we assume that $V(G)$ is a totally ordered set, and use uv instead of $\{u, v\}$ to denote the edges of G. For $X \subseteq V(G)$, we let $G[X]$ denote the subgraph of G induced by X. Given an edge $e = uv$ of a graph G, we let G/e denote the graph obtained from G by *contracting* the edge e, which amounts to deleting the endpoints of e, introducing a new vertex w_{uv}, and making it adjacent to all vertices in $(N(u) \cup N(v)) \setminus \{u, v\}$. For an edge $e = uv$, by *contracting v into u*, we mean contracting e and renaming the vertex w_{uv} by u. For a function $f\colon X \to Y$ and a set $X' \subseteq X$ we will define $f(X') = \{f(x) \mid x \in X'\}$. A *rooted graph* $G = (V, E, r)$ is a graph with the specified *universal vertex* $r \in V(G)$ which is connected to every other vertex of G. A *rooted tree* is a tree with a specially designated node known as the *root*. Let T be a rooted tree with root r and let $x \in V(T)$. Then an *ancestor* of x is any node $a \neq x$ on the path from r to x. Similarly a *descendant* of x is any node $d \neq x$ on a path from x to a leaf of T. A *rooted forest* is a disjoint union of rooted trees. Whenever we refer to a forest we will mean a rooted forest. For a node x in a tree T of a forest, the *depth* of x is the number of vertices in the path from the root of T to x (thus the depth of the root is one). The *height of a forest* is the maximum depth of a node of the forest, a forest with a single node therefore has height one. The *closure* $\mathrm{clos}(F)$ of a forest F is the graph $G = (\bigcup_{T \in F} V(T), E = \{xy \mid x \text{ is an ancestor of } y \text{ in } F\})$. Let x be a node of a tree T and let S be all the descendants of x in T. Then the *subtree of T rooted at x*, denoted by T_x, is the subtree of T induced by the node set $S \cup \{x\}$ with root x. Furthermore if C is a set of children of x in T and the set S' contains all descendants of nodes of C in T, the tree denoted by T_x^C, is the subtree of T induced by the node set $S' \cup C \cup \{x\}$ with root x. The height of a node x of a tree T is the height of T_x.

A *treedepth decomposition* of a graph G is a pair (F, ψ), where F is a rooted forest and $\psi\colon V(G) \to V(F)$ in an injective mapping such that if $uv \in E(G)$ then either $\psi(u)$ is an ancestor of $\psi(v)$ or vice versa. Whenever we deal with treedepth decompositions in this paper, the mapping ψ will usually be implicit as we will have $V(G) \subseteq V(F)$. The *treedepth* $\mathbf{td}(G)$ of a graph G is then the minimum height of any treedepth decomposition of G.

A treedepth decomposition of a graph is not unique. One can always add extra vertices to a treedepth decomposition without increasing its height. We introduce the notion of *trivially improvable treedepth decomposition* to differentiate between treedepth decomposition that have such unnecessary nodes and those who do not.

Definition 1 (Trivially Improvable Treedepth Decompositions). *A treedepth decomposition T of a graph G is* trivially improvable *if $V(G) \subsetneq V(T)$.*

We will also extensively use a special kind of treedepth decomposition that we will call *nice treedepth decomposition.* This notion is similar to that of *minimal trees* introduced in [10].

Definition 2 (Nice Treedepth Decomposition). *A treedepth decomposition T of G is* nice *if T is not trivially improvable and for every node $x \in V(T)$, the subgraph of G induced by the nodes in T_x is connected.*

Every graph admits a nice treedepth decomposition of minimal height. A proof of this fact is provided in the appendix.

3 Dynamic Programming Algorithm

In this section we present an algorithm that takes as input a graph G, a tree decomposition $\mathcal{T}$ of G of width w, and an integer t, and decides whether $\mathbf{td}(G) \leqslant t$ in time $2^{O(wt)} \cdot n$. For yes-instances, the algorithm can be modified to output a treedepth decomposition by backtracking. Later we will show how this algorithm can easily be used to achieve the three claimed results. The algorithm is a dynamic programming algorithm whose tables contain so-called *partial decompositions* for every bag of the tree decomposition. Each partial decomposition will correspond to a number of treedepth decompositions of that portion of the graph that was already processed during the dynamic programming. Those partial decompositions just store a minimally necessary amount of information about the implicitly constructed treedepth decompositions: The structure of subtrees of a treedepth decomposition that contain no vertex of the current bag is irrelevant for the remaining computation and thus such subtrees can be represented by a single number, namely their height. To retain the information about ancestor relationships between the already processed vertices, we maintain a single tree per partial decomposition. It turns out that for this purpose, it suffices to only store trees whose leaves are all contained in the current bag. Formally a partial decomposition is defined as follows.

Definition 3 (Partial Decomposition). *A* partial decomposition *is a triple (F, X, h), where F is a forest of rooted trees with $X \subseteq V(F)$ and $h\colon V(F) \to \mathbb{N}^+$ is a* height function *which obeys the property that for nodes $x, y \in V(F)$ with x an ancestor of y, $h(x) > h(y)$. Let R be the set of all roots in F. The height of (F, X, h) is* $\max_{x \in R} h(x)$.

It is important for the running time of our algorithm that the table sizes are small. We achieve this by introducing a notion of equivalency and subsequently only store one partial decomposition of every equivalence class.

Definition 4 (Partial Decomposition Equivalency). *Two partial decompositions (F_1, X_1, h_1) and (F_2, X_2, h_2) are* equivalent *if and only if $X_1 = X_2$ and there exists a bijective function $\psi\colon V(F_1) \to V(F_2)$ such that the following holds: The function ψ is an isomorphism between F_1 and F_2. For all $x \in X_1$, $\psi(x) = x$, that is, ψ is the identity map when restricted to the set X_1. For every node v in the forest F_1, $h_1(v) = h_2(\psi(v))$.*

Clearly two equivalent partial decompositions have the same height. We will keep a representative for each equivalence class during the dynamic programming. These partial decompositions will represent all pertinent treedepth decompositions of the portions of the graph seen so far. The way in which we will connect treedepth decompositions to partial decompositions will be based on the following operation.

Definition 5 (Restriction of a Partial Decomposition). *The* restriction of a partial decomposition (F, X, h) to $X' \subseteq X$ *is the partial decomposition (F', X', h'), where F' is obtained by iteratively deleting the leaves of the forest F that are* not *in X'. The height function h' is the restriction of h to $V(F')$.*

As we move from the leaves to the root of the tree decomposition the concept of a *topological generalization* will provide a relationship between the previous table of partial decompositions and the new ones we compute by enforcing that the predecessor relationship is maintained.

Definition 6 (Topological Generalization). *Let F_1, F_2 be rooted forests and let X be a set of vertices such that $X \subseteq V(F_1) \cap V(F_2)$. We say F_1* topologically generalizes *F_2 under X if there exists an injective mapping $f\colon V(F_2) \to V(F_1)$ where $f|_X = id$ and for any node $x \in V(F_2)$ and an ancestor y of x, $f(y)$ is an ancestor of $f(x)$ in F_1. We say that a partial decomposition (F_1, X_1, h_1) topologically generalizes a partial decomposition (F_2, X_2, h_2) if $X_2 \subseteq X_1$ and F_1 topologically generalizes F_2 under X_2.*

This definition does not capture the relation between the respective height functions that we want when dealing with partial decompositions. We therefore introduce the following notion of compatibility.

Definition 7 (Compatible Height Functions). *Let (F, X, h) be a partial decomposition and $(F', X', h'), X' \subseteq X$ such that F generalizes F' topologically as witnessed by the mapping $f\colon V(F') \to V(F)$. We say that h* is compatible with *h'* under *f if for every node $z \in F$ with children C in F it holds that*

$$h(z) = \begin{cases} \max\{1 + \max_{c \in C} h(c), h'(f^-1(z))\} & \textit{if } f^-1(z) \textit{ exists} \\ 1 + \max_{c \in C} h(c) & \textit{otherwise} \end{cases}$$

where we define the maximum over the empty set to be zero. Note that for every f, h' there always exists exactly one compatible height function which can easily be constructed via a bottom-up computation on F.

We will also show that it suffices to work on rooted graphs (i.e. a graph with a universal vertex r). This will significantly simplify both the algorithms and the proof of correctness, as we then may assume that a minimal treedepth decomposition has r as its root. As a consequence, we only need to consider treedepth decompositions that are trees. The main algorithm is as follows:

Algorithm 1. *The input is a graph G, an integer t, and a tree decomposition $\mathcal{T}$ of G. Add a universal node r to the graph and create a new tree decomposition $\mathcal{T}'$ where r has been added to every bag and a bag containing just r is added as a child to every leaf. Call Algorithm 2 on (G, r), $t+1$, $\mathcal{T}'$ and the root bag of $\mathcal{T}'$ and return true only if it does not return the empty set.*

Algorithm 2. treedepth-rec

```
Input: A rooted graph $G = (V, E, r)$, an integer $t$ and a nice tree decomposition
       $\mathcal{T}$ of $G$ containing $r$ in every bag and a bag $X$ of $\mathcal{T}$.
Output: A set $R$ of partial decompositions.
$R = \emptyset$;
if $X$ is a leaf then
    $r =$ the only vertex contained in $X$;
    $F =$ a tree consisting of just the node $r$;
    $h$ is a function which is only defined for $r$ and $h(r) = 1$;
    $R = \{(F, \{r\}, h)\}$;
else if $X$ is a forget bag then
    $u =$ forgotten vertex;
    $X' =$ the child of $X$;
    $R' =$ treedepth-rec$(G, t, \mathcal{T}, X')$;
    $R = \mathit{forget}(R', X', u)$;
else if $X$ is an introduce bag then
    $u =$ introduced vertex;
    $X' =$ the child of $X$;
    $R' =$ treedepth-rec$(G, t, \mathcal{T}, X')$;
    $R = \mathit{intro}_t(R', X', u, G)$;
else if $X$ is a join bag then
    $\{X_1, X_2\} =$ the set of children of $X$;
    $R_1 =$ treedepth-rec$(G, t, \mathcal{T}, X_1)$;
    $R_2 =$ treedepth-rec$(G, t, \mathcal{T}, X_2)$;
    $R = \mathit{join}_t(X, R_1, R_2, G)$;
return $R$;
```

In the above algorithm, the *introduce*, *forget*, and *join* bags refer, of course, to the bags in a nice tree decomposition, the definition of which has been omitted due to space constraints (refer to [8] for formal definitions). The definitions of functions *forget*, *intro* and *join* used in Algorithm 2 follow. We use the same variable names as in said algorithm for the sake of consistency.

Definition 8 (Forgetting a Vertex from a Partial Decomposition). *Let G be a graph, let $X' \subseteq V(G)$ and let R' be a set of partial decompositions on the set X'. Let further $u \in X'$ be a vertex.*

The result of the forget operation *on u denoted by $forget(R', X', u)$ is defined as a set R of pairwise non-equivalent partial decompositions of height at most t with the following property: for every partial decomposition $(F', X', h') \in R'$, the restriction of (F', X', h') to $X' \setminus \{u\}$ is equivalent to some partial decomposition contained in R.*

Note that the set R is not unique and that it contains only non-equivalent partial decompositions. Further, it is easy to see that it can be computed by taking restrictions to $X' \setminus \{u\}$ of every single partial decomposition in R' and adding it to R if no equivalent partial decomposition is already present.

The introduce operation is somewhat more involved. Given a set R' of partial decompositions of the form (F', X', h') where $X' \subseteq V(G)$, the result of introducing $u \in V(G) \setminus X'$ is again a set R of partial decompositions. The broad outline of this step is the following (a complete description can be found in Definition 9):

1. Guess every forest F that complies with certain conditions.
2. Find a partial decomposition $(F', X', h') \in R'$ such that F topologically generalizes F'. If no such partial decomposition exists, discard F.
3. Otherwise, create for every function f that witnesses F topologically generalizing F' a partial decomposition of the form $(F, X = X' \cup \{u\}, h)$ where h meets certain additional requirements.
4. Add (F, X, h) to R if its height is smaller than t and there is no equivalent partial decomposition already contained in R.

Definition 9 (Vertex Introduction into a Partial Decomposition). *Let $G = (V, E, r)$ be a rooted graph, let $X' \subseteq V(G)$ and let R' be a set of partial decompositions. For a vertex $u \in V(G) \setminus X'$ and an integer t, the result of the* introduction operation *on u denoted by $intro_t(R', X', u, G)$ is defined to be a set R of pairwise non-equivalent partial decompositions of height at most t with the following properties.*

For every $(F', X', h') \in R'$ and every tree F with the following properties

(i) r is the root of F;
(ii) $X \subseteq V(F)$;
(iii) All leaves of F are in X;
(iv) $E(G[X]) \subseteq E(clos(F)[X])$.

and for every surjective $f\colon V(F') \to V(F) \setminus \{u\}$ which witnesses that F topologically generalizes F' on $X \setminus \{u\}$, the set R contains a partial decomposition that is equivalent to (F, X, h) where h is the height function compatible with h' under f.

The join operation looks for partial decomposition in a similar fashion as the introduce operations, with the added difficulty that it has to find two partial decompositions in the previous tables which are both topologically generalized at the same time.

Definition 10 (Joining Partial Decompositions). *Let $G = (V, E, r)$ be a rooted graph. Let R_1 and R_2 be two sets of partial decompositions on $X \subseteq V(G)$. Let t be an integer. The result of the* join operation *denoted by $join_t(X, R_1, R_2, G)$ is a set R of pairwise non-equivalent partial decompositions of height at most t with the following properties.*

For every $(F_1, X, h_1) \in R_1$ and $(F_2, X, h_2) \in R_2$ and every tree F with the following properties (i) r is the root of F; (ii) $X \subseteq V(F)$; (iii) All leaves of F are in X; (iv) $E(G[X]) \subseteq E(clos(F)[X])$; and for every pair of functions $f_1 : V(F_1) \to V(F)$, $f_2 : V(F_2) \to V(F)$ with

(i) f_i witnesses that F topologically generalizes F_i on X, $i \in \{1, 2\}$
(ii) $f_1(F_1) \cap f_2(F_2) = X$
(iii) $f_1(F_1) \cup f_2(F_2) = V(F)$

the set R contains a partial decomposition equivalent to (F, X, h), with the following height function h: Given the height functions $\hat{h}_1, \hat{h}_2$ for F such that $\hat{h}_i$ is the height function compatible with h_i under f_i for $i \in \{1, 2\}$, the function h is defined as $h(z) = \max\{\hat{h}_1(z), \hat{h}_2(z)\}$ for every $z \in V(F)$.

Again we can compute the set R for a join by guessing every locally feasible tree F and trying to generating every partial decomposition from it that meets the above conditions.

We claim that this algorithm correctly decides, given an n-vertex graph G and a tree decomposition of width at most w, whether G has treedepth at most t in time $2^{O(wt)} \cdot n$. A sketch of the proof follows, the complete proof can be found in the appendix.

4 Proof Sketch

We will say that a partial decomposition (F, X, h) represents the treedepth decomposition T if it is a restriction of $(T, V(T), \text{height}_T)$ on X. Notice that then the height of the partial decomposition equals the height of the treedepth decompositions it represents. We will call a partial decomposition which represents a treedepth decomposition T *a restriction of* T. The proof works by showing the following statements consecutively. This proof has been omitted due to lack of space.

1. Every graph admits a nice treedepth decomposition of height $\mathbf{td}(G)$.
2. It is sufficient to work with rooted graphs that allow an optimal nice treedepth decomposition whose root is the root of graph.
3. For every nice treedepth decomposition T of the graph, our tables contain a restriction of T (Lemma 1).
4. Every partial decomposition contained in the table is a restriction of a treedepth decomposition of the graph (Lemma 2).

All the partial decompositions in our tables will be restrictions on the contents of a bag. We will provide a short sketch of the proof of the main Lemmas 1 and 2.

Lemma 1. *Let Algorithm 2 be called on $(G, t, \mathcal{T}, X)$, where G is a graph rooted at r, the remaining parameters $t, \mathcal{T}, X$ are as described in the algorithm. Then for every nice treedepth decomposition T of height at most t (that is rooted at r) of $G[V(\mathcal{T}_X)]$, the set R returned by the algorithm contains a restriction of T to the set X.*

Proof sketch. Since the forget case is rather simple we will omit it here.

We show that in the introduce case, the introduced vertex u can only be in a restricted number of positions in a nice treedepth decomposition: the vertex u can only be adjacent to vertices present in the current bag X and the edges present there must be contained in the closure of the guessed tree F. That the remaining edges are covered by all treedepth decompositions represented by a guessed restriction is proved analogously to the join case outlined below.

Let us now look at this last case: Let T be a nice treedepth decomposition of $G[V(\mathcal{T}_X)]$. We show that if (F, X, h) is the restriction of T to X our previous tables must contain partial decompositions (F_1, X, h_1) and (F_2, X, h_2) such that F is a topological generalization of F_1 and F_2. We do this by first converting T to nice treedepth decompositions T_1 and T_2 of $G[V(\mathcal{T}_{X_1})]$ and $G[V(\mathcal{T}_{X_2})]$. By induction we assume that R_1 and R_2 contain a restriction of T_1 and T_2 respectively. Let these be (F_1, X, h_1) and (F_2, X, h_2). We show that by the way we computed T_1 and T_2, the tree F will topologically generalize F_1 and F_2, thus we add a partial decomposition (F, X, h) to the result set of the join operation. Finally we prove that the height function h correctly reflects the height of T when computed from h_1 and h_2. This uses the fact that since T is nice and X is a separator, a subtree of T_1 or T_2 must be a single subtree in T, or else this subtree would contradict the niceness properties. □

Lemma 1 is not sufficient for the correctness of the algorithm, as the restrictions computed in the dynamic programming might not all correspond to actual treedepth decompositions. This part of the correctness proof is handled in the following lemma.

Lemma 2. *Let Algorithm 2 be called on $(G, t, \mathcal{T}, X)$, where G is a graph rooted at r, the remaining parameters $t, \mathcal{T}, X$ are as described in the algorithm. Then every member of R returned by the algorithm is a restriction of a treedepth decomposition of $G[V(\mathcal{T}_X)]$ to X.*

Proof sketch. We will only sketch the proof for the join case. Assume that we add a partial decomposition (F, X, h) to the table because F topologically generalizes F_1 and F_2. By induction we assume that $(F_1, X, h_1) \in R_1$ and $(F_2, X, h_2) \in R_2$ are restrictions of treedepth decompositions T_1 and T_2 of $G[V(\mathcal{T}_{X_1})]$ and $G[V(\mathcal{T}_{X_2})]$ respectively. We show we can construct a treedepth decomposition T of the graph $G[V(\mathcal{T}_X)]$ from (F_1, X, h_1), (F_2, X, h_2), T_1 and T_2 such that (F, X, h) is a restriction of it. □

We then show that there is a treedepth decompositions of depth t of the graph G only if the call from Algorithm 1 to Algorithm 2 does not return the empty set. Finally we prove the running time by simple counting arguments on partial decompositions. The only properties we use to show that there are at most $2^{O(tw)}$ entries in the tables is that the height function of a partial decomposition must increase monotonically, that they are all restrictions on the contents of a bag of the tree decomposition and that we do not keep two equivalent ones. Thus we will arrive at the following theorem.

Theorem 1. *Let G be a graph of size n and t an integer. Given a tree decomposition of G of width w, one can decide in time and space $2^{O(wt)} \cdot n$ whether G has treedepth at most t and if so, output a treedepth decomposition of that height.*

5 Algorithms Parameterized by Treedepth

Simple Algorithm. We can now use Theorem 1 to answer the problem posed by Ossona de Mendez and Nešetřil in [18]:

> Is there a simple linear time algorithm to check $td(G) \leqslant t$ for fixed t? Is there a simple linear time algorithm to compute a rooted forest Y of height t such that $G \subseteq clos(Y)$ (provided that such a rooted forest exists)?

The problem is motivated by the fact that treedepth—being a minor-closed property—can be expressed in monadic second order logic and thus one can employ Courcelle's theorem [5] to compute the treedepth of a graph of bounded treewidth in linear time. The above problem is motivated by the fact that the running time of this approach is unclear. More specifically, the standard proof of Courcelle's involves creating a tree-automaton whose size cannot be bounded by any elementary function in the formula size unless P=NP [11]. The algorithm presented in Chapter 3 can be extended to give a much more direct and simpler algorithm.

Theorem 2. *There is an algorithm that takes a graph G and a parameter t as input, and decides whether the treedepth of G is at most t in time and space $2^{2^{O(t)}} \cdot n$. If this is indeed the case, the algorithm outputs a treedepth decomposition of depth at most t.*

Proof. A DFS of graph with treedepth at most t has height at most 2^t and easily converts to a path decomposition of width at most 2^t [18]. An algorithm which first computes this path decomposition and then runs Algorithm 1 on it fulfills the requirements of the lemma. □

Fast Algorithm. The algorithm from Section 3 can be adapted to obtain an algorithm with a running time that is better than $2^{2^{O(t)}} \cdot n$. Instead of computing a treedepth decomposition using DFS, we use an approximation algorithm for the *treewidth* of a graph. Recently, a 5-approximation for treewidth was developed which runs in single exponential time [2]. This, together with the fact that $\mathbf{tw}(G) \leqslant \mathbf{td}(G)$, gives us all we need.

Theorem 3. *Deciding whether a graph G with n vertices has $\mathbf{td}(G) \leqslant t$ and constructing a treedepth decomposition of that height can be done in time $2^{O(t^2)} \cdot n$.*

Proof. First use the 5-approximation for treewidth ([2]) and check whether the tree decomposition it outputs has width $< 5t$. This takes time $2^{O(\mathbf{tw}(G))} \cdot n$. Since $\mathbf{tw}(G) \leqslant \mathbf{td}(G)$, this is actually $2^{O(\mathbf{td}(G))} \cdot n$. If $\mathbf{tw}(G) > 5t$ then $\mathbf{td}(G) > t$ and we say "no"; otherwise, run Algorithm 1 on this tree decomposition. The running time of this call will be $2^{O(t \cdot 5t)} \cdot n = 2^{O(t^2)} \cdot n$. □

For chordal graphs we can make use of the fact that the *clique tree* of a chordal graph can be converted into a nice tree decomposition whose bags induce cliques. Since the presence of a complete subgraph on more than t vertices implies that the treedepth is larger than t, the width of the clique tree and thus the treewidth is bounded by t for non-trivial instances. It then turns out that during the dynamic programming, the partial restrictions must necessarily be simple paths instead of forests: the vertices of each bag must all lie on a single path in order for their closure to form a clique. As the number of such restrictions is significantly lower, a simple modification to the algorithm (generating only paths as candidates for restrictions during join and introduce) yields a running time of $2^{O(t \log t)} n$.

6 Conclusions and Further Research

We provide a simple and self contained algorithm, i.e. an algorithm which does not rely on any other complex results, which decides whether a graph has treedepth at most t in time linear in the input size for every fixed t. This answers an open question posed in [18]. We also provide an explicit algorithm to decide the treedepth or construct a minimal treedepth decomposition of a given graph in time $2^{O(t^2)} n$.

A natural question that arises is whether there exists a constant-factor approximation algorithm for the Treedepth problem that runs in single-exponential time. Such an algorithm would remove the dependency of our algorithm from the treewidth-approximation algorithm (hoping that a direct approximation of treedepth would be simpler).

On the topic of width-measures, it still remains open whether graphs of low treedepth admit fast algorithms that are impossible on graphs of low pathwidth. This is motivated further by the fact that the construction proving lower bounds on graphs of bounded pathwidth clearly contain very long paths and thus have high treedepth [16]. Proving similar bounds for graphs of bounded treedepth would be equally insightful.

References

1. Bodlaender, H.L., Deogun, J.S., Jansen, K., Kloks, T., Kratsch, D., Müller, H., Tuza, Z.: Rankings of graphs. SIAM Journal of Discrete Mathematics 11(1), 168–181 (1998)
2. Bodlaender, H.L., Drange, P.G., Dregi, M.S., Fomin, F.V., Lokshtanov, D., Pilipczuk, M.: A O(c^k n) 5-approximation algorithm for treewidth. CoRR, abs/1304.6321 (2013)

3. Bodlaender, H.L., Gilbert, J.R., Hafsteinsson, H., Kloks, T.: Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. Journal of Algorithms 18(2), 238–255 (1995)
4. Bodlaender, H.L., Kratsch, D.: Personal communication (2014)
5. Courcelle, B.: The Monadic Second-Order Theory of Graphs. I. Recognizable Sets of Finite graphs. Information and Computation 85, 12–75 (1990)
6. Deogun, J.S., Kloks, T., Kratsch, D., Müller, H.: On vertex ranking for permutations and other graphs. In: Enjalbert, P., Mayr, E.W., Wagner, K.W. (eds.) STACS 1994. LNCS, vol. 775, pp. 747–758. Springer, Heidelberg (1994)
7. Dereniowski, D., Nadolski, A.: Vertex rankings of chordal graphs and weighted trees. Information Processing Letters 98, 96–100 (2006)
8. Diestel, R.: Graph Theory, 4th edn. Springer, Heidelberg (2010)
9. Duff, I.S., Reid, J.K.: The multifrontal solution of indefinite sparse symmetric linear equations. ACM Transactions on Mathematical Software 9, 302–325 (1983)
10. Fomin, F.V., Giannopoulou, A.C., Pilipczuk, M.: Computing tree-depth faster than 2^n. In: Gutin, G., Szeider, S. (eds.) IPEC 2013. LNCS, vol. 8246, pp. 137–149. Springer, Heidelberg (2013)
11. Frick, M., Grohe, M.: The complexity of first-order and monadic second-order logic revisited. Annals of Pure and Applied Logic 130(1-3), 3–31 (2004)
12. Katchalski, M., McCuaig, W., Seager, S.: Ordered colourings. Discrete Mathematics 142(1-3), 141–154 (1995)
13. Kaya, K., Uçar, B.: Constructing elimination trees for sparse unsymmetric matrices. SIAM Journal on Matrix Analysis and Applications 34(2), 345–354 (2013)
14. Leiserson, C.E.: Area-efficient graph layouts (for VLSI). In: FOCS, pp. 270–281 (1980)
15. Liu, J.W.H.: The role of elimination trees in sparse factorization. SIAM Journal on Matrix Analysis and Applications 11(1), 134–172 (1990)
16. Lokshtanov, D., Marx, D., Saurabh, S.: Known algorithms on graphs on bounded treewidth are probably optimal. In: Randall, D. (ed.) Proc. of 22nd SODA, pp. 777–789. SIAM (2011)
17. Nešetřil, J., Ossona de Mendez, P.: Grad and classes with bounded expansion I. Decompositions. European Journal of Combinatorics 29(3), 760–776 (2008)
18. Nešetřil, J., Ossona de Mendez, P.: Sparsity: Graphs, Structures, and Algorithms. Algorithms and Combinatorics, vol. 28. Springer, Heidelberg (2012)
19. Pothen, A.: The complexity of optimal elimination trees. Technical Report CS-88-13, Pennsylvannia State University (1988)
20. Pothen, A., Simon, H.D., Liou, K.-P.: Partitioning sparse matrices with eigenvectors of graphs. SIAM Journal of Matrix Analysis and Applications 11(3), 430–452 (1990)
21. Robertson, N., Seymour, P.D.: Graph minors XIII. The disjoint paths problem. Journal of Combinatorial Theory, Series B 63, 65–110 (1995)
22. Schäffer, A.A.: Optimal node ranking of trees in linear time. Information Processing Letters 33(2), 91–96 (1989)
23. Spielman, D.A., Teng, S.-H.: Spectral partitioning works: Planar graphs and finite element meshes. In: FOCS, pp. 96–105 (1996)

Pseudorandom Graphs in Data Structures

Omer Reingold[1], Ron D. Rothblum[2], and Udi Wieder[1]

[1] Microsoft Research
omer.reingold@microsoft.com
[2] Weizmann Institute
ron.rothblum@weizmann.ac.il, uwieder@microsoft.com

Abstract. We prove that the hash functions required for several data structure applications could be instantiated using the hash functions of Celis *et al.* (SIAM J. Comput., 2013). These functions simultaneously enjoy short description length as well as fast evaluation time. The applications we consider are: (1) Cuckoo Hashing, (2) Cuckoo Hashing with Stash and (3) the Power of Two Choices paradigm for load balancing. Our analysis relies on a notion of sparse pseudorandom graphs that are similar to random graphs in having no large connected component and no dense subgraph. Such graphs may be of independent interest. Relating pseudorandom graphs to the two-choice paradigm relies on a very simple new proof we give (at the price of somewhat worse parameters).

1 Introduction

A common assumption in the design of data structures is the availability of perfectly random hash functions, functions that are assumed to require no space to store and take unit time to compute. A large body of work is dedicated to the removal of this assumption by designing constructive and explicit families of functions with well defined properties. In this paper we show that the family of functions put forth by Celis *et al.* [CRSW13] can be used for variants of Cuckoo Hashing, and could also be used to place balls into bins in the "power of two choices" scheme. A function from this class can be represented using $O(\log n \log\log n)$ bits and evaluated in $O((\log\log n)^2)$ operations in a unit-RAM model. In both cases this is the first construction where the description length is lower than $O(\log^2 n)$ which is the bound obtained by $\log n$-wise independent functions. The analysis is done by showing that a random graph built using these functions shares structural properties with truly random graphs, a result that could have more applications and be of independent interest.

Cuckoo Hashing: In the dictionary problem the goal is to build a data structure that represents a set S of size n taken from a universe $\mathcal{U}$. The data structure should support look-ups, as well as updates. In an influential work, Pagh and Rodler [PR04] constructed a dictionary, referred to as *cuckoo hashing* which has worst case constant lookup time, amortized expected constant update time, and uses slightly more than $2n$ words for storing n elements. Additional attractive

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 943–954, 2014.

features of cuckoo hashing are that no dynamic memory allocation is performed (except for when the tables have to be resized), and the lookup procedure queries only two memory entries which are independent and can be queried in parallel.

Cuckoo hashing uses two tables T_1 and T_2, each consisting of $r \geq (1+\epsilon)n$ words for some small constant $\epsilon > 0$, and two hash functions $h, g : \mathcal{U} \to \{0, \ldots, r-1\}$. An element $x \in \mathcal{U}$ is stored either in entry $h(x)$ of table T_1 or in entry $g(x)$ of table T_2, but never in both. The lookup procedure is straightforward: when given an element $x \in \mathcal{U}$, query the two possible memory entries in which x may be stored. For insertions, Pagh and Rodler [PR04] suggested the "cuckoo approach", kicking other elements out until every element has its own "nest". Insertion is shown to run in expected constant time and w.h.p no more than $O(\log n)$ time when the functions h and g are truly random, that is, they sample an element in $[r]$ uniformly and independently.

Random graphs: A random graph $G(m, n)$ over m nodes is obtained by sampling uniformly and independently n unordered pairs of nodes.[1] The sampling is done without replacement so we allow multiple edges and self loops. When the graph is sparse, i.e. when $n < (1-\epsilon)m/2$ for some $\epsilon > 0$, it is known that $G(m, n)$ is likely to have some distinctive properties: connected components are small and each connected component is sparse (we leave for now the exact definition of sparseness). These properties had been used in the analysis of Cuckoo Hashing in a natural way.

To analyze cuckoo hashing, let G be a graph with $(1+\epsilon)2n$ nodes, associated with the locations of the two tables T_1, T_2. For each item x in the set S that is represented by the data structure, add an edge $(h(x), g(x))$ (the graph as described here is bipartite, but this makes little difference in the analysis). Now it is not hard to observe that insertion succeeds as long as each connected component in G has at most one cycle. Pagh and Rodler proved that this is indeed the case w.h.p in this parameter regime. Furthermore, the expected and worst-case sizes of a connected component determine the expected and worst-case insertion time.

It follows that one way to prove that a specific (explicit) family of hash functions $\mathcal{H}$ could be used for cuckoo hashing is to show that the graph obtained by sampling h, g and constructing G as above has the same structural properties as a random graph. The goal is that the family $\mathcal{H}$ be as small as possible (so that the description length of each function is short) and that the functions be easy to evaluate. In this paper we show how to instantiate this approach with the function of [CRSW13]. We also find a surprising application for it to the power of two-choice paradigm.

[1] We use a non-standard notation where m denotes the number of nodes and n the number of edges for sake of consistency with the applications to data structures described below.

The power of two choices: In the Greedy[d] process (sometimes called the d-choice process), balls are placed sequentially into d bins with the following rule: each ball is placed by uniformly and independently sampling d bins and assigning the ball to the least loaded of the d bins. We are interested in the load, measured in number of balls, of the most loaded bin. The case $d = 1$, when balls are placed uniformly at random in the bins, is well understood. When n balls are thrown, the bin with the largest number of balls will have $\Theta(\log n / \log \log n)$ balls w.h.p. In an influential paper Azar *et al.* [ABKU99] showed that when $d > 1$ and n balls are thrown the maximum load is $\log \log n / \log d + O(1)$ w.h.p. The proof in [ABKU99] uses an inductive argument which relies on the assumption of full randomness.

1.1 Previous Work

A natural explicit family of functions that is useful in many applications is that of $\log n$-wise (almost) independent functions. These are functions where every $\log n$ evaluations are (almost) uniformly distributed. A $\log n$-wise independent family could be represented using $O(\log^2 n)$ bits, say by specifying the coefficients of a degree $\log n$ polynomial, and naively could be evaluated in $O(\log n)$ time. Recently, Meka *et al.* [MRRR13] constructed a family of $\log(n)$-wise almost independent *Boolean* hash function that can be evaluated in $O(\log \log n)$ time.

Intuitively it is clear why $O(\log n)$-independence should suffice for Cuckoo Hashing; the connected components of $G(m, n)$ are of logarithmic size, so in some sense all the 'interesting' events occur over a logarithmic number of variables. Indeed, the proof in [PR04] carefully counts subgraphs of a certain type (long paths and large trees) and eliminates them one by one using a union bound. Such an approach could be transformed almost as-is to handle $\log n$-wise independent functions.

A similar result for the 'power of two' case is not so straightforward, the proof in [ABKU99] uses an inductive argument that relies on the assumption of full randomness. A different proof was discovered by Vöcking' [Vöc03] which uses a construct called 'witness trees'. These are essentially labeled subgraphs of a naturally defined random graph. Vöcking's goal was to analyze a variant of the process called GoLeft, but as a byproduct he implicitly showed that $O(\log n)$-wise independent functions could be used. We show a different and much simpler way of using random graphs to prove this result - albeit with a larger constant in front of the $\log \log n$ term.

It is natural to ask whether one can construct a family of hash functions that improves both the time to evaluate a function and the space complexity (i.e., the description length). Ideally we would like functions that can be represented by $O(\log n)$ bits and evaluated in $O(1)$ time in the unit RAM model. It is worth mentioning that there exist important data structures for which such functions are known. A notable example is that of the linear probing data structure where Pagh et.al. [PPR11] show that any 5-wise independent hash function suffices. Mitzenmacher and Vadhan [MV08] showed that pairwise independent functions could be used for Cuckoo Hashing, providing the keys themselves come with sufficient entropy. For arbitrary input it is not known whether there exists a constant k such that any k-wise independent construct could be used for Cuckoo Hashing or balls-into-bins, and this remains an interesting open problem.

There is an impressive body of literature presenting functions with a running time of $O(1)$, at the expense of the space complexity. Starting with Siegel [Sie04], and continuing with e.g. Dietzfelbinger and Woelfel [DW03], Woelfel [Woe06], Patrascu and Thorup [PT12] and Thorup [Tho13]. All these solutions need $n^{\Theta(1)}$ bits of space to represent a function. In this paper we are more concerned with the space complexity. Can we construct a family of functions where a function can be represented using $o(\log^2 n)$ bits? A step in this direction is the load-balancing functions of [CRSW13] which are the starting point of our work.

1.2 Our Contributions

Our main contribution is to show that the family of functions put forth in [CRSW13] (henceforth, the CRSW hash functions) can be used for cuckoo hashing and for the power of two choices. The space needed to represent a function is $O(\log n \log\log n)$ bits, so these functions constitute the only family we are aware of that suffices for these applications with space complexity lower than that of $\log n$-wise independence. Additionally, it was recently shown by Meka *et al.* [MRRR13] that hash function from this family can be evaluated efficiently, in $O((\log\log n)^2)$ time.

Our analysis is done by proving that the random graph derived by this family has structural properties similar to those of a truly random graph; a result that may find further applications.

Let $\mathcal{U}$ be a universe of size $\mathsf{poly}(n)$ and let $\mathcal{H}$ be a family of functions from $\mathcal{U} \to V$. Every set $S \subset \mathcal{U}$ and pair of functions $h, g \in \mathcal{H}$ that are sampled uniformly and independently from $\mathcal{H}$, define a graph whose node set is V and with the edge set $\{(h(x), g(x)) : x \in S\}$. Our main result is that if $\mathcal{H}$ is the CRSW hash function family, $|S| = n$ and $|V| = O(n)$, then the graph G has the following two properties:

- **Small Components:** For every $v \in V$, let $C(v)$ be the connected component of v in G. Then the expected size of $C(v)$ is $O(1)$, and is $O(\log n)$ w.h.p.
- **Sparse Components:** W.h.p the number of edges in $C(v)$ is bounded by $2|C(v)|$ (where 2 is an arbitrary constant that can be made as close to 1 as we like).

As alluded to before, these two properties underlie the analysis of cuckoo hashing and could be used to prove double logarithmic bounds on the load of Greedy$[d]$. We show that $\mathcal{H}$ can be used for the following data structures.

1. **Cuckoo Hashing:** A cuckoo hashing scheme with $O(n)$ memory cells and two items per bucket. The idea to allow more than one item in a memory cell is due to Diezfelbinger and Weidling [DW07]. They show that this approach increases the memory utility of the scheme. We note that we do not match the space efficiency obtained with fully random functions and observe that it is not known whether $\log n$-wise independent function obtain it either.

2. **Cuckoo Hashing with Stash:** One drawback of cuckoo hashing is that with some fixed polynomially small probability an insertion operation might fail. Kirsch *et al.* showed in [KMW09] that allocating a special location in memory (stash) which may hold any data item and is searched on every look-up, indeed reduces the failure probability significantly, even when the stash is of constant size. Aumüller *et al.* [ADW12] show that $O(\log n)$-wise independent functions would work in this case. Our functions also benefit from the use of stash, that is, using only a constant size stash, we can guarantee an arbitrarily small polynomial probability of failure.
3. **The Power of Two Choices:** when n balls are placed in n bins using two random functions from $\mathcal{H}$, the maximum load is $O(\log \log n)$ w.h.p. We observe that we lose a constant factor over truly random functions in which case the load is $\log \log n + O(1)$. On the plus side our proof is remarkably simple (and of course also holds for the truly random case). Our simple idea is as follows: assume that instead of throwing n balls to n bins we throw $n/10$ balls. Now consider the graph with n vertices (corresponding to the bins) and $n/10$ edges (where an edge corresponding to a ball is adjacent to the two bins it can land at). The observation is that this graph is now sparse enough (or in other words, is in the sub-critical regime) and can have the pseudorandom properties of no large connected component and no dense subgraph. A very simple argument, based on the 'witness trees' of Vöcking' [Vöc03] implies an $O(\log \log n)$ load. We finally give a simple reduction from the case of n balls to n bins to the case of $n/10$ balls to n bins. The reduction also increases the maximum load by a constant factor (in fact this is where the main loss in parameters comes from). It is interesting to note that while the parameters are a bit weaker the final result is a bit stronger in the additional power it gives the adversary: the result holds even if (after the hash functions are made public) the adversary has full control over the order in which the balls arrive in each batch of $n/10$ balls.

1.3 Main Technical Ideas for the Pseudorandom Graphs

We sketch the main idea behind the CRSW functions. Each function maps an item from $\mathcal{U}$ to $[n]$, so the output is $\log n$ bits long, but it is often convenient to think of the output as one of n bins. The function h is obtained by sampling $d = \Theta(\log \log n)$ *different* functions and concatenating their output so that $h(x) = h_1(x) \circ h_2(x) \circ \cdots \circ h_d(x)$, where $\circ$ denotes concatenation. Each h_i is a k_i-independent function with an output length of ℓ_i bits, such that $\sum \ell_i = \log n$. It was shown in [CRSW13] that parameters could be chosen so that the total memory needed to represent h is $\log n \log \log n$ bits. The main idea is that as i increases the independence also increases (exponentially), so $k_1 = O(1)$ while $k_d = O(\log n)$. On the other hand, the output ℓ_i decreases with i, so that the space needed to represent the function remains small.

As shown in [CRSW13], the key property that these functions enjoy is that they are 'balancing'. Let $S \subset \mathcal{U}$ be a set of size n. Fix some $j \leq d$ and consider the first j functions $h_1, ..., h_j$. Their output is concatenated to produce a string of $\sum_{i \le j} \ell_i$ bits. The guarantee is that w.h.p h distributes the n values of S across these $2^{\sum_{i \le j} \ell_i}$ values as well as a random function. In particular, when $j = d$ we get that the most heavily loaded of the n bins receives at most $O(\log n / \log \log n)$ items (which was the main result of [CRSW13]).

How can we use this property to argue on the structure of graphs? Our goal is to show that certain types of subgraphs are not likely to occur. Specifically, we want to exclude logarithmically large trees and dense subgraphs. As each of these graphs have at most $O(\log n)$ edges, we could select h and g to be $O(\log n)$-wise independent and rule out the existence of each specific graph. If the parameters are set correctly, a union bound over all possible "bad" graphs would give the desired result. However, when we sample h and g from the CRSW family of functions then we only have $\log n$ wise independence for a short suffix of these functions output, which is insufficient for this analysis.

Instead, we apply the following strategy. Consider h as in the CRSW functions. It is the concatenation of $h_1 \dots h_d$. Assume we want to rule out bad graphs with t edges (where t is $O(\log n)$ but can also be as small as a constant). Let $j < d$ be minimal such that $h_{j+1} \dots h_d$ are all at least t-wise independent. Let $\ell' = \sum_{i \le j} \ell_i$, so ℓ' is the length of the concatenated output of the first j functions. We can take the vertices of G and collapse every $2^{j'}$ vertices which share the same prefix of length j' into one 'super vertex'. Now, the load balancing property tells us that the degree of each of these super vertices should be close to its expectation. We show that such load balancing is all we need from $h_1, \dots, h_j$. Now $h_{j+1} \dots h_d$ are not only independent of $h_1, \dots, h_j$ but also have sufficient independence for us to complete the proof.

Organization. In Section 2 we show how the CRSW hash functions can be used to produce graphs with strong random-like properties. In Section 3 we describe our applications to cuckoo hashing. Finally, in Section 4 we describe our applications to the power of two choice. We refer the reader to the full version for standard definitions (e.g., limited independence and tail inequalities) and an overview of the CRSW hash function.

2 Pseudorandom Graphs

Fix $n \le m \le u$ such that m is a power of two and $m = \alpha n$ for some sufficiently large fixed constant $\alpha > 1$. We consider graphs that are constructed by taking two hash functions and adding edges that correspond to evaluations of the hash functions in the following natural way:

Definition 2.1. *Let $h, g : [u] \to \{0, \dots, m-1\}$ be functions and let $S \subseteq [u]$ be a set of size n. We define the (multi-)graph $G_{h,g}(S) \overset{\text{def}}{=} \big(\{0, \dots, m-1\}, E\big)$, where $E \overset{\text{def}}{=} \{(h(x), g(x)) : x \in S\}$.*

Let $\mathcal{H}_{\mathsf{CRSW}}$ be the CRSW hash function family (see the full version for the precise definition). We show that for every $S \subseteq [u]$, with high probability, a random graph $G_{h,g}(S)$, where h and g are chosen at random from $\mathcal{H}_{\mathsf{CRSW}}$, does not contain any large connected subgraphs nor small connected subgraphs that are "dense". This is formalized by Theorems 2.2 and 2.3 below. In addition, we prove in Theorem 2.4 that in *expectation* each connected component has constant size.

Theorem 2.2. *For every constant $c_1 > 0$ and for any sufficiently large constant $c_3 > 0$, there exists a suitable setting of constants for $\mathcal{H}_{\mathsf{CRSW}}$ such that for every $S \subseteq [u]$, with probability $1 - n^{-c_1}$ over the choice of $h, g \in_R \mathcal{H}_{\mathsf{CRSW}}$, the graph $G_{h,g}(S)$ does not contain a connected subgraph of size greater than $c_3 \log n$.*

Theorem 2.3. *For every constant $c_2 > 0$ there exists a suitable setting of constants for $\mathcal{H}_{\mathsf{CRSW}}$ such that for every $S \subseteq [u]$, with probability $1 - O(1/n)$ over the choice of $h, g \in_R \mathcal{H}_{\mathsf{CRSW}}$, for every connected subgraph (V, E) of the graph $G_{h,g}(S)$ it holds that $|E| < (1 + c_2) \cdot |V|$.*

Theorem 2.4. *There exists a suitable setting of constants for $\mathcal{H}_{\mathsf{CRSW}}$ such that for every $S \subseteq [u]$ and for every $x \in S$, the expected size of the connected component that contains x (in $G_{h,g}(S)$) is $O(1)$.*

Remark. In the theorem statements, by a suitable setting of constants for $\mathcal{H}_{\mathsf{CRSW}}$, we mean that we can set the constants so that the k_i's and d are sufficiently large and so that the δ_i's are sufficiently small. In particular, we can set the constants so that all three theorems hold simultaneously.

To prove Theorems 2.2 to 2.4 we will consider appropriate connected graphs (that are either too large or too dense) and bound the probability that they appear in $G_{h,g}(S)$. The theorems follow by taking a union bound over all such graphs.

In order to bound the probability that a particular connected graph (V, E) appears in $G_{h,g}(S)$ we consider blocks of roughly $m^{1/|E|}$ consecutive vertices in $G_{h,g}(S)$. Observe that vertices in each such block all have the same $(1 - 1/|E|) \cdot \log(m)$-bit prefix and therefore, by the load balancing guarantee of $\mathcal{H}_{\mathsf{CRSW}}$ (for the appropriate prefix length), the number of edges that are connected to each block is roughly $n^{1/|E|}$ (i.e., as it should be if the functions were truly random). Moreover, the internal edges inside each block depend only on the $\log(m)/|E|$-bit suffix of the functions, which are $|E|$-wise (almost) independent. Since we consider events that involve at most $|E|$ items, we can treat this internal mapping inside the blocks as though it were fully random. These two properties allow us to bound the probability that connected graphs (V, E) that are either too big (i.e., $|V| = \Omega(\log n)$) or too dense (e.g., $|E| > 2|V|$). See the full version for the proofs of Theorems 2.2 to 2.4.

3 Cuckoo Hashing

Let $n \leq m \leq u$ and $\alpha > 1$ be as in Section 2. We consider a variant of cuckoo hashing where (as usual) we want to store n items in $m = \alpha n$ bins, but each bin can contain *two* items (in contrast to "vanilla" cuckoo hashing in which each bin holds at most one ball). As in vanilla cuckoo hashing, two hash functions $h, g : [u] \to [m]$ are chosen at random and each item x is stored either in bin $h(x)$ or bin $g(x)$. The hash functions h and g are chosen at random from the collection $\mathcal{H}_{\mathsf{CRSW}}$.

At any given point, if the data structure contains the items $S \subseteq [u]$, we can consider the cuckoo graph $G_{h,g}(S)$. Recall that this graph contains an edge $(h(x), g(x))$ for every $x \in S$. We view the graph as being directed where edges are directed from $h(x)$ to $g(x)$ if x is contained in the bin $h(x)$.

We define the insertion procedure as follows. To insert an element x, we start a depth first search (DFS) starting at $h(x)$ on the (directed) cuckoo graph. When taking a step on edge $(h(y), g(y))$ we switch the orientation of the edge, which corresponds to moving the item, say, from $h(y)$ to $g(y)$. Once we reach a vertex that has out-degree less than two, we stop the process (since the current item can be stored in the bin that corresponds to this vertex while maintaining a load of at most two items).

Theorem 3.1. *There exists a setting of parameters for $\mathcal{H}_{\mathsf{CRSW}}$ such that for every subset $S \subseteq [u]$ of size n, if we choose $h, g \in_R \mathcal{H}_{\mathsf{CRSW}}$ then, the items in S can be allocated to m bins such that:*

1. *Every item $x \in S$ is contained either in the bin $h(x)$ or $g(x)$.*
2. *Each bin holds at most two items.*
3. *With probability $1 - O(1/n)$ the insertion procedure takes time $O(\log n)$, and the expected insertion time is $O(1)$.*

Proof. Items 1 and 2 above follow directly from the construction. We proceed to prove that item 3 holds.

Let c_3 be a sufficiently large constant. We say that the graph $G_{h,g}(S)$ is good if it contains no connected subgraph (V, E) such that $|V| \geq c_3 \log n$ or $|E| \geq \frac{3}{2}|V|$. By Theorem 2.2 and Theorem 2.3, the graph $G_{h,g}(S)$ is good, with probability $1 - O(1/n)$ (over the choice of $h, g \in_R \mathcal{H}_{\mathsf{CRSW}}$).

Claim 3.2. *Let $S \subseteq [u]$ be a set of n items and suppose that the graph $G_{h,g}(S)$ is good. Then there is a vertex v in the connected component of $h(x)$ in the directed cuckoo graph with out-degree less than* 2.

Proof. Suppose that some of the items have been allocated and let $x \in S$ be an item that has not yet been allocated. Let $G_u = G_{h,g}(S)$ be the *undirected* cuckoo graph and let G_d be the *directed* cuckoo graph, where edges are directed from the bin that holds an item to the bin that does not (if an item has not yet been allocated then the corresponding edge does not appear in G_d). Let V be the connected component in G_u to which $h(x)$ belongs and suppose that each vertex in V has degree 3 (in G_u). Hence, the total number of edges in V is at least $\frac{3}{2}|V|$, in contradiction to $G_{h,g}(S)$ being good.

Hence, there exists a vertex v in the connected component of $h(x)$ (in G_u) with degree at most 2. If v has out-degree less than 2 in G_d then we are done. Otherwise (i.e., if v has out-degree exactly 2 in G_d), we remove v and recursively apply the argument to the induced subgraph. □

Hence, with probability $1 - O(1/n)$ the connected component of $h(x)$ in the directed cuckoo graph has a vertex with degree less than 2 (and the graph is good) and so the insertion time is at most the time for the DFS to go over the entire connected component. Since each connected component has size at most $O(\log n)$ with $O(\log n)$ edges, we obtain the worst case guarantee.

We proceed to prove that insertion takes expected time $O(1)$. Let I_x be the insertion time of x and let C_x be the size of the connected component that contains x (both I_x and C_x are random variables).

$$\begin{aligned}\mathbf{E}[I_x] &= \Pr[G_{h,g}(S) \text{ good}] \cdot \mathbf{E}[I_x | G_{h,g}(S) \text{ good}] + \Pr[G_{h,g}(S) \text{ not good}] \cdot \mathbf{E}[I_x | G_{h,g}(S) \text{ not good}] \\ &\leq \Pr[G_{h,g}(S) \text{ good}] \cdot \mathbf{E}[I_x | G_{h,g}(S) \text{ good}] + O(1/n) \cdot O(n) \\ &= \Pr[G_{h,g}(S) \text{ good}] \cdot O(\mathbf{E}[C_x | G_{h,g}(S) \text{ good}]) + O(1) \\ &\leq O(\mathbf{E}[C_x]) + O(1) \\ &= O(1)\end{aligned}$$

where the first inequality follows from the fact that connected component of x has (in the very worst case) $O(n)$ vertices and edges and the last inequality follows from Theorem 2.4. □

3.1 Cuckoo Hashing with Constant Size Stash

Theorem 3.1 shows that the probability for a "failure" (i.e., insertion that takes super logarithmic time) when using our variant of cuckoo hashing is at most $O(1/n^2)$. For some applications it is useful to obtain a probability of error that is inversely proportional to an *arbitrarily* large polynomial.

The same problem affects standard cuckoo hashing (with one item per bin and totally random functions). Kirsch *et al.* [KMW09] suggested to solve this problem by introducing a stash in which items may also be stored. The idea of [KMW09] is that if insertion takes more than logarithmic time, then the item is stored in the stash (for further details, see [KMW09]). To check whether an item $x \in [u]$ is the data structure, one simply checks if x is contained in $h(x)$, $g(x)$, or in the (constant size) stash.

In this section we show that one can decrease the error probability to be inversely proportional to an arbitrary polynomial using a constant size stash also in our variant of cuckoo hashing (i.e., with two items per bin and using the [CRSW13] hash functions).

Theorem 3.3. *For every $c_1 > 0$, there exists a constant $s > 0$ and a setting of parameters for $\mathcal{H}_{\mathsf{CRSW}}$ such that for every subset $S \subseteq [u]$ of size n, if we choose $h, g \in_R \mathcal{H}_{\mathsf{CRSW}}$ then, the items in S can be allocated to m bins and a stash of size s such that:*

1. *Every item $x \in S$ is contained either in the bin $h(x)$ or $g(x)$, or in the stash.*
2. *Each bin holds at most two items.*
3. *With probability $1 - n^{-c_1}$ insertion takes time $O(\log n)$, and the expected insertion time is $O(1)$.*

See the full version for the proof of Theorem 3.3.

4 Two Choice Hashing

Let $n \leq m \leq u$ and α be as in Section 2. In this section we consider a data structure that holds n items in $m = \alpha n$ bins, where each bin may hold multiple items. Later, in Section 4.1 we consider the classical setting where we wish to store m items (rather than just $n < m$ items) in the m bins.

To store the item, two hash functions $h, g : [u] \to [m]$ are chosen at random from the collection $\mathcal{H}_{\mathsf{CRSW}}$ and each item x is stored either in bin $h(x)$ or bin $g(x)$. When inserting, we simply insert x into the least loaded of the two bins $h(x)$ and $g(x)$ (while breaking ties by favoring, say, the bin $h(x)$).

As discussed in the introduction, we rely on the witness trees approach of [Vöc03], though we manage to significantly simplify the proof (at the price of somewhat worse parameters).

Lemma 4.1. *For every constant $\epsilon > 0$, there exists a setting of parameters for $\mathcal{H}_{\mathsf{CRSW}}$ such that, with probability $1 - O(1/n)$, each bin contains less than $(1 + \epsilon) \log \log n$ items.*

Proof. Suppose that we are storing a set S of n items. Let G_d be the directed graph on m vertices, where each vertex corresponds to a bin and each edge corresponds to an item in the following way: for every item $x \in S$ we have an edge directed from $h(x)$ to $g(x)$ (resp., from $g(x)$ to $h(x)$) if the item is stored in bin $h(x)$ (resp., $g(x)$). Note that the undirected version of G_d corresponds to the graph $G_{h,g}(S)$ (see Theorem 2.1). We will show that if x generates a load of $\Omega(\log \log n)$, then the connected component (V, E) of $G_{h,g}(S)$ to which the edge $(h(x), g(x))$ belongs to must be either large or dense. Since n is sufficiently smaller than m, these events were shown in Section 2 to occur with only small probability.

Fix $c_1 = 2$, let $c_2 > 0$ be a sufficiently small constant (that depends on ϵ) and let $c_3 > 0$ be sufficiently large so that Theorem 2.2 and Theorem 2.3 hold with respect to c_1, c_2 and c_3. Let $\ell \overset{\text{def}}{=} (1 + \epsilon) \log \log n$. We consider the event that there exists a bin $i \in [m]$ with ℓ items and bound the probability for this event. Consider the last item x that was inserted into bin i and suppose without loss of generality that $h(x) = i$ (the case that $g(x) = i$ is completely symmetric). We iteratively mark vertices and edges in G_d by the following process.

Initially we mark only the vertex $h(x)$ and none of the edges. Next we mark the vertex $g(x)$ and the edge $(h(x), g(x))$. Observe that the bin $g(x)$ contains at least $\ell - 1$ (unmarked) items (since otherwise x would have been inserted into $g(x)$), and therefore the vertex $g(x)$ is connected to at least $\ell - 1$ unmarked edges. We mark the vertex $g(x)$. Note that each marked vertex is now connected to at least $\ell - 1$ unmarked edges. We proceed iteratively, where at iteration i each marked vertex (i.e., bin) has $\ell - i$ unmarked neighbors (i.e., items) and in the next iteration we mark all edges that correspond to $(\ell - i)$-th ball that was inserted into each of the marked bins. Let V_i be the marked vertices in iteration i and let E_i be the marked edges in iteration i. Then,

$$\begin{aligned} |V_0| &= 1 \\ |E_0| &= 0 \\ |E_{i+1}| &= |E_i| + |V_i| \end{aligned}$$

for every $i \in [\ell]$ (where the last equality follows from the fact that each edge is marked at most once and at iteration i we mark V_i new edges).

We now show that if each of the graphs (V_i, E_i) is not dense (an event that by Theorem 2.3 happens with high probability) then the last graph (V_ℓ, E_ℓ) has size exponential in ℓ. Since $\ell = \Omega(\log\log n)$, we obtain a connected component in $G_{h,g}(S)$ with $\Omega(\log n)$ size (an event that by Theorem 2.2 only happens with small probability). We proceed to the actual proof.

By Theorems 2.2 and 2.3, with probability $1 - O(1/n)$ all connected subgraphs of $G_{h,g}(S)$ are not dense nor large. In particular, $|V_\ell| \leq c_3 \log n$ and for every $i \in \{0, \ldots, \ell\}$ it holds that $|E_i| \leq (1 + c_2)|V_i|$. Hence, $|E_\ell| \geq |E_{\ell-1}| + |V_{\ell-1}| \geq \frac{2+c_2}{1+c_2} \cdot |E_{\ell-1}| \geq \cdots \geq \left(\frac{2+c_2}{1+c_2}\right)^{\ell-1}$ and so the size of the last subgraph is:

$$|V_\ell| \geq \frac{1}{1+c_2} \cdot |E_\ell| \geq \frac{1}{1+c_2} \cdot \left(\frac{2+c_2}{1+c_2}\right)^{\ell-1} > c_3 \log n$$

where the last inequality follows by setting c_2 to be sufficiently small. Since such a connected subgraph of size $c_3 \log n$ only appears with probability $O(1/n)$, we conclude that the probability that there exists a bin with more than ℓ items is at most $O(1/n)$. □

4.1 m Items in m Bins

We first consider a slight generalization of Theorem 4.1 (that follows directly from Theorem 4.1).

Corollary 4.2. *Suppose that initially each of the m bins contains at most k items and then we add n items following the two choice paradigm, with respect to functions $h, g \in_R \mathcal{H}_{\mathsf{CRSW}}$. Then, with probability $1 - O(1/n)$, each bin contains less than $k + O(\log\log n)$ items.*

Proof. By induction. Suppose that after the t-th insertion the load of each bin i is at most $k + b_i$, where b_i denotes the load of bin i after the t-th insertion in the *original* process (i.e., when all bins are initially empty). Now insert the $(t+1)$-th item, which is hashed to bins i and j. Assume that originally (i.e., when bins are initially empty) the item is stored in bin i. Hence, $b_i \leq b_j$. If the item is still stored in i when the bins are first loaded with (at most) k items each then the induction trivially holds. On the other hand, if the item is stored in bin j then the load of j is smaller than $b_i + k \leq b_j + k$ and therefore even after this insertion the load is at most $k + b_j$ (i.e., only k more than in the original process). □

By partitioning the m items into α sets of m/α balls each and applying Theorem 4.2 on each set iteratively (thereby increasing the load by an additive $O(\log \log n)$ factor on each iteration), we obtain the following theorem:

Theorem 4.3. *Suppose that we hash a set* $S \subseteq [u]$ *of* m *items into* m *bins following the two choice paradigm with respect to* $h, g \in_R \mathcal{H}_{\mathsf{CRSW}}$*. Then, with probability* $1 - O(1/n)$*, each bin contains less than* $O(\log \log n)$ *items.*

Acknowledgments. We thank the anonymous reviewers for useful comments and for pointing out an error that appeared in a previous version.

References

ABKU99. Azar, Y., Broder, A., Karlin, A., Upfal, E.: Balanced allocations. SIAM J. Computing 29(1), 180–200 (1999)

ADW12. Aumüller, M., Dietzfelbinger, M., Woelfel, P.: Explicit and efficient hash families suffice for cuckoo hashing with a stash. In: Epstein, L., Ferragina, P. (eds.) ESA 2012. LNCS, vol. 7501, pp. 108–120. Springer, Heidelberg (2012)

CRSW13. Celis, L.E., Reingold, O., Segev, G., Wieder, U.: Balls and bins: Smaller hash families and faster evaluation. SIAM J. Comput. 42(3), 1030–1050 (2013)

DW03. Dietzfelbinger, M., Woelfel, P.: Almost random graphs with simple hash functions. In: STOC, pp. 629–638 (2003)

DW07. Dietzfelbinger, M., Weidling, C.: Balanced allocation and dictionaries with tightly packed constant size bins. Theor. Comput. Sci. 380(1-2), 47–68 (2007)

KMW09. Kirsch, A., Mitzenmacher, M., Wieder, U.: More robust hashing: Cuckoo hashing with a stash. Siam J. Computing 39(4), 1543–1561 (2009)

MRRR13. Meka, R., Reingold, O., Rothblum, G.N., Rothblum, R.D.: Fast pseudorandomness for independence and load balancing (2013) (manuscript)

MV08. Mitzenmacher, M., Vadhan, S.: Why simple hash functions work: Exploiting the entropy in a data stream. In: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008,, pp. 746–755 (2008)

PPR11. Pagh, A., Pagh, R., Ruzic, M.: Linear probing with 5-wise independence. SIAM Review 53(3), 547–558 (2011)

PR04. Pagh, R., Rodler, F.F.: Cuckoo hashing. Journal of Algorithms 51(2), 122–144 (2004)

PT12. Patrascu, M., Thorup, M.: The power of simple tabulation hashing. J. ACM 59(3), 14 (2012)

Sie04. Siegel, A.: On universal classes of extremely random constant-time hash functions. SIAM J. Comput. 33(3), 505–543 (2004)

Tho13. Thorup, M.: Simple tabulation, fast expanders, double tabulation, and high independence. In: FOCS, pp. 90–99 (2013)

Vöc03. Vöcking, B.: How asymmetry helps load balancing. J. ACM 50(4), 568–589 (2003)

Woe06. Woelfel, P.: Asymmetric balanced allocation with simple hash functions. In: SODA, pp. 424–433 (2006)

Sampling-Based Proofs of Almost-Periodicity Results and Algorithmic Applications

Eli Ben-Sasson[1,⋆], Noga Ron-Zewi[1,⋆⋆], Madhur Tulsiani[2,⋆⋆⋆], and Julia Wolf[3]

[1] Department of Computer Science, Technion
{eli,nogaz}@cs.technion.ac.il
[2] TTI Chicago
madhurt@ttic.edu
[3] Centre de Mathématiques Laurent Schwartz, École Polytechnique
julia.wolf@cantab.net

Abstract. We give new and simple combinatorial proofs of *almost-periodicity results* for sumsets of sets with small doubling in the spirit of Croot and Sisask [7], whose almost-periodicity lemma has had far-reaching implications in additive combinatorics. We provide an alternative point of view which relies only on Chernoff's bound for sampling, and avoids the need for L^p-norm estimates used in the original proof of Croot and Sisask.

We demonstrate the usefulness of our new approach by showing that one can easily deduce from it two significant recent results proved using Croot and Sisask almost-periodicity – the *quasipolynomial Bogolyubov-Ruzsa lemma* due to Sanders [22] and a result on large subspaces contained in sumsets of dense sets due to Croot, Laba and Sisask [6].

We then turn to algorithmic applications, and show that our approach allows for almost-periodicity proofs to be converted in a natural way to probabilistic algorithms that decide membership in almost-periodic sumsets of dense subsets of $\mathbb{F}_2^n$. Exploiting this, we give a new algorithmic version of the quasipolynomial Bogolyubov-Ruzsa lemma.

Together with the results by the last two authors [27], this implies an algorithmic version of the *quadratic Goldreich-Levin theorem* in which the number of terms in the quadratic Fourier decomposition of a given function, as well as the running time of the algorithm, are quasipolynomial in the error parameter ε. The algorithmic version of the quasipolynomial Bogolyubov-Ruzsa lemma also implies an improvement in running time and performance of the self-corrector for the Reed-Muller code of order 2 at distance $1/2 - \varepsilon$ in [27].

[⋆] The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 240258.

[⋆⋆] Research supported by a scholarship from the Israel Ministry of Science and Technology and by the US-Israel Binational Science Foundation. Part of the research was conducted while the author was a visiting researcher at Microsoft Research New England. The research leading to these results has also received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 257575.

[⋆⋆⋆] Research supported by NSF Career Award CCF-1254044.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 955–966, 2014.

1 Introduction

When Croot and Sisask introduced “A probabilistic technique for finding almost-periods of convolutions” in 2009 [7], it created quite a splash in the additive combinatorics community. Roughly speaking, their main result says that if $A \subseteq \mathbb{F}_2^n$ is a set whose sumset $A+A = \{a+a' : a, a' \in A\}$ is small, then there exists a dense set X such that the convolution $\mathbb{1}_A * \mathbb{1}_A(\cdot)$ of the indicator function of A with itself, and its translates $\mathbb{1}_A*\mathbb{1}_A(\cdot+x)$ for $x \in X$, are almost indistinguishable in the L^2 and higher L^p norms. This set X may then be referred to as the set of “almost periods”.

The two main combinatorial applications of the above technique were the proof of the quasipolynomial Bogolyubov-Ruzsa lemma due to Sanders [22] and a result saying that sumsets of dense sets contain large subspaces due to Croot, Łaba and Sisask [6]. Both applications made crucial use of the L^p-norm estimates of Croot and Sisask, where p was taken to be very large (a function of the density α of the set $A \subseteq \mathbb{F}_2^n$ under investigation, such as $\log \alpha^{-1}$).

Our main result is a simple combinatorial proof of almost-periodicity results in the spirit of Croot and Sisask that proceeds entirely without recourse to L^p-norms, instead only relying on the Chernoff bound for sampling. This is in contrast to Croot and Sisask's approach which obtained L^p-norm estimates using a simple sampling technique combined with tailbounds for a multinomial distribution, which Sanders replaced by the Marcinkiewicz-Zygmund inequality. It is our hope that this proof will appeal to a larger part of the theoretical computer science community than the currently existing ones, thereby increasing the likelihood of further novel applications of the almost-periodicity techniques.

We prove our almost-periodicity results in Section 4. We stress that our almost-periodicity approach works for arbitrary abelian groups, but for simplicity we state our results only over $\mathbb{F}_2^n$. We illustrate the use of our new approach by presenting simplified combinatorial proofs of known additive combinatorics results as well as new algorithmic applications. Due to space limitations both the combinatorial and the algorithmic applications are omitted. Let us describe these in more detail.

1.1 Combinatorial Applications

The quasipolynomial Bogolyubov-Ruzsa lemma. In its original form, the Bogolyubov-Ruzsa lemma states that if $A \subseteq \mathbb{F}_2^n$ is a set of density α, then $4A := A+A+A+A$ contains a subspace of codimension at most $2\alpha^{-2}$. One of the first applications Croot and Sisask gave of their new technique was a *quasipolynomial* Bogolyubov-Ruzsa lemma, which asserted that $4A$ contains iterated sumsets, of density at least $2^{-O(\log^4(1/\alpha))}$ inside $\mathbb{F}_2^n$. It was quickly recognized by Sanders [22] that the latter result could be boot-strapped, using a little Fourier analysis, to a quasipolynomial version of the Bogolyubov-Ruzsa lemma in which the codimension of the subspace that is found within $4A$ is at most $O(\log^4(\alpha^{-1}))$.

This result has important implications for the bounds in Freiman's theorem, which describes the structure of sets with small sumsets [20], and to the inverse theorem for the Gowers U^3 norm [21,16]. It is also a crucial ingredient in Sanders's groundbreaking upper bound of $C(\log\log N)^5 N/\log N$ for the size of a subset of $\{1,\ldots,N\}$ not containing any 3-term arithmetic progressions [24]. An improvement to $O(\log(1/\alpha))$ of the bound on the codimension of the subspace which is contained in $4A$ implies the *polynomial Freiman-Ruzsa conjecture* in additive combinatorics, which has found several applications to complexity theory so far [2,1,3]. See the survey of Green [14] for more information on the polynomial Freiman-Ruzsa conjecture and its combinatorial applications.

Sumsets of Dense Sets Contain Large Subspaces. Our second combinatorial application concerns the problem of finding large subspaces within sumsets of a dense set. Green [14] had shown that if $A \subseteq \mathbb{F}_2^n$ has density α, then $A + A$ contains a subspace of dimension $\Omega(\alpha^2 n)$. This was improved by Sanders who proved in [23] using a Fourier-iteration lemma that this subspace must be of dimension at least $\Omega(\alpha n)$. Croot, Łaba and Sisask [6], who addressed the more general problem in the integers, asking for long arithmetic progressions in sumsets of dense sets, remarked that a slightly worse bound of the form $\Omega\big(\frac{\alpha}{\log^3(1/\alpha)}n\big)$ follows implicitly from their techniques. Due to space limitations this part is omitted from this version, but can be found in the full-length version.

1.2 Algorithmic Applications

An advantage of our sampling-based almost-periodicity proofs is that they can be turned into algorithms that decide membership in almost-periodic sumsets of dense subsets of $\mathbb{F}_2^n$ in a rather natural way. In particular, using our new techniques we present in an algorithmic version of the quasipolynomial Bogolyubov-Ruzsa lemma. This is an algorithm which runs in time polynomial in n and quasipolynomial in $1/\alpha$, which, given an oracle access to an indicator function of a set A of density at least α inside $\mathbb{F}_2^n$, finds, with high probability, a basis to a subspace $V \subseteq 4A$ of codimension $O(\log^4(1/\alpha))$.

The main problem we encounter when converting our almost-periodicity proofs into such an algorithm is that our combinatorial proofs produce large subsets of $\mathbb{F}_2^n$ of size exponential in n. Since we are interested in an algorithm which runs in time polynomial in n, we cannot afford to store these sets or compute with them explicitly. Instead, we develop probabilistic procedures to efficiently test membership in such sets. We elaborate on these procedures in Section 2.

Combined with the results of [27], the algorithmic version of the quasipolynomial Bogolyubov-Ruzsa lemma implies an improvement in the running time and performance guarantee of a self-correction procedure for the Reed-Muller code of order 2 at distance $1/2 - \varepsilon$, given in [27]. This in turn leads to an improved quadratic Goldreich-Levin theorem in which the running time of the algorithm, as well as the number of terms in the quadratic Fourier decomposition of a given function, are quasipolynomial in the error parameter ε. We discuss these applications below.

One major difficulty we faced when trying to deduce the above applications from the algorithmic version of the quasipolynomial Bogolyubov-Ruzsa lemma, encountered already in [27], was that the individual subroutines in these applications, which correspond to algorithmic versions of theorems in additive combinatorics, are probabilistic in nature. Since they are applied in sequence, this means that the input for the next subroutine comes with a certain amount of noise, and it is therefore necessary to prove robust algorithmic versions of the theorems from additive combinatorics. This applies in particular to the quasipolynomial Bogolyubov-Ruzsa lemma, of which we prove such a robust version.

An improved self-corrector for the Reed-Muller code of order 2. A central ingredient in the quadratic Goldreich-Levin theorem of [27] is a self-correction procedure for the Reed-Muller code of order 2 at distance $1/2 - \varepsilon$. More precisely, the authors present a procedure which runs in time polynomial in n and exponential in $1/\varepsilon$, which given a function $f : \mathbb{F}_2^n \to \{-1, 1\}$ of distance at most $1/2 - \varepsilon$ from a quadratic phase $(-1)^q$ (which is a codeword of the Reed-Muller code of order 2), finds a quadratic phase $(-1)^{q'}$ which has distance at most $1/2 - \eta(\varepsilon)$ from f for $\eta(\varepsilon) = \exp(-1/\varepsilon)$.

This self-correction procedure is essentially an algorithmic version of the proof of the inverse theorem for the U^3 norm [21,16], which states that if a bounded function f has large U^3 norm, then it correlates with a quadratic phase. As stated above, the Bogolyubov-Ruzsa lemma is crucial in the proof of the inverse theorem, and hence plugging our new algorithmic proof of the quasipolynomial Bogolyubov-Ruzsa lemma into the self-correction procedure of [27] we improve the running time of the procedure, as well as the parameter η, to depend only quasipolynomially on $1/\varepsilon$.

We remark that the list decoding radius of the Reed-Muller code of order 2 is $1/4$ [10,9], and hence at distance $1/2 - \varepsilon$ one cannot expect to find all codewords of distance $1/2 - \varepsilon$ from a given codeword. Instead our self-correction procedure (as well as that of [27]) returns only a single codeword that correlates with the original codeword.

An improved quadratic Goldreich-Levin theorem. As mentioned above, the self-correction procedure for the Reed-Muller code of order 2 at distance $1/2 - \varepsilon$ plays a substantial role in the work on quadratic decomposition theorems by the last two authors. The aim of such theorems is to decompose any bounded function $f : \mathbb{F}_2^n \to \mathbb{C}$ as a sum $g + h$, where g is quadratically uniform, in the sense that the Gowers U^3 norm $\|g\|_{U^3}$ is small, and h is quadratically structured, in the sense that it is a bounded sum of quadratic phases. These types of decompositions constitute a higher-order analogue of classical Fourier decompositions, and they have found several number-theoretic applications [4,12,13,11,17]. Such decomposition theorems had previously been obtained in an abstract and non-constructive way (either using a form of the Hahn-Banach theorem [13], or a so-called energy increment approach [15]).

From a computer science perspective, it is a natural question to ask whether such a decomposition could be computed efficiently. In [27], the authors gave a probabilistic algorithm that, given any function $f : \mathbb{F}_2^n \to \mathbb{C}$, would with high probability compute, in time polynomial in n and exponential in $1/\varepsilon$, a quadratic decomposition for that function with a specified U^3 error ε, in which the number of quadratic terms is exponential in $1/\varepsilon$. This essentially amounts to computing a "quadratic Fourier decomposition" for f, and was therefore termed a *quadratic Goldreich-Levin theorem* in analogy with the well-known linear case [8].

The quadratic Goldreich-Levin algorithm consists of two parts: a deterministic part which is able to construct the quadratically structured part of f under the assumption that we have an algorithm which provides some quadratic phase function that f correlates with (if there is no such phase function, we just set $g = f$). The algorithm for finding a quadratic phase function, which constitutes the second part of the overall algorithm, is basically the self-correction procedure for the Reed-Muller code of order 2 at distance $1/2 - \varepsilon$ described above. Using our improved self-correction procedure we improve the running time, as well as the number of terms that are obtained in the final quadratic decomposition, to depend only quasipolynomially in the uniformity parameter ε.

2 Techniques

2.1 Sampling-Based Proofs of Almost-Periodicity Results

The following is the precise statement of the original almost-periodicity lemma of Croot and Sisask (Proposition 1.3 in [7]). Since it is valid for general abelian groups G, it is written in multiplicative notation. For a pair of functions $f, g : \mathbb{F}_2^n \to \mathbb{C}$, their convolution is defined by $f * g(x) = \mathbb{E}_{y \in \mathbb{F}_2^n} f(y)g(x-y)$.

Proposition 1 (Croot-Sisask Lemma, L^p Local Version). *Let $\varepsilon > 0$ and let $m \geq 1$ be an integer. Let G be an abelian group and let $A, B \subseteq G$ be finite subsets such that $|B \cdot A| \leq K|B|$. Then there is a set $X \subseteq A$ of size $|X| \geq |A|/(2K)^{50m/\varepsilon}$ such that for each $x \in XX^{-1}$,*

$$\|1_A * 1_B(yx) - 1_A * 1_B(y)\|_{2m}^{2m} \leq \max\{\varepsilon^m |AB||B|^m, \|1_A * 1_B\|_m^m\}\varepsilon^m |B|^m.$$

It is known that L^p bounds and Chernoff's inequality are, in a certain sense, equivalent. Specifically, a random variable X obeys a Chernoff-type tail bound of the form
$$\mathbb{P}[|X| \geq t\|X\|_2] \leq C\exp(-\Omega(t^2))$$
if and only if its L^p-norm satisfies

$$\|X\|_p \leq C\sqrt{p}\|X\|_2$$

for all $p \in [2, \infty)$, the latter representing a Khinchine-type inequality (from which Marcinkiewicz-Zygmund can be derived). For a proof of this statement we refer the reader to the excellent lecture notes by Sanders [25].

Thus it is natural to ask whether one could formulate an 'L^p-norm free' almost-periodicity statement that suffices for applications and whether such a statement could be proven without recourse to L^p-norms. In Section 4.1 we answer this question in the affirmative by proving Proposition 2 below. We stress again that this proposition holds over any abelian group, but for simplicity we state it only for the special case of $\mathbb{F}_2^n$.

To state Proposition 2 we start with fixing some notation. For a subset $A \subseteq \mathbb{F}_2^n$ we let $\mathbb{1}_A$ denote the indicator function of A and μ_A denote the function $\mathbb{1}_A \cdot (2^n/|A|)$. For two real numbers α, β we write $\alpha \approx_\varepsilon \beta$ to denote $|\alpha - \beta| \leq \varepsilon$ and if $|\alpha - \beta| > \varepsilon$ we write $\alpha \not\approx_\varepsilon \beta$. Given subsets $A, B \subset \mathbb{F}_2^n$, define the *measure of additive containment* $\rho_{A\to B} : \mathbb{F}_2^n \to [0,1]$ by

$$\rho_{A\to B}(y) := \mathop{\mathbb{P}}_{a\in A}[y + a \in B] = \frac{|(y + A) \cap B|}{|A|} = \mu_A * \mathbb{1}_B(y), \tag{1}$$

for each $y \in \mathbb{F}_2^n$. Notice that $\rho_{A\to B}(y) = 1$ when $y + A \subseteq B$ and $\rho_{A\to B}(y) = 0$ when $(y + A) \cap B = \emptyset$.

Proposition 2 (Almost-Periodicity of Sumsets). *If $A \subset \mathbb{F}_2^n$ satisfies $|2A| \leq K|A|$, then for every integer t and set $B \subseteq \mathbb{F}_2^n$ there exists a set X with the following properties.*

1. *The set X is contained in an affine shift of A.*
2. *The size of X is at least $|A|/(2K^{t-1})$.*
3. *For all $x \in X$ and for all subsets $S \subseteq \mathbb{F}_2^n$,*

$$\mathop{\mathbb{P}}_{y\in S}[\rho_{A\to B}(y) \approx_{2\varepsilon} \rho_{A\to B}(y + x)] \geq 1 - 8\frac{|A + B|}{|S|} \cdot \exp\left(-2\varepsilon^2 t\right). \tag{2}$$

Our proof differs from the original proof of Croot and Sisask in that it disposes of L^p-norms and tail bounds for a multinomial distribution (or the Marcinkiewicz-Zygmund inequality), and replaces them with sampling arguments relying on the Chernoff-Hoeffding bound.

We now sketch the proof. To obtain X we replace $\rho_{A\to B}$ by an estimator function computed by taking a sequence of t independent random samples distributed uniformly over A. Denoting the sample sequence by $\mathbf{a} = (a_1, \ldots, a_t)$, we estimate $\rho_{A\to B}(y)$ by the fraction of $a_i \in \mathbf{a}$ satisfying $y + a_i \in B$. Denote the estimator function corresponding to $\mathbf{a}$ by $\hat{\rho}_{\mathbf{a}}$. Fixing y, the Chernoff-Hoeffding bound says that the probability that $\hat{\rho}_{\mathbf{a}}(y)$ differs from $\rho_{A\to B}(y)$ by more than ε, i.e., the probability of the event "$\rho_{A\to B}(y) \not\approx_\varepsilon \hat{\rho}_{\mathbf{a}}(y)$" when $\mathbf{a} = (a_1, \ldots, a_t)$ is distributed uniformly over A^t, is at most $\exp\left(-\Omega\left(\varepsilon^2 t\right)\right)$.

The key observation in the construction of the set X is that there are many pairs of good estimator sequences $\mathbf{a} = (a_1, \ldots, a_t), \hat{\mathbf{a}} = (\hat{a}_1, \ldots, \hat{a}_t)$ for which there exists a "special" element $x \in \mathbb{F}_2^n$ such that $\hat{\mathbf{a}} = x + \mathbf{a}$, where $x + \mathbf{a} := (x + a_1, \ldots, x + a_t)$. Such x are called "special" for the following reason. We say y is "good" if both of the following conditions hold,

$$\rho_{A\to B}(y) \approx_\varepsilon \hat{\rho}_{\hat{\mathbf{a}}}(y) \quad \text{and} \quad \rho_{A\to B}(y + x) \approx_\varepsilon \hat{\rho}_{\mathbf{a}}(y + x). \tag{3}$$

Now if $\hat{\mathbf{a}} = x + \mathbf{a}$, then we have

$$\hat{\rho}_{\hat{\mathbf{a}}}(y) = \hat{\rho}_{\hat{\mathbf{a}}}(y + x + x) = \hat{\rho}_{\hat{\mathbf{a}}+x}(y + x) = \hat{\rho}_{\mathbf{a}}(y + x),$$

and combining this with (3) implies that for "good" y we have $\rho_{A\to B}(y) \approx_{2\varepsilon} \rho_{A\to B}(y+x)$. Thus, to prove the proposition we only need to bound from below the number of "special" elements x, which is done based on the assumption that A has small doubling.

For applications, one would like a version of almost-periodicity in which the set X is replaced with a subspace. It was observed by Sanders [22] that such a version could be deduced from Proposition 1 above using Fourier analysis arguments. In Section 4.2 we use Sanders's approach to deduce such a version also from our Proposition 2. We now turn to describing the techniques used in our algorithmic version of almost-periodicity results.

2.2 Algorithmic Versions of Almost-Periodicity Results

With a view to algorithmic applications, we prove an algorithmic version of our almost-periodicity results. As previously noted, the main difficulty in obtaining such an algorithmic version is that the combinatorial proofs of these results use the description of large subsets of $\mathbb{F}_2^n$, whose size is exponential in n. Since we are interested in an algorithm which runs in time polynomial in n we do not have time to describe and inspect these sets as a whole. Instead, we use random sampling methods to decide efficiently membership in such sets.

For instance, one of the first issues we need to deal with is that for our algorithmic applications we need to compute the measure $\rho_{A\to 2A}(y) = \mathbb{P}_{a\in A}[a + y \in 2A]$. In order to compute this measure algorithmically one has to have access to the indicator function for the sumset $2A$, whereas we only have oracle access to the indicator function of A. In order to deal with this, we observe that an element $y \in \mathbb{F}_2^n$ satisfies $a + y \in 2A$ if and only if $\mathbb{1}_A * \mathbb{1}_A(a + y) > 0$, and that the latter quantity can be estimated using sampling. In order to handle possible error in the estimation of $\mathbb{1}_A * \mathbb{1}_A(a + y) > 0$ we need to make some further modifications in the combinatorial proof.

Another core issue that we need to deal with is how to efficiently find the set X of almost periods. In the combinatorial proof the existence of the set X is shown in a non-constructive way using the pigeonhole principle. In order to find the set X in a constructive manner we prove that for a random string $\hat{\mathbf{a}} = (a_1, \ldots, a_t) \in (\mathbb{F}_2^n)^t$ many translates $\hat{\mathbf{a}}+x := (a_1+x, \ldots, a_t+x)$ of $\hat{\mathbf{a}}$, as well as $\hat{\mathbf{a}}$ itself, will be good estimator sequences. In particular, one can take the set X to be the set of all $x \in \mathbb{F}_2^n$ such that $\hat{\mathbf{a}} + x$ is a good estimator sequence. For this one needs an efficient procedure for testing whether a given sequence is a good estimator sequence, and we obtain such a procedure using the above-mentioned idea for estimating $\rho_{A\to 2A}(y)$.

Finally, in our combinatorial proof we show that X can be approximated by a subspace V using Fourier analysis. It follows that in order to find the subspace V, it suffices to inspect the large Fourier coefficients of $\mathbb{1}_X$. This can be done efficiently using the (standard) Goldreich-Levin theorem.

3 Preliminaries

In this section we fix our notation and collect some results that we shall use throughout the paper. Fundamental to our approach will be the following Chernoff-type tail bound for sampling [26].

Lemma 1 (Hoeffding Bound for Sampling). *If* $\mathbf{X}$ *is a random variable with* $|\mathbf{X}| \leq 1$ *and* $\hat{\mu}$ *is the empirical average obtained from* t *samples, then*

$$\mathbb{P}\left[|\mathbb{E}\left[\mathbf{X}\right] - \hat{\mu}| \ > \ \gamma\right] \ \leq \ 2\exp(-2\gamma^2 t).$$

Throughout the paper we shall make use of the discrete Fourier transform, which we define as follows. For $f : \mathbb{F}_2^n \to \mathbb{C}$, let

$$\widehat{f}(t) = \mathbb{E}_{x\in\mathbb{F}_2^n} f(x)(-1)^{x\cdot t}$$

for any $t \in \widehat{\mathbb{F}_2^n} = \mathbb{F}_2^n$, where $\mathbb{E}_{x\in\mathbb{F}_2^n}$ simply stands for the normalized sum $2^{-n}\sum_{x\in\mathbb{F}_2^n}$ and $x \cdot t = \sum_{i=1}^n x_i t_i$ for a pair of vectors $x = (x_1, \ldots, x_n), t = (t_1, \ldots, t_n) \in \mathbb{F}_2^n$. The inversion formula states that

$$f(x) = \sum_{t\in\mathbb{F}_2^n} \widehat{f}(t)(-1)^{x\cdot t}$$

for all $x \in \mathbb{F}_2^n$, and Parseval's identity takes the form

$$\mathbb{E}_{x\in\mathbb{F}_2^n} f(x)\overline{g(x)} = \sum_{t\in\mathbb{F}_2^n} \widehat{f}(t)\overline{\widehat{g}(t)},$$

for any two functions $f, g : \mathbb{F}_2^n \to \mathbb{C}$. Finally, the convolution of two such functions is defined by

$$f * g(x) = \mathbb{E}_{y\in\mathbb{F}_2^n} f(y)g(x-y),$$

and the fact that the Fourier transform diagonlizes the convolution operator is expressed via the idenity

$$\widehat{f * g}(t) = \widehat{f}(t)\widehat{g}(t),$$

which holds for all $t \in \mathbb{F}_2^n$.

The set of large Fourier coefficients determines the value of a function to a significant extent, and for many arguments it is important to be able to estimate its size and determine its structure. For a function $f : \mathbb{F}_2^n \to \mathbb{C}$, let

$$\mathrm{Spec}_\rho(f) = \{t \in \mathbb{F}_2^n : |\widehat{f}(t)| \geq \rho\|f\|_1\}. \tag{4}$$

For a subset $A \subseteq \mathbb{F}_2^n$ we let $\mathbb{1}_A$ denote the indicator function of A and μ_A denote the function $\mathbb{1}_A \cdot (2^n/|A|)$ so that $\mathbb{E}_{x\in\mathbb{F}_2^n}[\mu_A(x)] = 1$. In the special case where $f = \mathbb{1}_A$ for a subset $A \subseteq \mathbb{F}_2^n$ of density α, Parseval's identity tells us that $|\mathrm{Spec}_\rho(\mathbb{1}_A)| \leq \rho^{-2} \cdot \alpha^{-1}$. A more precise result is known: Chang's theorem [5] states that $\mathrm{Spec}_\rho(\mathbb{1}_A)$ is in fact contained in a subspace of dimension at most $C\rho^{-2}\log\alpha^{-1}$.

Theorem 1 (Chang's Theorem). *Let $\rho \in (0,1]$ and $A \subseteq \mathbb{F}_2^n$. Then there is a subspace V of $\mathbb{F}_2^n$ such that* $\mathrm{Spec}_\rho(\mathbb{1}_A) \subseteq V$ *and*

$$\dim(V) \leq 8\frac{\log(2^n/|A|)}{\rho^2}.$$

For an elegant recent proof of this result using entropy, see Impagliazzo et al. [18].

Finally, for two real numbers α, β we write $\alpha \approx_\varepsilon \beta$ to denote $|\alpha - \beta| \leq \varepsilon$ and if $|\alpha - \beta| > \varepsilon$ we write $\alpha \not\approx_\varepsilon \beta$. All logarithms in this paper are taken to base 2.

4 Sampling-Based Proofs of Almost-Periodicity Results

4.1 Croot-Sisask Almost-Periodicity

By an inductive application of Proposition 2 (using the triangle inequality and the union bound), one can prove the following iterated version.

Corollary 1 (Almost-Periodicity of Sumsets, Iterated). *If $A \subset \mathbb{F}_2^n$ satisfies $|2A| \leq K|A|$, then for every integer t and set $B \subseteq \mathbb{F}_2^n$, there exists a set X with the following properties.*

1. *The set X is contained in an affine shift of A.*
2. *The size of X is at least $|A|/(2K^{t-1})$.*
3. *For all $x_1, \ldots, x_\ell \in X$ and for all subsets $S \subseteq \mathbb{F}_2^n$,*

$$\mathop{\mathbb{P}}_{y \in S}\left[\rho_{A \to B}(y) \approx_{2\varepsilon\ell} \rho_{A \to B}(y + x_1 + \ldots + x_\ell)\right] \geq 1 - 8\ell\frac{|A+B|}{|S|} \cdot \exp\left(-2\varepsilon^2 t\right). \tag{5}$$

4.2 Almost-Periodicity Over a Subspace

For applications one would like a version of Proposition 2 in which the set X of periods is in fact a subspace. It was observed by Sanders [22] that one can use iterated almost-periodicity statements such as Corollary 1, combined with some Fourier analysis, to obtain such a subspace. Here we use Sanders's argument to deduce the following statement from Corollary 1.

Corollary 2 (Almost-Periodicity of Sumsets Over a Subspace). *If $A \subset \mathbb{F}_2^n$ is a subset of density α, then for every integer t and set $B \subseteq \mathbb{F}_2^n$ there exists a subspace V of codimension* $\mathrm{codim}(V) \leq 32\log(2/\alpha^t)$ *with the following property.*

For every $v \in V$, for all subsets $S \subseteq \mathbb{F}_2^n$ and for every $\varepsilon, \eta > 0$ and integer ℓ,

$$\mathop{\mathbb{P}}_{y \in S}\left[\rho_{A \to B}(y) \approx_{\varepsilon'} \rho_{A \to B}(y + v)\right] \geq 1 - 16\frac{\ell}{\eta}\frac{|A+B|}{|S|} \cdot \exp\left(-2\varepsilon^2 t\right), \tag{6}$$

where $\varepsilon' = 4\varepsilon\ell + 2\eta + 2^{-\ell}\sqrt{|B|/|A|}$.

The proof of the quasipolynomial Bogolyubov-Ruzsa lemma follows easily from the above corollary, and the result of Croot, Laba and Sisask on the existence of subspaces in sumsets of dense sets follows easily from a refinement of the above corollary. Note that for the proof of Corollary 2 we need the stronger assumption that A has density at least α in $\mathbb{F}_2^n$, instead of the doubling hypothesis $|2A| \leq K|A|$.

The idea of the proof of Corollary 2 is the following. Let X be the subset guaranteed by Corollary 1 for $K = 1/\alpha$, and define the subspace V as $V = \mathrm{Spec}_{1/2}(X)^{\perp}$ (see Section 3 for the definition of Spec_{ρ}). The intuition is that if X were a subspace then $\mathrm{Spec}_{1/2}(X) = V^{\perp}$, and hence $V = X$. Thus V serves as an "approximate subspace" for X. Since A is dense in $\mathbb{F}_2^n$, by Corollary 1 we also have that X is dense in $\mathbb{F}_2^n$ and hence Chang's theorem (Theorem 1) implies that the subspace V is also dense in $\mathbb{F}_2^n$ (this is the only place where we need the stronger assumption on the density of A).

In order to show that (6) holds we first show, using Corollary 1, a simple averaging argument and the triangle inequality, that for most $y \in S$,

$$\mathop{\mathbb{E}}_{x_1,\ldots,x_\ell \in X} [\rho_{A\to B}(y + x_1 + \ldots + x_\ell)] \approx_{2\varepsilon\ell+\eta} \rho_{A\to B}(y)\,. \tag{7}$$

Similarly, for all $v \in V$ and for most $y \in S$,

$$\mathop{\mathbb{E}}_{x_1,\ldots,x_\ell \in X} [\rho_{A\to B}(y + v + x_1 + \ldots + x_\ell)] \approx_{2\varepsilon\ell+\eta} \rho_{A\to B}(y + v)\,. \tag{8}$$

We then use Fourier analysis, following Sanders's argument closely, to show that for *all* $y \in \mathbb{F}_2^n$,

$$\begin{aligned}&\mathop{\mathbb{E}}_{x_1,\ldots,x_\ell \in X} [\rho_{A\to B}(y + x_1 + \ldots + x_\ell)]\\ &\quad\approx_{2^{-\ell}\sqrt{|B|/|A|}} \mathop{\mathbb{E}}_{x_1,\ldots,x_\ell \in X} [\rho_{A\to B}(y + v + x_1 + \ldots + x_\ell)]\,,\end{aligned} \tag{9}$$

where v is again an arbitrary element of V. The final conclusion follows from (7), (8) and (9) using the union bound and the triangle inequality. We first establish (7) and (8) using Markov's inequality.

Lemma 2. *Let $\varepsilon, \delta > 0$, and let $A, B, X, S \subseteq \mathbb{F}_2^n$ be such that for all $x_1, \ldots, x_\ell \in X$,*

$$\mathop{\mathbb{P}}_{y\in S} [\rho_{A\to B}(y) \approx_{\varepsilon} \rho_{A\to B}(y + x_1 + \ldots + x_\ell)] \geq 1 - \delta\,.$$

Then for every $\eta > 0$ we have that

$$\mathop{\mathbb{P}}_{y\in S} \left[\rho_{A\to B}(y) \approx_{\varepsilon+\eta} \mathop{\mathbb{E}}_{x_1,\ldots,x_\ell \in X} [\rho_{A\to B}(y + x_1 + \ldots + x_\ell)]\right] \geq 1 - \delta/\eta\,.$$

The next lemma establishes (9).

Lemma 3. *Let $X \subseteq \mathbb{F}_2^n$, and let $V \subseteq \mathrm{Spec}_{1/2}(X)^{\perp}$. Then for all $y \in \mathbb{F}_2^n$ and $v \in V$,*

$$\mathop{\mathbb{E}}_{x_1,\ldots,x_\ell \in X} [\rho_{A\to B}(y + x_1 + \ldots + x_\ell)] \approx_{\varepsilon''} \mathop{\mathbb{E}}_{x_1,\ldots,x_\ell \in X} [\rho_{A\to B}(y + v + x_1 + \ldots + x_\ell)]\,, \tag{10}$$

where $\varepsilon'' = 2^{-\ell}\sqrt{|B|/|A|}$.

Acknowledgements. The last two authors would like to thank Tom Sanders for numerous helpful remarks and discussions. The first two authors would like to thank Shachar Lovett for sharing with them his view of Sanders's proof of the quasipolynomial Bogolyubov-Ruzsa lemma (cf. [19]).

An application of Chang's theorem now proves Corollary 2.

References

1. Ben-Sasson, E., Lovett, S., Ron-Zewi, N.: An additive combinatorics approach relating rank to communication complexity. In: FOCS, pp. 177–186. IEEE Computer Society (2012)
2. Ben-Sasson, E., Zewi, N.: From affine to two-source extractors via approximate duality. In: Fortnow, L., Vadhan, S.P. (eds.) STOC, pp. 177–186. ACM (2011)
3. Bhowmick, A., Dvir, Z., Lovett, S.: Lower bounds on vector matching codes. In: STOC. ACM (2013)
4. Candela, P.: On the structure of steps of three-term arithmetic progressions in a dense set of integers. Bull. Lond. Math. Soc. 42(1), 1–14 (2010)
5. Chang, M.-C.: A polynomial bound in Freiman's theorem. Duke Math. J. 113(3), 399–419 (2002)
6. Croot, E., Łaba, I., Sisask, O.: Arithmetic progressions in sumsets and L^{p}-almost-periodicity (March 2011), http://arxiv.org/abs/1103.6000v1
7. Croot, E., Sisask, O.: A probabilistic technique for finding almost-periods of convolutions. Geom. Funct. Anal. 20(6), 1367–1396 (2010)
8. Goldreich, O., Levin, L.: A hard-core predicate for all one-way functions. In: STOC, pp. 25–32 (1989)
9. Gopalan, P.: A fourier-analytic approach to reed-muller decoding. In: FOCS, pp. 685–694. IEEE Computer Society (2010)
10. Gopalan, P., Klivans, A.R., Zuckerman, D.: List-decoding reed-muller codes over small fields. In: Dwork, C. (ed.) STOC, pp. 265–274. ACM (2008)
11. Gowers, T., Wolf, J.: Linear forms and quadratic uniformity for functions on $\mathbb{Z}_N$. J. Anal. Math., arXiv:1002.2210 (2010) (to appear)
12. Gowers, T., Wolf, J.: The true complexity of a system of linear equations. Proc. Lond. Math. Soc. (3),100(1), 155–176 (2010)
13. Gowers, T., Wolf, J.: Linear forms and quadratic uniformity for functions on $\mathbb{F}_p^n$. Mathematika 57(2), 215–237 (2012)
14. Green, B.: Finite field models in additive combinatorics. In: Surveys in Combinatorics 2005. London Math. Soc. Lecture Note Ser, vol. 327, pp. 1–27. Cambridge Univ. Press, Cambridge (2005)
15. Green, B.: Montréal notes on quadratic Fourier analysis. In: Additive Combinatorics. CRM Proc. Lecture Notes, vol. 43, pp. 69–102. Amer. Math. Soc., Providence (2007)
16. Green, B., Tao, T.: An inverse theorem for the Gowers $U^3(G)$ norm. Proc. Edinb. Math. Soc (2), 51(1), 73–153 (2008)
17. Hatami, H., Lovett, S.: Higher-order Fourier analysis of $\mathbb{F}_p^n$ and the complexity of systems of linear forms. Geom. Func. Anal (2011) (to appear)
18. Impagliazzo, R., Moore, C., Russell, A.: An Entropic Proof of Chang's Inequality (May 2012), http://arxiv.org/abs/1205.0263v1
19. Lovett, S.: An exposition of Sanders's quasi-polynomial Freiman-Ruzsa theorem. Electronic Colloquium on Computational Complexity (ECCC) 19, 29 (2012)

20. Ruzsa, I.: An analog of Freiman's theorem in groups. Astérisque xv(258), 323–326 (1999); Structure theory of set addition
21. Samorodnitsky, A.: Low-degree tests at large distances. In: STOC, pp. 506–515 (2007)
22. Sanders, T.: On the Bogolyubov-Ruzsa lemma. Anal. PDE (2010) (to appear)
23. Sanders, T.: Green's sumset problem at density one half. Acta Arith. 146(1), 91–101 (2011)
24. Sanders, T.: On Roth's theorem on progressions. Ann. of Math (2), 174(1), 619–636 (2011)
25. Sanders, T.: Lecture notes on applications of commutative harmonic analysis (July 2012), `http://people.maths.ox.ac.uk/~sanders`
26. Tao, T., Vu, V.: Additive combinatorics. Cambridge University Press (2006)
27. Tulsiani, M., Wolf, J.: Quadratic Goldreich-Levin theorems. In: FOCS, pp. 619–628 (2011)

The Mondshein Sequence*

Jens M. Schmidt

TU Ilmenau

Abstract. Canonical orderings [STOC'88, FOCS'92] have been used as a key tool in graph drawing, graph encoding and visibility representations for the last decades. We study a far-reaching generalization of canonical orderings to non-planar graphs that was published by Lee Mondshein in a PhD-thesis at M.I.T. as early as 1971.

Mondshein proposed to order the vertices of a graph in a sequence such that, for any i, the vertices from 1 to i induce essentially a 2-connected graph while the remaining vertices from $i+1$ to n induce a connected graph. Mondshein's sequence generalizes canonical orderings and became later and independently known under the name *non-separating ear decomposition.* Currently, the best known algorithm for computing this sequence achieves a running time of $O(nm)$; the main open problem in Mondshein's and follow-up work is to improve this running time to a subquadratic time.

In this paper, we present the first algorithm that computes a Mondshein sequence in time and space $O(m)$, improving the previous best running time by a factor of n. In addition, we illustrate the impact of this result by deducing linear-time algorithms for several other problems, for which the previous best running times have been quadratic.

In particular, we show how to compute three independent spanning trees in a 3-connected graph in linear time, improving a result of Cheriyan and Maheshwari [J. Algorithms 9(4)]. Secondly, we improve the preprocessing time for the output-sensitive data structure by Di Battista, Tamassia and Vismara [Algorithmica 23(4)] that reports three internally disjoint paths between any given vertex pair from $O(n^2)$ to $O(m)$. Thirdly, we improve the computation of 3-partitioning of a 3-connected graph to linear time. Finally, we show how a very simple linear-time planarity test can be derived once a Mondshein sequence is computed.

1 Introduction

Canonical orderings are a fundamental tool used in graph drawing, graph encoding and visibility representations; we refer to [1] for a wealth of applications. For maximal planar graphs, canonical orderings were first introduced by de Fraysseix, Pach and Pollack [6,7] in 1988. Kant then generalized canonical orderings to arbitrary 3-connected planar graphs [12,13].

Surprisingly, the concept of canonical orderings can be traced back much further, namely to a long-forgotten PhD-thesis at M.I.T. by Lee F. Mondshein [15]

* This research was partly done at Max Planck Institute for Informatics, Saarbrücken.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 967–978, 2014.

in 1971. In fact, Mondshein proposed a sequence that generalizes canonical orderings to non-planar graphs, hence making them applicable to arbitrary 3-connected graphs. Mondshein's sequence was, independently and in a different notation, found later by Cheriyan and Maheshwari [4] under the name *non-separating ear decompositions.*

Computationally, it is an intriguing question how fast a Mondshein sequence can be computed. Mondshein himself gave an involved algorithm with running time $O(m^2)$. Cheriyan showed that it is possible to achieve a running time of $O(nm)$ by using a theorem of Tutte that proves the existence of non-separating cycles in 3-connected graphs [20]. Both works (see [15, p 1.2] and [4, p. 532]) state as main open problem, whether it is possible to compute a Mondshein sequence in subquadratic time.

We present the first algorithm that computes a Mondshein sequence in time and space $O(m)$, hence solving the above 40-year-old problem. The interest in such a computational result stems from the fact that 3-connected graphs play a crucial role in algorithmic graph theory; we illustrate this in four direct applications by giving linear-time (and hence optimal) algorithms for several problems, for two of which the previous best running times have been quadratic.

In particular, we show how to compute three independent spanning trees in a 3-connected graph in linear time, improving a result of Cheriyan and Maheshwari [4]. Second, we improve the preprocessing time from $O(n^2)$ to $O(m)$ for a data structure by Di Battista, Tamassia and Vismara [8] that reports three internally disjoint paths in a 3-connected graph between any given vertex pair in time $O(\ell)$, where ℓ is the total length of these paths. Finally, we illustrate the usefulness of Mondshein's sequence by giving a very simple linear-time planarity test, once a Mondshein sequence is computed.

We start by giving an overview of Mondshein's work and its connection to canonical orderings and non-separating ear decompositions in Section 3. Section 4 sketches the main ideas for our linear-time algorithm that computes a Mondshein sequence. Section 5 covers four applications of this linear-time algorithm.

2 Preliminaries

We use standard graph-theoretic terminology and assume that all graphs are simple.

Definition 1 ([14,23])**.** An *ear decomposition* of a 2-connected graph $G = (V, E)$ is a sequence $(P_0, P_1, \dots, P_k)$ of subgraphs of G that partition E such that P_0 is a cycle and every P_i, $1 \leq i \leq k$, is a path that intersects $P_0 \cup \dots \cup P_{i-1}$ in exactly its end points. Each P_i is called an *ear*. An ear is *short* if it is an edge and *long* otherwise.

According to Whitney [23], every ear decomposition has exactly $m - n + 1$ ears. For any i, let $G_i = P_0 \cup \dots \cup P_i$ and $\overline{V_i} := V - V(G_i)$. We write $\overline{G_i}$ to denote the graph induced by $\overline{V_i}$. We observe that $\overline{G_i}$ does not necessarily contain all edges in $E - E(G_i)$; in particular, there may be short ears in $E - E(G_i)$ that have both of their endpoints in G_i.

For a path P and two vertices x and y in P, let $P[x, y]$ be the subpath in P from x to y. A path with endpoints v and w is called a *vw-path.* A vertex x in a vw-path P is an *inner vertex* of P if $x \notin \{v, w\}$. For convenience, every vertex in a cycle is an inner vertex of that cycle.

The set of inner vertices of an ear P is denoted as $inner(P)$. The inner vertex sets of the ears in an ear decomposition of G play a special role, as they partition $V(G)$. Every vertex of G is contained in exactly one long ear as inner vertex. This gives readily the following characterization of $\overline{V_i}$.

Observation 2. *For every i, $\overline{V_i}$ is the union of the inner vertices of all long ears P_j with $j > i$.*

We will compare vertices and edges of G by their first occurrence in a fixed ear decomposition.

Definition 3. Let $D = (P_0, P_1, \ldots, P_{m-n})$ be an ear decomposition of G. For an edge $e \in G$, let $birth_D(e)$ be the index i such that P_i contains e. For a vertex $v \in G$, let $birth_D(v)$ be the minimal i such that P_i contains v (thus, $P_{birth_D(v)}$ is the ear containing v as an inner vertex). Whenever D is clear from the context, we will omit D.

Clearly, for every vertex v, the ear $P_{birth(v)}$ is long, as it contains v as an inner vertex.

3 Generalizing Canonical Orderings

We give a compact rephrasing of canonical orderings in terms of non-separating ear decompositions. This will allow for an easier comparison of a canonical ordering and its generalization to non-planar graphs, as the latter is also based on ear decompositions. We assume that the input graphs are 3-connected and, when talking about canonical orderings, planar. It is well-known that maximal planar graphs, which were considered in [6], form a subclass of 3-connected graphs (apart from the triangle-graph).

Definition 4. An ear decomposition is *non-separating* if, for $0 \leq i \leq m - n$, every inner vertex of P_i has a neighbor in $\overline{G_i}$ unless $\overline{G_i} = \emptyset$.

The name *non-separating* refers to the following helpful property.

Lemma 5. *In a non-separating ear decomposition D, $\overline{G_i}$ is connected for every i.*

Proof. Let u be an inner vertex of the last long ear in D. If $\overline{G_i} = \emptyset$, the claim is true. Otherwise, consider any vertex x in $\overline{G_i}$. In order to show connectedness, we exhibit a path from x to u in $\overline{G_i}$. If x is an inner vertex of $P_{birth(u)}$, this path is just the path $P_{birth(u)}[x, u]$. Otherwise, $birth(x) < birth(u)$. Then x has a neighbor in $\overline{G_{birth(x)}}$, since D is non-separating, and, according to Observation 2, this neighbor is an inner vertex of some ear P_j with $j > birth(x)$. Applying induction on j gives the desired path to u. □

A *plane graph* is a graph that is embedded into the plane. In particular, a plane graph has a fixed outer face. We define canonical orderings as special non-separating ear decompositions.

Definition 6 (canonical ordering)**.** Let G be a 3-connected plane graph having the edges tr and ru in its outer face. A *canonical ordering* with respect to tr and ru is an ear decomposition D of G such that

1. $tr \in P_0$,
2. $P_{birth(u)}$ is the last long ear, contains u as its only inner vertex and does not contain ru, and
3. D is non-separating.

The original definition of canonical orderings by Kant [13] states several additional properties, all of which can be deduced from the ones given in Definition 6. E.g., it is easy to see for every i that the outer face C_i of G_i forms a cycle containing tr.

The fact that D is non-separating plays a key role for both canonical orderings and their generalization to non-planar graphs. E.g., for canonical orderings, Lemma 5 implies that the plane graph G can be constructed from P_0 by successively inserting the ears of D to only one dedicated face of the current embedding, a routine that is heavily applied in graph drawing and embedding problems.

Our definition of canonical orderings uses planarity only in one place: $tr \cup ru$ is assumed to be part of the outer face of G. Note that the essential art of this assumption is that $tr \cup ru$ is part of some face of G, as we can always choose an embedding for G having this face as outer face. By dropping this assumption, our definition of canonical orderings can be readily generalized to non-planar graphs: We merely require tr and ru to be edges in the graph.

This is in fact equivalent to the definition Mondshein used 1971 to define a *(2,1)-sequence* [15, Def. 2.2.1], but which he gave in the notation of a special vertex ordering. This vertex ordering actually refines the partial order $inner(P_0), \ldots, inner(P_{m-n})$ by enforcing an order on the inner vertices of each path according to their occurrence on that path (in any direction). For conciseness, we will instead stick to the following short ear-based definition, which is similar to the one given in [4] but does not need additional degree-constraints.

Definition 7 ([15,4])**.** Let G be a graph with an edge ru. A *Mondshein sequence avoiding* ru (see Figure 3a) is an ear decomposition D of G such that

1. $r \in P_0$,
2. $P_{birth(u)}$ is the last long ear, contains u as its only inner vertex and does not contain ru, and
3. D is non-separating.

An ear decomposition D that satisfies Conditions 1 and 2 is said to *avoid* ru. Put simply, this forces ru to be "added last" in D, i.e., strictly after the last long ear $P_{birth(u)}$ has been added. Note that Definition 7 implies $u \notin P_0$, as $P_{birth(u)}$ contains only one inner vertex. As a direct consequence of this and the fact that D is non-separating, G must have minimum degree 3 in order to have a Mondshein sequence. Mondshein proved that every 3-connected graph has a Mondshein sequence. In fact, also the converse is true.

Theorem 8. *[4,24] Let $ru \in E(G)$. Then G is 3-connected if and only if G has a Mondshein sequence avoiding ru.*

We state two additional facts about Mondshein sequences. Since we replaced the assumption that $tr \cup ru$ is in the outer face of G with the very small assumption that ru is an edge of G (which does not assume anything about t at all), it is natural to ask how we can extract t (and thus, a canonical ordering) from a Mondshein sequence when G is plane. We choose t as any neighbor of r in P_0. Since P_0 is non-separating and the non-separating cycles of a 3-connected plane graph are precisely its faces [20], this satisfies Definition 6 and leads to the following observation.

Observation 9. *Let D be a Mondshein sequence avoiding ru of a planar graph G and let t be a neighbor of r in P_0. Then D is a canonical ordering of the planar embedding of G whose outer face contains $tr \cup ru$.*

Once having a Mondshein sequence, we can aim for a slightly stronger structure. A *chord* of an ear P_i is an edge in G that joins two non-adjacent vertices of P_i. Let a Mondshein sequence be *induced* if P_0 is induced in G and every ear $P_i \neq P_0$ has no chord in G, except possibly the chord joining the endpoints of P_i. The following lemma shows that we can always expect Mondshein sequences that are induced. We omit the proof.

Lemma 10. *Every Mondshein sequence can be transformed to an induced Mondshein sequence in linear time.*

4 Computing a Mondshein Sequence

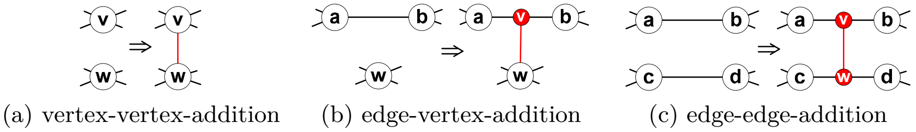
Mondshein gave an involved algorithm [15] that computes his sequence in time $O(m^2)$. Independently, Cheriyan and Maheshwari gave an algorithm that runs in time $O(nm)$ and which is based on a theorem of Tutte. At the heart of our linear-time algorithm is the following classical construction sequence for 3-connected graphs due to Barnette and Grünbaum [2] and Tutte [21, Thms. 12.64 and 12.65].

Definition 11. The following operations on simple graphs are *BG-operations* (see Figure 1).

(a) *vertex-vertex-addition*: Add an edge between two distinct non-adjacent vertices

(b) *edge-vertex-addition*: Subdivide an edge ab, $a \neq b$, by a vertex v and add the edge vw for a vertex $w \notin \{a, b\}$

(c) *edge-edge-addition*: Subdivide two distinct edges by vertices v and w, respectively, and add the edge vw



(a) vertex-vertex-addition (b) edge-vertex-addition (c) edge-edge-addition

Fig. 1. BG-operations

Theorem 12 ([2,21])**.** *A graph is 3-connected if and only if it can be constructed from K_4 using BG-operations.*

Hence, applying an BG-operation on a 3-connected graphs preserves it to be simple and 3-connected. Let a *BG-sequence* of a 3-connected graph G be a sequence of BG-operations that constructs G from K_4. It has been shown that such a BG-sequence can be computed efficiently.

Theorem 13 ([17, Thms. 6.(2) and 52])**.** *A BG-sequence of a* 3*-connected graph can be computed in time* $O(m)$*.*

The outline of our algorithm is as follows. We start with a Mondshein sequence of K_4, which is easily obtained, and compute a BG-sequence of our 3-connected input graph by using Theorem 13. The crucial part is now a careful analysis that a Mondshein sequence of a 3-connected graph G can be modified to one of G', where G' is obtained from G by applying a BG-operation.

This last step is the main technical contribution of this paper and depends on the various positions in the sequence in which the vertices and edges that are involved in the BG-operation can occur. We will prove that there is always a modification that is local in the sense that the only long ears that are modified are the ones containing a vertex that is involved in the BG-operation.

Lemma 14 (Path Replacement Lemma)**.** *Let G be a* 3*-connected graph with an edge ru. Let $D = (P_0, P_1, \dots, P_{m-n})$ be a Mondshein sequence avoiding ru of G. Let G' be obtained from G by applying a single BG-operation Γ and let ru' be the edge of G' corresponding to ru. Then a Mondshein sequence D' of G' avoiding ru' can be computed from D using only constantly many constant-time modifications.*

However, the complete description of these modifications goes beyond the scope of this extended abstract. We will therefore state precise modifications only for the very first cases of vertex-vertex- and edge-vertex-additions and omit everything else.

We need some notation for describing the modifications. Let vw be the edge that was added by Γ such that, if applicable, v subdivides $ab \in E(G)$ and w subdivides $cd \in E(G)$. Then the edge ru' of G' that *corresponds* to ru in G is either ru, rv or rw. Whenever we consider the edge ab or cd, e.g. in a statement about $birth(ab)$, we assume that Γ subdivides ab, respectively, cd. W.l.o.g., we further assume that $birth(a) \leq birth(b)$, $birth(c) \leq birth(d)$ and

$birth(d) \leq birth(b)$. If not stated otherwise, the $birth$-operator refers always to D in this section. Let $S \subseteq \{av, vb, vw, cw, wd\}$ be the set of new edges in G'.

We state the detailed replacement scheme that plays a key-role in proving the above Path Replacement Lemma.

Lemma 15. *There is a Mondshein sequence $D' = (P'_0, P'_1, \ldots, P'_{m-n+1})$ of G' avoiding ru (respectively, rv or rw if Γ subdivides ru) that can be obtained from D by performing the following four modifications:*

M1) replacing the long ear $P_{birth(b)}$ with $1 \leq i \leq 3$ consecutive long ears P'_{b_1}, P'_{b_2} and P'_{b_3}, each of which consists of edges in $P_{birth(b)} \cup S$ (for notational convenience, we assume that all three ears exist such that $P'_{b_j} := P'_{b_i}$ for every $j > i$)

M2) if $P_{birth(cd)}$ is long and $birth(d) < birth(b)$, replacing $P_{birth(cd)}$ with the long ear P'_{cwd} that is obtained from $P_{birth(cd)}$ by subdividing cd with w (in particular, $birth(cd) = birth(d) < birth(b)$ in this case)

M3) if $P_{birth(ab)}$ is short, deleting or replacing $P_{birth(ab)}$ with an edge in $\{va, vb, vw\}$; if $P_{birth(cd)}$ is short, deleting or replacing $P_{birth(cd)}$ with an edge in $\{wc, wd\}$

M4) possibly adding vw as new last ear.

In particular, D' can be constructed from D as follows (Figure 2 determines the new ears P'_{b_1}–P'_{b_3} in $M1$).

(1) Γ is a vertex-vertex-addition

Obtain D' from D by adding the new ear vw at the end.

(2) Γ is an edge-vertex-addition

(a) $birth(b) = birth(ab)$

Let a' and b' be the endpoints of $P_{birth(b)}$ such that a' is closer to a than to b on $P_{birth(b)}$ (a' may be a, but $b' \neq b$).

(i) $w \notin G_{birth(b)}$ ▷ *$birth(w) > birth(b)$*

Obtain D' from D by subdividing $ab \subseteq P_{birth(b)}$ with v and adding the new ear vw at the end.

(ii) $w \in G_{birth(b)} - P_{birth(b)}$ ▷ *$birth(w) < birth(b)$ and $w \notin \{a', b'\}$*

Let Z be the path obtained from $P_{birth(b)}$ by replacing ab with $av \cup vb$. Let Z_1 be the $a'w$-path in $Z \cup vw$. Obtain D' from D by replacing $P_{birth(b)}$ with the two ears Z_1 and $Z[v, b']$ in that order.

(iii) $w \in P_{birth(b)}$ ▷ *$birth(w) = birth(b)$ or $w \in \{a', b'\}$*

Let Z be obtained from $P_{birth(b)}$ by replacing ab with $av \cup vb$. Let Z_2 be the vw-path in Z (if $birth(b) = 0$, Z is a cycle and there are two vw-paths; we then choose one that does not contain r as an inner vertex). Let Z_1 be obtained from Z by replacing Z_2 with the edge vw. Obtain D' from D by replacing $P_{birth(b)}$ with the two ears Z_1 and Z_2 in that order.

We omit a concise proof of the correctness of Lemma 15 and, thus, of the Path Replacement Lemma 14. Applying Lemma 14 iteratively for each operation in a BG-sequence gives immediately a linear-time algorithm for constructing

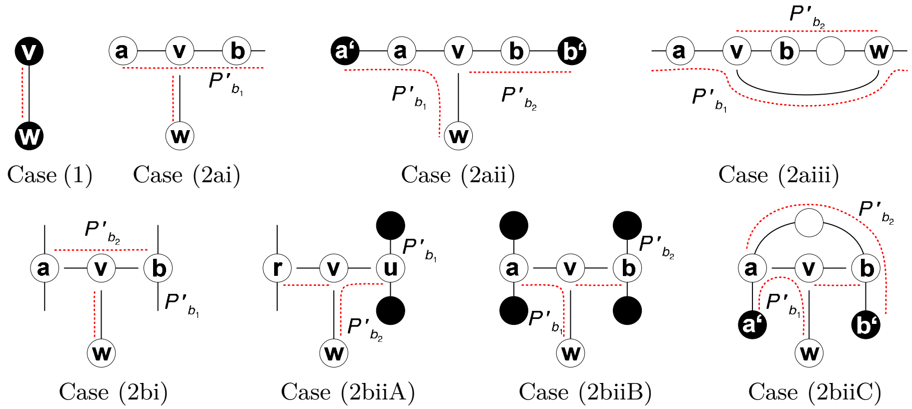


Fig. 2. Cases (1) and first subcases of (2) of Lemma 15. Black vertices are endpoints of ears that are contained in $G_{birth(b)}$. The dashed paths depict (parts of) the ears in D'.

a Mondshein sequence, as each step can be computed in constant time. We conclude the following theorem.

Theorem 16. *Given an edge ru of a 3-connected graph G, a Mondshein sequence of G avoiding ru can be computed in time $O(m)$.*

We now discuss four applications where Theorem 16 leads immediately to linear-time solutions. For three of these problems only quadratic algorithms have been known.

5 Applications

Application 1: Independent Spanning Trees
Let k spanning trees of a graph be *independent* if they all have the same root vertex r and, for every vertex $x \neq r$, the paths from r to x in the k spanning trees are *internally disjoint* (i.e., vertex-disjoint except for their endpoints). The following conjecture from 1988 due to Itai and Rodeh [11] has received considerable attention in graph theory throughout the past decades.

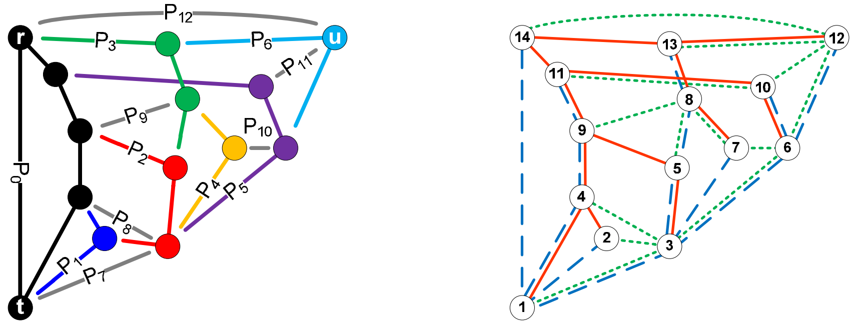
Conjecture (Independent Spanning Tree Conjecture [11])**.** Every k-connected graph contains k independent spanning trees.

The conjecture has been proven for $k \leq 2$ [11], $k = 3$ [4,24] and $k = 4$ [5], with running times $O(m)$, $O(n^2)$ and $O(n^3)$, respectively, for computing the corresponding independent spanning trees. For $k \geq 5$, the conjecture is open. For planar graphs, the conjecture has been proven by Huck [10].

We show how to compute three independent spanning trees in linear time, using an idea of [4]. This improves the previous best running time by a factor of n. It may seem tempting to compute the spanning trees directly and without using a Mondshein sequence, e.g. by local replacements in an induction over BG-operations or inverse contractions. However, without additional structure it can be proven that this is bound to fail.

Compute a Mondshein sequence avoiding ru, as described in Theorem 16. Choose r as the common root vertex of the three spanning trees and let $x \neq r$ be an arbitrary vertex.

First, we show how to obtain two internally disjoint paths from x to r that are both contained in the subgraph $G_{birth(x)}$. An *st-numbering* π is an ordering $v_1 < \cdots < v_n$ of the vertices of a graph such that $s = v_1$, $t = v_n$, and every other vertex has both a higher-numbered and a lower-numbered neighbor. Let π be *consistent* to a Mondshein sequence if π is an st-numbering for every graph G_i, $0 \leq i \leq m - n$. Let $t \neq u$ be a neighbor of r in P_0. A consistent tr-numbering π can be easily computed in linear time [3]. According to π, we can start with x and iteratively traverse to a higher-numbered and lower-numbered neighbor, respectively, without leaving $G_{birth(x)}$. This gives two internally disjoint paths from x to r and t; the path to t is then extended to the desired path ending at r by appending the edge tr. The traversed edges of this procedure for every $x \neq r$ give the first two independent spanning trees T_1 and T_2.



(a) A Mondshein sequence of a non-planar 3-connected graph G.

(b) Three independent spanning trees in G (vertex numbers depict a consistent st-numbering).

Fig. 3.

We construct the third independent spanning tree. Since a Mondshein sequence is non-separating, we can start with any vertex $x \neq r$, traverse to a neighbor in $\overline{G_{birth(x)}}$ and iterate this procedure until we end at u. The traversed edges of this procedure for every $x \neq r$ form a tree that is rooted at u and that can be extended to a spanning tree T_3 that is rooted at r by adding the edge ur. T_3 is independent from T_1 and T_2, as, for every $x \neq r$, the path from x to u intersects $G_{birth(x)}$ only in x.

Application 2: Output-Sensitive Reporting of Disjoint Paths
Given two vertices x and y of an arbitrary graph, a *k-path query* reports k internally disjoint paths between x and y or outputs that these do not exist. Di Battista, Tamassia and Vismara [8] give data structures that answer k-path queries for $k \leq 3$. A key feature of these data structures is that every k-path query has an *output-sensitive* running time, i.e., a running time of $O(\ell)$ if the total length of the reported paths is ℓ (and running time $O(1)$ if the paths do not exist). The preprocessing time of these data structures is $O(m)$ for $k \leq 2$ and $O(n^2)$ for $k = 3$.

For $k = 3$, Di Battista et al. show how the input graph can be restricted to be 3-connected using a standard decomposition. For every 3-connected graph we can compute a Mondshein sequence, which allows us to compute three independent spanning trees T_1–T_3 in a linear preprocessing time, as shown in Application 1. If x or y is the root r of T_1–T_3, this gives a straight-forward output-sensitive data structure that answers 3-path queries: we just store T_1–T_3 and extract one path from each tree per query.

In order to extend these queries to k-path queries between arbitrary vertices x and y, [8] gives a case distinction that shows that the desired paths can be efficiently found in the union of the six paths in T_1–T_3 that join x with r and y with r. This case distinction can be used for the desired output-sensitive reporting in time $O(\ell)$ without changing the preprocessing. We conclude a linear preprocessing time for all k-path queries with $k \leq 3$.

Application 3: Planarity Testing
We give a conceptually very simple planarity test based on Mondshein's sequence for any 3-connected graph G in time $O(n)$.

The 3-connectivity requirement is not really crucial, as the planarity of G can be reduced to the planarity of all 3-connected components of G, which in turn are computed as a side-product for the BG-sequence in Theorem 13; alternatively, one can use standard algorithms [9,16] for reducing G to be 3-connected. We compute an induced Mondshein sequence D avoiding an arbitrary edge ru in time $O(n)$. Let t be a neighbor of r in P_0.

We start with a planar embedding M_0 of P_0 and assume with Observation 9 w.l.o.g. that the last vertex u will be embedded in the outer face. We will first ignore short ears. Step by step, we attempt to augment M_i with the next long ear P_j in D in order to construct a planar embedding M_j of G_j.

Once the current embedding M_i contains u, we have added all the vertices of G and are done. Otherwise, u is contained in $\overline{G_i}$, according to Definition 6.2. Then $\overline{G_i}$ contains a path from each inner vertex of P_j to u, according to Lemma 5. Since u is contained in the outer face of the final embedding, adding the long ear P_j to M_i can preserve planarity only when it is embedded into the outer face f of M_i. Thus, we only have to check that both endpoints of P_j are contained in f (this is easy to test by maintaining the vertices of the current outer face). If yes, we embed P_j into f. Otherwise, we output "not planar"; if desired, a Kuratowski-subdivision can then be extracted in linear time.

Until now we ignored short ears, but have already constructed a planar embedding M' of a spanning subgraph of G. In order to test whether the addition of the short ears to M' can make the embedding non-planar, we pass through the construction of M' once more, this time adding short ears. Whenever a long ear P_j is embedded, we test whether all short ears that join a vertex of $inner(P_j)$ with a vertex of G_{j-1} can be embedded while preserving a planar embedding. Note that if D is a canonical ordering of M, G_{j-1} must be 2-connected and the outer face of G_{j-1} must be a cycle, according to [19, Corollary 1.3]. The last fact allows for an easy test whether adding the short ears preserves a planar embedding.

Application 4 (Bonus Application): The *k-partitioning* problem
At the time of submission, the author was pointed to the following problem. Given vertices $v_1, \ldots, v_k$ of a graph G and natural numbers $n_1, \ldots, n_k$ with $n_1 + \cdots + n_k = n$, find a partition of V into sets $V_1, \ldots, V_k$ with $|V_i| = n_i$ for every i such that every set V_i induces a connected graph in G.

For general graphs, this problem is NP-hard even for $k = 2$. However, for 3-connected graphs, the 3-partitioning problem can be solved in linear time if the input graph is *planar*. As suggested in [22], this problem (as well as a related extension) can be solved directly, once a non-separating ear decomposition has been computed. For planar graphs, we can thus use the (well-established) canonical ordering instead, which simplifies previous algorithms considerably.

More importantly, the fastest algorithm for the 3-partitioning problem in arbitrary 3-connected graphs runs in time $O(n^2)$ [18]. Combining a Mondshein sequence through with a simple assignment of the vertices on ears to V_1, V_2 and V_3 (as shown in [22]) gives the first $O(m)$ algorithm for this problem.

Acknowledgments. I wish to thank Joseph Cheriyan for valuable hints and helpfully providing a scan of his PhD-thesis, the anonymous person who drew my attention to Lee F. Mondshein's work, and the anonymous reviewer that pointed me to the k-partitioning problem.

References

1. Badent, M., Brandes, U., Cornelsen, S.: More canonical ordering. J. Graph Algorithms Appl. 15(1), 97–126 (2011)
2. Barnette, D.W., Grünbaum, B.: On Steinitz's theorem concerning convex 3-polytopes and on some properties of planar graphs. In: Many Facets of Graph Theory, pp. 27–40 (1969)
3. Brandes, U.: Eager *st*-ordering. In: Möhring, R.H., Raman, R. (eds.) ESA 2002. LNCS, vol. 2461, pp. 247–256. Springer, Heidelberg (2002)
4. Cheriyan, J., Maheshwari, S.N.: Finding nonseparating induced cycles and independent spanning trees in 3-connected graphs. J. Algorithms 9(4), 507–537 (1988)
5. Curran, S., Lee, O., Yu, X.: Finding four independent trees. SIAM Journal on Computing 35(5), 1023–1058 (2006)

6. de Fraysseix, H., Pach, J., Pollack, R.: Small sets supporting fary embeddings of planar graphs. In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC 1988), pp. 426–433 (1988)
7. de Fraysseix, H., Pach, J., Pollack, R.: How to draw a planar graph on a grid. Combinatorica 10(1), 41–51 (1990)
8. Di Battista, G., Tamassia, R., Vismara, L.: Output-sensitive reporting of disjoint paths. Algorithmica 23(4), 302–340 (1999)
9. Hopcroft, J.E., Tarjan, R.E.: Dividing a graph into triconnected components. SIAM J. Comput. 2(3), 135–158 (1973)
10. Huck, A.: Independent trees in planar graphs. Graphs and Combinatorics 15(1), 29–77 (1999)
11. Itai, A., Rodeh, M.: The multi-tree approach to reliability in distributed networks. Information and Computation 79, 43–59 (1988)
12. Kant, G.: Drawing planar graphs using the lmc-ordering. In: Proceedings of the 33th Annual Symposium on Foundations of Computer Science (FOCS 1992), pp. 101–110 (1992)
13. Kant, G.: Drawing planar graphs using the canonical ordering. Algorithmica 16(1), 4–32 (1996)
14. Lovász, L.: Computing ears and branchings in parallel. In: Proceedings of the 26th Annual Symposium on Foundations of Computer Science (FOCS 1985), pp. 464–467 (1985)
15. Mondshein, L.F.: Combinatorial Ordering and the Geometric Embedding of Graphs. PhD thesis, M.I.T. Lincoln Laboratory / Harvard University (1971), Technical Report, available at `http://www.dtic.mil/cgi-bin/GetTRDoc?AD=AD0732882`
16. Mutzel, P.: The SPQR-tree data structure in graph drawing. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 34–46. Springer, Heidelberg (2003)
17. Schmidt, J.M.: Contractions, removals and certifying 3-connectivity in linear time. SIAM Journal on Computing 42(2), 494–535 (2013)
18. Suzuki, H., Takahashi, N., Nishizek, T., Miyano, H., Ueno, S.: An algorithm for tripartitioning 3-connected graphs. Information Processing Society of Japan (IPSJ) 31(5), 584–592 (1990) (in Japanese).
19. Thomassen, C.: Kuratowski's theorem. J. Graph Theory 5(3), 225–241 (1981)
20. Tutte, W.T.: How to draw a graph. Proc. Lond. Math. Soc. 13, 743–767 (1963)
21. Tutte, W.T.: Connectivity in graphs. In: Mathematical Expositions, vo. 15, University of Toronto Press (1966)
22. Wada, K., Kawaguchi, K.: Efficient algorithms for tripartitioning triconnected graphs and 3-edge-connected graphs. In: van Leeuwen, J. (ed.) WG 1993. LNCS, vol. 790, pp. 132–143. Springer, Heidelberg (1994)
23. Whitney, H.: Non-separable and planar graphs. Trans. Amer. Math. Soc. 34(1), 339–362 (1932)
24. Zehavi, A., Itai, A.: Three tree-paths. J. Graph Theory 13(2), 175–188 (1989)

Balanced Allocations: A Simple Proof for the Heavily Loaded Case

Kunal Talwar and Udi Wieder

Microsoft Research
{kunal,uwieder}@microsoft.com

Abstract. We give a new proof for the fact that the expected gap between the maximum load and the average load in the two-choice process is bounded by $(1 + o(1)) \log \log n$, irrespective of the number of balls thrown. The original proof of this surprising result, due to Berenbrink et al. in [2], uses tools from Markov chain theory, and a sophisticated induction proof involving computer-aided calculations. We provide a significantly simpler and more elementary proof. The new technique allows us to generalize the result and derive new and often tight bounds for the case of weighted balls. The simplification comes at a cost of larger lower order terms and a weaker tail bound for the probability of deviating from the expectation.

1 Introduction

Balls-and-Bins processes are a name for randomized allocations processes, typically used to model the performance of hashing or more general load balancing schemes. Suppose there are m balls (think items) to be thrown into n bins (think hash buckets). We want a simple process that will keep the loads balanced, while allowing quick decentralized lookup. One of the simplest such process is the Balls-and-Bins process where balls are placed sequentially via some simple randomized allocation process, and not moved once placed. There are other approaches to the problem that we will not discuss here. A significant body of work had been amassed on the analysis of simple and natural versions of these Balls-and-Bins processes. In this work we present a simpler analysis for the heavily loaded case for many of these processes.

In the Greedy[d] process (sometimes called the d-choice process), balls are placed sequentially into n bins with the following rule: Each ball is placed by uniformly and independently sampling d bins and assigning the ball to the least loaded of the d bins[1]. In this paper we are interested in the *gap* of the allocation, which is the difference between the number of balls in the heaviest bin, and the average. The case $d = 1$, when balls are placed uniformly at random in the bins, is well understood. In particular when n balls are thrown, the bin with the largest number of balls will have $\Theta(\log n / \log \log n)$ balls w.h.p. Since the

[1] Assume for simplicity and w.l.o.g that ties are broken according to some fixed ordering of the bins.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 979–990, 2014.

average is 1, asymptotically this is also the gap. If $m \gg n$ balls are thrown the heaviest bin will have $m/n + \Theta(\sqrt{m \log n/n})$ balls w.h.p. [10]. In other words $\mathrm{Gap}(m) = \Theta(\sqrt{m \log n/n})$ w.h.p.

In an influential paper Azar et al. [1] showed that when n balls are thrown and $d > 1$ the gap[2] is $\log\log n/\log d + O(1)$ w.h.p. The case $d = 2$ is implicitly shown in Karp et al. [5]. The proof by Azar et al. uses a simple but clever induction; in our proof here we take the same approach. The proof in [1] breaks down once the number of balls is super-linear in the number of bins. Two other approaches to prove this result, namely, using differential equations or witness trees, also fail when the number of balls is large (see for example the survey [6]). A breakthrough was achieved by Berenbrink et al. in [2]:

Theorem 1 ([2]). *For every $c > 0$ there is a $\gamma = \gamma(c)$ so that for any given $m \in \mathbb{N}$,*

$$\Pr[\mathit{Gap}(m) \geq \frac{\log\log n}{\log d} + \gamma] \leq n^{-c}$$

Thus the additive gap remains at $\log\log n$ even after $m \gg n$ balls are thrown! Contrast this with the one choice case in which the gap diverges with the number of balls. At a (very) high level their approach was the following: first they show that the gap after m balls are thrown is distributed similarly to the gap after only $\mathit{poly}(n)$ balls are thrown. This is done by bounding the mixing time of an underlying Markov chain. The second step is to extend the induction technique of [1] to the case of $\mathit{poly}(n)$ balls. This turns out to be a major technical challenge which involves four inductive invariants and computer-aided calculations. Furthermore, whenever the model is tweaked, this technically difficult part of the proof needs to be redone, making such changes challenging. As such, finding a simpler proof has remained an interesting open problem [8].

1.1 Our Contributions

In this paper we present a simple proof for a bound similar to that of Theorem 1.

Theorem 2. *For any m, the expected gap between maximum and average of Greedy$[d]$ is at most $\frac{\log\log n}{\log d} + O(\log\log\log n)$. Moreover for an absolute constant c,*

$$\Pr[\mathit{Gap}(m) > \frac{\log\log n}{\log d} + c\log\log\log n] \leq c(\log\log n)^{-4}$$

Our proof builds on the *layered induction* approach of Azar et al. [1]. The basic approach bounds the number of bins containing at least h balls, by using an induction on h. This approach runs into several issues when trying to go beyond $O(n)$ balls, the most crucial of these is establishing the base case for the induction: When the number of balls is n it trivially holds that the number

[2] Unless otherwise stated, all logs in this paper are base 2.

of bins that received at least 2 balls is at most $n/2$. When $m >> n$ there is no straightforward argument to claim that the number of bins with load above the average is at most $n/2$. We show that the potential function approach of [9] allows us to surmount these hurdles: the bound from the potential function lets us restrict ourselves to the last $\tilde{O}(n \log n)$ balls, and also gives us a suitable base case for the layered induction.

Our proof is relatively short and accessible. This simplicity comes at a price. Our bound is slightly weaker than Theorem 1, it has larger lower order terms and a weaker tail bound on the probability of deviation from expectation.

On the positive side the simple proof structure allows for easier generalization. We obtain bounds on similar processes without much added difficulty. These include a bound on Vöcking's Left$[d]$ process (also shown in [2]) which we present in Section 4, as well as tight bounds on processes with weighted balls, which were previously unknown. For instance suppose that each ball has a weight sampled uniformly from the set $\{1, 2\}$. in Section 3, we show that the gap is $2 \log \log n$ up to lower order terms. This improves on the previously best known bound of $O(\log n)$. We also show lower bounds for the weighted case that match our upper bounds for several interesting distributions. In particular, for the case of weights in $\{1, 2\}$, we show that the upper bound of $2 \log \log n$ is tight up to lower order terms.

Another way to characterize the d-choice process is by defining the probability a ball is placed in one of the i heaviest bins (at the time when it is placed) to be exactly $(i/n)^d$. We remark that using this characterization, there is no need to assume that d is a natural number. While the process is algorithmically simpler to describe when d is an integer, natural cases arise in which d is not an integer, c.f. [13]. Our approach, being based on layered induction, naturally extends to this setting for any $d > 1$.

2 The Main Proof

We define the normalized *load vector* X^t to be an n dimensional vector where X_i^t is the difference between the load of the i'th bin after tn balls are thrown and the average t, (so that a load of a bin can be negative and $\sum X_i = 0$). We also assume without loss of generality that the vector is sorted so that $X_1^t \geq X_2^t \geq ... \geq X_n^t$. We will consider a Markov chain with state space X^t, where one step of the chain consists of throwing n balls according to the d-choice scheme and then sorting and normalizing the load vector.

The main tool we use is the following Theorem proven in [9].

Theorem 3 ([9]). *For every $d > 1$ there exist positive constants a and b such that for all n and all t,*

$$\mathbb{E}\left[\sum_i \exp\left(a|X_i^t|\right)\right] \leq bn.$$

Let $G^t \stackrel{def}{=} X_1^t$ denote the gap between maximum and average when sampling from X^t. Applying Markov's inequality to Theorem 3 implies the following:

Lemma 1. *For any t, any $c \geq 0$, $\Pr[G^t \geq \frac{c \log n}{a}] \leq \frac{bn}{n^c}$. Thus for every c there is a $\gamma = \gamma(c)$ such that $\Pr[G^t \geq \gamma \log n] \leq n^{-c}$.*

We remark that Theorem 3 is a statement about the absolute values of the X_i^t's and thus a version of Lemma 1 holds also for the gap between the *minimum* and the average. This bound is tight up to constant factors: the lightest bin indeed trails the average by a logarithmic number of balls (see e.g. [9]). The challenge is therefore to use a different technique to "sharpen" the bound on the gap between maximum and average. We do this next by showing that if the gap is indeed bounded by $\log n$, then after additional $n \log n$ balls are thrown the gap is reduced to $\log \log n$.

The crucial lemma, that we present next, says that if the gap at time t is L, then after throwing another nL balls, the gap becomes $\log \log n + O(\log L)$ with probability close to 1. Roughly speaking, our approach is to apply the lemma twice, first with $L = O(\log n)$ taken from Theorem 3. This reduces the bound to $O(\log \log n)$. A second application of the lemma with $L = O(\log \log n)$ implies Theorem 2. While Theorem 2 holds for any $d > 1$, for ease of exposition we assume in the following that $d = 2$. Generalizing for any $d > 1$ requires nothing more than choosing the constants appropriately and is done in the full version of the paper.

Lemma 2. *There is a universal constant γ such that the following holds: for any t, ℓ, L such that $1 \leq \ell \leq L \leq n^{\frac{1}{4}}$ and $\Pr[G^t \geq L] \leq \frac{1}{2}$,*

$$\Pr[G^{t+L} \geq \log \log n + \ell + \gamma] \leq \Pr[G^t \geq L] + \frac{16bL^3}{\exp(a\ell)} + \frac{1}{n^2},$$

where a, b are the constants froms Theorem 3.

Intuition: We use the layered induction technique: For a specific ball to increase the number of balls in a bin from i to $i+1$, it must pick two bins that already contain at least i balls. If we assume inductively that the fraction of bins with at least i balls when this ball is placed is at most β_i, then this probability is at most β_i^2 and thus there are (on expectation) at most $nL\beta_i^2$ bins with load at least $i+1$. Roughly speaking this implies that $\beta_{i+1} \approx L\beta_i^2$. While the β_i's are a function of time, they are monotonically increasing and using the final β_i value would give us an upper bound on the probability of increase. The main challenge is to obtain a base case for the induction. Theorem 3 provides us with such a base case, for bins with ℓ more balls than the average in X^{t+L}. For simplicity, the reader may think of L as $O(\log n)$ and ℓ as $O(\log \log n)$. With these parameters Theorem 3 implies that the fraction of bins with load at least $\ell = O(\log \log n)$ (at time $t + L$) is at most $\frac{1}{4 \log n}$, so the β's shrink in each induction step even though $n \log n$ balls are thrown. As mentioned above, we will use the lemma a second time for $L = O(\log \log n)$ and $\ell = O(\log \log \log n)$.

Proof (Lemma 2). We sample an allocation X^t and let G^t be its gap. Now take an additional L steps of the Markov chain to obtain X^{t+L}, in other words, an additional nL balls are thrown by the 2-choice process. For brevity, we will use

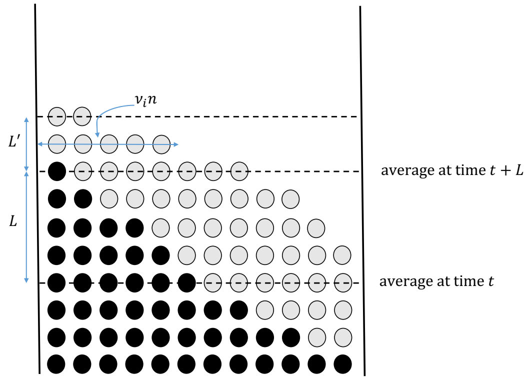


Fig. 1. Black balls are in X, nL white balls are thrown to obtain X'

X, G, X', G' to denote $X^t, G^t, X^{t+L}, G^{t+L}$ respectively. We condition on $G < L$ and we prove the bound for G'. Let $L' = \log\log n + \ell + \gamma$. Observe that:

$$\Pr[G' \geq L'] \leq \Pr[G' \geq L' \mid G < L] + \Pr[G \geq L] \tag{1}$$

It thus suffices to prove that $\Pr[G' \geq L' \mid G < L] \leq \frac{16bL^3}{\exp(a\ell)} + \frac{1}{n^2}$. We do this using a layered induction similar to the one in [1].

Let ν_i be the fraction of bins with normalized load at least i in X' (i.e. containing $t + L + i$ balls or more), we will define a series of numbers β_i such that $\nu_i \leq \beta_i$ with high probability. To convert an expectation bound to a high probability bound, we will use a Chernoff-Hoeffding tail bound as long as $\beta_i n$ is large enough (at least $\log n$). The case for larger i will be handled separately.

By Theorem 3 and along with the assumption $\Pr[G < L] \geq \frac{1}{2}$, Markov's inequality implies that,

$$\Pr[\nu_\ell \geq \frac{1}{8L^3} \mid G < L] \leq \frac{16bL^3}{\exp(a\ell)}. \tag{2}$$

We will set $\beta_\ell = \frac{1}{8L^3}$ as the base of the layered induction. We next define the series β_i.

Let $i^* = \ell + \log\log n$. Recall that we set $\beta_\ell = \frac{1}{8L^3}$. For $i = \ell, \ldots, i^* - 1$ we set $\beta_{i+1} = \max(2L\beta_i^2, 18\log n/n)$. It is easy to check that $\beta_{i^*} = 18\log n/n$. Indeed suppsose that the truncation does not come into play until i^*. Then the recurrence

$$\log \beta_\ell = -3\log(2L),$$
$$\log \beta_{i+1} = 2\log \beta_i + \log(2L)$$

solves to $\log \beta_{\ell+k} = (\log 2L)(-3 \cdot 2^k + (2^k - 1))$ so that $\log \beta_{i^*} = \log \beta_{\ell + \log\log n} \leq (\log 2L)(-2\log n)$. This is at most $-2\log n$ as $L \geq 1$ so that $\beta_{i^*} \leq \frac{1}{n^2}$ which is smaller than the truncation threshold, contradicting the assumption.

The inductive step is encapsulated in the next lemma. The proof is an expectation computation, followed by an application of the Chernoff-Hoeffding bound. Let $B(n,p)$ denote a binomially distributed variable with parameters n and p.

Lemma 3. *For* $i \in [\ell, i^* - 1]$, *we have* $\Pr[\nu_{i+1} > \beta_{i+1} \mid \nu_i \leq \beta_i, G < L] \leq \frac{1}{n^3}$.

Proof. For convenience, let the balls existing in X be black, and let the new nL balls thrown be white. We define the *height* of a ball to be the load of the bin in which it was placed relative to X', that is, if the ball was the k'th ball to be placed in the bin, the ball's height is defined to be $k - (t + L)$. Notice that the conditioning that $G < L$ implies that all the black balls have a negative height. We use μ_i to denote the number of white balls with height $\geq i$. Thus for any $i \geq 0$, we have $\nu_i n \leq \mu_i$ and thus it suffices to bound μ_i.

For a ball to have a height of at least $i + 1$, it should pick two bins that have load at least i when the ball is placed, and hence at least as much in X'. Thus the probability that a ball has height at least $i+1$ is at most $\nu_i^2 \leq \beta_i^2 \leq \beta_{i+1}/2L$ under our conditioning. Since we place nL balls, the number of balls with height at least $i+1$ is dominated by a $B(nL, \beta_{i+1}/2L)$ random variable. Chernoff bounds (e.g. Theorem 1.1 in [4]) imply that the probability that $\Pr[B(n,p) \geq 2np] \leq \exp(-np/3)$. Thus

$$\begin{aligned}\Pr[\nu_i \geq \beta_{i+1}^2 \mid \nu_i \leq \beta_i] &\leq \Pr[B(nL, \beta_{i+1}/2L) \geq \beta_{i+1} n] \\ &\leq \exp(-\beta_{i+1} n/6) \\ &\leq 1/n^3.\end{aligned}$$

since $\beta_{i+1} n \geq 18 \log n$. □

It remains to bound the number of balls with height $\geq i^*$. To this end we condition on $\nu_{i^*} \leq \beta_{i^*}$, and let H be the set of bins of height at least i^* in X'. Once a bin reaches this height, an additional ball falls in it with probability at most $(2\beta_{i^*} n + 1)/n^2$. The probability that any specific bin in H gets at least 4 balls after reaching height i^* is then at most $\Pr[B(nL, (2\beta_{i^*} n + 1)/n^2) \geq 4]$. Recalling that $\Pr[B(n,p) \geq k] \leq \binom{n}{k} p^k \leq (enp/k)^k$. Using this estimate and applying a union bound over the bins in H, we conclude that

$$\begin{aligned}Pr[\nu_{i^*+4} > 0 \mid \nu_{i^*} \leq \beta_{i^*}, G < L] &\leq (18 \log n) \times (eL(32 \log n + 1)/4n)^4) \\ &\leq \frac{1}{2n^2}, \qquad (3)\end{aligned}$$

as long as n exceeds an absolute constant n_0. On the other hand, Lemma 1 already implies that for $n \leq n_0$, Lemma 2 holds with $\gamma = O(\log n_0)$ so that this assumption is without loss of generality.

Finally a union bound using (2) and Lemma 3 and (3), we get that

$$\begin{aligned}
&\Pr[\nu_{i^*+4} > 0 \mid G < L] \\
&\quad \leq \Pr[\nu_\ell \geq \beta_\ell \mid G < L] + \sum_{i=\ell}^{i^*-1} \Pr[\nu_{i+1} > \beta_{i+1} \mid \nu_i \leq \beta_i, G < L] \\
&\qquad + Pr[\nu_{i^*+4} > 0 \mid \nu_{i^*} \leq \beta_{i^*}, G < L] \\
&\quad \leq \frac{16bL^3}{\exp(a\ell)} + \frac{\log\log n}{n^3} + \frac{1}{2n^2} \\
&\quad \leq \frac{16bL^3}{\exp(a\ell)} + \frac{1}{n^2}.
\end{aligned}$$

This concludes the proof of Lemma 2. □

Lemma 2 allows us to bound $\Pr[G^{t+L} \geq \log\log n + O(\log L)]$ by $\Pr[G^t \geq L] + \frac{1}{poly(L)}$. Since $\Pr[G^t \geq O(\log n)]$ is small, we can conclude that $\Pr[G^{t+O(\log n)} \geq O(\log\log n)]$ is small. Another application of the lemma, now with $L = O(\log\log n)$ then gives that $\Pr[G^{t+O(\log n)+O(\log\log n)} \geq \log\log n + O(\log\log\log n)]$ is small. We formalize these corollaries next.

Corollary 1. *There is a universal constant γ such that for any $t \geq (12\log n)/a$, $\Pr[G^t \geq (5 + \frac{10}{a}) \cdot \log\log n + \gamma] \leq \frac{2}{n^2} + \frac{1}{\log^4 n}$.*

Proof. Set $L = 12\log n/a$, and use Lemma 1 to bound $\Pr[G^{t-L} \geq L]$. Set $\ell = \log(16bL^3\log^4 n)/a$ in Lemma 2 to derive the result. □

Corollary 2. *There are universal constants γ, α such that for any $t \geq \omega(\log n)$, $\Pr[G^t \geq \log\log n + \alpha\log\log\log n + \gamma] \leq \frac{3}{n^2} + \frac{1}{\log^4 n} + \frac{1}{(\log\log n)^4}$.*

Proof. Set $L = \log(16b(\frac{12\log n}{a})^3\log^4 n)/a = \frac{7\log\log n}{a} + O_{a,b}(1)$ and use Corollary 1 to bound $\Pr[G^{t-L} \geq L]$. Set $\ell = \log(16bL^3(\log\log n)^4)/a$ to derive the result. □

This proves that with probability $(1 - o(1))$, the gap is at most $\log\log n + o(\log\log n)$. We can also use Lemma 2 to upper bound the expected gap. Towards this end, we prove slight generalizations of the above corollaries:

Corollary 3. *There is a universal constant γ such that for any $k \geq 0$, $t \geq (12\log n)/a$, $\Pr[G^t \geq (5 + \frac{10}{a}) \cdot \log\log n + k + \gamma] \leq \frac{2}{n^2} + \frac{\exp(-ak)}{\log^4 n}$.*

Proof. Set $L = 12\log n/a$, and use Lemma 1 to bound $\Pr[G^{t-L} \geq L]$. Set $\ell = k + \log(16bL^3\log^4 n)/a$ to derive the result. □

Corollary 4. *There are universal constants γ, α such that for any $k \geq 0$, $t \geq \omega(\log n)$, $\Pr[G^t \geq \log\log n + \alpha\log\log\log n + k + \gamma] \leq \frac{3}{n^2} + \frac{1}{\log^4 n} + \frac{\exp(-ak)}{(\log\log n)^4}$.*

Proof. Set $L = \log(16b(\frac{12\log n}{a})^3 \log^4 n)/a = \frac{7\log\log n}{a} + O_{a,b}(1)$ and use Corollary 3 with k=0 to bound $\Pr[G^{t-L} \geq L]$. Set $\ell = k + \log(16bL^3(\log\log n)^4)/a$ to derive the result. □

Using the above results, we can now prove

Corollary 5. *There are universal constants γ, α such that for $t \geq \omega(\log n)$ $\mathbb{E}[G^t] \leq \log\log n + \alpha \log\log\log n + \gamma$.*

Proof. Let $\ell_1 = \log\log n + \alpha \log\log\log n + \gamma_1$ for α, γ_1 from Corollary 4, and let $\ell_2 = (5 + \frac{10}{a}) \cdot \log\log n + \gamma_2$ for γ_2 from Corollary 3. Finally, let $\ell_3 = 12\log n/a$. We bound

$$\mathbb{E}[G^t] \leq \ell_1 + \int_{\ell_1}^{\ell_2} \Pr[G^t \geq x]\,\mathrm{d}x + \int_{\ell_2}^{\ell_3} \Pr[G^t \geq x]\,\mathrm{d}x + \int_{\ell_3}^{\infty} \Pr[G^t \geq x]\,\mathrm{d}x$$

Each of the three integrals are bounded by constants, using Corollaries 4 and 3 and Lemma 1 respectively. □

All that remains to prove the $d = 2$ case of Theorem 2 is to show that the lower bound condition on t is unnecessary.

Lemma 4. *For $t \geq t'$, $G^{t'}$ is stochastically dominated by G^t. Thus $\mathbb{E}[G^{t'}] \leq \mathbb{E}[G^t]$ and for every k, $\Pr[G^{t'} \geq k] \leq \Pr[G^t \geq k]$.*

Proof (sketch). We use the notion of majorization, which is a variant of stochastic dominance. See for example [1] for definitions. Observe that trivially X^0 is majorized by $X^{t-t'}$. Now throw nt' balls with a standard coupling and get that $X^{t'}$ is majorized by X^t. By definition this implies the stochastic dominance of the maximum and the bounds on the expectation and the tail follow.

3 The Weighted Case

So far we assumed all balls are identical. Often balls-and-bins processes model scenarios where items are of heterogenous size. A natural way to extend the model is to assign weights to the balls drawn from some distribution. We use the model proposed in [11] and also used in [9]. Every ball comes with a weight W independently sampled from a non-negative weight distribution $\mathcal{W}$. The weight of a bin is the sum of weights of balls assigned to it. The *gap* is now defined as the difference between the weight of the heaviest bin and the average bin. We observe that by multiplying all weights by the appropriate constant we can normalize the distribution so that $\mathbb{E}[\mathcal{W}] = 1$. In [11] it is shown that if $\mathcal{W}$ has a bounded second moment and satisfies some additional mild smoothness condition, then the expected gap does not depend on the number of balls. However, no explicit bounds on the gap are shown. In [9] it is shown that if $\mathcal{W}$ satisfies $\mathbb{E}[\exp(\lambda W)] < \infty$ for some $\lambda > 0$, then the gap is bounded by $O(\log n)$ (with λ effecting the hidden constant in O notation). For some distributions, such as the exponential distribution, this bound is tight. A bound of $O(\log n)$ does not necessarily remain tight as the distribution becomes more concentrated.

Consider for example the case where the size of each ball is drawn uniformly from $\{1, 2\}$. Previous techniques such as [2] fail to prove an $O(\log \log n)$ bound in this case, and the best bound prior to this work is the $O(\log n)$ via the potential function argument of [9].

The fact that Theorem 3 holds means that the techniques of this paper can be applied. The modifications needed are straightforward. The layered induction argument works as is, with the only change being that we go up in steps of size two instead of one. This shows a bound of $2 \log \log n + O(\log \log \log n)$ for this distribution, which we soon show is tight up to lower order terms.

Generalizing the argument, for a weight distribution W with a bounded exponential moment generating function, let M be the smallest value such that $\Pr[W \geq M] \leq \frac{1}{n \log n (\log \log n)^5}$ (the constant 5 here is somewhat arbitrary, and will only affect the probability of the gap exceeding the desired bound). Then carrying out a proof analogous to Lemma 2, with step size M gives a bound of $M(\log \log n + O(\log \log \log n))$ with probability $(1 - \frac{3}{(\log \log n)^4})$. This follows since by the definition of M, the probability that any of the last $O(n \log n)$ exceeds size M is $O(\frac{1}{(\log \log n)^5})$, and conditioning on this event the proof goes through unchanged except for the fact that we go up in increments of M.

Indeed, when we use the lemma with $L = O(\log n)$, the base of the induction as before gives us for $\ell = O(\log \log n)$, the fraction of bins with load at least ℓ is at most $\frac{1}{L^3}$. By the argument in Lemma 3, no more than $\beta_{i_L+1} n$ balls will fall in bins that already have at least this load. Since we condition on the $O(n \log n)$ white balls being of size at most M, the number of bins of load $\ell + M$ is at most $\beta_{i_L+1} n$. Continuing in this fashion, with step size M in each step of the induction, we get that there are at most $O(\log n)$ bins of load larger than $O(\log \log n) + M \log_2 \log n$. Finally, as before, we can complete the argument with an additional overhead of $O(M)$ as each of these bins is unlikely to get more than a constant number of balls. Finally, a second application of the Lemma gives us the claimed bound.

We next instantiate this bound for some specific distributions. As remarked above, for an exponential or a geometric distribution, the gap is $\Theta(\log n)$ and this induction approach will not help us prove a better bound. Consider a half-normal weight distribution with mean 1 (i.e. W is the absolute value of an $N(0, \frac{\pi}{2})$ random variable. Then $M = \sqrt{\pi} erf^{-1}(1 - \frac{1}{n \log n (\log \log n)^5}) = \Theta(\sqrt{\log n})$. This gives a bound of $O(\sqrt{\log n} \log \log n)$ instead of $O(\log n)$ that we get from [9]. On the other hand, as we show in the next section, a lower bound of $\Omega(\sqrt{\log n})$ is easily proved.

Similarly, if the weight distribution is uniform in $[a, b]$, normalizing the expectation to 1 makes $b = 2 - a \leq 2$. An upper bound of $b \log \log n \leq 2 \log \log n$ follows immediately.

We note that Lemma 4 does not hold when balls are weighted (c.f [11],[3]). As a result this proof leaves a "hole" between n and $n \log n$. It proves the bound on the gap when $O(n)$ or $\Omega(n \log n)$ balls are thrown but does not cover for example $\Theta(n\sqrt{\log n})$ balls.

Lower Bounds. If weights are drawn uniformly from $\{1, 2\}$ one might hope the maximum load to be $3/2 \log\log n + O(1)$. It is true that $n/2$ balls of weight 2 already cause a gap of $2 \log\log n$ but one hopes that the balls of weight 1 would reduce this gap. Our first lower bound shows that this intuition is not correct and that the maximum load is indeed $2 \log\log -O(1)$.

Theorem 4. *Suppose that the weight distribution $\mathcal{W}$ satisfies $\Pr[W \geq s] \geq \epsilon$ for some $s \geq 1, \epsilon > 0$ and $\mathbb{E}[W] = 1$. For large enough n, for every $m \geq n/\epsilon$, the gap of Greedy$[d]$ is at least $s(\log\log n/ \log d) - O(s)$ with constant probability.*

A similar lower bound is proven in [1] for the case $m = n$ and uniform weights. In the uniform case, majorization (similar to Lemma 4) extends the lower bound to any $m > n$. The same could not be said in the weighted case. For the $m = n$ case the weighted case is almost as simple as a variable change in the proof of [1]. The extension to all $m \geq n$ is done, similarly to the upper bound, by using Theorem 3 to provide a base case for the inductive argument.

Proof. It is convenient to think of time m as time 0 and count both load and time with respect to the m'th ball, so when we say a bin has load i in time t it actually means it has load $w(m)/n + i$ at time $m + t$, where $w(m)$ is the total weight of the first m balls. The bound will be proven for time $m + n/\epsilon$ which is time n/ϵ in our notation. Intuitively, in this amount of time we will see about n balls of weight at least s which would cause the maximum load to increase by $s(\log\log n - O(1))$. The average however would increase only by $O(\frac{1}{\epsilon}) = O(s)$, hence the gap would be at least $s \log\log n - O(s)$.

We follow the notation set in [6], with appropriate changes. The variable $\nu_j(t)$ indicates the number of bins with load in $[(j - 1)s, \infty)$ at time t. We will set a series of numbers γ_i and times t_i (to be specified later) and an event $\mathcal{F}_i := \{\nu_i(t_i) \geq \gamma_i\}$. For the base case of the induction we set $\gamma_0 = n/ \log^2 n$ and $t_0 = 0$. We observe that Theorem 3 implies that for large enough n, $\Pr[\nu_0(0) \geq \gamma_0] \geq 1 - 1/n^2$, so $\mathcal{F}_0$ occurs with high probability. Indeed Theorem 3 implies that for the normalized load vector, $|X^t|_\infty \leq c \log n$ for an absolute constant c. If half the X^t_i's are at least $-s$, we are already done. If not then then $\sum_{i:X^t_i<-s} |X^t_i|$ is at least $\frac{ns}{2}$. Thus the sum $\sum_{i:X^t_i\geq 0} |X^t_i| = \sum_{i:X^t_i<0} |X^t_i| \geq \frac{ns}{2}$. The bound on $|X^t|_\infty$ then implies that at least $ns/c \log n$ X^t_i's are non-negative. Since $s \geq 1$, the base case is proved.

Our goal is to show that $\Pr[\mathcal{F}_{i+1}|\mathcal{F}_i]$ is large. To this end, we define $t_i = (1 - 2^{-i})\frac{n}{\epsilon}$ and the range $R_i := \left[(1 - 2^{-i})\frac{n}{\epsilon}, (1 - 2^{-(i+1)})\frac{n}{\epsilon}\right]$. Finally fix an $i > 0$ and for $t \in R_i$ define the binary random variable

$$Z_t = 1 \text{ iff ball } t \text{ pushes load of a bin above } is \text{ or } \nu_{(i+1)}(t - 1) \geq \gamma_{i+1}.$$

As long as $\nu_{(i+1)}(t-1) < \gamma_{i+1}$ it holds that for $Z_t = 1$ it suffices that a ball of weight at least s is placed in a bin of load $h \in [s(i-1), si)$. Conditioned on $\mathcal{F}_i$, the probability of that is at least $\epsilon\left((\frac{\gamma_i}{n})^d - (\frac{\gamma_{i+1}}{n})^d\right) \geq \frac{\epsilon\gamma_i^d}{2n^d}$ since we will set $\gamma_{i+1} \leq \gamma_i/2$. Denote $p_i := \frac{\epsilon\gamma_i^d}{2n^d}$ and by $B(n,p)$ a variable distributed according to the Binomial distribution. We have: $\Pr\left[\sum_{i\in R_i} Z_i \leq k \mid \mathcal{F}_i\right] \leq \Pr\left[B\left(\frac{n}{\epsilon 2^{i+1}}, p_i\right) \leq k\right]$. We continue exactly as in [6] by choosing $\gamma_{i+1} = \frac{\gamma_i^d}{2^{i+3}n^{d-1}}$. Now Chernoff bounds imply that as long as $\frac{np_i}{\epsilon 2^{i+1}} \geq 17\log n$ it holds that $\Pr\left[B\left(\frac{n}{\epsilon 2^{i+1}}, p_i\right) \leq \gamma_{i+1}\right] = o(1/n^2)$. The tail inequality holds as long as $i \leq \log\log n/\log d - O(1)$, at which point the load had increased by $s(\log\log n/\log d) - O(s)$. The average increased by at most $4/\epsilon \leq 4s$ with probability 3/4, and the theorem follows. □

We note that the uniform distribution on $\{1,2\}$ (when normalized by a factor of $\frac{2}{3}$) satisfies the conditions of this Theorem with $s = 2, \epsilon = \frac{1}{2}$. Thus the gap is $2\log\log n - O(1)$.

Another, rather trivial lower bound applies to distributions with heavier tails.

Theorem 5. *Let $\mathcal{W}$ be a weight distribution with $\mathbb{E}_{W\sim\mathcal{W}}[W] = 1$. Let M be such that $\Pr_{W\sim\mathcal{W}}[W \geq M] \geq \frac{1}{n}$. Then for any allocation scheme, the gap is at least $M - O(1)$ with constant probability.*

Proof. After throwing n balls, the probability that we do not see a ball of weight M or more is at most $(1-\frac{1}{n})^n \leq \frac{1}{2}$. Moreover, by Markov's, the average is at most 4 except with probability $\frac{1}{4}$. Thus with probability at least $\frac{1}{4}$, the maximum is at least M and the average is at most 4. □

We note that this implies an $\Omega(\log n)$ lower bound for an exponential distribution, and an $\Omega(\sqrt{\log n})$ lower bound for the half normal distribution.

4 The Left[*d*] Scheme

Next we sketch a tight bound for Vöcking's Left[d] process [12]. The result had been shown in [2], though there they had to redo large sections of the proof, while here we only require minor changes. Recall that in Left[d], the bins are partitioned into d sets of n/d bins each (we assume n is divisible by d). When placing a ball, one bin is sampled uniformly from each set and the ball is placed in the least loaded of the d bins. The surprising feature of this process is that ties are broken according to a fixed ordering of the sets (we think of the sets as ordered from left to right and ties are broken "to the left", hence the name of the scheme). Surprisingly, the gap now drops from $\frac{\log\log n}{\log d}$ to $\frac{\log\log n}{d\ln\phi_d}$ where $\phi_d = \lim_{k\to\infty}(F_d(k))^{\frac{1}{k}} \in [1.61, 2)$ is the base of the order d Fibonacci number.

The key ingredient in the proof is Theorem 3 from [9]. The exponential potential function is Schur-Convex and therefore the theorem holds for any process which is majorized by the Greedy[d] process. It is indeed the case that Vöcking's Left[d] process [12] is majorized by Greedy[d] (see the proof in [2]). All that remains is to prove the analog of Lemma 2. For this we follow the analysis of Mitzenmacher and Vöcking in [7]. Let X_{jd+k} be the number of bins of load at least j from the k'th set, and set $x_i = X_i/n$. It is easy to verify that $\mathbb{E}[x_i|x_{<i}] \leq d^d \prod_{j=i-d}^{i-1} x_j$. From here the proof is similar to that of Lemma 2 and is left as an exercise.

5 Discussion

An interesting corollary from Theorem 3 is that the Markov chain X^t has a stationary distribution and that the bounds hold also for the stationary distribution itself. In that sense, while in [2] the mixing of the chain was used to show that the interesting events happen at the beginning, and thus an induction on the first $poly(n)$ suffices, in our technique we look directly at the distant "future" and argue on the stationary distribution itself. When balls are unweighted a majorization based argument shows that moving closer in time can only improve the bounds on the gap (this is Lemma 4). Unfortunately, a similar Lemma does not hold when balls are weighted (see [3]), so we need to specify the time periods we look at. Indeed, while our results hold when considering a large number of balls, we curiously have a 'hole' for a number of balls that is smaller than $n \log n$.

References

1. Azar, Y., Broder, A., Karlin, A., Upfal, E.: Balanced allocations. SIAM J. Computing 29(1), 180–200 (1999)
2. Berenbrink, P., Czumaj, A., Steger, A., Vöcking, B.: Balanced allocations: The heavily loaded case. SIAM J. Computing 35(6), 1350–1385 (2006)
3. Berenbrink, P., Friedetzky, T., Hu, Z., Martin, R.: On weighted balls-into-bins games. Theor. Comput. Sci. 409(3), 511–520 (2008)
4. Dubhashi, D., Panconesi, A.: Concentration of Measure for the Analysis of Randomized Algorithms. Cambridge University Press (2009)
5. Karp, R., Luby, M., Meyer auf der Heide, F.: Efficient pram simulation on a distributed memory machine. In: STOC, pp. 318–326 (1992)
6. Mitzenmacher, M., Richa, A.W., Sitaraman, R.: The power of two random choices: A survey of techniques and results. In: Handbook of Randomized Computing, pp. 255–312. Kluwer (2000)
7. Mitzenmacher, M., Vcking, B.: The asymptotics of selecting the shortest of two, improved. In: Allerton, pp. 326–327 (1998)
8. Pagh, R.: Hashing 2. Slides for the MADALGO Summer School on Data Structures (2013), `http://www.madalgo.au.dk/html_sider/2_5_Events/SS2013/Course_material2013.html`
9. Peres, Y., Talwar, K., Wieder, U.: The (1 + beta)-choice process and weighted balls-into-bins. In: SODA 2010, pp. 1613–1619 (2010)
10. Raab, M., Steger, A.: "Balls into bins" - A simple and tight analysis. In: Rolim, J.D.P., Serna, M., Luby, M. (eds.) RANDOM 1998. LNCS, vol. 1518, pp. 159–170. Springer, Heidelberg (1998)
11. Talwar, K., Wieder, U.: Balanced allocations: the weighted case. In: STOC, pp. 256–265 (2007)
12. Vöcking, B.: How asymmetry helps load balancing. J. ACM 50(4), 568–589 (2003)
13. Wieder, U.: Ballanced allocations with heterogenous bins. In: SPAA, pp. 188–193 (2007)

Close to Uniform Prime Number Generation with Fewer Random Bits

Pierre-Alain Fouque[1] and Mehdi Tibouchi[2]

[1] Université de Rennes 1 and Institut universitaire de France
pierre-alain.fouque@ens.fr
[2] NTT Secure Platform Laboratories
tibouchi.mehdi@lab.ntt.co.jp

Abstract. In this paper, we analyze several variants of a simple method for generating prime numbers with fewer random bits. To generate a prime p less than x, the basic idea is to fix a constant $q \propto x^{1-\varepsilon}$, pick a uniformly random $a < q$ coprime to q, and choose p of the form $a + t \cdot q$, where only t is updated if the primality test fails. We prove that variants of this approach provide prime generation algorithms requiring few random bits and whose output distribution is close to uniform, under less and less expensive assumptions: first a relatively strong conjecture by H. Montgomery, made precise by Friedlander and Granville; then the Extended Riemann Hypothesis; and finally fully unconditionally using the Barban–Davenport–Halberstam theorem.

We argue that this approach has a number of desirable properties compared to previous algorithms. In particular:

- it uses much fewer random bits than both the "trivial algorithm" (testing random numbers less than x for primality) and Maurer's almost uniform prime generation algorithm;
- the distance of its output distribution to uniform can be made arbitrarily small, unlike algorithms like PRIMEINC (studied by Brandt and Damgård), which we show exhibit significant biases;
- all quality measures (number of primality tests, output entropy, randomness, etc.) can be obtained under very standard conjectures or even unconditionally, whereas most previous nontrivial algorithms can only be proved based on stronger, less standard assumptions like the Hardy–Littlewood prime tuple conjecture.

Keywords: Number Theory, Cryptography, Prime Number Generation.

1 Introduction

There are several ways in which we could assess the quality of a random prime generation algorithm, such as its speed (time complexity), its accuracy (the probability that it outputs numbers that are in fact composite), its statistical properties (the regularity of the output distribution), and the number of bits of randomness it consumes to produce a prime number (as good randomness is crucial to key generation and not easy to come by [9]).

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 991–1002, 2014.

In a number of works in the literature, cryptographers have proposed faster prime generation algorithms [4,3,16,15] or algorithms providing a proof that the generated numbers are indeed prime numbers [17,18,19].

A number of these works also prove lower bounds on the entropy of the distribution of prime numbers they generate, usually based on very strong conjectures on the regularity of prime numbers, such as the prime r-tuple conjecture of Hardy–Littlewood [13]. However, such bounds on the entropy do not ensure that the resulting distribution is statistically close to the uniform distribution: for example, they do not preclude the existence of efficient distinguishers from the uniform distribution, which can indeed be shown to exist in most cases.

But some cryptographic protocols (including most schemes based on the Strong RSA assumption, such as Cramer–Shoup signatures [5]) specifically require uniformly distributed prime numbers for the security proofs to go through.

Moreover, some cryptographers, like Maurer [17], have argued that even for more common uses of prime number generation, like RSA key generation, one should preferably generate primes that are almost uniform, so as to avoid biases in the RSA moduli N themselves, even if it is not immediately clear how such biases can help an adversary trying to factor N. This view is counterbalanced by results of Mihăilescu [20] stating in particular that, provided the biases are not too large (a condition that is satisfied by the algorithms with large output entropy mentioned above, if the conjectures used to establish those entropy bounds hold), then, asymptotically, they can give at most a polynomial advantage to an adversary trying to factor N. This makes the problem of uniformity in prime number generation somewhat comparable to the problem of tightness in security reductions.

To the authors' knowledge, the only known prime generation algorithms for which the statistical distance to the uniform distribution can be bounded are the one proposed by Maurer [17,18] on the one hand, and the trivial algorithm (viz. pick a random odd integer in the desired interval, return it if it is prime, and try again otherwise) on the other hand. The output distribution of the trivial algorithm is exactly uniform (or at least statistically close, once one accounts for the compositeness probability of the underlying randomized primality checking algorithm), and the same can be said for at least some variants of Maurer's algorithm, but both of those algorithms have the drawback of consuming a very large amount of random bits.

By contrast, the PRIMEINC algorithm studied by Brandt and Damgård [3] (basically, pick a random number and increase it until a prime is found) only consumes roughly as many random bits as the size of the output primes, but we can show that its output distribution, even if it can be shown to have high entropy if the prime r-tuple conjecture holds, is also provably quite far from uniform, as we demonstrate in the full version of this paper [10]. It is likely that most algorithms that proceed deterministically beyond an initial random choice, including those of Joye, Paillier and Vaudenay [16,15], exhibit similar distributional biases.

The goal of this paper is to achieve in some sense the best of both worlds: construct a prime generation algorithm that consumes much fewer random bits than the trivial algorithm while being efficient and having an output distribution that is provably close to the uniform one.

We present such an algorithm in §3: to generate a prime p, the basic idea is to fix a constant $q \sim x^{1-\varepsilon}$, pick a uniformly random $a < q$ coprime to q, and choose p of the form $a + t \cdot q$, where only t is updated if the primality test fails. We prove that variants of this approach provide prime generation algorithms requiring few random bits and whose output distribution is close to uniform, under less and less expensive assumptions: first a relatively strong conjecture by H. L. Montgomery, made precise by Friedlander and Granville; then the Extended Riemann Hypothesis; and finally fully unconditionally using the Barban–Davenport–Halberstam theorem.

2 Preliminaries

2.1 Regularity Measures of Finite Probability Distributions

In this subsection, we give some definitions on distances between random variables and the uniform distribution on a finite set. We also provide some relations which will be useful to bound the entropy of our prime generation algorithms. These results can be found in [24].

Definition (Entropy and Statistical Distance). Let X and Y be two random variables on a finite set S. The *statistical distance* between them is defined as the ℓ_1 norm:[1]

$$\Delta_1(X;Y) = \sum_{s\in S} \Big| \Pr[X=s] - \Pr[Y=s] \Big|.$$

We simply denote by $\Delta_1(X)$ the statistical distance between X and the uniform distribution on S:

$$\Delta_1(X) = \sum_{s\in S} \Big| \Pr[X=s] - \frac{1}{|S|} \Big|,$$

and say that X is *statistically close to uniform* when $\Delta_1(X)$ is negligible.[2]

The *squared Euclidean imbalance* of X is the square of the ℓ_2 norm between X and the uniform distribution on the same set:

$$\Delta_2^2(X) = \sum_{s\in S} \Big| \Pr[X=s] - 1/|S| \Big|^2.$$

[1] An alternate definition frequently found in the literature differs from this one by a constant factor 1/2. That constant factor is irrelevant for our purposes.

[2] For this to be well-defined, we of course need a family of random variables on increasingly large sets S. Usual abuses of language apply.

We also define the *collision probability* of X as:

$$\beta(X) = \sum_{s \in S} \Pr[X = s]^2,$$

and the *collision entropy* (also known as the Rényi entropy) of X is then $H_2(X) = -\log_2 \beta(X)$. Finally, the *min-entropy* of X is $H_\infty(X) = -\log_2 \gamma(X)$, where $\gamma(X) = \max_{s \in S}(\Pr[X = s])$.

Lemma A. *Suppose X is a random variable of a finite set S. The quantities defined above satisfy the following relations:*

$$\gamma(X)^2 \le \beta(X) = 1/|S| + \Delta_2^2(X) \le \gamma(X) \le 1/|S| + \Delta_1(X), \tag{1}$$

$$\Delta_1(X) \le \Delta_2(X)\sqrt{|S|}. \tag{2}$$

2.2 Prime Numbers in Arithmetic Progressions

All algorithms proposed in this paper are based on the key idea that, for any given integer $q > 1$, prime numbers are essentially equidistributed among invertible classes modulo q. The first formalization of that idea is de la Vallée Poussin's *prime number theorem for arithmetic progressions* [8], which states that for any fixed $q > 1$ and any a coprime to q, the number $\pi(x; q, a)$ of prime numbers $p \le x$ such that $p \equiv a \pmod q$ satisfies:

$$\pi(x; q, a) \underset{x \to +\infty}{\sim} \frac{\pi(x)}{\varphi(q)}. \tag{3}$$

De la Vallée Poussin established that estimate for constant q, but it is believed to hold uniformly in a very large range for q. In fact, H. L. Montgomery conjectured [21,22] that for any $\varepsilon > 0$:[3]

$$\left|\pi(x; q, a) - \frac{\pi(x)}{\varphi(q)}\right| \ll_\varepsilon (x/q)^{1/2+\varepsilon} \qquad \big(q < x,\ (a, q) = 1\big),$$

which would imply that (3) holds uniformly for $q \ll x/\log^{2+\varepsilon} x$. However, Friedlander and Granville showed [11] that conjecture to be overly optimistic, and proposed the following corrected estimate.

Conjecture B (Friedlander–Granville–Montgomery). For $q < x$, $(a, q) = 1$ and all $\varepsilon > 0$, we have:

$$\left|\pi(x; q, a) - \frac{\pi(x)}{\varphi(q)}\right| \ll_\varepsilon (x/q)^{1/2} \cdot x^\varepsilon.$$

In particular, the estimate (3) holds uniformly for $q \ll x^{1-3\varepsilon}$.

[3] As is usual in analytic number theory and related subjects, we use the notations $f(u) \ll g(u)$ and $f(u) = O\big(g(u)\big)$ interchangeably. A subscripted variable on $\ll$ or O means that the implied constant depends only on that variable.

That conjecture is much more precise than what can be proved using current techniques, however. The best unconditional result of the same form is the Siegel–Walfisz theorem [26], which only implies that (3) holds in the much smaller range $q \ll (\log x)^A$ (for any $A > 0$).

Stronger estimates can be established assuming the Extended Riemann Hypothesis (i.e. the Riemann Hypothesis for L-functions of Dirichlet characters), which gives [6, p. 125]:

$$\left|\pi(x;q,a) - \frac{\pi(x)}{\varphi(q)}\right| \ll x^{1/2}\log x \qquad \big(q < x,\ (a,q) = 1\big).$$

This implies (3) in the range $q \ll x^{1/2}/\log^{2+\varepsilon} x$, which is again much smaller than the one from Conjecture B. The range can be extended using averaging, however. The previous result under ERH is actually deduced from estimates on the character sums $\pi(x,\chi) = \sum_{p\le x}\chi(p)$ for nontrivial Dirichlet characters $\chi \bmod q$, and more careful character sum arguments allowed Turán to obtain the following theorem.

Theorem C (Turán [25]). *The Extended Riemann Hypothesis implies that for all $q < x$:*

$$\sum_{a\in(\mathbb{Z}/q\mathbb{Z})^*}\left|\pi(x;q,a) - \frac{\pi(x)}{\varphi(q)}\right|^2 \ll x(\log x)^2$$

where the implied constant is absolute.

That estimate is nontrivial in the large range $q \ll x/\log^{4+\varepsilon}$, and implies that (3) holds for all q in that range and *almost all* $a \in (\mathbb{Z}/q\mathbb{Z})^*$.

Averaging over the modulus as well, it is possible to obtain fully unconditional estimates valid in a similarly wide range: this is a result due to Barban [2] and Davenport and Halberstam [7]. We will use the following formulation due to Gallagher [12], as stated in [6, Ch. 29].

Theorem D (Barban–Davenport–Halberstam). *For any fixed $A > 0$ and any Q such that $x(\log x)^{-A} < Q < x$, we have:*

$$\sum_{q\le Q}\ \sum_{a\in(\mathbb{Z}/q\mathbb{Z})^*}\left|\pi(x;q,a) - \frac{\pi(x)}{\varphi(q)}\right|^2 \ll_A \frac{xQ}{\log x}.$$

Finally, we will also need a few classical facts regarding Euler's totient function (for example, [14, Th. 328 & 330]).

Lemma E. *The following asymptotic estimates hold:*

$$\varphi(q) \gg \frac{q}{\log\log q}, \tag{4}$$

$$\Phi(x) := \sum_{q\le x}\varphi(q) = \frac{3x^2}{\pi^2} + O(x\log x). \tag{5}$$

Algorithm 1. Our basic algorithm

```
Fix $q \propto x^{1-\varepsilon}$
$a \xleftarrow{\$} (\mathbb{Z}/q\mathbb{Z})^*$        ▷ considered as an element of $\{1, \dots, q-1\}$
repeat forever
    $t \xleftarrow{\$} \{0, \dots, \lfloor \frac{x-a}{q} \rfloor\}$
    $p \leftarrow a + t \cdot q$
    if $p$ is prime then return $p$
end repeat
```

3 Close-to-Uniform Prime Number Generation with Fewer Random Bits

3.1 Basic Algorithm

A simple method to construct obviously uniformly distributed prime numbers up to x is to pick random numbers in $\{1, \dots, \lfloor x \rfloor\}$ and retry until a prime is found. However, this method consumes $\log_2 x$ bits of randomness per iteration (not counting the amount of randomness consumed by primality testing), and hence an expected amount of $(\log x)^2/\log 2$ bits of randomness to produce a prime, which is quite large.

As mentioned in the introduction, we propose the following algorithm to generate almost uniform primes while consuming fewer random bits: first fix an integer

$$q \propto x^{1-\varepsilon} \tag{6}$$

and pick a random $a \in (\mathbb{Z}/q\mathbb{Z})^*$. Then, search for prime numbers $\leq x$ of the form $p = a + t \cdot q$. This method, described as Algorithm 1, only consumes $\log_2 t = \varepsilon \log_2 x$ bits of randomness per iteration, and the probability of success at each iteration is $\sim \frac{\pi(x;q,a)}{x/q}$. Assuming that Conjecture B is true, which ensure that (3) holds in the range (6), this probability is about $q/\big(\varphi(q)\log x\big)$, and the algorithm should thus consume roughly:

$$\varepsilon \cdot \frac{\varphi(q)}{q} \cdot \frac{(\log x)^2}{\log 2} \tag{7}$$

bits of randomness on average: much less than the trivial algorihm. Moreover, we can also show, under the same assumption, that the output distribution is statistically close to uniform and has close to maximal entropy.

We establish those results in §3.2, and show in §3.3 that Turán's theorem can be used to obtain nearly the same results under the Extended Riemann Hypothesis. ERH is not sufficient to prove that Algorithm 1 terminates almost surely, or to bound the expectation of the number of random bits it consumes, due to the possibly large contribution of negligibly few values of a. We can avoid these problems by modifying the algorithm slightly, as discussed in §3.4. Finally, in §3.5, we show that unconditional results of the same type can be obtained using the Barban–Davenport–Halberstam theorem, for another slightly different variant of the algorithm.

Before turning to these analyses, let us make a couple of remarks on Algorithm 1. First, note that one is free to choose q in any convenient way in the range (6). For example, one could choose q as the largest power of 2 less than $x^{1-\varepsilon}$, so as to make Step 2 very easy. It is preferable, however, to choose q as a (small multiple of a) primorial, to minimize the ratio $\varphi(q)/q$, making it as small as $\propto 1/\log\log q \sim 1/\log\log x$; this makes the expected number of iterations and the expected amount (7) of consumed randomness substantially smaller. In that case, Step 2 becomes slightly more complicated, but this is of no consequence.

Indeed, our second observation is that Step 2 is always negligible in terms of running time and consumed randomness compared to the primality testing loop that follows. Indeed, even the trivial implementation (namely, pick a random $a \in \{0, \dots, q-1\}$ and try again if $\gcd(a, q) \neq 1$) requires $q/\varphi(q) \ll \log\log q$ iterations on average. It is thus obviously much faster than the primality testing loop, and consumes $\ll \log x \log\log x$ bits of randomness, which is negligible compared to (7). Furthermore, an actual implementation would take advantage of the known factorization of q and use a unit generation algorithm such as the one proposed by Joye and Paillier [15], which we can show requires only $O(1)$ iterations on average.

Finally, while we will not discuss the details of the primality test of Step 6, and shall pretend that it returns exact results (as the AKS algorithm [1] would, for example), we note that it is fine (and in practice preferable) to use a probabilistic compositeness test such as Miller–Rabin [23] instead, provided that the number of rounds is set sufficiently large as to make the error probability negligible. Indeed, the output distribution of our algorithm then stays statistically close to uniform, and the number of iterations is never larger.

3.2 Analysis under the Friedlander–Granville–Montgomery Conjecture

As mentioned above, it is straightforward to deduce from the Friedlander–Granville–Montgomery conjecture that Algorithm 1 terminates almost surely, and to bound its expected number of iterations and amount of consumed randomness.

Theorem 3.2.1. *Assume that Conjecture B holds. Then Algorithm 1 terminates almost surely, requires $(1 + o(1))\varphi(q)/q \cdot \log x$ iterations of the main loop on average, and consumes $\big(\varepsilon + o(1)\big) \cdot \frac{\varphi(q)}{q} \cdot \frac{(\log x)^2}{\log 2}$ bits of randomness on average.*

Proof. Indeed, fix $q \propto x^{1-\varepsilon}$. Conjecture B implies, uniformly over $a \in (\mathbb{Z}/q\mathbb{Z})^*$:

$$\left|\pi(x; q, a) - \frac{\pi(x)}{\varphi(q)}\right| \ll (x/q)^{1/2} \cdot x^{\varepsilon/4} \propto x^{3\varepsilon/4},$$

which is negligible compared to $\pi(x)/\varphi(q) \gg x^{\varepsilon}/\log x$. As a result, we get $\pi(x;q,a) = (1+o(1))\pi(x)/\varphi(q) = (1+o(1))/\varphi(q) \cdot x/\log x$ uniformly over a, and the success probability of the main loop becomes:

$$\frac{\pi(x;q,a)}{1+\lfloor\frac{x-a}{q}\rfloor} = \frac{q}{\varphi(q)} \cdot \frac{1+o(1)}{\log x}$$

which implies the stated results immediately. □

Now let X be the output distribution of Algorithm 1, i.e. the distribution on the set of prime numbers $\leq x$ such that Algorithm 1 outputs a prime p with probability exactly $\Pr[X=p]$. Clearly, we have, for all $(a,q)=1$ and all t such that $a+t\cdot q \leq x$ is prime:

$$\Pr[X = a+t\cdot q] = \frac{1}{\varphi(q)} \cdot \frac{1}{\pi(x;q,a)}.$$

As a result, the squared Euclidean imbalance of X is:

$$\begin{aligned}
\Delta_2^2(X) &= \sum_{a\in(\mathbb{Z}/q\mathbb{Z})^*} \sum_{a+tq\leq x \text{ prime}} \left|\Pr[X=a+tq] - \frac{1}{\pi(x)}\right|^2 + \sum_{p|q} \frac{1}{\pi(x)^2} \\
&= \sum_{a\in(\mathbb{Z}/q\mathbb{Z})^*} \pi(x;q,a)\left|\frac{1}{\varphi(q)} \cdot \frac{1}{\pi(x;q,a)} - \frac{1}{\pi(x)}\right|^2 + \sum_{p|q}\frac{1}{\pi(x)^2} \\
&= \frac{1}{\pi(x)^2} \sum_{a\in(\mathbb{Z}/q\mathbb{Z})^*} \frac{1}{\pi(x;q,a)}\left|\pi(x;q,a) - \frac{\pi(x)}{\varphi(q)}\right|^2 + \sum_{p|q}\frac{1}{\pi(x)^2} \\
&\ll \frac{1}{\pi(x)^2} \sum_{a\in(\mathbb{Z}/q\mathbb{Z})^*} \frac{\log x}{x^{\varepsilon}} \cdot x^{3\varepsilon/2} \ll \frac{\log^3 x}{x^2} \cdot \varphi(q) x^{\varepsilon/2} \ll \frac{\log^3 x}{x^{1+\varepsilon/2}} \ll \frac{1}{x^{1+\varepsilon/3}}.
\end{aligned}$$

We can then deduce the following.

Theorem 3.2.2. *Assume that Conjecture B holds. Then the output distribution of Algorithm 1 is statistically close to uniform, and its collision entropy is only negligibly smaller than that of the uniform distribution.*

Proof. Indeed, by (2), the statistical distance to the uniform distribution satisfies:

$$\Delta_1(X) \leq \Delta_2(X)\sqrt{\pi(x)} \ll \frac{1}{x^{1/2+\varepsilon/6}}\sqrt{\frac{x}{\log x}} \ll x^{-\varepsilon/6},$$

which is negligible. Moreover, the collision probability is:

$$\beta(X) = \frac{1}{\pi(x)} + \Delta_2^2(X) = \frac{1}{\pi(x)}\left(1 + O\Big(\frac{\pi(x)}{x^{1+\varepsilon/3}}\Big)\right) = \frac{1}{\pi(x)}\left(1+o\big(x^{-\varepsilon/3}\big)\right).$$

Hence:

$$H_2(X) = \log_2\big(\pi(x)\big) - \log_2\big(1+o(x^{-\varepsilon/3})\big) = (H_2)_{\max} - o(x^{-\varepsilon/3})$$

as required. □

3.3 Analysis under the Extended Riemann Hypothesis

Assume the Extended Riemann Hypothesis, and denote by α the fraction of all possible choices of $a \in (\mathbb{Z}/q\mathbb{Z})^*$ such that the error term $E(x;q,a) := \left|\pi(x;q,a) - \pi(x)/\varphi(q)\right|$ satisfies $E(x;q,a) > x^{3\varepsilon/4}$. Then, Turán's theorem asserts that:

$$\sum_{a \in (\mathbb{Z}/q\mathbb{Z})^*} E(x;q,a)^2 \ll x(\log x)^2,$$

and the left-hand side is greater or equal to $\alpha\varphi(q) \cdot x^{3\varepsilon/2}$ by definition of α. As a result, we get:

$$\alpha \ll \frac{x^{1-3\varepsilon/2}(\log x)^2}{\varphi(q)} \ll \frac{(\log x)^2 \log\log x}{x^{\varepsilon/2}}$$

and hence α is negligible. Therefore, for all except at most a negligible fraction of choices of $a \in (\mathbb{Z}/q\mathbb{Z})^*$, we obtain that $E(x;q,a) \le x^{3\varepsilon/4}$, and since $\pi(x)/\varphi(q) \gg x^{\varepsilon}/\log x$, this implies $\pi(x;q,a) = (1+o(1))\pi(x)/\varphi(q)$ as before. As a result, under ERH, we obtain an analogue of Theorem 3.2.1 valid with overwhelming probability on the choice of a.

Theorem 3.3.1. *Assume ERH holds. Then Algorithm 1 terminates with overwhelming probability. Moreover, except for a negligible fraction of choices of the class* $a \bmod q$, *it requires* $(1+o(1))\varphi(q)/q \cdot \log x$ *iterations of the main loop on average, and consumes* $\big(\varepsilon + o(1)\big) \cdot \frac{\varphi(q)}{q} \cdot \frac{(\log x)^2}{\log 2}$ *bits of randomness on average.*

Moreover, in the full version of this paper [10], using Turán's theorem and the Cauchy–Schwarz inequality, we also establish under ERH alone the following analogue of Theorem 3.2.2, regarding the output distribution of the algorithm.

Theorem 3.3.2. *Assume ERH holds. Then the output distribution of Algorithm 1 is statistically close to uniform, and its collision entropy is no more than* $O(\log\log x)$ *bits smaller than that of the uniform distribution.*

3.4 Achieving Almost Sure Termination under ERH

Theorem 3.3.1 above is somewhat unsatisfactory, as we have to ignore a negligible but possibly nonzero fraction of all values $a \bmod q$ to obtain a bound on the average number of iterations and on the randomness consumed by Algorithm 1 under ERH. But this is unavoidable for that algorithm: as mentioned above, it is not known whether ERH implies that for $q \propto x^{1-\varepsilon}$, all $a \in (\mathbb{Z}/q\mathbb{Z})^*$ satisfy $\pi(x;q,a) \neq 0$. And if an a exists such that $\pi(x;q,a) = 0$, the choice of that a in Step 2 of Algorithm 1, however unlikely, is a case of non-termination: as a result, the existence of such an a prevents any nontrivial bound on average running time or average randomness.

We propose to circumvent that problem by falling back to the trivial algorithm (pick a random $p < x$, check whether it is prime and try again if not) in case too

Algorithm 2. A variant which terminates almost surely under ERH

```
Fix $q \propto x^{1-\varepsilon}$
$a \xleftarrow{\$} (\mathbb{Z}/q\mathbb{Z})^*$        ▷ considered as an element of $\{1, \dots, q-1\}$
repeat $T = \log^2 x$ times
    $t \xleftarrow{\$} \{0, \dots, \lfloor \frac{x-a}{q} \rfloor\}$
    $p \leftarrow a + t \cdot q$
    if $p$ is prime then return $p$
end repeat
repeat forever
    $p \xleftarrow{\$} \{1, \dots, \lfloor x \rfloor\}$
    if $p$ is prime then return $p$
end repeat
```

many iterations of the main loop have been carried out. This variant is presented as Algorithm 2.

Clearly, since Algorithm 2 is the same as Algorithm 1 except for the possible fallback to the trivial algorithm, which has a perfectly uniform output distribution, the output distribution of the variant is at least as close to uniform as the original algorithm. In other words, the analogue of Theorem 3.3.2 holds, with the same proof.

Theorem 3.4.1. *Assume ERH holds. Then the output distribution of Algorithm 2 is statistically close to uniform, and its collision entropy is no more than $O(\log\log x)$ bits smaller than that of the uniform distribution.*

Moreover, as claimed above, we can obtain the following stronger analogue of Theorem 3.3.1, proved in the full version of this paper [10].

Theorem 3.4.2. *Assume ERH holds. Then Algorithm 2 terminates almost surely, requires $(1+o(1))\varphi(q)/q \cdot \log x$ iterations of the main loop on average, and consumes $\big(\varepsilon + o(1)\big) \cdot \frac{\varphi(q)}{q} \cdot \frac{(\log x)^2}{\log 2}$ bits of randomness on average.*

3.5 An Unconditional Algorithm

Finally, we propose yet another variant of our algorithm for which both almost sure termination and uniformity bounds can be established unconditionally. The idea is to no longer use a fixed modulus q, but to pick it uniformly at random instead in the range $\{1, \dots, Q\}$ where $Q \propto x(\log x)^{-A}$; uniformity bounds can then be deduced from the Barban–Davenport–Halberstam theorem. Unfortunately, since Q is only polynomially smaller than x, we can no longer prove that the output distribution is statistically close to uniform: the statistical distance is polynomially small instead, with an arbitrarily large exponent depending only on the constant A. On the other hand, termination is obtained as before by falling back to the trivial algorithm after a while, and since q is often very close to x, we get an even better bound on the number of consumed random bits. Details are provided in the full version of this paper [10].

4 Comparison with other Prime Number Generation Algorithms

In the full version of this paper [10], we present a detailed discussion of how our method compares with other prime number generation algorithms, and specifically Brandt and Damgård's PRIMEINC [3] as well as Maurer's algorithm [17,18]. Our main observations are as follows.

- Brandt and Damgård analyze their PRIMEINC algorithm and prove a lower bound on its output entropy under the prime r-tuple conjecture of Hardy and Littlewood. We show, however, that if that conjecture holds, then the output distribution of PRIMEINC is quite far from uniform. The statistical distance Δ_1 satisfies $\Delta_1 > 0.86 + o(1)$.
- On the other hand, Maurer's algorithm has an output distribution that is arguably close to uniform (although it is somewhat difficult to quantify *how* close), but it consumes a large amount of random bits. In particular, we prove that it consumes at least $c(\log x)^2$ random bits for some absolute constant $c > 0$, so about as many as the trivial algorithm. Compared to our method, however, it has the significant advantage of producing primality certificates.

References

1. Agrawal, M., Kayal, N., Saxena, N.: PRIMES is in P. Ann. Math. 160(2), 781–793 (2004)
2. Barban, M.B.: The "large sieve" method and its application to number theory. Uspehi Mat. Nauk 21, 51–102 (1966)
3. Brandt, J., Damgård, I.: On generation of probable primes by incremental search. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 358–370. Springer, Heidelberg (1993)
4. Brandt, J., Damgård, I., Landrock, P.: Speeding up prime number generation. In: Matsumoto, T., Imai, H., Rivest, R.L. (eds.) ASIACRYPT 1991. LNCS, vol. 739, pp. 440–449. Springer, Heidelberg (1993)
5. Cramer, R., Shoup, V.: Signature schemes based on the strong RSA assumption. ACM Trans. Inf. Syst. Secur. 3(3), 161–185 (2000)
6. Davenport, H.: Multiplicative Number Theory, 2nd edn. Graduate Texts in Mathematics, vol. 74. Springer (1980)
7. Davenport, H., Halberstam, H.: Primes in arithmetic progressions. Michigan Math. J. 13, 485–489 (1966)
8. De la Vallée Poussin, C.-J.: Recherches analytiques sur la théorie des nombres premiers. Ann. Soc. Sci. Bruxelles 20, 281–397 (1896)
9. Eastlake, D., Schiller, J., Crocker, S.: Randomness Requirements for Security. RFC 4086 (Best Current Practice) (June 2005)
10. Fouque, P.-A., Tibouchi, M.: Close to uniform prime number generation with fewer random bits. arXiv.org e-Print archive (2014), Full version of this paper
11. Friedlander, J.B., Granville, A.: Limitations to the equi-distribution of primes I. Ann. Math. 129, 363–382 (1989)
12. Gallagher, P.X.: The large sieve. Mathematika 14(1), 14–20 (1967)

13. Hardy, G.H., Littlewood, J.E.: Some problems of 'partitio numerorum': III. on the expression of a number as a sum of primes 44, 1–70 (1922)
14. Hardy, G.H., Wright, E.M.: An Introduction to the Theory of Numbers, 4th edn. Clarendon Press (1960)
15. Joye, M., Paillier, P.: Fast generation of prime numbers on portable devices: An update. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 160–173. Springer, Heidelberg (2006)
16. Joye, M., Paillier, P., Vaudenay, S.: Efficient generation of prime numbers. In: Koç, Ç.K., Paar, C. (eds.) CHES 2000. LNCS, vol. 1965, pp. 340–354. Springer, Heidelberg (2000)
17. Maurer, U.M.: Fast generation of secure RSA-moduli with almost maximal diversity. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 636–647. Springer, Heidelberg (1990)
18. Maurer, U.M.: Fast generation of prime numbers and secure public-key cryptographic parameters. J. Cryptology 8(3), 123–155 (1995)
19. Mihăilescu, P.: Fast generation of provable primes using search in arithmetic progressions. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 282–293. Springer, Heidelberg (1994)
20. Mihăilescu, P.: Security of biased sources for cryptographic keys. In: Cryptography and computational number theory (Singapore, 1999). Progr. Comput. Sci. Appl. Logic, vol. 20, pp. 287–302. Birkhäuser, Basel (2001)
21. Montgomery, H.L.: Topics in Multiplicative Number Theory. Lecture Notes in Mathematics, vol. 227. Springer (1971)
22. Montgomery, H.L.: Problems concerning prime numbers. Proc. Symp. Pure Math. 28, 307–310 (1976)
23. Rabin, M.: Probabilistic algorithms for testing primality 12, 128–138 (1980)
24. Shoup, V.: A Computational Introduction to Number Theory and Algebra (Version 2). Cambridge University Press (2008)
25. Turán, P.: Über die Primzahlen der arithmetischen Progression. Acta Sci. Math. Szeged 8(4), 226–235 (1936)
26. Walfisz, A.: Zur additiven Zahlentheorie II. Mathematische Zeitschrift 40(1), 592–607 (1936)

Optimal Strong Parallel Repetition for Projection Games on Low Threshold Rank Graphs

Madhur Tulsiani[1,*], John Wright[2], and Yuan Zhou[2,**]

[1] TTI Chicago
[2] Carnegie Mellon University
madhurt@ttic.edu, {jswright,yuanzhou}@cs.cmu.edu

Abstract. Given a two-player one-round game G with value $\mathrm{val}(G) = (1-\eta)$, how quickly does the value decay under parallel repetition? If G is a projection game, then it is known that we can guarantee $\mathrm{val}(G^{\otimes n}) \leq (1-\eta^2)^{\Omega(n)}$, and that this is optimal. An important question is under what conditions can we guarantee that *strong* parallel repetition holds, i.e. $\mathrm{val}(G^{\otimes}) \leq (1-\eta)^{\Omega(n)}$?

In this work, we show a strong parallel repetition theorem for the case when G's constraint graph has low threshold rank. In particular, for any $k \geq 2$, if σ_k is the k-th largest singular value of G's constraint graph, then we show that

$$\mathrm{val}(G^{\otimes n}) \leq \left(1 - \frac{\sqrt{1-\sigma_k^2}}{k} \cdot \eta\right)^{\Omega(n)} .$$

This improves and generalizes upon the work of [RR12], who showed a strong parallel repetition theorem for the case when G's constraint graph is an expander.

1 Introduction

A *two-prover one-round game* G is given by two sets of questions U, V, a distribution on $U \times V$, a set of possible answers Σ and a set of predicates $\Pi = \{\pi_{u,v}\}_{u \in U, v \in V}$. A verifier samples a pair of questions (u, v) according to the given distribtion, and gives u to the first prover and v is given to the second prover. The provers respond with answers $f(u), g(v) \in \Sigma$. The verifier accepts the answers if they satisfy the predicate $\pi_{u,v}(f(u), g(v))$ associated to the pair of questions (u, v). The value of the game, denoted by $\mathrm{val}(G)$, is defined to the maximum probability with which the provers can make the verifier accept.

We say G is a *projection game* if for every predicate $\pi_{u,v}$ and every $\beta \in \Sigma$, there is at most one $\alpha \in \Sigma$ such that $\pi_{u,v}(\alpha, \beta)$ is satisfied. The *constraint*

[*] Research supported by NSF Career Award CCF-1254044.

[**] Supported in part by a grant from the Simons Foundation (Award Number 252545). Work done in part when visiting TTI-Chicago.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 1003–1014, 2014.

graph of G is the bipartite graph H with vertex set $U \cup V$ corresponding to the distribution of questions. See Section 2 for the precise definitions.

Perhaps the most fundamental result in the area of two-prover one-round games is the parallel repetition theorem of Raz [Raz98]. In its version for projection games by Rao [Rao11], it states:

Theorem 1. *Let G be a projection game. If* $\mathrm{val}(G) \leq 1 - \eta$*, then* $\mathrm{val}(G^{\otimes n}) \leq (1 - \eta^2)^{\Omega(n)}$.

Stated contrapositively, if $\mathrm{val}(G^{\otimes n}) \geq (1 - \eta)^n$, then $\mathrm{val}(G) \geq 1 - O(\sqrt{\eta})$. Naively, one would expect a faster rate of decay: that is, if $\mathrm{val}(G) \leq 1 - \eta$, then $\mathrm{val}(G^{\otimes n})$ should satisfy $\mathrm{val}(G^{\otimes n}) \leq (1 - \eta)^{\Omega(n)}$. A parallel repetition bound of this form is known as *strong* parallel repetition, and it was open whether a strong parallel repetition theorem was true in general for projection games, or even for interesting restricted classes of games (e.g. Unique Games; see [FKO07]), until Raz [Raz11] showed that strong parallel repetition fails on the so-called Odd Cycle Game. In particular, he showed that $\mathrm{val}(G^{\otimes n})$ for the Odd Cycle Game matches exactly the upper bound given in Theorem 1.

One interesting subclass not covered by the Odd Cycle Game is the class of *expanding* games. This is the class of two-player one-round games G in which the constraint graph of G is an expander. This is an interesting class of two-player one-round games which appears frequently in the hardness of approximation literature (see [AKK$^+$08] and [SS07] for example). Following the papers of [AKK$^+$08] and [BRR$^+$09], which proved strong parallel repetition theorems for certain special cases of expanding games, Raz and Rosen [RR12] showed the following strong parallel repetition theorem for expanding projection games:

Theorem 2. *Let G be a projection game, and let σ_2 be the second largest singular value of G's constraint graph. If* $\mathrm{val}(G) \leq (1 - \eta)$*, then* $\mathrm{val}(G^{\otimes n}) \leq (1 - (1 - \sigma_2)^2 \cdot \eta)^{\Omega(n)}$.

In particular, if σ_2 is a constant, then $\mathrm{val}(G^{\otimes n})$ decays like $(1 - \eta)^{\Omega(n)}$. This result motivates a natural question, for which we need the following defintion.

Definition 1. *Let G be a two-prover one-round game with constraint graph H. Suppose the singular values of H are $\sigma_1 \geq \sigma_2 \geq \ldots \geq 0$. Then the τ-*threshold rank *of G is the number of singular values σ_i of H for which $\sigma_i \geq \tau$.*

Thus, at a high level, a graph with *low threshold rank* is one whose k-th largest singular value σ_k is bounded away from 1, where k is small (say, a constant). It is natural to ask if a strong parallel repetition theorem hold for graphs with low threshold rank?

1.1 Parallel Repetition and Cheeger's Inequality

In this work, we answer this question using the new parallel repetition framework of [DS14]. They study a relaxation to the value of the game, called $\mathrm{val}_+(G)$, which, roughly speaking, is the best value achieved on the game by a *distribution* over "fractional assignments". These "fractional assignments" are required to collectively look like a valid assignment, but individually they may look very different than a valid assignment. This relaxation $\mathrm{val}_+(G)$ enjoys several nice analytic properties: for example, it upper-bounds $\mathrm{val}(G)$, and it is multiplicative under parallel repetition. From here, they give a new proof of Theorem 1 using the following three-step process:

1. Supposing $\mathrm{val}(G^{\otimes n}) \geq (1-\eta)^n$, then $\mathrm{val}_+(G) \geq (1-\eta)$. This gives a distribution of "fractional assignments" with "fractional value" at least $(1-\eta)$.
2. Round fractional assignments to a 0/1-assignments using Cheeger rounding.
3. Combine these 0/1-assignments into a single 0/1 assignment using the correlated sampling approach of [BHH$^+$08].

Supposing that $\mathrm{val}(G^{\otimes n}) \geq (1-\eta)^n$, then this will produce a solution to G with value $(1 - O(\sqrt{\eta}))$.

This proof has revealed a strong connection between the parallel repetition theorem and Cheeger's inequality. We begin with some necessary definitions. Given a d-regular graph $H = (V, E)$, the *conductance* of a set $S \subseteq V$ is

$$\phi(S) = \frac{|E(S, \overline{S})|}{d \cdot \min\{|S|, |\overline{S}|\}},$$

where $E(S, \overline{S})$ is the set of edges in H which cross from S to $\overline{S}$. The conductance of H is defined to be $\phi(H) := \min_{S \subset V}(\phi(S))$. Cheeger's inequality states:

Cheeger's Inequality. *Given a graph $H = (V, E)$, let λ_2 be the second-smallest eigenvalue of its normalized Laplacian. Then*

$$\frac{\lambda_2}{2} \leq \phi(G) \leq \sqrt{2\lambda_2}.$$

The upper-bound $\phi(G) \leq \sqrt{2\lambda_2}$ is shown by taking the second-largest eigenvector v_2, which has a "fractional conductance" of λ_2, and using a rounding algorithm (which we refer to as Cheeger rounding) to produce a set S of conductance at most $\sqrt{2\lambda_2}$. This is the same rounding used in step 2 of the parallel repetition proof, and is the reason that we can only prove $\mathrm{val}(G) \geq (1 - \sqrt{\eta})$ if $\mathrm{val}_+(G) \geq (1 - \eta)$. In both Cheeger's inequality and in the parallel repetition theorem, we "lose a square root" for the same reason.

Thus, to prove a strong parallel repetition theorem, it seems we need a "strong" Cheeger's inequality, one that rounds vectors of "fractional conductance" η to sets of conductance $O(\eta)$. Many recent works have shown how to modify/improve Cheeger's inequality to account for the higher eigenvalues of the graph (for example, see [LOGT12] and [LRTV12]). One of these is indeed a "strong" Cheeger's inequality for the case when the graph has low threshold rank [KLL$^+$13]:

Theorem 3. *Given a graph $H = (V, E)$, let $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{|V|}$ be the eigenvalues of its normalized Laplacian. Then for every $k \geq 2$,*

$$\phi(H) \leq O(k) \cdot \frac{\lambda_2}{\sqrt{\lambda_k}}.$$

In particular, if λ_k is large for a constant k (in other words, if H has low threshold rank), then an eigenvector of H with eigenvalue λ_2 can be rounded into a cut of sparsity $\approx \lambda_2$. This theorem gives hope for a strong parallel repetition theorem for the low threshold rank case.

1.2 Our Results

We may now state our main theorem.

Theorem 4. *Let G be a biregular projection game, and let its constraint graph have singular values $1 = \sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{|V|}$. For any $k \geq 2$, if* $\mathrm{val}(G) \leq (1-\eta)$, *then*

$$\mathrm{val}(G^{\otimes n}) \leq \left(1 - \frac{\sqrt{1-\sigma_k^2}}{k} \cdot \eta\right)^{\Omega(n)} .$$

At a high level, our proof of this comes from combining Theorem 3 with the parallel repetition framework of [DS14]. However, the interface between these two isn't clean, and some care must be taken in combining them. Furthermore, we can't just apply Theorem 3 all at once; instead, we have to wait to apply its various components at the appropriate time (at a high level, this is because whereas the normal Cheeger rounding preservers marginals, this higher-order Cheeger rounding only *approximately* preserves marginals).

In addition, we give another proof of strong parallel repetition for the special case of expanding games. We show that

$$\mathrm{val}(G^{\otimes n}) \leq \left(1 - (1-\sigma_2^2) \cdot \eta\right)^{\Omega(n)} .$$

This is stronger than the bound from [RR12], though it is weaker than the $k = 2$ case of Theorem 4. However, the proof is much simpler. Independently, David Steurer [Ste13] also proved a strong parallel repetition theorem for expanding games in the framework of [DS14].

1.3 Tightness of Our Results

The dependence on k in Theorem 4 is tight, as certified by the Odd Cycle Game.

Let $k \geq 0$, and set $m := 2k + 1$. Let $G_m = (V_m, E_m)$ be the cycle on m vertices. In the Odd Cycle Game, the two players $\mathsf{P1}$ and $\mathsf{P2}$ try to convince the verifier that the graph G_m is 2-colorable. Formally, each player Pi is given as questions the vertices in V_m, and verifier expects answers $b_1, b_2 \in \{0, 1\}$. The constraints are distributed as follows:

- With probability 1/2, both players are given a random vertex $v \in V_m$. The constraint is that $b_1 = b_2$.
- With probability 1/2, an edge $(v_1, v_2) \in E_m$ is selected uniformly at random; $\mathsf{P1}$ is given v_1, and $\mathsf{P2}$ is given v_2. The constraint is that $b_1 \neq b_2$.

It is easy to see that $\mathrm{val}(G) = 1 - \frac{1}{2m}$. In [Raz11], Raz lower bounded the value of G under parallel repetition:

Theorem 5. *For large enough n,* $\mathrm{val}(G^{\otimes n}) \geq \left(1 - \frac{1}{m^2}\right)^{O(n)}$.

Let us now calculate the bound our Theorem 4 gives for the Odd Cycle Game. The constraint graph of the Odd Cycle Game is w/prob. 1/2 the identity mapping and w/prob. 1/2 a random step on the cycle graph. As the cycle graph has eigenvalues $\cos(2\pi k/m)$ for each $k \in \{0, \ldots, m-1\}$ (see, for example, [Tre11]) the constraint graph of the Odd Cycle Game has singular values $1/2 + \cos(2\pi k/m)/2$ for each $k \in \{0, \ldots, m-1\}$. For small values of k, this means that the k-th largest singular value of the constraint graph is $\approx 1 - \left(\frac{k}{m}\right)^2$. Plugging this into our Theorem 4, we see that it gives $\mathrm{val}(G^{\otimes n}) \leq \left(1 - \frac{1}{m^2}\right)^{\Omega(n)}$, exactly matching Theorem 5.

2 Preliminaries

2.1 Two-Prover One-Round Games

A *two-prover one-round game G* consists of a bipartite graph $H = (U, V, E)$ (known as the *constraint graph*) and a set Σ of answers (or labels). We will reserve $\alpha \in \Sigma$ to refer to labels on the U side and $\beta \in \Sigma$ to refer to labels on the V side. Each edge $(u, v) \in E$ is associated with a predicate $\pi_{uv} : \Sigma \times \Sigma \to \{0, 1\}$. We will write $(u, v) \sim E$ to denote that the edge (u, v) is sampled uniformly at random from E. The game, with two provers P1 and P2, is played as follows:

- Sample $(u, v) \sim E$. Give u to P1 and v to P2.
- Receive answers α from P1 and β from P2. Accept iff $\pi_{uv}(\alpha, \beta) = 1$.

We can associate the responses of player P1 with an *assignment* $f : U \to \Sigma$ and the responses of player P2 with an assignment $g : V \to \Sigma$. This gives us our main definition:

Definition 2. *Given a two-prover one-round game G, the* value *of two assignments $f : U \to \Sigma$ and $g : V \to \Sigma$ is* $\mathrm{val}(f, g; G) := \mathbf{Pr}_{(u,v)\sim E}[\pi_{uv}(f(u), g(v)) = 1]$. *The value of the game G is* $\mathrm{val}(G) := \max_{f,g} \mathrm{val}(f, g; G)$.

Rather than focusing on general two-player one-round games, we will focus on the special case of *biregular projection games*. G is saidf to be a *projection game* if for every predicate π_{uv} and every $\beta \in \Sigma$, there is at most one $\alpha \in \Sigma$ such that $\pi_{uv}(\alpha, \beta) = 1$. G is *biregular* if its constraint graph $H = (U, V, E)$ is biregular, i.e. if each vertex $u \in U$ has the same degree d_U, and each vertex $v \in V$ has the same degree d_V. Given two two-prover one-round games G_1 and G_2, we denote the *parallel repetition* of G_1 and G_2 by $G_1 \otimes G_2$. This is the two-prover one-round game which is played as follows:

1. Sample $(u_1, v_1) \sim G_1$ and $(u_2, v_2) \sim G_2$.
2. Give question (u_1, u_2) to P1 and (v_1, v_2) to P2.
3. Receive answers (α_1, α_2) and (β_1, β_2).
4. Accept iff $\pi_{u_1v_1}(\alpha_1, \beta_1) = \pi_{u_2v_2}(\alpha_2, \beta_2) = 1$.

We will write $G^{\otimes k}$ to denote the k-fold parallel repetition of G with itself.

2.2 Vectors

Definition 3. *Given an index set $\mathcal{J}$, a* vector *is a function $f : \mathcal{J} \to \mathbb{R}$.*

Vectors in this paper will be indexed in one of three ways: either by vertices $v \in V$, by labels $\beta \in \Sigma$, or by vertex/label pairs $(v, \beta) \in V \times \Sigma$. Given two vectors $f, g : \mathcal{J} \to \mathbb{R}$, their inner product is defined as:

$$\langle f, g \rangle := \sum_{x \in \mathcal{J}} f(x) \cdot g(x) = f^\top g.$$

(We note that this differs from the way [DS14] defines their inner products. There, inner products between vectors with index set V or $V \times \Sigma$ use an expectation over $v \sim V$ rather than a summation.) We will also define the 2-norm of a vector as: $\|f\| := \sqrt{\langle f, f \rangle}$.

The following types of vectors will be especially important for us in this paper.

Definition 4. *As above, given a vertex set V and a label set Σ, an* assignment *is a function $f : V \to \Sigma$. We will associate with this assignment the vector $f : V \times \Sigma \to \mathbb{R}$ (which we will also call an assignment) for which*

$$f(v, \beta) := \begin{cases} \frac{1}{\sqrt{|V|}} & \textit{if } f(v) = \beta, \\ 0 & \textit{otherwise.} \end{cases}$$

A partial assignment *is a vector $f : V \times \Sigma \to \left\{0, \frac{1}{\sqrt{|V|}}\right\}$ in which for each $v \in V$, $f(v, \beta)$ is nonzero for at most one $\beta \in \Sigma$. a* fractional assignment *is a vector $f : V \times \Sigma \to \mathbb{R}^{\geq 0}$ in which for each $v \in V$, $f(v, \beta)$ is nonzero for at most one $\beta \in \Sigma$.*

Fractional assignments admit the following decomposition, which we will use repeatedly:

Fact 6. *Suppose $g : V \times \Sigma \to \mathbb{R}$ is a fractional assignment. Then we can write $g(v, \beta) = h(v) \cdot I(v, \beta)$, where $h : V \to \mathbb{R}^{\geq 0}$ and $I : V \times \Sigma \to \{0, 1\}$ is a $0/1$-indicator. For convenience, we will require that for each $v \in V$, there exists a $\beta \in \Sigma$ such that $I(v, \beta) = 1$.*

It is also easy to see that $\|h\|^2 = \|g\|^2$.

Definition 5. *Given $f : V \times \Sigma \to \mathbb{R}$ and a vertex $v \in V$, we will write $f(v, \cdot)$ for the vector mapping $\Sigma \to \mathbb{R}$ which, on input $\beta \in \Sigma$, outputs $f(v, \beta)$.*

2.3 The Projection Game Operator

Let G be a two-prover one-round game with constraint graph $H = (U, V, E)$. As before, we will write d_U for the degree of the U vertices and d_V for the degree of the V vertices. We will associate with G and H a pair of matrices which allow us to analyze the parallel repetition of our games via linear algebra.

To H we associate the $|U| \times |V|$ matrix (also named H) defined as

$$H_{u,v} := \begin{cases} \frac{1}{\sqrt{d_U \cdot d_V}} & \text{if } u \sim v, \\ 0 & \text{otherwise.} \end{cases}$$

This is just the normalized adjacency matrix of H. To G we associate the $|U \times \Sigma| \times |V \times \Sigma|$ matrix (also named G) defined as

$$G_{(u,\alpha),(v,\beta)} := \begin{cases} \frac{1}{\sqrt{d_U \cdot d_V}} & \text{if } u \sim v \text{ and } \pi_{u,v}(\alpha,\beta) = 1, \\ 0 & \text{otherwise.} \end{cases}$$

This is the *projection game* operator. The matrix we use for H is a natural choice, and the next proposition shows that the definition of the G matrix is natural as well.

Proposition 1. *Suppose f and g are assignments. Then $\langle f, Gg\rangle = f^\top Gg = \mathrm{val}(f, g; G)$.*

A primary focus for us in this paper will be the constraint graph H and its spectrum. For this, we use the singular value decomposition.

Singular Value Decomposition. *Suppose the rank of H is d. Then we can write*

$$H = \sum_{i=1}^{d} \sigma_i \cdot l_i \cdot r_i^\top . \tag{1}$$

Here, $\sigma_1 \geq \ldots \geq \sigma_d \geq 0$ are the singular values *of H, $l_1, \ldots, l_d$ are an orthonormal set of vectors called the* left-singular vectors, *and $r_1, \ldots, r_d$ are an orthonormal set of vectors called the* right-singular vectors. *Furthermore, the l_i's are of type $U \to \mathbb{R}$, and the r_i's are of type $V \to \mathbb{R}$.*

Fact 7. *If G (equivalently, H) is biregular, then $\sigma_i \in [0,1]$ for all $i \in \{1,\ldots,d\}$. In addition, $\sigma_1 = 1$, $l_1 = \left(\frac{1}{\sqrt{|U|}}, \ldots, \frac{1}{\sqrt{|U|}}\right)$, and $r_1 = \left(\frac{1}{\sqrt{|V|}}, \ldots, \frac{1}{\sqrt{|V|}}\right)$.*

One of our main uses for G and H is to project assignments "on the V side" to assignments "on the U side". For a vector $g : V \times \Sigma \to \mathbb{R}$, this involves projecting it to the vector (Gg) of type $U \times \Sigma \to \mathbb{R}$, which is specified by the equation

$$(Gg)(u,\alpha) = \frac{1}{\sqrt{d_U \cdot d_V}} \sum_{v \sim u} \sum_{\beta : \pi_{uv}(\alpha,\beta)=1} g(v,\beta).$$

The following proposition shows that this is a reasonable way to project g "to the U side".

Proposition 2. *Suppose g is a fractional assignment. Then*

$$(Gg)(u,\alpha) = \frac{1}{\sqrt{U}} \mathop{\mathbf{Pr}}_{v \sim u} \left[\pi_{uv}(\alpha, g(v)) = 1\right].$$

For a vector $h : V \to \mathbb{R}$, its projection is the vector (Hh) of type $U \to \mathbb{R}$, which is specified by the equation

$$(Hh)(u) = \frac{1}{\sqrt{d_U \cdot d_V}} \sum_{v \sim u} h(v).$$

We will now introduce some simplifying notation. Fix adjacent vertices u and v. Write $G_{u \leftarrow v}$ for the $|\Sigma| \times |\Sigma|$ matrix defined as

$$(G_{u \leftarrow v})_{\alpha,\beta} := \begin{cases} 1 \text{ if } \pi_{u,v}(\alpha,\beta) = 1, \\ 0 \text{ otherwise.} \end{cases}$$

Recalling the $g(v,\cdot)$ notation from Definition 5, we have that

$$\left(G_{u \leftarrow v} g(v,\cdot)\right)(\alpha) = \sum_{\beta : \pi_{uv}(\beta) = \alpha} g(v,\beta).$$

Using this, we can rewrite the action of the projection game operator G on g as follows:

Proposition 3. $(Gg)(u,\alpha) = \frac{1}{\sqrt{d_U \cdot d_V}} \sum_{v \sim u} \left(G_{u \leftarrow v} g(v,\cdot)\right)(\alpha)$.

Given a vector $g : V \times \Sigma \to \mathbb{R}$, we will be especially interested in the quadratic form $\|Gg\|^2$. By Proposition 3, we can write this as

$$\|Gg\|^2 = \frac{1}{d_U \cdot d_V} \sum_{u \in U} \sum_{v_1, v_2 \sim u} \langle G_{u \leftarrow v_1} g(v_1,\cdot), G_{u \leftarrow v_2} g(v_2,\cdot) \rangle. \tag{2}$$

We can use this expression to show that G is norm reducing on fractional assignments.

Proposition 4. *Suppose $g : V \times \Sigma \to \mathbb{R}$ is a fractional assignment. Then $\|Gg\|^2 \le \|g\|^2$.*

Proposition 5. *Let $g : V \times \Sigma \to \mathbb{R}^{\ge 0}$ be a nonnegative fractional assignment, and define h and I as in Fact 6. Then $\|Hh\|^2 \ge \|Gg\|^2$.*

So as not to conflict with the notation for the indicator I from Fact 6, we will write Id for the identity operator.

2.4 Projection Games Relaxation

The paper of [DS14] introduced a convenient relaxation for val(G). Before we state what it is, let us begin with a definition.

Definition 6. *A* finite measure space *is a finite set Ω along with a measure $\mu : \Omega \to \mathbb{R}^{\ge 0}$ on that set. Typically, we will keep the set explicit while the measure will be implicit. Given a quantity $q : \Omega \to \mathbb{R}$, we define the expectation with respect to Ω as*

$$\mathop{\mathbf{E}}_{\omega \sim \Omega}[q(\omega)] = \sum_{\omega \in \Omega} \mu(\omega) \cdot q(\omega).$$

We note that μ need not be a probability measure, i.e. $\sum_{\omega \in \Omega} \mu(\omega)$ need not be 1.

$$\begin{array}{ll} \text{maximize} & \mathrm{val}_+(f,g) := \mathop{\mathbf{E}}_{\omega\sim\Omega}\langle f_\omega, Gg_\omega\rangle \\ \text{subject to} & f_\omega \text{ and } g_\omega \text{ are fractional assignments for all } \omega \in \Omega \\ & \mathop{\mathbf{E}}_{\omega\sim\Omega}\|f_\omega(u,\cdot)\|^2 \le \frac{1}{|U|} \quad \forall u \\ & \mathop{\mathbf{E}}_{\omega\sim\Omega}\|g_\omega(v,\cdot)\|^2 \le \frac{1}{|V|} \quad \forall v \end{array}$$

Now, we can specify the relaxation.

Let $\mathrm{val}_+(G)$ denote the value of the relaxation. The following proposition is implicit in [DS14]

Proposition 6. *For a bipartite projection game G, $\mathrm{val}(G^{\otimes n}) \le \mathrm{val}_+(G)^{n/2}$.*

We also need the following lemma which is a combination of Claim 2.1 and Lemma 3.1 in [DS14]

Lemma 1. *Suppose $\mathrm{val}(G^{\otimes n}) \ge (1-\eta)^n$. Then there exists a fractional assignment g such that $\|Gg\|^2 \ge 1-2\eta$ and $\|g\| = 1$.*

3 Parallel Repetition for Low Threshold Rank Graphs

In this section, we prove the following theorem.

Theorem 8. *Let G be a biregular projection game with constraint graph $H = (U, V, E)$. For $k \ge 2$, let σ_k be H's k-th singular value. If $\mathcal{S} = (f, g, \Omega)$ is a solution to the projection games relaxation with $\mathrm{val}_+(\mathcal{S}) \ge 1-\eta$, then*

$$\mathrm{val}(G) \ge 1 - \frac{64k\eta}{\sqrt{1-\sigma_k^2}}.$$

Combining this with Proposition 6 yields our main theorem. We now prove Theorem 8.

Proof (of Theorem 8). As $\mathrm{val}_+(\mathcal{S})$ is monotically increasing in f and g, we can assume that

$$\mathop{\mathbf{E}}_{\omega\sim\Omega}\|f_\omega\|^2 = \mathop{\mathbf{E}}_{\omega\sim\Omega}\|g_\omega\|^2 = 1.$$

In particular, this means that for each $u \in U$ and $v \in V$,

$$\mathop{\mathbf{E}}_{\omega\sim\Omega}\|f_\omega(u,\cdot)\|^2 = \frac{1}{|U|} \quad \text{and} \quad \mathop{\mathbf{E}}_{\omega\sim\Omega}\|g_\omega(v,\cdot)\|^2 = \frac{1}{|V|}.$$

We begin our proof by focusing in on g.

Proposition 7. $\mathbf{E}_{\omega\sim\Omega}\|Gg_\omega\|^2 \ge 1-2\eta$.

Proof. By Cauchy-Schwarz,

$$1 - \eta \leq \mathop{\mathbf{E}}_{\omega \sim \Omega} \langle f_\omega, G g_\omega \rangle \leq \mathop{\mathbf{E}}_{\omega} \|f_\omega\| \cdot \|G g_\omega\| \leq \frac{\mathbf{E}_\omega \|f_\omega\|^2 + \mathbf{E}_\omega \|G g_\omega\|^2}{2}.$$

Since $\mathbf{E}_\omega \|f_\omega\|^2 = 1$, this inequality is satisfied only if $\mathbf{E} \|G g_\omega\|^2 \geq 1 - 2\eta$.

Our next step is to convert g into a distribution on partial assignments (see Definition 4). We will need the following definition.

Definition 7. *A* partial assignment distribution (g'_ω, Ω') *is a finite measure space* Ω' *and a partial assignment* g'_ω*, for each* $\omega \in \Omega'$*.*

The next lemma shows that it is indeed possible to convert g into a partial assignment distribution, the one caveat being that $\mathbf{E}_{\omega \sim \Omega'} \|G g'_\omega\|^2$ is large only in relation to $\mathbf{E}_{\omega \sim \Omega'} \|g'_\omega\|^2$.

Lemma 2. *There is a partial assignment distribution* (g'_ω, Ω') *with*

$$\mathop{\mathbf{E}}_{\omega \sim \Omega'} \|G g'_\omega\|^2 \ \geq \ \mathop{\mathbf{E}}_{\omega \sim \Omega'} \|g'_\omega\|^2 - \frac{4\eta}{\sqrt{1 - \sigma_k^2}}.$$

Furthermore, $\frac{1}{8k} \cdot \frac{1}{|V|} \leq \mathbf{E}_{\omega \sim \Omega'} \|g'_\omega(v, \cdot)\|^2$*, for each* $v \in V$*.*

With (g'_ω, Ω') in hand, the rest of the proof will closely follow the proof of Lemma 5.5 from [DS14]. The main difference is that $\mathbf{E}_{\omega \sim \Omega'} \|G g'_\omega\|^2$ is guaranteed to be large only in relation to $\mathbf{E}_{\omega \sim \Omega'} \|g'_\omega\|^2$, and not just in isolation. However, we have the guarantee that g'_ω assigns each $v \in V$ a value a nonnegligible fraction of the time (i.e. with measure at least $\frac{1}{8k}$). This fact will allow us to extract a good assignment from (g'_ω, Ω').

Lemma 3. *There exists an assignment* $A : V \times \Sigma \to \left\{0, \frac{1}{\sqrt{|V|}}\right\}$ *such that*

$$\|GA\|^2 \geq 1 - \frac{64k\eta}{\sqrt{1 - \sigma_k^2}}.$$

We defer the proof of this lemma to the full version. The proof is largely based on the proof of Lemma 5.5 in [DS14]. There, they use the *correlated sampling* approach of [BHH$^+$08] to combine the various g'_ωs into a single assignment.

Finally, we can show an assignment (f, A) for which $\langle f, GA \rangle$ is large. This is because

$$\|GA\|^2 = \langle GA, GA \rangle \leq \max_f \langle f, GA \rangle \leq \mathrm{val}(G),$$

where the max is taken over assignments to U. Thus, $\|GA\|^2$ is a lower bound on the value of G, and we are done.

Acknowledgments. We would like to thank David Steurer for helpful discussions.

References

[ABS10] Arora, S., Barak, B., Steurer, D.: Subexponential algorithms for Unique Games and related problems. In: Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science, pp. 563–572 (2010)

[AKK$^+$08] Arora, S., Khot, S.A., Kolla, A., Steurer, D., Tulsiani, M., Vishnoi, N.K.: Unique Games on expanding constraint graphs are easy. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, pp. 21–28 (2008)

[BHH$^+$08] Barak, B., Hardt, M., Haviv, I., Rao, A., Regev, O., Steurer, D.: Rounding parallel repetitions of Unique Games. In: Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science, pp. 374–383 (2008)

[BRR$^+$09] Barak, B., Rao, A., Raz, R., Rosen, R., Shaltiel, R.: Strong parallel repetition theorem for free projection games. In: Proceedings of the 13th Annual International Workshop on Randomization and Computation, pp. 352–365 (2009)

[DS14] Dinur, I., Steurer, D.: Analytical approach to parallel repetition. In: Proceedings of the 46th Annual ACM Symposium on Theory of Computing (2014)

[FKO07] Feige, U., Kindler, G., O'Donnell, R.: Understanding parallel repetition requires understanding foams. In: Proceedings of the 22nd Annual IEEE Conference on Computational Complexity, pp. 179–192 (2007)

[GS11] Guruswami, V., Sinop, A.K.: Lasserre hierarchy, higher eigenvalues, and approximation schemes for graph partitioning and quadratic integer programming with PSD objectives. In: Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science, pp. 482–491 (2011)

[GT13] Gharan, S.O., Trevisan, L.: A new regularity lemma and faster approximation algorithms for low threshold rank graphs. In: Proceedings of the 16th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, pp. 303–316 (2013)

[KLL$^+$13] Kwok, T.C., Lau, L.C., Lee, Y.T., Gharan, S.O., Trevisan, L.: Improved Cheeger's inequality: analysis of spectral partitioning algorithms through higher order spectral gap. In: Proceedings of the 45th Annual ACM Symposium on Theory of Computing, pp. 11–20 (2013)

[Kol11] Kolla, A.: Spectral algorithms for Unique Games. Computational Complexity 20(2), 177–206 (2011)

[KT07] Kolla, A., Tulsiani, M.: Playing random and expanding unique games(2007) (manuscript)

[LOGT12] Lee, J.R., Gharan, S.O., Trevisan, L.: Multi-way spectral partitioning and higher-order Cheeger inequalities. In: Proceedings of the 44th Annual ACM Symposium on Theory of Computing, pp. 1117–1130 (2012)

[LRTV12] Louis, A., Raghavendra, P., Tetali, P., Vempala, S.: Many sparse cuts via higher eigenvalues. In: Proceedings of the 44th Annual ACM Symposium on Theory of Computing, pp. 1131–1140 (2012)

[Rao11] Rao, A.: Parallel repetition in projection games and a concentration bound. SIAM Journal on Computing 40(6), 1871–1891 (2011)

[Raz98] Raz, R.: A parallel repetition theorem. SIAM Journal on Computing 27(3), 763–803 (1998)

[Raz11] Raz, R.: A counterexample to strong parallel repetition. SIAM Journal on Computing 40(3), 771–777 (2011)

[RR12] Raz, R., Rosen, R.: A strong parallel repetition theorem for projection games on expanders. In: Proceedings of the 27th Annual IEEE Conference on Computational Complexity, pp. 247–257 (2012)

[RST12] Raghavendra, P., Steurer, D., Tulsiani, M.: Reductions between expansion problems. In: Proceedings of the 27th Annual IEEE Conference on Computational Complexity, pp. 64–73 (2012)

[SS07] Safra, S., Schwartz, O.: On parallel-repetition, unique-game and max-cut (2007) (manuscript)

[Ste13] Steurer, D.: Talk at the Simons Institute for the Theory of Computing (2013)

[Tre11] Trevisan, L.: Lecture 6 from CS359G: graph partitioning and expanders(2011) http://theory.stanford.edu/~trevisan/cs359g/lecture06.pdf

Sparser Random 3-SAT Refutation Algorithms and the Interpolation Problem[⋆]

(Extended Abstract)

Iddo Tzameret[⋆⋆]

The Institute for Interdisciplinary Information Sciences (IIIS)
Tsinghua University, Beijing

Abstract. We formalize a combinatorial principle, called *the 3XOR principle*, due to Feige, Kim and Ofek [12], as a family of unsatisfiable propositional formulas for which refutations of small size in any propositional proof system that possesses the feasible interpolation property imply an efficient *deterministic* refutation algorithm for random 3SAT with n variables and $\Omega(n^{1.4})$ clauses. Such small size refutations would improve the state of the art (with respect to the clause density) efficient refutation algorithm, which works only for $\Omega(n^{1.5})$ many clauses [13].

We demonstrate polynomial-size refutations of the 3XOR principle in resolution operating with disjunctions of quadratic equations with small integer coefficients, denoted R(quad); this is a weak extension of cutting planes with small coefficients. We show that R(quad) is weakly automatizable iff R(lin) is weakly automatizable, where R(lin) is similar to R(quad) but with linear instead of quadratic equations (introduced in [25]). This reduces the problem of refuting random 3CNF with n variables and $\Omega(n^{1.4})$ clauses to the interpolation problem of R(quad) and to the weak automatizability of R(lin).

1 Introduction

In the well known *random 3-SAT model* one usually considers a distribution on formulas in conjunctive normal form (CNF) with m clauses and three literals each, where each clause is chosen independently with repetitions out of all possible $2^3 \cdot \binom{n}{3}$ clauses with n variables (cf. [1]). The *clause density* of such a 3CNF is m/n. When m is greater than cn for sufficiently large c, that is, when the clause density is greater than c, it is known (and easily proved for e.g. $c \geq 5.2$) that with high probability a random 3CNF is unsatisfiable.

A *refutation algorithm* for random kCNFs is an algorithm that receives a kCNF (with sufficiently large clause density) and outputs either "`unsatisfiable`" or "`don't know`"; if the algorithm answers "`unsatisfiable`"

[⋆] The full version is available at: `http://arxiv.org/abs/1305.0948`

[⋆⋆] Supported in part by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the National Natural Science Foundation of P. R. China; Grants 61033001, 61061130540, 61073174, 61373002.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 1015–1026, 2014.

then the kCNF is required to be indeed unsatisfiable; moreover, the algorithm should output "`unsatisfiable`" with high probability (namely, with probability $1 - o(1)$ over the input kCNFs).

We can view the problem of determining the complexity of (deterministic) refutation algorithms as an average-case version of the **P** vs. **coNP** problem: a polynomial-time refutation algorithm for random kCNFs (for a small enough clause density) can be interpreted as showing that "$\mathbf{P} = \mathbf{coNP}$ in the average-case"; while a polynomial-time *nondeterministic* refutation algorithm (again, for a small enough clause density) can be interpreted as "$\mathbf{NP} = \mathbf{coNP}$ in the average-case".

Refutation algorithms for random kCNFs were investigated in Goerdt and Krivelevich [15] and subsequent works by Goerdt and Lanka [16], Friedman, Goerdt and Krivelevich [14], Feige and Ofek [13], Feige [11] and Coja-Oghlan et al. [8] (among other works). For random 3CNFs, the best (with respect to the clause density) polynomial-time refutation algorithm to date works for formulas with at least $\Omega(n^{1.5})$ clauses [13]. On the other hand, Feige, Kim and Ofek [12] considered efficient *nondeterministic* refutation algorithms; namely, short *witnesses* for unsatisfiability of 3CNFs that can be checked for correctness in polynomial-time. They established the current best (again, with respect to the clause density) efficient, alas *nondeterministic*, refutation procedure: they showed that with probability converging to 1 a random 3CNF with n variables and $\Omega(n^{1.4})$ clauses has a witness of size polynomial in n.

Since the current state of the art random 3CNF refutation algorithm works for $\Omega(n^{1.5})$ clauses, while the best nondeterministic refutation algorithm works already for $O(n^{1.4})$, determining whether a deterministic polynomial-time (or even a quasipolynomial-time) refutation algorithm for random 3CNFs with n variables and $\Omega(n^{1.4})$ clauses exists is to a certain extent the frontier open problem in the area of efficient refutation algorithms.

1.1 Results

In this work we reduce the problem of devising an efficient deterministic refutation algorithm for random 3CNFs with $\Omega(n^{1.4})$ clauses to the interpolation problem in propositional proof complexity. For a refutation system $\mathcal{P}$, the *interpolation problem for $\mathcal{P}$* is the problem that asks, given a $\mathcal{P}$-refutation of an unsatisfiable formula $A(x, y) \wedge B(x, z)$, for x, y, z mutually disjoint sets of variables, and an assignment α for x, to return 0 or 1, such that if the answer is 0 then $A(\alpha, y)$ is unsatisfiable and if the answer is 1 then $B(\alpha, z)$ is unsatisfiable. If the interpolation problem for a refutation system $\mathcal{P}$ is solvable in time $T(n)$ we say that $\mathcal{P}$ has *interpolation in time $T(n)$*.[1] When $T(n)$ is a polynomial we say that $\mathcal{P}$ has *feasible interpolation*. The notion of feasible interpolation was proposed in [18] and developed further in [27,6,19].

[1] We do not distinguish in this paper between proofs and refutations: proof systems prove tautologies and refutation systems refute unsatisfiable formulas (or, equivalently prove the negation of unsatisfiable formulas).

We present a family of unsatisfiable propositional formulas, ***denoted*** Υ_n and called *the 3XOR principle formulas*, expressing a combinatorial principle, such that for any given refutation system $\mathcal{P}$ that admits short refutations of Υ_n, solving efficiently the interpolation problem for $\mathcal{P}$ provides an efficient *deterministic* refutation algorithm for random 3CNFs with $\Omega(n^{1.4})$ clauses. In other words, we have the following:

Theorem 1. *If there exists a propositional proof system $\mathcal{P}$ that has interpolation in time $T(n)$ and that admits $s(n)$-size refutations of Υ_n, then there is a deterministic refutation algorithm for random 3CNF formulas with n variables and $\Omega(n^{1.4})$ clauses that runs in time $T(s(n))$. In particular, if $\mathcal{P}$ has feasible interpolation and admits polynomial-size refutations of Υ_n then the refutation algorithm runs in polynomial-time.*

The argument is based on the following: we show that the *computationally hard part* of the Feige, Kim and Ofek nondeterministic refutation algorithm (namely, the part we do not know how to efficiently compute deterministically) corresponds to a disjoint **NP**-pair. Informally, the pair $(\mathbf{A}, \mathbf{B})$ of disjoint **NP** sets is the following: **A** is the set of 3CNFs that have a certain combinatorial property, that is, they contain a collection of sufficiently many *inconsistent even k-tuples*, as defined by Feige et al. (see Definition 2); and **B** is the set of 3CNFs with m clauses for which there exists an assignment that satisfies more than $m - \ell$ clauses as 3XORs (for ℓ a certain function of the number of variables n).

Theorem 1 then follows from the known relation between disjoint **NP**-pairs and feasible interpolation [26,24]: in short, if **A** and **B** are two disjoint **NP** sets and $A(x, y)$ and $B(x, z)$ are the two polynomial-size Boolean formulas corresponding to **A** and **B**, respectively (i.e., for all x, there exists a short y such that $A(x, y) = 1$ iff $x \in \mathbf{A}$; and similarly for **B**), then short refutations of $A(x, y) \wedge B(x, z)$ imply a polynomial-size algorithm that separates **A** from **B**. For more on the relation between disjoint **NP**-pairs and propositional proof complexity see, e.g., [24,3].

In general, we observe that every efficient refutation algorithm (deterministic or not) corresponds directly to a disjoint **NP**-pair as follows: every efficient refutation algorithm is based on some property P of CNFs that can be witnessed (or better, found) in polynomial-time. Thus, every efficient refutation algorithm corresponds to a family of formulas $P(x) \to \neg\mathrm{SAT}(x)$, expressing that if the input CNF has the property P then x is unsatisfiable; thus, $P(x)$ and $\mathrm{SAT}(x)$ are two disjoint **NP** predicates. In the case of the refutation algorithm of Feige, Kim and Ofek, $P(x)$ expresses simply that the 3CNF x has the Feige et al. witness. *However, the disjoint* **NP***-pair* $(\mathbf{A}, \mathbf{B})$ *we work with is not of this type.* Namely, **A** is not the predicate $P(x)$ for the full Feige, Kim and Ofek witnesses, rather a specific combinatorial predicate (mentioned above) that is only one ingredient in the definition of the Feige et al. witness; and **B** is not $\mathrm{SAT}(x)$. This saves us the trouble to formalize and prove in a weak propositional proof system the full Feige et al. argument (such a formalization was done recently in [22]; see Sec. 1.2 for a comparison with [22]).

In the second part of this paper we reduce the problem of determinizing the Feige et al. nondeterministic refutation algorithm to the interpolation problem of a concrete and apparently weak refutation system. Specifically, we demonstrate polynomial-size refutations for Υ_n in a refutation system denoted R(quad) that extends both cutting planes with small coefficients[2] (cf. [9,6,23]) and Res(2) (for any natural k, the system Res(k) is resolution that operates with kDNFs instead of clauses, introduced by Krajíček [20]). We note also that R(quad) is a subsystem of TC^0-Frege.

An R(quad) refutation (see full version [28] for the definition) over the variables $\{x_1, \ldots, x_n\}$ operates with *disjunctions* of quadratic equations, where each quadratic equation is of the form:

$$\sum_{i,j\in[n]} c_{ij}x_ix_j + \sum_{i\in[n]} c_ix_i + c_0 = a,$$

in which all c_i, c_{ij} and a are integers written in unary. The system R(quad) has the following derivation rule, which can be viewed as a generalized resolution rule: from two disjunctions of quadratic equations $\bigvee_i L_i \vee (L = a)$ and $\bigvee_j L_j \vee (L' = b)$ one can derive:

$$\bigvee_i L_i \vee \bigvee_j L_j \vee (L - L' = a - b).$$

We also add axioms that force our variables to be $0, 1$. An R(quad) refutation of an unsatisfiable set S of disjunctions of quadratic equations is a sequence of disjunctions of quadratic equations (called *proof-lines*) that terminates with $1 = 0$, and such that every proof-line is either an axiom, or appears in S, or is derived from previous lines by the derivation rules.

We show the following:

Theorem 2. R(quad) *admits polynomial-size refutations of the 3XOR principle formulas* Υ_n.

This polynomial upper bound on the refutation size of the 3XOR principle is non-trivial because the encoding of the 3XOR formula is complicated in itself and further the refutation system is very restrictive.

By Theorem 1, we get the reduction from determinizing Feige et al. work to the interpolation problem for R(quad). In other words:

Corollary 1. *If* R(quad) *has feasible interpolation then there is a deterministic polynomial-time refutation algorithm for random 3CNFs with n variables and $\Omega(n^{1.4})$ clauses.*

Next we reduce the problem of determinizing the Feige et al. refutation algorithm to the weak automatizability of a weaker system than R(quad), namely R(lin), as explained in what follows.

[2] A refutation in *cutting planes with small coefficients* is a restriction of cutting planes in which all intermediate inequalities are required to have coefficients bounded in *value* by a polynomial in n, where n is the size of the formula to be refuted (see [6]).

The concept of automatizability, introduced by Bonet, Pitassi and Raz [7] (following the work of [21]), is central to proof-search algorithms. The *proof-search problem* for a refutation system $\mathcal{P}$ asks, given an unsatisfiable formula τ, to find a $\mathcal{P}$-refutation of τ. A refutation system $\mathcal{P}$ is *automatizable* if for any unsatisfiable τ the proof-search problem for $\mathcal{P}$ is solvable in time polynomial in the smallest $\mathcal{P}$-refutation of τ (or equivalently, if there exists a polynomial-time algorithm that on input τ and a number m in unary, outputs a $\mathcal{P}$-refutation of τ of size at most m, in case such a refutation exists). Following Atserias and Bonet [3], we say that a refutation system $\mathcal{P}$ is *weakly automatizable* if there exists an automatizable refutation system $\mathcal{P}'$ that polynomially simulates $\mathcal{P}$. Note that if $\mathcal{P}$ is not automatizable, it does *not* necessarily follow that also $\mathcal{P}'$ is not automatizable. Hence, from the perspective of proof-search algorithms, weak automatizability is a more natural notion than automatizability (see [24] on this).

In [25], the system R(lin) was introduced which is similar to R(quad), except that all equations are *linear* instead of quadratic. In other words, R(lin) is resolution over linear equations with small coefficients. We show the following:

Theorem 3. R(quad) *is weakly automatizable iff* R(lin) *is weakly automatizable.*

The proof of this theorem follows a similar argument to Pudlák [24]. Since weak automatizability of a proof system implies that the proof system has feasible interpolation [7,24], we obtain the following:

Corollary 2. *If* R(lin) *is weakly automatizable then there is a deterministic refutation algorithm for random 3CNFs with n variables and $\Omega(n^{1.4})$ clauses.*

1.2 Consequences and Relations to Previous Work

The key point of this work is the relation between constructing an efficient refutation algorithm for the clause density $\Omega(n^{0.4})$ and proving upper bounds in weak enough propositional proof systems for the 3XOR principle (namely, proof systems possessing feasible interpolation); as well as establishing such upper bounds in relatively weak proof systems.

There are two ways to view our results: either as **(i)** proposing an approach to improve the current state of the art in refutation algorithms via proof complexity upper bounds; or conversely as **(ii)** providing a *new kind* of important computational consequences that will follow from feasible interpolation and weak automatizability of weak proof systems. Indeed, the consequence that we provide is of a different kind from the group of important recently discovered algorithmic-game-theoretic consequences shown by Atserias and Maneva [4], Huang and Pitassi [17] and Beckmann, Pudlák and Thapen [5]. In what follows we explain these two views in more details.

(i) Our results show that by proving that R(quad) has feasible interpolation or by demonstrating a short refutation of the 3XOR principle in some refutation system that admits feasible interpolation, one can advance the state of the art in refutation algorithms. We can hope that if feasible interpolation of R(quad) does not hold, perhaps interpolation in quasipolynomial-time holds (either for R(quad) or for any other system admitting short refutations of the 3XOR principle), which would already improve exponentially the running time of the current best deterministic refutation algorithm for 3CNFs with $\Omega(n^{1.4})$ clauses, since the current algorithm works in time $2^{O(n^{0.2}\log n)}$ [12].

As mentioned above, R(quad) is a common extension of Res(2) and cutting planes with small coefficients (though it is apparently not the weakest such common extension because already R(lin) polynomially simulates both Res(2) and cutting planes with small coefficients). Whether Res(2) has feasible interpolation (let alone, interpolation in quasi-polynomial time) is open and there is no conclusive evidence for or against it. Note that by Atserias and Bonet [3], Res(2) has feasible interpolation iff resolution is weakly automatizable. However this does not necessarily constitute strong evidence against the feasible interpolation of Res(2), because the question of whether resolution is *weakly* automatizable is itself open, and there is no strong evidence ruling out a positive answer to this question.[3] Similarly, there is no strong evidence that rules out the possibility that cutting planes is weakly automatizable.

(ii) Even if our suggested approach is not expected to lead to an improvement in refutation algorithms, it is still interesting in the following sense. The fact that R(quad) has short refutations of the 3XOR principle provides new evidence that (weak extensions of) Res(2) and cutting planes with small coefficients may not have feasible interpolation, or at least that it would be highly non-trivial to prove they do have feasible interpolation; the reason for this is that establishing feasible interpolation for such proof systems would entail quite strong algorithmic consequences, namely, a highly non-trivial improvement in refutation algorithms. This algorithmic consequence adds to other recently discovered and important algorithmic-game-theoretic consequences that would follow from feasible interpolation of weak proof systems.

Specifically, in recent years several groups of researchers discovered connections between feasible interpolation and weak automatizability of small depth Frege systems to certain game-theoretic algorithms: Atserias and Maneva [4] showed that solving *mean payoff games* is reducible to the weak automatizability of depth-2 Frege (equivalently, Res(n)) systems and to the feasible interpolation of depth-3 Frege systems (actually, depth-3 Frege where the bottom fan-in of formulas is at most two). Subsequently, Huang and Pitassi [17] showed that if depth-3 Frege system is weakly automatizable, then *simple stochastic games* are solvable in polynomial time. Finally, Beckmann, Pudlák and Thapen [5] showed that weak automatizability of resolution implies a polynomial-time algorithm for the *parity game.*

Comparison with Müller and Tzameret [22]. In [22] a polynomial-size TC^0-Frege proof of the correctness of the Feige et al. witnesses was shown. However

[3] It is known that, based on reasonable hardness assumptions from parameterized complexity, resolution is not *automatizable* by Alekhnovich and Razborov [2], which is, as the name indicates, a stronger property than *weak automatizability.*

the goal of [22] was different from the current paper. In [22] the goal was to construct short propositional refutations for random 3CNFs (with sufficiently small clause density). Accordingly, the connection to the interpolation problem was not made in [22]; and further, it is known by [7] that TC^0-Frege does not admit feasible interpolation (under cryptographical assumptions). On the other hand, the current paper aims to demonstrate that certain short refutations will have *algorithmic* consequences (for refutation algorithms). Indeed, since we are not interested here to prove the correctness of the full Feige et al. witnesses, we are isolating the computationally hard part of the witnesses from the easy (polytime computable) parts, and formalize the former part (i.e., the 3XOR principle) as a propositional formula in a way that is suitable for the reduction to the interpolation problem.

One advantage of this work over [22] is that Theorem 2 gives a more concrete logical characterization of parts of the Feige et al. witnesses (because the proofs in [22] were conducted indirectly, via a general translation from first-order proofs in bounded arithmetic), and this characterization is possibly *tighter* (because R(quad) is apparently strictly weaker than TC^0-Frege).

Organization of the extended abstract. In the preliminaries we review some necessary background from proof complexity and refutation algorithms. In Sec. 3 we describe the connection between feasible interpolation and refutations algorithms. Due to lack of space, readers who wish to read the full details involved in the short R(quad) refutations of the 3XOR principle, as well as the reduction to weak automatizability of R(lin), are referred to the full version available on-line [28].

2 Preliminaries

We usually assume that a 3CNF has n variables $X = \{x_1, \dots, x_n\}$ and m clauses.

2.1 Disjoint NP-pairs and Feasible Interpolation of Propositional Proofs

A *disjoint **NP**-pair* is simply a pair of languages in **NP** that are disjoint. Let L, N be a disjoint **NP**-pair such that $R(x,y)$ is the corresponding relation for L and $Q(x,z)$ is the corresponding relation for N; namely, there exists polynomials p, q such that $R(x,y)$ and $Q(x,z)$ are polynomial-time relations where $x \in L$ iff $\exists y, |y| \le p(|x|) \wedge R(x,y) = \texttt{true}$ and $x \in N$ iff $\exists z, |z| \le q(|x|) \wedge Q(x,z) = \texttt{true}$.

Since both polynomial-time relations $R(x,y)$ and $Q(x,z)$ can be converted into a family of polynomial-size Boolean circuits, they can be written as a family of polynomial-size (in n) CNF formulas. Thus, let $A_n(\overline{x}, \overline{y})$ be a polynomial-size CNF in the variables $\overline{x} = (x_1, \dots, x_n)$ and $\overline{y} = (y_1, \dots, y_\ell)$, that is true iff $R(\overline{x}, \overline{y})$ is true, and let $B_n(\overline{x}, \overline{z})$ be a polynomial-size CNF in the variables $\overline{x}$ and $\overline{z} = (z_1, \dots, z_m)$, that is true iff $Q(\overline{x}, \overline{z})$ is true (for some ℓ, m that are polynomial in n). For every $n \in \mathbb{N}$, we define the following unsatisfiable CNF formula in three mutually disjoint vectors of variables $\overline{x}, \overline{y}, \overline{z}$:

$$F_n := A_n(\overline{x}, \overline{y}) \wedge B_n(\overline{x}, \overline{z}). \quad (1)$$

Note that because $\overline{y}$ and $\overline{z}$ are disjoint vectors of variables and $A_n(\overline{x}, \overline{y}) \wedge B_n(\overline{x}, \overline{z})$ is unsatisfiable, it must be that given any $\overline{x} \in \{0,1\}^n$, either $A_n(\overline{x}, \overline{y})$ or $B_n(\overline{x}, \overline{z})$ is unsatisfiable (or both).

A *propositional proof system* $\mathcal{P}$ is a polynomial-time relation $V(\pi, \tau)$ such that for every propositional formula τ, τ is a tautology iff there exists a binary string π with $V(\pi, \tau) = \texttt{true}$. A propositional proof system $\mathcal{P}$ *polynomially-simulates* another propositional proof system $\mathcal{Q}$ if there is a polynomial-time computable function f that maps $\mathcal{Q}$-proofs to $\mathcal{P}$-proofs of the same tautologies.

Consider a family of unsatisfiable formulas $F_n := A_n(\overline{x}, \overline{y}) \wedge B_n(\overline{x}, \overline{z})$, $i \in \mathbb{N}$, in mutually disjoint vectors of variables, as in (1). We say that the Boolean function $f(\overline{x})$ is *the interpolant of* F_n if for every n and every assignment $\overline{\alpha}$ to $\overline{x}$:

$$\begin{array}{lcll} f(\overline{\alpha}) = 1 & \Longrightarrow & A_n(\overline{\alpha}, \overline{y}) & \text{is unsatisfiable; and} \\ f(\overline{\alpha}) = 0 & \Longrightarrow & B_n(\overline{\alpha}, \overline{z}) & \text{is unsatisfiable.} \end{array} \tag{2}$$

Note that L (as defined above) is precisely the set of those assignments $\overline{\alpha}$ for which $A(\overline{\alpha}, \overline{y})$ is satisfiable, and N is precisely the set of those assignments $\overline{\alpha}$ for which $B(\overline{\alpha}, \overline{z})$ is satisfiable, and L and N are disjoint by assumption, and so $f(\overline{x})$ *separates* L from N.

Definition 1 (Interpolation Property). *A propositional proof system $\mathcal{P}$ is said to have the* interpolation property in time $T(n)$ *if the existence of a size $s(n)$ $\mathcal{P}$-refutation of a family F_n as in (1) above implies the existence of an algorithm computing $f(\overline{x})$ in $T(s(n))$ time. When a proof system $\mathcal{P}$ has the interpolation property in time* poly(n) *we say that $\mathcal{P}$ has the* feasible interpolation property*, or simply that $\mathcal{P}$ has* feasible interpolation*.*

Definition 2 (Inconsistent Even k-tuple (Feige et al. [12]). *An* even k-tuple *is a tuple of k many 3-clauses in which every variable appears an even number of times. An* inconsistent even k-tuple *is an even k-tuple in which the total number of negative literals is odd.*

Note that for any even k-tuple, k must be an even number (since by assumption the total number of variable occurrences $3k$ is even). The following is the combinatorial principle, due to Feige et al. [12] that we consider in this work:

The 3XOR Principle 1. *Let K be a 3CNF over the variables X. Let S be t inconsistent even k-tuples from K, such that every clause from K appears in at most d inconsistent even k-tuples in S. Then, given any Boolean assignment to the variables X, the number of clauses in K that are unsatisfied by the assignment as 3XOR is at least $\lceil t/d \rceil$.*

The correctness of the 3XOR principle follows directly from the following proposition (the proof of which follows by counting modulo 2) and the fact that every clause in K appears in at most d even k-tuples in S:

Proposition 1 ([12]). *For any inconsistent even k-tuple (over the variables X) and any Boolean assignment A to X, there must be a clause in the k-tuple that is unsatisfied as 3XOR.*

3 From Short Proofs to Refutation Algorithms

In this section we demonstrate that polynomial-size proofs of (encodings of the) 3XOR principle in a proof system that has the feasible interpolation property yield deterministic polynomial-time refutation algorithms for random 3CNF formulas with $\Omega(n^{1.4})$ clauses.

3.1 The Witness for Unsatisfiability

The Feige, Kim and Ofek nondeterministic refutation algorithm [12] is based on the existence of a polynomial-size witness of unsatisfiability for most 3CNF formulas with sufficiently large clause to variable ratio. The witness has several parts, but as already observed in [12], apart from the t inconsistent even k-tuples (Def. 2), all the other parts of the witness are known to be computable in polynomial-time. In what follows we define the witnesses for unsatisfiability.

Let K be a 3CNF with n variables $x_1, \ldots, x_n$ and m clauses. The *imbalance* of a variable x_i is the absolute value of the difference between the number of its positive occurrences and the number of its negative occurrences. The *imbalance of* K is the sum over the imbalances of all variables, in K, denoted $I(K)$. We define $M(K)$ to be an $n \times n$ rational matrix M as follows: let $i, j \in [n]$, and let d be the number of clauses in K where x_i and x_j appear with different signs and s be the number of clauses where x_i and x_j appear with the same sign. Then $M_{ij} := \frac{1}{2}(d - s)$. In other words, for each clause in K in which x_i and x_j appear with the same sign we add $\frac{1}{2}$ to M_{ij} and for each clause in K in which x_i and x_j appear with different signs we subtract $\frac{1}{2}$ from M_{ij}. Let λ be a rational approximation of the biggest eigenvalue of $M(K)$. We shall assume that the additive error of the approximation is $1/n^c$ for a constant c independent of n; i.e., $|\lambda - \lambda'| \leq 1/n^c$, for λ' the biggest eigenvalue of $M(K)$; see [22].

Definition 3 (FKO Witness). *Given a 3CNF K, the* FKO witness for the unsatisfiability of K *is defined to be the following collection:*

1. *the imbalance $I(K)$;*
2. *the matrix $M(K)$ and the (polynomially good) rational approximation λ of its largest eigenvalue;*
3. *a collection S consisting of $t < n^2$ inconsistent even k-tuples such that every clause in K appears in at most d many even k-tuples, for some positive natural k;*
4. *the inequality $t > \frac{d \cdot (I(K) + \lambda n)}{2} + o(1)$ holds.*

(The $o(1)$ above stands for a specific rational number b/n^c, for c and b constants independent of n).

Theorem 4 ([12]). *There are constants c_0, c_1 such that for a random 3CNF K with n variables and $\Omega(n^{1.4})$ clauses, with probability converging to 1 as n tends to infinity there exist natural numbers k, t, d such that $t = \Omega(n^{1.4})$ and*

$$k \leq c_0 \cdot n^{0.2} \quad \textit{and} \quad t < n^2 \quad \textit{and} \quad d \leq c_1 \cdot n^{0.2}, \tag{3}$$

and K has a witness for unsatisfiability as in Definition 3.

Inspecting the argument in [12], it is not hard to see that it is sufficient to replace part 3 in the witness with a witness for the following:

> *3'. No assignment can satisfy more than* $m - \lceil t/d \rceil - 1$ *clauses in* K *as 3XORs.*

Therefore, since $I(K)$, $M(K)$ and λ are all polynomial-time computable (see [12] for this), in order to determinize the nondeterministic refutation algorithm of [12] it is sufficient to provide an algorithm that almost surely determines (correctly) that part 3' above holds (when also t and d are such that part 4 in the witness holds). In other words, in order to construct an efficient refutation algorithm for random 3CNFs (with $\Omega(n^{1.4})$ clauses) it is sufficient to have a deterministic algorithm A that on every input 3CNF (and for t and d such that part 4 in the witness holds) answers either "`condition 3' is correct`" or "`don't know`", such that A is never wrong (i.e., if it says "`condition 3' is correct`" then condition 3' holds) and with probability $1 - o(1)$ over the input 3CNFs A answers "`condition 3' is correct`". Note that we do <u>not</u> need to actually find the Feige et al. witness <u>nor</u> do we need to decide if it exists or not (it is possible that condition 3' holds but condition 3 does not, meaning that there is *no* Feige et al. witness). The relation between unsatisfiability and bounding the number of clauses that can be satisfied as 3XOR in a 3CNF was introduced by Feige in [10] (and used in [13] as well as in [12]).

3.2 The Disjoint NP-pair Corresponding to the 3XOR Principle

We define the corresponding *3XOR principle disjoint* ***NP****-pair* as the pair of languages (L, N), where k, t, d are natural numbers given in *unary*:

$$\begin{aligned} L := \{\langle X, k, t, d\rangle \mid X &\text{ is a 3CNF with } n \text{ variables and Equation (3) holds} \\ &\text{for } k, t, d \text{ and there exist } t \text{ inconsistent even } k\text{-tuples such that} \\ &\text{each clause of } X \text{ appears in no more than } d \text{ many } k\text{-tuples}\}, \end{aligned}$$

$$\begin{aligned} N := \big\{\langle X, k, t, d\rangle \mid X &\text{ is a 3CNF with } n \text{ variables and } m \text{ clauses and} \\ &\text{Equation (3) holds for } k, t, d \text{ and there exists an assignment} \\ &\text{that satisfies at least } m - \lceil t/d \rceil \text{ clauses in } X \text{ as 3XOR}\big\}. \end{aligned}$$

It is easy to verify that both L and N are indeed **NP** sets and that, by the 3XOR principle, $L \cap N = \emptyset$.

Using the same notation as in Section 2.1, we denote by $R(x, y)$ and $Q(x, z)$ the polynomial-time relations for L and N, respectively. Further, for every $n \in \mathbb{N}$, there exists an *unsatisfiable* CNF formula in three mutually disjoint sets of variables $\overline{x}, \overline{y}, \overline{z}$:

$$\Upsilon_n := A_n(\overline{x}, \overline{y}) \wedge B_n(\overline{x}, \overline{z}), \tag{4}$$

where $A_n(\overline{x}, \overline{y})$ and $B_n(\overline{x}, \overline{z})$ are the CNF formulas expressing that $R(x, y)$ and $Q(x, z)$ are true for x of length n, respectively.

Theorem 1. *Assume that there exists a propositional proof system that has interpolation in time $T(n)$ and that admits size $s(n)$ refutations of Υ_n. Then, there is a deterministic refutation algorithm for random 3CNF formulas with $\Omega(n^{1.4})$ clauses running in time $T(s(n))$.*

Proof. By the assumption, and by the definition of the feasible interpolation property, there exists a deterministic polynomial-time interpolant algorithm A that on input a 3CNF K and three natural numbers k, t, d given in unary, if $\mathsf{A}(K, k, t, d) = 1$ then $\langle K, k, t, d\rangle \notin L$ and if $\mathsf{A}(K, k, t, d) = 0$ then $\langle K, k, t, d\rangle \notin N$.

The desired refutation algorithm works as follows: it receives the 3CNF K and for each 3-tuple of natural numbers $\langle k, t, d\rangle$ for which Equation (3) holds it runs $\mathsf{A}(K, k, t, d)$. Note there are only $O(n^3)$ such 3-tuples. If for one of these runs $\mathsf{A}(K, k, t, d) = 0$ then we know that $\langle K, k, t, d\rangle \notin N$; in this case we check (in polynomial-time) that the inequality in Part 4 of the FKO witness (Definition 3) holds, and if it does, we answer "`unsatisfiable`". Otherwise, we answer "`don't know`".

The correctness of this algorithm stems from the following two points:

(i) If we answered "`unsatisfiable`", then there exist k, t, d such that $\langle K, k, t, d\rangle \notin N$ and Part 4 in the FKO witness holds, and so Condition 3' (from Section 3.1) is correct, and hence, by the discussion in 3.1, K is unsatisfiable.

(ii) For almost all 3CNFs we will answer "`unsatisfiable`". This is because almost all of them will have an FKO witness (by Theorem 4), which means that $\langle K, k, t, d\rangle \in L$ for some choice of $t < n^2, d, k$ (in the prescribed ranges) and hence the interpolant algorithm A must output 0 in at least one of these cases (because $\mathsf{A}(K, k, t, d) = 1$ means that $\langle K, k, t, d\rangle \notin L$).

Acknowledgments. I wish to thank Jan Krajíček for very useful comments related to this work as well as Albert Atserias and Neil Thapen for useful related discussions. Thanks also to the anonymous reviewers who helped a lot to improve the exposition.

References

1. Achlioptas, D.: Random satisfiability. In: Handbook of Satisfiability, pp. 245–270 (2009)
2. Alekhnovich, M., Razborov, A.A.: Resolution is not automatizable unless W[P] is tractable. SIAM J. Comput. 38(4), 1347–1363 (2008)
3. Atserias, A., Bonet, M.L.: On the automatizability of resolution and related propositional proof systems. Information and Computation 189, 182–201 (2004)
4. Atserias, A., Maneva, E.: Mean-payoff games and propositional proofs. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6198, pp. 102–113. Springer, Heidelberg (2010)
5. Beckmann, A., Pudlák, P., Thapen, N.: Parity games and propositional proofs. ACM Transactions on Computational Logic (to appear)
6. Bonet, M.L., Pitassi, T., Raz, R.: Lower bounds for cutting planes proofs with small coefficients. The Journal of Symbolic Logic 62(3), 708–728 (1997)

7. Bonet, M.L., Pitassi, T., Raz, R.: On interpolation and automatization for Frege systems. SIAM J. Comput. 29(6), 1939–1967 (2000)
8. Coja-Oghlan, A., Goerdt, A., Lanka, A.: Strong refutation heuristics for random k-SAT. Combinatorics, Probability & Computing 16(1), 5–28 (2007)
9. Cook, W., Coullard, C.R., Turan, G.: On the complexity of cutting plane proofs. Discrete Applied Mathematics 18, 25–38 (1987)
10. Feige, U.: Relations between average case complexity and approximation complexity. In: STOC, pp. 534–543 (2002)
11. Feige, U.: Refuting smoothed 3CNF formulas. In: Proceedings of the IEEE 48th Annual Symposium on Foundations of Computer Science, pp. 407–417. IEEE Computer Society (2007)
12. Feige, U., Kim, J.H., Ofek, E.: Witnesses for non-satisfiability of dense random 3CNF formulas. In: Proceedings of the IEEE 47th Annual Symposium on Foundations of Computer Science (2006)
13. Feige, U., Ofek, E.: Easily refutable subformulas of large random 3CNF formulas. Theory of Computing 3(1), 25–43 (2007)
14. Friedman, J., Goerdt, A., Krivelevich, M.: Recognizing more unsatisfiable random k-SAT instances efficiently. SIAM J. Comput. 35(2), 408–430 (2005)
15. Goerdt, A., Krivelevich, M.: Efficient recognition of random unsatisfiable k-SAT instances by spectral methods. In: Annual Symposium on Theoretical Aspects of Computer Science, pp. 294–304 (2001)
16. Goerdt, A., Lanka, A.: Recognizing more random unsatisfiable 3-SAT instances efficiently. Electronic Notes in Discrete Mathematics 16, 21–46 (2003)
17. Huang, L., Pitassi, T.: Automatizability and simple stochastic games. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part I. LNCS, vol. 6755, pp. 605–617. Springer, Heidelberg (2011)
18. Krajíček, J.: Lower bounds to the size of constant-depth propositional proofs. The Journal of Symbolic Logic 59(1), 73–86 (1994)
19. Krajíček, J.: Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. Jour. Symb. Logic 62(2), 457–486 (1997)
20. Krajíček, J.: On the weak pigeonhole principle. Fund. Math. 170(1-2), 123–140 (2001)
21. Krajíček, J., Pudlák, P.: Some consequences of cryptographical conjectures for S^1_2 and EF. Inform. and Comput. 140(1), 82–94 (1998)
22. Müller, S., Tzameret, I.: Short propositional refutations for dense random 3CNF formulas. In: Proceedings of the 27th Annual ACM-IEEE Symposium on Logic In Computer Science (LICS) (2012)
23. Pudlák, P.: Lower bounds for resolution and cutting plane proofs and monotone computations. The Journal of Symbolic Logic 62(3), 981–998 (1997)
24. Pudlák, P.: On reducibility and symmetry of disjoint NP pairs. Theoret. Comput. Sci. 295, 323–339 (2003)
25. Raz, R., Tzameret, I.: Resolution over linear equations and multilinear proofs. Ann. Pure Appl. Logic 155(3), 194–224 (2008)
26. Razborov, A.A.: On provably disjoint NP-pairs. ECCC 1(6) (1994)
27. Razborov, A.A.: Unprovability of lower bounds on circuit size in certain fragments of bounded arithmetic. Izv. Ross. Akad. Nauk Ser. Mat. 59(1), 201–224 (1995)
28. Tzameret, I.: Sparser random 3-SAT refutation algorithms and the interpolation problem (2014), http://arxiv.org/abs/1305.0948

On Learning, Lower Bounds and (un)Keeping Promises

Ilya Volkovich*

Computer Science Department and Center for Computational Intractability,
Princeton University, Princeton NJ
ilyav@cs.princeton.edu

Abstract. We extend the line of research initiated by Fortnow and Klivans [6] that studies the relationship between efficient learning algorithms and circuit lower bounds. In [6], it was shown that if a Boolean circuit class $\mathcal{C}$ has an efficient *deterministic* exact learning algorithm, (i.e. an algorithm that uses membership and equivalence queries) then $\mathsf{EXP}^{\mathsf{NP}} \not\subseteq \mathsf{P/poly}[\mathcal{C}]$[1]. Recently, in [14] $\mathsf{EXP}^{\mathsf{NP}}$ was replaced by $\mathsf{DTIME}(n^{\omega(1)})$. Yet for the models of *randomized* exact learning or Valiant's PAC learning, the best result so far is a lower bound against BPEXP (the exponential-time analogue of BPP). In this paper, we derive stronger lower bounds as well as some other consequences from *randomized* exact learning and PAC learning algorithms, answering an open question posed in [6] and [14]. In particular, we show that

1. If a Boolean circuit class $\mathcal{C}$ has an efficient *randomized* exact learning algorithm or an efficient PAC learning algorithm[2] then $\mathsf{BPTIME}(n^{\omega(1)})/1 \not\subseteq \mathsf{P/poly}[\mathcal{C}]$.
2. If a Boolean circuit class $\mathcal{C}$ has an efficient *randomized* exact learning algorithm then no strong pseudo-random generators exist in $\mathsf{P/poly}[\mathcal{C}]$.

We note that in both cases the learning algorithms need not be *proper*[3]. The extra bit of advice comes to accommodate the need to keep the promise of bounded away probabilities of acceptance and rejection. The exact same problem arises when trying to prove lower bounds for BPTIME or MA [3,7,16,20]. It has been an open problem to remove this bit. We suggest an approach to settle this problem in our case. Finally, we slightly improve the result of [5] by showing a subclass of MAEXP that requires super-polynomial circuits. Our results combine and extend some of the techniques previously used in [6,14] and [20].

* Research partially supported by NSF Award CCF 0832797. Full version available at http://eccc.hpi-web.de/report/2014/058/.

[1] $\mathsf{P/poly}[\mathcal{C}]$ stands for the class of all languages that can be computed by polynomial-size circuits from the class $\mathcal{C}$.

[2] In fact, our result hold even for a more general model of PAC learning with membership queries.

[3] A learning algorithm is proper if it outputs a hypothesis from the class it learns.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 1027–1038, 2014.

1 Introduction

Revealing a hidden function from a set of examples is a natural and basic problem. As in any other problem, identifying the obstacles along your trail is a fundamental task in achieving your goal. In this paper, we continue to identify the obstacles to efficient learnability following the line of research initiated by Fortnow and Klivans [6]. Several results [23,13,15,8] exhibit a two-way connection between learning and cryptography: basing the hardness of learning on cryptography and constructing cryptographic primitives based on hardness of learning. In this paper we identify the obstacles in the form of circuit lower bounds and relationships between complexity classes.

Angluin's model of Exact Learning [2] consists of a (computationally bounded) learner and a (all-powerful) teacher. The learner's goal is to output a target function f from a given function class $\mathcal{C}$. To do so, the learner is allowed to query the value $f(\bar{x})$ on any input $\bar{x}$ (membership query). In addition, the learner is also allowed to propose a hypothesis $\hat{f}$ and ask the teacher whether it is equivalent to f (equivalence query). If this is indeed the case, the learner has achieved his goal. Otherwise, the teacher presents the learner with a counterexample $\bar{a}$ for which $f(\bar{a}) \neq \hat{f}(\bar{a})$. We say that a function class $\mathcal{C}$ is *exactly learnable* if there exists a learner which given any $f \in \mathcal{C}$, in time polynomial in n and $|f|$ (the size of f in the representation scheme) outputs a hypothesis $\hat{f}$ such that $f(\bar{x}) = \hat{f}(\bar{x})$ for all $\bar{x}$, using membership and equivalence queries. In the *randomized* Exact Learning model the learner is allowed to toss coins and, given any $f \in \mathcal{C}$, it must output a correct hypothesis, with high probability. We say that a class $\mathcal{C}$ is *exactly learnable with high probability* (w.h.p) if there exists a learner which given any $f \in \mathcal{C}$, in time polynomial in n and $|f|$ w.h.p outputs a hypothesis $\hat{f}$ such that $f(\bar{x}) = \hat{f}(\bar{x})$ for all $\bar{x}$, using membership and equivalence queries.

In Valiants PAC learning model [23], we (again) have a (computationally bounded) learner that is given a set of samples of the form $(\bar{x}, f(\bar{x}))$ from some fixed function $f \in \mathcal{C}$, where $\bar{x}$ is chosen according to some unknown distribution D. Given $\varepsilon > 0$ and $\delta > 0$, the learner's goal is to output, with probability $1 - \varepsilon$ a hypothesis $\hat{f}$ such that $\hat{f}$ is a $1 - \delta$ close to f under D. We say that a function class $\mathcal{C}$ is PAC *learnable* if there exists a learner which given any $f \in \mathcal{C}$, $\varepsilon > 0$ and $\delta > 0$ in time polynomial in $n, 1/\varepsilon, 1/\delta, |f|$ outputs a hypothesis as required. In a more general model, the learner is allowed membership queries (as in the exact learning model). In this case, we say that $\mathcal{C}$ is PAC *learnable* with *membership queries*.

A learning algorithm is said to be *proper* if it outputs a hypothesis from the class it learns. It is known [1] that both randomized and exact learners can be used to obtain a PAC learner with membership queries. Thus, hardness results for the PAC learning model entail hardness results for the randomized exact learning model. In [6], Fortnow and Klivans have shown that if a Boolean circuit class $\mathcal{C}$ has an efficient *deterministic* exact learning algorithm, then $\mathsf{EXP}^{\mathsf{NP}} \not\subseteq \mathsf{P/poly}[\mathcal{C}]$. Subsequently, Harkins and Hitchcock [10] removed the NP oracle replacing $\mathsf{EXP}^{\mathsf{NP}}$ by EXP, using techniques from resource bounded measure. Further improvement was shown in [14] where $\mathsf{EXP}^{\mathsf{NP}}$ was replaced by $\mathsf{DTIME}(n^{\omega(1)})$ using simpler, diagonalization type of techniques. However, given an efficient *randomized* exact learning algorithm or a PAC learner, the above techniques fail to produce a hard function. Indeed, the best known result so far [6,14] is a lower bound against BPEXP (the exponential-time analogue of BPP), therefore leaving an open question for improvement. In this paper we derive stronger (matching) lower bounds as well as some other consequences from *randomized* exact learning and PAC learning algorithms, answering the open question.

1.1 Our Results

In this section we go briefly over our results comparing them with the previous ones. We now present the first result of the paper, which gives lower bounds against $\mathsf{BPTIME}(n^{\omega(1)})/1$ and $\mathsf{PromiseBPTIME}(n^{\omega(1)})$ assuming an efficient randomized exact learner or an efficient PAC learner with queries. We note that the learning algorithm need not be proper. .

Theorem 1. *Let $\mathcal{C}$ be a circuit class. If $\mathcal{C}$ is exactly learnable w.h.p or is* PAC *learnable with membership queries, then* $\mathsf{BPTIME}(n^{\omega(1)})/1 \not\subseteq \mathsf{P/poly}[\mathcal{C}]$ *and* $\mathsf{PromiseBPTIME}(n^{\omega(1)}) \not\subseteq \mathsf{P/poly}[\mathcal{C}]$.

This matches the recent result of [14], where a lower bound against $\mathsf{DTIME}(n^{\omega(1)})$ was produced assuming a deterministic efficient learner. In fact, the results of [14] also imply fixed polynomial-size circuit lower bounds against P. That is, for each k, $\mathsf{P} \not\subseteq \mathsf{SIZE}(n^k)[\mathcal{C}]$, where $\mathsf{SIZE}(n^k)[\mathcal{C}]$ stands for languages accepted by size $\mathcal{O}(n^k)$ circuits from the class $\mathcal{C}$. We match this result as well.

Theorem 2. *Let $\mathcal{C}$ be a circuit class. If $\mathcal{C}$ is exactly learnable w.h.p or is* PAC *learnable with membership queries, then for any $k \geq 1$:* $\mathsf{BPP}/1 \not\subseteq \mathsf{SIZE}(n^k)[\mathcal{C}]$ *and* $\mathsf{PromiseBPP} \not\subseteq \mathsf{SIZE}(n^k)[\mathcal{C}]$.

Next, we show that efficient randomized exact learner for a circuit class $\mathcal{C}$ gives rise to a P/poly-natural property useful against $\mathsf{P/poly}[\mathcal{C}]$. Following the celebrated result of Razborov & Rudich [19] (and its extensions) this implies that no strong pseudo-random generators exist in $\mathsf{P/poly}[\mathcal{C}]$. For the case of $\mathcal{C} = \mathsf{P/poly}$ or even a smaller class of TC^0 of constant-depth threshold functions such an outcome is very unlikely [18]. Again, the learning algorithm need not be proper.

Theorem 3. *Let $\mathcal{C}$ be a circuit class. If $\mathcal{C}$ is exactly learnable w.h.p then no strong pseudo-random generators exist in* $\mathsf{P/poly}[\mathcal{C}]$.

In a similar fashion to the previous lower bounds and hierarchy theorems for the "bounded" probabilistic classes [3,7,16] and MA [20] we require an extra bit of advice to keep the promise of bounded away probabilities. It has been an open problem to remove this bit. We suggest an approach to settle this problem by trying to "unkeep" the promise using fixed oracles. More specifically, recall that $\mathsf{BPP} = \mathsf{P}^{\mathsf{PromiseBPP}} = \mathsf{P}^{\mathrm{CA}}$ [4]. That is, a language L is in BPP iff it can be decided by a P machine with an oracle to CA. Note, thought, that for some queries the answer of the CA oracle can be arbitrary (e.g. for balanced circuits). We eliminate the need of an advice bit for the cases when oracle is fixed.

Theorem 4. *Let O be an oracle consistent with* CA*. That is, O accepts circuit from CA_{YES} and rejects circuit from CA_{NO}. Let $\mathcal{C}$ be a circuit class. If $\mathcal{C}$ is exactly learnable w.h.p or is* PAC *learnable with membership queries, then for any $k \geq 1$: $\mathsf{P}^{O} \not\subseteq \mathsf{SIZE}(n^k)[\mathcal{C}]$.*

In [20], Santhanam proved lowers bounds for $\mathsf{MA}/1$ unconditionally. In fact, our conditional results use several techniques from this paper (partially described in Section 1.2). Applying the same idea unconditionally and recalling that $\mathsf{MA} = \mathsf{NP}^{\mathsf{PromiseBPP}} = \mathsf{NP}^{\mathrm{CA}}$ (see e.g. [9]) we obtain the following (unconditional) result:

Theorem 5. *Let O be an oracle consistent with* CA*. That is, O accepts circuit from CA_{YES} and rejects circuit from CA_{NO}. Then for any $k \geq 1$: $\mathsf{NP}^{O} \not\subseteq \mathsf{SIZE}(n^k)$.*

The smallest complexity class for which unconditional super-polynomial lower bounds are known is MAEXP [5,17]. We show that this is still true for a subclass of MAEXP, as $\mathsf{MAEXP} = \mathsf{NEXP}^{\mathsf{PromiseBPP}}$. Due to space limitation the proof of this theorem is omitted.

Theorem 6. *There exists a language* $\mathrm{LA} \in \mathsf{PromiseBPP}$ *s.t.* $\mathsf{NEXP}^{\mathrm{LA}} \not\subseteq \mathsf{P/poly}$.

1.2 Our Techniques and Ideas

We now describe our main techniques and ideas.

Learning a "Conveniently Hard" Language. As in various previous papers dealing with conditional and unconditional lower bounds [11,12,6,20,14], we require a "conveniently hard" language L. That is, a language L that possesses some "nice" structural properies (downward self-reducibility, self-testability and self-correctability) and yet can be used to compute a "hard" function (for a formal definition see Definition 6). The "nice" properties of L make it easily learnable given an efficient learner. More specifically, they allow answering the learner's queries efficiently. We now describe the idea in a nutshell. We combine several techniques from [11,6,20,14] and extend them.

Given an efficient learner for a circuit class $\mathcal{C}$, the idea is to learn a non-uniform circuit $C \in \mathcal{C}$ for L and then use C to compute the hard function within L. This puts a hard function in BPP and, obviously, can be carried out only if L is computable by a small-sized family of circuits in $\mathcal{C}$. Similar idea appeared in

[4] CA (Circuit Approximation) is the natural $\mathsf{PromiseBPP}$-complete problem $\mathrm{CA}_{YES} = \left\{C \mid \Pr_{\bar{a}\in\{0,1\}^n}[C(\bar{a}) = 1] \geq 3/4\right\}$ and $\mathrm{CA}_{NO} = \left\{C \mid \Pr_{\bar{a}\in\{0,1\}^n}[C(\bar{a}) = 1] \leq 1/4\right\}$.

[11]. The other option is that L requires large circuits from $\mathcal{C}$. At this point, the approach taken in [6,14] is to combine both hardness results into a single lower bound. This results in a lower bound against BPEXP. We take the approach suggested in [20]: learn a padded version of L. Intuitively, padding allows us to increase the input size and thus allowing more learning time, without actually increasing the "real" size of the input. We try different amounts of padding until the learner has "enough running time" to learn L. Note, that the learning algorithm is guaranteed to succeed w.h.p only given enough running time/samples. Consequently, when executed prior to reaching the "right" amount of padding the acceptance/rejection probabilities of the algorithm can be arbitrarily close to half. Yet in order to remain within the framework of BPP, the algorithm must keep the promise of bounded away acceptance and rejection probabilities. The solution of [7,16,20] was to add one bit of advice indicating whether or not we have reached the necessarily amount of padding. To finish the argument, we show that small circuits for a sufficiently padded version of L imply small circuits for L itself.

(un)Keeping the Promise. For the sake of simplicity, we describe here the intuition behind removing the advice from $\mathsf{MA}/1$. We follow along the lines of the previous section with one change: instead of using the learning algorithm to a learn a non-uniform circuit C for L, we "guess" it. Next, we need to verify that C indeed computes L via a probabilistic procedure referred to as "self-testability" (Property 2 Definition 6). This procedure can detect an error in C only if C if "far enough" from L. Thus, we are dealing with an oracle call to $\mathsf{PromiseBPP}$, which can replaced by an oracle call to CA. However, the answer of the oracle is undefined for circuits that are "close" to L but not equal L. We suggest to circumvent this problem by replacing an oracle call to CA with a fixed oracle O that is consistent with CA. Given a fixed oracle O every call has a defined answer. Repeating the previous reasoning we establish a hard language in NP^O. Yet, for each O it might be a different language. Our glint of hope comes come from the following relation, which follows from the definition of $\mathsf{NP}^{\mathsf{PromiseBPP}}$ (see e.g. [4]): $\mathsf{MA} = \mathsf{NP}^{\mathsf{PromiseBPP}} \triangleq \bigcap_{O \text{ is consistent with CA}} \mathsf{NP}^O$. If one could show that a hard language in each of the NP^O terms on the RHS implies a hard language in their intersection, we would be done. We formalize this approach in Section 5.

1.3 Organization

We start by some basic definitions and notation in Section 2. In Section 3 we give our main result showing that efficient randomized learning algorithms entail circuit lower bounds, proving Theorems 1 and 2. In Section 4 we prove Theorem 3 show that for several circuit classes the very existence of efficient randomized exact learning algorithm "bumps" into the natural barier of Razborov & Rudich. In Section 5 we propose an approach for removing the extra bit of advice. Finally, we discuss some open questions in Section 6.

2 Preliminaries

Definition 1 (Functions, Circuits, Languages). *In this paper we deal with Boolean functions, that is $f : \{0,1\}^n \to \{0,1\}$. Let $f, g : \{0,1\}^n \to \{0,1\}$. We define the* relative distance $\Delta(f,g) \overset{\Delta}{=} \Pr_{\bar{a}\in\{0,1\}^n}[f(\bar{a}) \neq g(\bar{a})]$. *For $\varepsilon \geq 0$, we say that f is ε-*close *to g if $\Delta(f,g) \leq \varepsilon$, otherwise we say that f is ε-*far *from g. Let $L \subseteq \{0,1\}^*$ be a language. We denote by $L|_n$ the set of the strings of length n in L. We say that L has circuits of size $a(n)$ and denote it by $L \in \mathsf{SIZE}(a(n))$ if for every $n \geq 1$ $L|_n$ can be computed by a Boolean circuit of size $\mathcal{O}(a(n))$. A* circuit class *$\mathcal{C}$ is a subset of all Boolean circuits (e.g. circuits with AND,OR and NOT gates, $\mathsf{AC}^0, \mathsf{ACC}, \mathsf{TC}^0, \mathsf{NC}^2$ etc.). We assume that the representation in $\mathcal{C}$ is chosen in way that a size s circuit can be described using* poly(s) *bits. In addition, given a circuit $C \in \mathcal{C}$ of size s the circuit $C|_{x_i=b}$ is also in $\mathcal{C}$ and of size at most s, when $C|_{x_i=b}$ is the circuit resulting from C by fixing the variable x_i to the bit $b \in \{0,1\}$. We denote by $\mathsf{SIZE}(a(n))[\mathcal{C}]$ the set of languages having circuits of size $\mathcal{O}(a(n))$ from the class $\mathcal{C}$.*

A Promise Problem is a relaxation of a language. Formally:

Definition 2 (Promise Problems). *$\Pi = (\Pi_{YES}, \Pi_{NO})$ is a* promise problem *if $\Pi_{YES} \cap \Pi_{NO} = \emptyset$. We say that a language O is* consistent *with Π iff $x \in \Pi_{YES} \implies x \in O$ and $x \in \Pi_{NO} \implies x \notin O$. The containment of O outside of $\Pi_{YES} \dot\cup \Pi_{NO}$ can be arbitrary.*

Let Π be a promise problem and let M be a deterministic (resp. nondeterministic) polynomial-time Turing machine with an oracle access to Π. Consider a (oracle) language O consistent with Π. By definition (see e.g. [4]), M's language should not depend on the answers of the oracle when a query $q \notin \Pi_{YES} \dot\cup \Pi_{NO}$ is made. Consequently, $\mathsf{P}^{\Pi} \subseteq \mathsf{P}^{O}$ (resp. $\mathsf{NP}^{\Pi} \subseteq \mathsf{NP}^{O}$). It turns out that the following holds as well, and in fact can be considered as an alternative definition for classes of languages computed by Turing machines with oracles to promise problems.

Definition 3 (Promise Problems as Oracles)

$$\mathsf{P}^{\Pi} \ (\mathit{resp.}\ \mathsf{NP}^{\Pi}) \overset{\Delta}{=} \bigcap_{O \text{ is consistent with } \Pi} \mathsf{P}^{O} \ (\mathit{resp.}\ \mathsf{NP}^{O})$$

For more details and discussion see e.g. [4].

Definition 4 (Lower Bounds for Promise Problems). *Let $\mathcal{C}$ be a circuit class and $f(n)$ be a function. Then $\Pi \notin \mathsf{SIZE}(f(n)) \iff \forall O$ consistent with Π: $O \notin \mathsf{SIZE}(f(n))$.*

Definition 5 (Complexity Classes). *A language L is in* BPP *if there exists a polynomial-time machine $M(x,r)$ such that: $x \in L \implies \Pr_r[M(x,r) = 1] \geq 3/4, x \notin L \implies \Pr_r[M(x,r) = 1] \leq 1/4$. A language $L \in \mathsf{BPP}/1$ if in addition the machine requires an auxiliary* advice *bit b_n for each input of length n. We note that given the complement advice bit $\bar{b}_n$ the machine is not guaranteed to* *preform correctly. In particular, given $\bar{b}_n$ as the advice bit there is no promise for bounded away acceptance and rejection probabilities. Other standard complexity classes include:* $\mathsf{P}, \mathsf{PSPACE}, \mathsf{RP}, \mathsf{NP}, \mathsf{MA}$.

For the core of our proofs we require a *conveniently hard* language. Intuitively, it is language that has "nice" structural properies and yet can be used to compute functions that require "large" circuits. It is not hard to see that every conveniently hard language is computable in PSPACE. In [22] a conveniently hard PSPACE-complete language was constructed via arithmezation of the proof that $\mathsf{PSPACE} = \mathsf{IP}$ [21]. In their construction, all the "nice" structural properties follow from the properties of low-degree polynomials and of TQBF. The Embedded Hardness (Part 4) is due to the fact that given a circuit class $\mathcal{C}$, in $\mathsf{DSPACE}(\text{poly}(s))$ one can diagonalize against all the circuits of size s from $\mathcal{C}$, thus obtaining a language that can not be computed by any size s circuit. Formally:

Definition 6. *We say that a language L is* conveniently hard *if it satisfies:*

1. ***Downward Self-Reducibility:*** *we say that a language L is* downward-self-reducible *if there is a deterministic polynomial-time algorithm* $\mathsf{COMPUTE}$ *such that for all $n \geq 1$:* $\mathsf{COMPUTE}^{L|_{n-1}} = L|_n$.
2. ***Self-Testability:*** *we say that a language L is* self-testable *if there is a probabilistic polynomial-time algorithm* TEST *such that for any Boolean function $f : \{0,1\}^n \to \{0,1\}$:*
   - *If $f = L|_n$ then* $\Pr[\mathsf{TEST}^f = 1] = 1$.
   - *If $\Delta(f, L|_n) > 1/n$ then* $\Pr[\mathsf{TEST}^f = 1] \leq 2^{-10n}$.
3. ***Self-Correctability:*** *we say that a language L is* self-correctable *if there is a probabilistic polynomial-time algorithm* $\mathsf{CORRECT}$ *such that, for any Boolean function $f : \{0,1\}^n \to \{0,1\}$ it holds that if $\Delta(f, L|_n) \leq 1/n$ then for all $\bar{x} \in \{0,1\}^n$:* $\Pr[\mathsf{CORRECT}^f(\bar{x}) \neq L|_n(\bar{x})] \leq 2^{-10n}$.
4. ***Embedded Hardness:*** *we say that a language L has an* Embedded Hardness *if for every circuit class $\mathcal{C}$ and $k \geq 1$:* $\mathsf{P}^L \not\subseteq \mathsf{SIZE}(n^k)[\mathcal{C}]$.

The following is immediate from the definition:

Observation 1. *Let C be an n-variate circuit of size s such that $\Delta(C, L|_n) \leq 1/n$. Then there exists an n-variate circuit C' of size* $\text{poly}(s, n)$ *such that $\Delta(C, L|_n) = 0$. Moreover, C' can be obtained from C in polynomial time w.h.p.*

3 Lower Bounds from Randomized Learning Algorithms

In this section we prove Theorems 1 and 2. Let L be the conveniently hard PSPACE-complete language of [22]. Following and extending definitions from [20] we define padded versions of L. For simplicity, throughout the section we fix a circuit class $\mathcal{C}$. We will assume w.l.o.g that $\mathsf{P} \subseteq \mathsf{P/poly}[\mathcal{C}]$ (otherwise there is nothing to prove). As $L \in \mathsf{PSPACE} \subseteq \mathsf{EXP}$, by a translation argument there exists $d \geq 1$ such that $L \in \mathsf{SIZE}(2^{n^d})$.

Definition 7 (Padded Languages)
For $r \geq 1$ let $s(r)$ denote the size of the smallest circuit from $\mathcal{C}$ that computes $L|_r$. By the preceding discussion $s(r)$ is well-defined and in particular $s(r) = \mathcal{O}(2^{r^d})$. Let $t(w) : \mathbb{N} \to \mathbb{N}$ be a constructible function. We define the padded version of L:

$$L'_{t(\cdot)} = \left\{ 1^m x \;\middle|\; \begin{array}{l} 1)\ m \text{ is power of } 2. \\ 2)\ r \stackrel{\Delta}{=} |x| \leq m. \\ 3)\ x \in L. \\ 4)\ s(r) \leq t(m). \end{array} \right\}$$

Remark: *Condition 4 can be restated as: there exists a circuit of size $t(m)$ that computes $L|_r$. In addition, observe that for $L'_{t(\cdot)}$ each input has a unique interpretation.*

The main property of the padded languages is that sufficiently small circuits for $L'_{t(\cdot)}$ can be used to construct small circuits for L.

Lemma 1. *Let $k \geq 1$ and $t(w) = \Omega(w^{2k})$. Suppose $L'_{t(\cdot)} \in \mathsf{SIZE}(n^k)$. Then $s(r) = \mathcal{O}(r^{2k})$.*

Proof. For $n \geq 1$ let C'_n be a circuit for $L'_{t(\cdot)}|_n$. Let $r \geq 1$. We will now construct a circuit that computes $L|_r$. Take m to be a minimal power of 2 such that $r \leq m$ and $s(r) \leq t(m)$. As $t(w) = \Omega(w^{2k})$, we have that $t(m) \geq \alpha \cdot m^{2k}$, for some $\alpha > 0$. Hence, it must be the case that $m \leq 2 \cdot \alpha^{-1/2k} \cdot s(r)^{1/2k} + 2r$. Now, set $n = r + m$ and consider the circuit $\tilde{C}$ resulting from C'_n when we hardwire the lower m bits of the input to 1^m. By definition, $\tilde{C}$ computes $L|_r$ and there exists $\beta > 0$ such that $\tilde{C}$ is of size at most $\beta \cdot n^k = \beta \cdot (r+m)^k \leq \beta \cdot (3r + 2 \cdot \alpha^{-1/2k} \cdot s(r)^{1/2k})^k \leq \beta \cdot \sqrt{\alpha} \cdot 6^k \cdot r^k \cdot \sqrt{s(r)}$. By recalling the definition of $s(r)$ we get that $s(r) \leq \gamma \cdot r^k \cdot \sqrt{s(r)}$ which implies $s(r) = \mathcal{O}(r^{2k})$.

We now give the main result of this section.

Proof (of Theorem 2). Let $\mathcal{A}$ be a PAC learner for $\mathcal{C}$. Fix k. Consider two cases.

Case 1: $L \in \mathsf{P/poly}[\mathcal{C}]$. We will show that $\mathsf{P}^L \subseteq \mathsf{BPP}$ and hence $\mathsf{BPP} \not\subseteq \mathsf{SIZE}(n^k)[\mathcal{C}]$. For this purpose will use $\mathcal{A}$ to learn circuits for L and then apply Property 4 of Definition 6 as follows:

- Begin with a lookup table $\tilde{C}_1 = C_1$ for $L|_1$.
- For $i \geq 2$, invoke $\mathcal{A}$ with $\varepsilon = 1/i^3$ and $\delta = 1/i$ to learn a circuit $\tilde{C}_i$ of size $s(i)$ for $L|_i$.
- Given a membership query for $L|_i$ invoke $\mathsf{COMPUTE}$ using $C_{i-1}(x)$.
- Set $C_i \stackrel{\Delta}{=} \mathsf{CORRECT}^{\tilde{C}_i}$

Analysis. We claim that C_i computes $L|_i$ with probability at least $1 - 2^{-10i}$. By induction on i. Basis $i = 1$ is clear. Now assume that hypothesis holds for i. By definition, w.h.p $\mathcal{A}$ will output $\tilde{C}_{i+1}$ to be $1/i$ close to $L|_i$ using Property 1 of Definition 6 to answer membership queries. By Property 3 C_{i+1} will compute $L|_{i+1}$ with probability at least $1-2^{-10(i+1)}$. The total number of steps is $\text{poly}(i)$ while each has an exponential probability error. Hence, for each i: $C_i \equiv L|_i$ w.h.p.

Running Time. Given an input of size n we learn the corresponding circuits of sizes $s(1), \ldots, s(n)$ which is $\text{poly}(n)$ by assumption. In addition, all the algorithms are polynomial-time.

Case 2: $L \notin \mathsf{P/poly}[\mathcal{C}]$. Set $t(w) = w^{2k}$. We show that $L'_{t(\cdot)} \in \mathsf{BPP}/1$ and conclude that in this case $L'_{t(\cdot)} \notin \mathsf{SIZE}(n^k)[\mathcal{C}]$. We follow the learning scheme described in the previous case to learn circuits for L with two changes: first, since each input of $L'_{t(\cdot)}$ has a unique interpretation, we could use the advice bit to determine whether $s(r) \leq t(m)$. If the advice bit is 0, we reject. Otherwise, we carry on with flow of the scheme barring a second change: invoke $\mathcal{A}$ to learn circuits $\tilde{C}$ of size $t(m)$ and use the resulting circuit C_r to decide if $x \in L$. Now, suppose that $L'_{t(\cdot)} \in \mathsf{SIZE}(n^k)$. By Lemma 1 we get that $s(r) = \mathcal{O}(r^{2k})$, which contradicts the fact that $L \notin \mathsf{P/poly}[\mathcal{C}]$.

The proof of Theorem 1 is essentially the same. We leave it as an exercise for the reader. To complete the picture, recall that a randomized exact learner can be used to obtain a PAC learner with membership queries [1] and observe that $\mathsf{BPTIME}\left(t(n)\right)/1 \notin \mathsf{SIZE}\left(f(n)\right)$ implies that $\mathsf{PromiseBPTIME}\left(t(n)\right) \notin \mathsf{SIZE}\left(f(n)\right)$ by adding the advice to the input.

4 Natural Property from Learning

We now describe the approach in more details. First, we recall some related definitions from [19].

Definition 8. *Let $\mathcal{P}$ be a property of Boolean functions. That is, a subset of all Boolean functions. Let Γ, Λ be complexity classes. We say that $\mathcal{P}$ is Γ-*natural *with density* δ_n *if there is a property $\mathcal{P}^* \subseteq \mathcal{P}$ such that the following holds:* ***(i) Constructivity:*** *given a function f by its truth table, $f \in \mathcal{P}^*$ can be decided in Γ.* ***(ii) Largeness:*** *for all n, $\mathcal{P}^*$ contains at least a δ_n fraction of all n-variate Boolean functions. We say that $\mathcal{P}$ is* ***useful*** *against Λ if for every family of Boolean functions $\{f_n\}_{n\geq 1} \subseteq \mathcal{P}$ there are infinitely many n-s such that $f_n \notin \Lambda$.*

The main results of [19] (and its extensions) states as follows:

Lemma 2 ([19] and Extensions). *Let $\mathcal{C}$ be a circuit class. If there exists a $\mathsf{P/poly}$-natural property with density $2^{-\mathcal{O}(n)}$ that is useful against $\mathsf{P/poly}[\mathcal{C}]$ then no strong pseudo-random generators exist in $\mathsf{P/poly}[\mathcal{C}]$.*

We show that we can turn an efficient randomized exact learner $\mathcal{A}$ for a circuit class $\mathcal{C}$ into a $\mathsf{P/poly}$-natural property useful against $\mathcal{C}$. First, we amplify the error probability and get a $\mathsf{P/poly}$ algorithm $\mathcal{A}'_s$ which succeeds in learning all the function computable by size s circuits from $\mathcal{C}$. Next, we define the property $\mathcal{P}$ as the set of all functions on which $\mathcal{A}'_s$ fails. For an appropriate choice of s the property is useful against $\mathsf{P/poly}[\mathcal{C}]$. Moreover, observe that if $\mathcal{A}'_s$ succeeds learning a function f then f must have a small circuit. A simple counting argument shows that the majority of Boolean functions require large circuits, thus $\mathcal{A}'_s$ succeeds only a small fraction of functions. Note that since the counting argument is valid for any circuit class, the learning algorithm $\mathcal{A}$ need not be proper. Due to space limitation we omit the formal proof of Theorem 3 from this version.

5 Towards Better Lower Bounds by Unkeeping Promises

The extra bit of advice in Theorems 1 and 2 as well as in the result of [20] (lower bounds for $\mathsf{MA}/1$) comes to accommodate the need to keep the promise of bounded away probabilities of acceptance and rejection required for the "bounded" probabilistic classes. As in other cases, removing this bit is an open problem. In this section we suggest an approach how to settle this problem in the case of conditional and unconditional lower bounds. As previously, we will use the conveniently hard PSPACE-complete language L of [22].

We raise the question whether lower bounds are preserved under consistency. More specifically, given a promise problem Π our hope is that one could show that the existence of hard language in each P^O (or NP^O) implies an existence hard language in their intersection, when O runs over all the consistent with Π languages. We now give a formal treatment to this intuition.

Definition 9. *Let $\mathcal{C}$ be a circuit class. We say that a promise problem Π is* deterministically *(resp.* nondeterministically*)* compact *w.r.t. $\mathcal{C}$ if:* P^Π *(resp.* NP^Π*)* $\subseteq \mathsf{SIZE}(f(n))[\mathcal{C}] \implies \exists O$ *consistent with Π s.t.* P^O *(resp.* NP^O*)* $\subseteq \mathsf{SIZE}(f(n)^{\mathcal{O}(1)})$ $[\mathcal{C}]$.

Note that the promise problems that are languages are trivially compact for all circuit classes in both settings. We can extend the definition to classes of promise problems requiring compactness for each problem in the class. We now give the proofs starting with the conditional case. Recall that CA is the natural $\mathsf{PromiseBPP}$-complete problem of *Circuit Approximation* defined as $\mathrm{CA} \triangleq (\mathrm{CA}_{YES}, \mathrm{CA}_{NO})$ where $(\mathrm{CA}_{YES} = \{C \mid \Delta(C, \bar{0}) \geq 3/4\}, \mathrm{CA}_{NO} = \{C \mid \Delta(C, \bar{0}) \leq 1/4\})$.

Lemma 3. *Let $\mathcal{C}$ be a circuit class. Suppose that* $\mathsf{PromiseBPP}$ *is deterministically compact w.r.t. $\mathcal{C}$. If $\mathcal{C}$ is* PAC *learnable with membership queries, then for any $k \geq 1$:* $\mathsf{BPP} \not\subseteq \mathsf{SIZE}(n^k)[\mathcal{C}]$.

Proof. Fix $k \geq 1$ and assume for a contradiction that $\mathsf{BPP} \subseteq \mathsf{SIZE}(n^k)[\mathcal{C}]$. As $\mathsf{P}^{\mathrm{CA}} \subseteq \mathsf{BPP}$, from compactness there exists $k' \geq 1$ and a language O consistent with CA such that $\mathsf{P}^O \subseteq \mathsf{SIZE}(n^{k'})[\mathcal{C}]$. By Theorem 2 there exists $\Pi \in \mathsf{PromiseBPP}$ such that $\Pi \notin \mathsf{SIZE}(n^{k'})[\mathcal{C}]$. Since CA is a $\mathsf{PromiseBPP}$-complete language there exists a polynomial-time function $g : \Pi \to \mathrm{CA}$ such that: $x \in \Pi_{YES} \implies g(x) \in \mathrm{CA}_{YES}, x \in \Pi_{NO} \implies g(x) \in \mathrm{CA}_{NO}$. Let us define $O' \triangleq \{x \mid g(x) \in O\}$. First, observe that $O' \in \mathsf{P}^O$. Next, we claim that O' is consistent with Π. That is: $x \in \Pi_{YES} \implies g(x) \in \mathrm{CA}_{YES} \subseteq O \implies x \in O', x \in \Pi_{NO} \implies g(x) \in \mathrm{CA}_{NO} \subseteq \bar{O} \implies x \notin O'$. Recalling Definition 4, $O' \notin \mathsf{SIZE}(n^{k'})[\mathcal{C}]$ thus leading to a contradiction.

The unconditional case is slightly more involved. We refer the reader to the full version of the paper.

6 Discussion and Open Questions

In this paper we show that efficient *randomized* learning algorithms imply circuit lower bounds against $\mathsf{BPTIME}(n^{\omega(1)})/1$, and some other hardness results. This (almost) solves the main open problem posed in [6] and [14], and matches the corresponding result of [14] that *deterministic* learning algorithms imply circuit lower bounds against $\mathsf{DTIME}(n^{\omega(1)})$. We would like to point out that those conditional lower bounds are nearly-optimal if we treat the learning algorithms as black-boxes. More specifically, all the above results only assume an existence of an efficient learning algorithm, which is invoked as a black-box regardless of the class $\mathcal{C}$ it learns. Consequently, the obtained lowers bounds are of form "$\mathcal{C}$ is learnable $\implies \Gamma \nsubseteq \mathsf{P/poly}[\mathcal{C}]$" for every circuit class $\mathcal{C}$ and some complexity class Γ. Given that, we cannot expect to obtain conditional lower bounds of the form: "P or $\mathsf{BPP}/1 \nsubseteq \mathsf{P/poly}[\mathcal{C}]$" since $\mathsf{P} \subseteq \mathsf{BPP}/1 \subseteq \mathsf{P/poly}$ (i.e. when $\mathcal{C}$ is the class of all Boolean circuits with AND,OR and NOT gates). We refer to this as the "black-box barrier". This barrier can be seen as an analog of the relativization barrier for proving lower bounds. So, one open question is to derive lower bounds of the form "P or $\mathsf{BPP} \nsubseteq \mathsf{P/poly}[\mathcal{C}]$" from an efficient learning algorithm for some specific families of circuit classes. In the light of the above, such a conditional lower bound will have to exercise a non black-box technique.

The other open question is, naturally, to remove the extra bit of advice appearing in both conditional and the unconditional bounds. We hope that the approach described in Section 5 will be a step in the right direction.

Acknowledgment. The author would like to extend his gratitude to Dieter van Melkebeek for sharing lots of his knowledge and commenting on the first version of the paper. The author would also like to thank Eric Allender and Nader Bshouty for answering his questions and many useful conversations. Finally, the author would like to thank Akinori Kawachi, Igor Oliveira and anonymous referees for their detailed comments and suggestions.

References

1. Angluin, D.: Learning regular sets from queries and counterexamples. Inf. Comput. 75(2), 87–106 (1987)
2. Angluin, D.: Queries and concept learning. Machine Learning 2, 319–342 (1988)

3. Barak, B.: A probabilistic-time hierarchy theorem for slightly non-uniform algorithms. In: Rolim, J.D.P., Vadhan, S.P. (eds.) RANDOM 2002. LNCS, vol. 2483, pp. 194–208. Springer, Heidelberg (2002)
4. Buhrman, H., Fortnow, L.: One-sided versus two-sided error in probabilistic computation. In: Meinel, C., Tison, S. (eds.) STACS 1999. LNCS, vol. 1563, pp. 100–109. Springer, Heidelberg (1999)
5. Buhrman, H., Fortnow, L., Thierauf, T.: Nonrelativizing separations. In: Proceedings of the 13th Annual IEEE Conference on Computational Complexity (CCC), pp. 8–12 (1998)
6. Fortnow, L., Klivans, A.R.: Efficient learning algorithms yield circuit lower bounds. J. Comput. Syst. Sci. 75(1), 27–36 (2009)
7. Fortnow, L., Santhanam, R.: Hierarchy theorems for probabilistic polynomial time. In: FOCS, pp. 316–324 (2004)
8. Gentry, C., Halevi, S.: Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In: Proceedings of the 52nd Annual FOCS, pp. 107–109 (2011)
9. Goldreich, O., Zuckerman, D.: Another proof that bpp $\subseteq$ ph (and more). In: Studies in Complexity and Cryptography, pp. 40–53 (2011)
10. Harkins, R.C., Hitchcock, J.M.: Exact learning algorithms, betting games, and circuit lower bounds. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part I. LNCS, vol. 6755, pp. 416–423. Springer, Heidelberg (2011)
11. Impagliazzo, R., Wigderson, A.: Randomness vs. time: De-randomization under a uniform assumption. In: FOCS, pp. 734–743 (1998)
12. Kabanets, V., Impagliazzo, R.: Derandomizing polynomial identity tests means proving circuit lower bounds. Computational Complexity 13(1-2), 1–46 (2004)
13. Kearns, M.J., Valiant, L.G.: Cryptographic limitations on learning boolean formulae and finite automata. J. ACM 41(1), 67–95 (1994)
14. Klivans, A., Kothari, P., Oliveira, I.: Constructing hard functions from learning algorithms. In: Proceedings of the 28th Annual IEEE Conference on Computational Complexity (CCC), pp. 86–97 (2013)
15. Klivans, A.R., Sherstov, A.A.: Cryptographic hardness for learning intersections of halfspaces. J. Comput. Syst. Sci. 75(1), 2–12 (2009)
16. van Melkebeek, D., Pervyshev, K.: A generic time hierarchy with one bit of advice. Computational Complexity 16(2), 139–179 (2007)
17. Miltersen, P.B., Vinodchandran, N.V., Watanabe, O.: Super-polynomial versus half-exponential circuit size in the exponential hierarchy. In: Asano, T., Imai, H., Lee, D.T., Nakano, S.-i., Tokuyama, T. (eds.) COCOON 1999. LNCS, vol. 1627, pp. 210–220. Springer, Heidelberg (1999)
18. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. J. ACM 51(2), 231–262 (2004)
19. Razboeov, A.A., Rudich, S.: Natural proofs. J. of Computer and System Sciences 55(1), 24–35 (1997)
20. Santhanam, R.: Circuit lower bounds for merlin–arthur classes. SIAM J. Comput. 39(3), 1038–1061 (2009)
21. Shamir, A.: IP=PSPACE. In: Proceedings of the Thirty First Annual Symposium on Foundations of Computer Science, pp. 11–15 (1990)
22. Trevisan, L., Vadhan, S.P.: Pseudorandomness and average-case complexity via uniform reductions. Computational Complexity 16(4), 331–364 (2007)
23. Valiant, L.G.: A theory of the learnable. Communications of the ACM 27(11), 1134–1142 (1984)

Certificates in Data Structures[*]

Yaoyu Wang and Yitong Yin[**]

State Key Laboratory for Novel Software Technology, Nanjing University, China
yaoyu.wang.nju@gmail.com, yinyt@nju.edu.cn

Abstract. We study certificates in static data structures. In the cell-probe model, certificates are the cell probes which can uniquely identify the answer to the query. As a natural notion of nondeterministic cell probes, lower bounds for certificates in data structures immediately imply deterministic cell-probe lower bounds. In spite of this extra power brought by nondeterminism, we prove that two widely used tools for cell-probe lower bounds: richness lemma of Miltersen *et al.* [9] and direct-sum richness lemma of Pătraşcu and Thorup [15], both hold for certificates in data structures with even better parameters. Applying these lemmas and adopting existing reductions, we obtain certificate lower bounds for a variety of static data structure problems. These certificate lower bounds are at least as good as the highest known cell-probe lower bounds for the respective problems. In particular, for approximate near neighbor (ANN) problem in Hamming distance, our lower bound improves the state of the art. When the space is strictly linear, our lower bound for ANN in d-dimensional Hamming space becomes $t = \Omega(d)$, which along with the recent breakthrough for polynomial evaluation of Larsen [7], are the only two $t = \Omega(d)$ lower bounds ever proved for any problems in the cell-probe model.

1 Introduction

In static data structure problems, a database is preprocessed to form a table according to certain encoding scheme, and upon each query to the database, an algorithm (decision tree) answers the query by adaptively probing the table cells. The complexity of this process is captured by the cell-probe model for static data structures. Solutions in this model are called cell-probing schemes.

The cell-probe model plays a central role in studying data structure lower bounds. The existing cell-probe lower bounds for static data structure problems can be classified into the following three categories according to the techniques they use and the highest possible lower bounds supported by these techniques:

- Lower bounds implied by asymmetric communication complexity: Classic techniques introduced in the seminal work of Miltersen *et al.* [9] see a cell-probing scheme as a communication protocol between the query algorithm

[*] The full version of this paper can be found on http://arxiv.org/abs/1404.5743

[**] Supported by NSFC grants 61272081, 61003023 and 61321491.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 1039–1050, 2014.

and the table, and the cell-probe lower bounds are implied by the asymmetric communication complexity lower bounds which are proved by the *richness lemma* or round eliminations. In the usual setting that both query and data items are points from a d-dimensional space, the highest time lower bound that can be proved in this way is $t = \Omega\left(\frac{d}{\log s}\right)$ with a table of s cells. This bound is a barrier for the technique, because a matching upper bound can always be achieved by communication protocols.

- Lower bounds proved by self-reduction using direct-sum properties: The seminal works of Pătraşcu and Thorup [14, 15] introduce a very smart idea of many-to-one self-reductions, using which and by exploiting the direct-sum nature of problems, higher lower bounds can be proved for a near-linear space. The highest lower bounds that can be proved in this way is $t = \Omega\left(d/\log \frac{sw}{n}\right)$ with a table of s cells each containing w bits. Such lower bounds grow differently with near-linear space and polynomial space, which is indistinguishable in the communication model.
- Higher lower bounds for linear space: A recent breakthrough of Larsen [7] uses a technique refined from the cell sampling technique of Panigrahy *et al.* [10,11] to prove an even higher lower bound for the polynomial evaluation problem. This lower bound behaves as $t = \Omega(d)$ when the space is strictly linear. This separates for the first time between the cell-probe complexity with linear and near-linear spaces, and also achieves the highest cell-probe lower bound ever known for any data structure problems.

In this paper, we consider an even stronger model: *certificates* in static data structures. A query to a database is said to have certificate of size t if the answer to the query can be uniquely identified by the contents of t cells in the table. This very natural notion represents the nondeterministic computation in cell-probe model and is certainly a lower bound to the complexity of deterministic cell-probing schemes. This nondeterministic model has been explicitly considered before in a previous work [18] of one of the authors of the current paper.

Surprisingly, in spite of the seemingly extra power brought by the nondeterminism, the highest cell-probe lower bound to date is in fact a certificate lower bound [7]. Indeed, we conjecture that for typical data structure problems, especially those hard problems, *the complexity of certifying the answer should dominate that of computing the answer.*[1] This belief has been partially justified in [18] by showing that a random static data structure problem is hard nondeterministically. In this paper, we further support this conjecture by showing that several mainstream techniques for cell-probe lower bounds in fact can imply as good or even higher certificate lower bounds.

[1] Interestingly, the only known exception to this conjecture is the predecessor search problem whose cell-probe complexity is a mild super-constant while the queries can be easily certified with constant cells in a sorted table.

Table 1. Certificate lower bounds proved in this paper

problem	certificate lower bound proved here	highest known cell-probe lower bound
bit-vector retrieval	$t = \Omega\left(\frac{m \log n}{\log s}\right)$	not known
lopsided set disjointness (LSD)	$t = \Omega\left(\frac{m \log n}{\log s}\right)$	$t = \Omega\left(\frac{m \log n}{\log s}\right)$ [1,9,13]
approximate near neighbor (ANN) in Hamming space	$t = \Omega\left(d/\log \frac{sw}{nd}\right)^{\diamond}$	$t = \Omega\left(d/\log \frac{sw}{n}\right)^{\star}$ [10,15]
partial match (PM)	$t = \Omega\left(d/\log \frac{sw}{n}\right)^{\star}$	$t = \Omega\left(d/\log \frac{sw}{n}\right)^{\star}$ [10,15]
3-ANN in ℓ_∞	$t = \Omega\left(d/\log \frac{sw}{n}\right)^{\star}$	$t = \Omega\left(d/\log \frac{sw}{n}\right)^{\star}$ [15]
reachability oracle 2D stabbing 4D range reporting	$t = \Omega\left(\log n/\log \frac{sw}{n}\right)^{\star}$	$t = \Omega\left(\log n/\log \frac{sw}{n}\right)^{\star}$ [13]
2D range counting	$t = \Omega\left(\log n/\log \frac{sw}{n}\right)^{\star}$	$t = \Omega\left(\log n/\log \frac{sw}{n}\right)^{\star}$ [12,13]
approximate distance oracle	$t = \Omega\left(\frac{\log n}{\alpha \log(s \log n/n)}\right)^{\star}$	$t = \Omega\left(\frac{\log n}{\alpha \log(s \log n/n)}\right)^{\star}$ [16]

$\star$: lower bound which grows differently with near-linear and polynomial space;
$\diamond$: lower bound which grows differently with linear, near-linear, and polynomial space.

1.1 Our Contributions

We make the following contributions:

1. We prove a richness lemma for certificates in data structures, which improves the classic richness lemma for asymmetric communication complexity of Miltersen *et al.* [9] in two ways: (1) when applied to prove data structure lower bounds, our richness lemma implies lower bounds for a stronger *nondeterministic* model; and (2) our richness lemma achieves better parameters than the classic richness lemma and may imply higher lower bounds.
2. We give a scheme for proving certificate lower bounds using a similar direct-sum based self-reduction of Pătraşcu and Thorup [15]. The certificate lower bounds obtained from our scheme is at least as good as before when the space is near-linear or polynomial. And for strictly linear space, our technique may support superior lower bounds, which was impossible for the direct-sum based techniques before.
3. By applying these techniques, adopting the existing reductions, and modifying the reductions in the communication model to be model-independent, we prove certificate lower bounds for a variety of static data structure problems, listed in Table 1. All these certificate lower bounds are at least as good as the highest known cell-probe lower bounds for the respective problems. And for approximate near neighbor (ANN), our $t = \Omega\left(d/\log \frac{sw}{nd}\right)$ lower bound improves the state of the art. When the space $sw = O(nd)$ is strictly linear, our lower bound for ANN becomes $t = \Omega(d)$, which along with the recent breakthrough for polynomial evaluation [7], are the only two $t = \Omega(d)$ lower bounds ever proved for any problems in the cell-probe model.

1.2 Related Work

The richness lemma, along with the round elimination lemma, for asymmetric communication complexity was introduced in [9]. The richness lemma was later widely used, for example in [2, 3, 6, 8], to prove lower bounds for high dimensional geometric problems, e.g. nearest neighbor search. In [1, 13], a generalized version of richness lemma was proved to imply lower bounds for (Monte Carlo) randomized data structures. A direct-sum richness theorem was first proved in the conference version of [15]. Similar but less involved many-to-one reductions were used in [13] and [16] for proving lower bounds for certain graph oracles.

The idea of cell sampling was implicitly used in [12] and independently in [10]. This novel technique was later fully developed in [11] for high dimensional geometric problems and in [7] for polynomial evaluation. The lower bound in [7] actually holds for nondeterministic cell probes, i.e. certificates. The nondeterministic cell-probe complexity was studied for dynamic data structure problems in [4] and for static data structure problems in [18].

2 Certificates in Data Structures

A data structure problem is a function $f : X \times Y \to Z$ with two domains X and Y. We call each $x \in X$ a *query* and each $y \in Y$ a *database*, and $f(x, y) \in Z$ specifies the result of query x on database y. A code $T : Y \to \Sigma^s$ with an alphabet $\Sigma = \{0,1\}^w$ transforms each database $y \in Y$ to a *table* $T_y = T(y)$ of s *cells* each containing w bits. We use $[s] = \{1, 2, \dots, s\}$ to denote the set of indices of cells, and for each $i \in [s]$, we use $T_y(i)$ to denote the content of the i-th cell of table T_y.

A data structure problem is said to have (s, w, t)-*certificates*, if any database can be stored in a table of s cells each containing w bits, so that the result of each query can be uniquely determined by contents of at most t cells. Formally, we have the following definition.

Definition 1. *A data structure problem $f : X \times Y \to Z$ is said to have (s, w, t)-*certificates*, if there exists a code $T : Y \to \Sigma^s$ with an alphabet $\Sigma = \{0,1\}^w$, such that for any query $x \in X$ and any database $y \in Y$, there exists a subset $P \subseteq [s]$ of cells with $|P| = t$, such that for any database $y' \in Y$, we have $f(x, y') = f(x, y)$ if $T_{y'}(i) = T_y(i)$ for all $i \in P$.*

Because certificates represent nondeterministic computation in data structures, it is obvious that it has stronger computational power than cell-probing schemes.

Proposition 2. *For any data structure problem f, if there is a cell-probing scheme storing every database in s cells each containing w bits and answering every query within t cell-probes, then f has (s, w, t)-certificates.*

In the full version [17], we state the equivalent formulations of data structure certificates as proof systems and certificates in decision trees of partial functions.

3 The Richness Lemma

From now on, we focus on the decision problems where the output is either 0 or 1. A data structure problem $f : X \times Y \to \{0, 1\}$ can be naturally treated as an $|X| \times |Y|$ matrix whose rows are indexed by queries $x \in X$ and columns are indexed by data $y \in Y$. The entry at the x-th row and y-th column is $f(x, y)$. For $\xi \in \{0, 1\}$, we say f has a *monochromatic ξ-rectangle* of size $k \times \ell$ if there is a combinatorial rectangle $A \times B$ with $A \subseteq X, B \subseteq Y, |A| = k$ and $|B| = \ell$, such that $f(x, y) = \xi$ for all $(x, y) \in A \times B$. A matrix f is said to be *(u, v)-rich* if at least v columns contain at least u 1-entries. The following richness lemma for cell-probing schemes is introduced in [9].

Lemma 3 (Richness Lemma [9]). *Let f be a (u, v)-rich problem. If f has an (s, w, t)-cell-probing scheme, then f contains a monochromatic 1-rectangle of size $\frac{u}{2^{t \log s}} \times \frac{v}{2^{wt+t \log s}}$.*

In [9], the richness lemma is proved for asymmetric communication protocols. A communication protocol between two parties Alice and Bob is called an $[A, B]$-protocol if Alice sends Bob at most A bits and Bob sends Alice at most B bits in total in the worst-case. The richness lemma states that existence of $[A, B]$-protocol for a (u, v)-rich problem f implies a submatrix of dimension $\frac{u}{2^A} \times \frac{v}{2^{A+B}}$ containing only 1-entries. An (s, w, t)-cell-probing scheme can imply an $[A, B]$-protocol with $A = t \log s$ and $B = wt$, so the above richness lemma for the cell-probing schemes follows.

3.1 Richness Lemma for Certificates

We prove a richness result for data structure certificates, with even a better reliance on parameters.

Lemma 4 (Richness Lemma for Data Structure Certificates). *Let f be a (u, v)-rich problem. If f has (s, w, t)-certificates, then f contains a monochromatic 1-rectangle of size $\frac{u}{\binom{s}{t}} \times \frac{v}{\binom{s}{t} 2^{wt}}$.*

Remark. Note that we always have $\log \binom{s}{t} = t \log \frac{s}{t} + O(t) \leq t \log s$. The bound in Lemma 4 is at least as good as the bound in classic richness lemma, even though now it is proved for nondeterministic computation. When s and t are close to each other, the bound in Lemma 4 is substantially better than that of classic richness lemma. Later in Section 4, this extra gain is used in direct-sum reductions introduced in [15] to achieve better time lower bounds for linear or near-linear space which match or improve state of the art. It is quite shocking to see all these achieved through a very basic reduction to the 1-probe case to be introduced later.

The classic richness lemma for asymmetric communication protocol is proved by a halving argument. Due to determinism of communication protocols (and cell-probing schemes), the combinatorial rectangle obtained from halving the universe are *disjoint*. This disjointness no longer holds for the rectangles obtained from certificates because of nondeterminism. We resolve this issue by exploiting combinatorial structures of rectangles obtained from data structure certificates.

The following preparation lemma is a generalization of the averaging principle.

Lemma 5. *Let $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_r \subset 2^V$ be partitions of V satisfying $|\mathcal{P}_i| \leq k$ for every $1 \leq i \leq k$. There must exist a $y \in V$ such that $|\mathcal{P}_i(y)| \geq \frac{|V|}{rk}$ for all $1 \leq i \leq r$, where $\mathcal{P}_i(y)$ denotes the partition block $B \in \mathcal{P}_i$ containing y.*

Proof. The lemma is proved by the probabilistic method. Let y be uniformly chosen from V. Fix an arbitrary order of partition blocks for each partition $\mathcal{P}_i$. Let w_{ij} be the cardinality of the j-th block in $\mathcal{P}_i$. Obviously the probability of $\mathcal{P}_i(y)$ being the j-th block in $\mathcal{P}_i$ is $\frac{w_{ij}}{|V|}$. By union bound, the probability that $|\mathcal{P}_i(y)| < w$ is bounded by $\sum_{j:w_{ij}<w} \frac{w_{ij}}{|V|} < |\{j : w_{ij} < w\}| \frac{w}{|V|}$. Since $|\mathcal{P}_i| \leq k$, for every i there are at most k many such j satisfying that $w_{ij} < w$, thus $\Pr\left[|\mathcal{P}_i(y)| < \frac{|V|}{rk}\right] < k \cdot \frac{|V|/rk}{|V|} = \frac{1}{r}$. Applying union bound again for all $\mathcal{P}_i$, we have $\Pr\left[\exists 1 \leq i \leq r, |\mathcal{P}_i(y)| < \frac{|V|}{rk}\right] < 1$, which means there exists a $y \in V$ such that $|\mathcal{P}_i(y)| \geq \frac{|V|}{rk}$ for all $1 \leq i \leq r$.

We first prove the richness lemma for the 1-probe case.

Lemma 6. *Let f be a (u, v)-rich problem. If f has $(s, w, 1)$-certificates, then f contains a monochromatic 1-rectangle of size $\frac{u}{s} \times \frac{v}{s \cdot 2^w}$.*

Proof. Let $T : Y \to \Sigma^s$ where $\Sigma = \{0,1\}^w$ be the code in the $(s, w, 1)$-certificates for f. Let $V \subseteq Y$ denote the set of v columns of f that each contains at least u 1-entries. For each cell $1 \leq i \leq s$, an equivalence relation $\sim_i$ on databases in V can be naturally defined as follows: for any $y, y' \in V$, $y \sim_i y'$ if $T_y(i) = T_{y'}(i)$, that is, if databases y and y' look same in the i-th cell. Let $\mathcal{P}_i$ denote the partition induced by the equivalence relation $\sim_i$. Each partition $\mathcal{P}_i$ classifies the databases in V according to the content of the i-th cell. Obviously $|\mathcal{P}_i| \leq 2^w$, because the content of a cell can have at most $|\Sigma| = 2^w$ possibilities, and we also have $\mathcal{P}_i(y) = \{y' \in V \mid T_{y'}(i) = T_y(i)\}$ being the set of databases indistinguishable from y by looking at the i-th cell, where $\mathcal{P}_i(y)$ denotes the partition block $B \in \mathcal{P}_i$ containing y. By Lemma 5, there always exists a bad database $y \in V$ such that $|\mathcal{P}_i(y)| \geq \frac{|V|}{s \cdot 2^w} = \frac{v}{s \cdot 2^w}$ for all $1 \leq i \leq s$.

For each database $y \in V$, let $X_1(y) = \{x \in X \mid f(x, y) = 1\}$ denote the set of positive queries on database y, and for a subset $A \subseteq V$ of databases, let $X_1(A) = \bigcap_{y \in A} X_1(y)$ denote the set of queries which are positive on all databases in A. Note that $X_1(y)$ and $X_1(A)$ are the respective 1-preimages of Boolean functions $f(\cdot, y)$ and $\bigwedge_{y \in A} f(\cdot, y)$. By definition, it is easy to see that $X_1(A) \times A$ is a monochromatic 1-rectangle for any $A \subseteq V$.

Claim: For any $y \in V$, it holds that $X_1(y) = \bigcup_{1 \leq i \leq s} X_1(\mathcal{P}_i(y))$.

It is easy to see the direction $\bigcup_{1 \leq i \leq s} X_1(\mathcal{P}_i(y)) \subseteq X_1(y)$ holds because $X_1(A) \subseteq X_1(y)$ for any A containing y and clearly $y \in \mathcal{P}_i(y)$. So we only need to prove the other direction. Since f has $(s, w, 1)$-certificates, for any positive query

x on database y (i.e. any $x \in X_1(y)$), there is a cell i such that all databases y' indistinguishable from y by looking at the i-th cell (i.e. all $y' \in \mathcal{P}_i(y)$) answer the query x positively (i.e. $f(x, y') = f(x, y) = 1$), which gives $x \in X_1(\mathcal{P}_i(y))$ by definition of $X_1(A)$. This proves the direction $X_1(y) \subseteq \bigcup_{i \in [s]} X_1(\mathcal{P}_i(y))$.

Consider the bad database $y \in V$ satisfying $|\mathcal{P}_i(y)| \geq \frac{|V|}{s \cdot 2^w} = \frac{v}{s \cdot 2^w}$ for all $1 \leq i \leq s$. Due to the above claim, we have

$$u \leq |X_1(y)| = \left| \bigcup_{1 \leq i \leq s} X_1(\mathcal{P}_i(y)) \right| \leq \sum_{1 \leq i \leq s} |X_1(\mathcal{P}_i(y))| \,.$$

By averaging principle, there exists a cell i such that $|X_1(\mathcal{P}_i(y))| \geq \frac{u}{s}$. This gives us a monochromatic 1-rectangle $X_1(\mathcal{P}_i(y)) \times \mathcal{P}_i(y)$ of size at least $\frac{u}{s} \times \frac{v}{s \cdot 2^w}$.

The richness lemma for general case can be derived from the 1-probe case by a one-line reduction.

Lemma 7. *If a data structure problem f has (s, w, t)-certificates, then f has $\left(\binom{s}{t}, wt, 1\right)$-certificates.*

Proof. Store every t-combination of cells with a new table of $\binom{s}{t}$ cells each of $w \cdot t$ bits.

In the full version [17] we apply Lemma 4 to prove certificate lower bounds for bit-vector retrieval problem and lopsided set disjointness problem.

4 Direct-Sum Richness Lemma

In this section, we prove a richness lemma for certificates using direct-sum property of data structure problems. Such a lemma was introduced in [15] for cell-probing schemes, which is used to prove some highest known cell-probe lower bounds with *near-linear spaces.*

Consider a vector of problems $\bar{f} = (f_1, \ldots, f_k)$ where every $f_i : X \times Y \to \{0, 1\}$ is defined on the same domain $X \times Y$. Let $\bigoplus^k \bar{f} : ([k] \times X) \times Y^k \to \{0, 1\}$ be a problem defined as follows: $\bigoplus^k \bar{f}((i, x), \bar{y}) = f_i(x, y_i)$ for every $(i, x) \in [k] \times X$ and every $\bar{y} = (y_1, y_2, \ldots, y_k) \in Y^k$. In particular, for a problem f we denote $\bigoplus^k f = \bigoplus^k \bar{f}$ where $\bar{f}$ is a tuple of k copies of problem f.

Lemma 8 (Direct-Sum Richness Lemma for Certificates). *Let $\bar{f} = (f_1, f_2 \ldots f_k)$ be a vector of problems such that for each $i = 1, 2, \ldots, k$, we have $f_i : X \times Y \to \{0, 1\}$ and f_i is $(\alpha|X|, \beta|Y|)$-rich. If problem $\bigoplus^k \bar{f}$ has (s, w, t)-certificates for a $t \leq \frac{s}{k}$, then there exists a $1 \leq i \leq k$ such that f_i contains a monochromatic 1-rectangle of size $\frac{\alpha^{O(1)}|X|}{2^{O(t \log \frac{s}{kt})}} \times \frac{\beta^{O(1)}|Y|}{2^{O(wt + t \log \frac{s}{kt})}}$.*

Remark 1. The direct-sum richness lemma proved in [15] is for asymmetric communication protocols as well as cell-probing schemes, and gives a rectangle size of $\frac{\alpha^{O(1)}|X|}{2^{O(t \log \frac{s}{k})}} \times \frac{\beta^{O(1)}|Y|}{2^{O(wt + t \log \frac{s}{k})}}$. Our direct-sum richness lemma has a better rectangle bound. This improvement may support stronger lower bounds which separate between linear and near-linear spaces.

Remark 2. A key idea to apply this direct sum based lower bound scheme is to exploit the extra power gained by the model from solving k problem instances in parallel. In [15], this is achieved by seeing cell probes as communications between query algorithm and table, and t-round adaptive cell probes for answering k parallel queries can be expressed in $t \log \binom{s}{k}$ bits instead of naively $kt \log s$ bits. For our direct-sum richness lemma for certificates, in contrast, we will see (in Lemma 9) that unlike communications, the parallel simulation of certificates does not give us any extra gain, however, in our case all extra gains are provided by the improved bound in Lemma 4, the richness lemma for certificates. Indeed, all our extra gains by "parallelism" are offered by the one-line reduction in Lemma 7, which basically says that the certificates for k instances of a problem can be expressed in $\log \binom{s}{kt}$ bits, even better than the $t \log \binom{s}{k}$-bit bound for communications. Giving up adaptivity is essential to this improvement on the power of parallelism, so that all kt cells can be chosen at once which gives the $\log \binom{s}{kt}$-bit bound: *we are now not even parallel over instances, but also parallel over time.*

The idea of proving Lemma 8 can be concluded as: (1) reducing the problem $\bigoplus^k \bar{f}$ from a direct-product problem $\bigwedge^k \bar{f}$ whose richness and monochromatic rectangles can be easily translated between $\bigwedge^k \bar{f}$ and subproblems f_i; and (2) applying Lemma 4, the richness lemma for certificates, to obtain large monochromatic rectangles for the direct-product problem.

We first define a direct-product operation on vector of problems. For $\bar{f} = (f_1, \dots, f_k)$ with $f_i : X \times Y \to \{0,1\}$ for every $1 \le i \le k$, let $\bigwedge^k \bar{f} : X^k \times Y^k \to \{0,1\}$ be a direct-product problem defined as: $\bigwedge^k \bar{f}(\bar{x}, \bar{y}) = \prod_i f_i(x_i, y_i)$ for every $\bar{x} = (x_1, \dots, x_k)$ and every $\bar{y} = (y_1, \dots, y_k)$.

Lemma 9. *For any $\bar{f} = (f_1, \dots, f_k)$, if $\bigoplus^k \bar{f}$ has (s, w, t)-certificates for a $t \le \frac{s}{k}$, then $\bigwedge^k \bar{f}$ has (s, w, kt)-certificates.*

Proof. Suppose that $T : Y^k \to \Sigma^s$ with $\Sigma = \{0,1\}^w$ is the code used to encode databases to tables in the (s, w, t)-certificates of $\bigoplus^k \bar{f}$. For problem $\bigwedge^k \bar{f}$, we use the same code T to prepare table. And for each input $(\bar{x}, \bar{y})$ of problem $\bigwedge^k \bar{f}$ where $\bar{x} = (x_1, \dots, x_k)$ and $\bar{y} = (y_1, \dots, y_k)$, suppose that for each $1 \le i \le k$, $P_i \subset [s]$ with $|P_i| = t$ is the set of t cells in table $T_{\bar{y}}$ to uniquely identify the value of $\bigoplus^k \bar{f}((i, x_i), \bar{y})$, then let $P = P_1 \cup P_2 \cup \dots \cup P_k$ so that $|P| \le kt$. It is easy to verify that the set P of at most kt cells in $T_{\bar{y}}$ uniquely identifies the value of $\bigwedge^k \bar{f}(\bar{x}, \bar{y}) = \bigwedge_{1 \le i \le k} \left(\bigoplus^k \bar{f}((i, x_i), \bar{y}) \right)$ because it contains all cells which can uniquely identify the value of $\bigoplus^k \bar{f}((i, x_i), \bar{y})$ for every $1 \le i \le k$. Therefore, problem $\bigwedge^k \bar{f}$ has (s, w, kt)-certificates.

The following two lemmas are from [15]. These lemmas give easy translations of richness and monochromatic rectangles between the direct-product problem $\bigwedge^k \bar{f}$ and subproblems f_i.

Lemma 10 (Pătraşcu and Thorup [15]). *For problem vector $\bar{f} = (f_1, f_2 \dots f_k)$, if $f_i : X \times Y \to \{0,1\}$ is $(\alpha|X|, \beta|Y|)$-rich for every $1 \le i \le k$, then $\bigwedge^k \bar{f}$ is $((\alpha|X|)^k, (\beta|Y|)^k)$-rich.*

Lemma 11 (Pătraşcu and Thorup [15]). *For any $\bar{f} = (f_1, \dots, f_k)$ with $f_i : X \times Y \to \{0,1\}$ for every $1 \le i \le k$, if $\bigwedge^k \bar{f}$ contains a monochromatic 1-rectangle of size $(\alpha|X|)^k \times (\beta|Y|)^k$, then there exists a $1 \le i \le k$ such that f_i contains a monochromatic 1-rectangle of size $(\alpha)^3|X| \times (\beta)^3|Y|$.*

The direct-sum richness lemma can be easily proved by combining the above lemmas with the richness lemma for certificates.

Proof (Proof of Lemma 8). If $\bigoplus^k \bar{f}$ has (s, w, t)-certificates, then by Lemma 9, the direct-product problem $\bigwedge^k \bar{f}$ has (s, w, kt)-certificates. Since every f_i in $\bar{f} = (f_1, f_2, \dots, f_k)$ is $(\alpha|X|, \beta|Y|)$-rich, by Lemma 10 we have that $\bigwedge^k \bar{f}$ is $((\alpha|X|)^k, (\beta|Y|)^k)$-rich. Applying Lemma 4, the richness lemma for certificates, problem $\bigwedge^k \bar{f}$ has a 1-rectangle of size $\frac{(\alpha|X|)^k}{\binom{s}{kt}} \times \frac{(\beta|Y|)^k}{\binom{s}{kt}2^{kwt}}$. Then due to Lemma 11, we have a problem f_i who contains a monochromatic 1-rectangle of size $\frac{\alpha^{O(1)}|X|}{2^{O(t \log \frac{s}{kt})}} \times \frac{\beta^{O(1)}|Y|}{2^{O(wt + t \log \frac{s}{kt})}}$.

4.1 Applications

We then apply the direct-sum richness lemma to prove lower bounds for two important high dimensional problems: approximate near neighbor (ANN) in hamming space and partial match (PM).

- For ANN in d-dimensional hamming space, we prove a $t = \Omega(d / \log \frac{sw}{nd})$ lower bound for (s, w, t)-certificates. The highest known cell-probing scheme lower bound for the problem is $t = \Omega(d / \log \frac{sw}{n})$. In a super-linear space, our certificate lower bound matches the highest known lower bound for cell-probing scheme; and for linear space, our lower bound becomes $t = \Omega(d)$, which gives a strict improvement, and also matches the highest cell-probe lower bound ever known for any problem (which has only been achieved for polynomial evaluation [7]).
- For d-dimensional PM, we prove a $t = \Omega(d / \log \frac{sw}{n})$ lower bound for (s, w, t)-certificates, which matches the highest known cell-probing scheme lower bound for the problem in [15].

Approximate Near Neighbor (ANN). The near neighbor problem NN_n^d in a d-dimensional metric space is defined as follows: a database y contains n points from a d-dimensional metric space, for any query point x from the same space and a distance threshold λ, the problem asks whether there is a point in database y within distance λ from x. The approximate near neighbor problem $\mathrm{ANN}_n^{\lambda,\gamma,d}$ is similarly defined, except upon a query x to a database y, answering "yes" if there is a point in database y within distance λ from x and "no" if all points in y are $\gamma\lambda$-far away from x (and answering arbitrarily if otherwise).

We first prove a lower bound for $\mathrm{ANN}_n^{\lambda,\gamma,d}$ in Hamming space $X = \{0,1\}^d$, where for any two points $x, x' \in X$ the distance between them is given by Hamming distance $h(x,x')$.

The richness and monochromatic rectangles of $\mathrm{ANN}_n^{\lambda,\gamma,n}$ were analyzed in [8].

Claim 12 (Claim 10 and 11 in [8]). *There is a $\lambda \le d$ such that $\mathrm{ANN}_n^{\lambda,\gamma,d}$ is $(2^{d-1}, 2^{nd})$-rich and $\mathrm{ANN}_n^{\lambda,\gamma,d}$ does not contain a 1-rectangle of size $2^{d-d/(169\gamma^2)} \times 2^{nd-nd/(32\gamma^2)}$.*

A model-independent self-reduction of ANN was constructed in [15].

Claim 13 (Theorem 6 in [15]). *For $D = d/(1+5\gamma) \ge \log n$, $N < n$ and $k = n/N$, there exist two functions ϕ_X, ϕ_Y such that ϕ_X (and ϕ_Y) maps each query (x,i) (and database $\bar{y}$) of $\bigoplus^k \mathrm{ANN}_N^{\lambda,\gamma,D}$ to a query x' (and database y') of $\mathrm{ANN}_n^{\lambda,\gamma,d}$ and it holds that $\bigoplus^k \mathrm{ANN}_N^{\lambda,\gamma,D}((x,i),\bar{y}) = \mathrm{ANN}_n^{\lambda,\gamma,d}(x',y')$.*

We then prove the following certificate lower bound for ANN.

Theorem 14. *For $\mathrm{ANN}_n^{\lambda,\gamma,d}$ in d-dimensional Hamming space, assuming $d \ge (1+5\gamma)\log n$, there exists a λ, such that if $\mathrm{ANN}_n^{\lambda,\gamma,d}$ has (s,w,t)-certificates, then $t = \Omega\left(\frac{d}{\gamma^3}/\log\frac{sw\gamma^3}{nd}\right)$.*

Proof. Due to the model-independent reduction from $\bigoplus^k \mathrm{ANN}_N^{\lambda,\gamma,D}$ to $\mathrm{ANN}_n^{\lambda,\gamma,d}$ of Claim 13, existence of (s,w,t)-certificates for $\mathrm{ANN}_n^{\lambda,\gamma,d}$ implies the existence of (s,w,t)-certificates for $\bigoplus^k \mathrm{ANN}_N^{\lambda,\gamma,D}$.

Note that for problem $\mathrm{ANN}_N^{\lambda,\gamma,D}$, the size of query domain is $|X| = 2^D$, and the size of data domain is $|Y| = 2^{ND}$, so applying Claim 12, the problem is $(|X|/2, |Y|)$-rich. Assuming that $t \le \frac{s}{k}$, by Lemma 8, $\mathrm{ANN}_N^{\lambda,\gamma,D}$ contains a 1-rectangle of size $2^D/2^{O(t\log\frac{s}{kt})} \times 2^{ND}/2^{O(wt+t\log\frac{s}{kt})}$. Due to Claim 12, and by a calculation, we have either $t = \Omega\left(\frac{D}{\gamma^2}/\log\frac{s}{kt}\right)$ or $t = \Omega\left(\frac{ND}{\gamma^2}/w\right)$. We then choose $N = w$. Note that such choice of N may violate the assumption $t \le \frac{s}{k}$ (that is, $N \ge \frac{tn}{s}$) only when it implies an even higher lower bound $t > \frac{sw}{n}$. With this choice of $N = w$, the bound $t = \Omega\left(\frac{D}{\gamma^2}/\log\frac{s}{kt}\right)$ is the smaller one in the two branches. Substituting $D = d/(1+5\gamma)$ and $k = n/N$ we have $t = \Omega\left(\frac{d}{\gamma^3}/\log\frac{sN}{nt}\right) = \Omega\left(\frac{d}{\gamma^3}/\log\frac{sw}{nt}\right)$. Multiplying both side by a $\Delta = \frac{sw}{nd}$ gives us $\Delta\cdot\gamma^3 = \Omega\left(\frac{\Delta d}{t}/\log\frac{\Delta d}{t}\right)$. Assuming $\Delta' = \frac{\Delta d}{t}$, we have $\frac{\Delta'}{\log\Delta'} = O(\Delta\gamma^3)$. The function $f(x) = \frac{x}{\log x}$ is increasing for $x > 1$, so we have $\Delta' = O(\Delta\gamma^3\log(\Delta\gamma^3))$, which gives us the lower bound $t = \Omega\left(\frac{d}{\gamma^3}/\log\frac{sw\gamma^3}{nd}\right)$.

For general space, when points are still from the Hamming cube $\{0,1\}^d$, for any two points $x, x' \in \{0,1\}^d$, the Hamming distance $h(x,x') = \|x-x'\|_1 = \|x-x'\|_2^2$. And by setting $\gamma = 1$, we have the following corollary for exact near neighbor.

Corollary 15. *There exists a constant C such that for problem NN_n^d with Hamming distance, Manhattan norm ℓ_1 or Euclidean norm ℓ_2, assuming $d \geq C \log n$, if NN_n^d has (s, w, t)-certificates, then $t = \Omega(d / \log \frac{sw}{nd})$.*

Partial Match. The partial match problem is another fundamental high-dimensional problem. The d-dimensional partial match problem PM_n^d is defined as follows: a database y contains n strings from $\{0, 1\}^d$, for any query pattern $x \in \{0, 1, *\}^d$, the problem asks whether there is a string z in database y matching pattern x, in such a way that $x_i = z_i$ for all $i \in [d]$ that $x_i \neq *$.

Theorem 16. *Assuming $d \geq 2 \log n$, if problem PM_n^d has (s, w, t)-certificates for a $w = d^{O(1)}$, then $t = \Omega\left(d / \log \frac{sd}{n}\right)$.*

The proof of this theorem is in the full version [17].

It is well known that partial match can be reduced to 3-approximate near neighbor in ℓ_∞-norm by a very simple reduction [5]. We write 3-$\mathrm{ANN}_n^{\lambda,d}$ for $\mathrm{ANN}_n^{\lambda,3,d}$.

Theorem 17. *Assuming $d \geq 2 \log n$, there is a λ such that if 3-$\mathrm{ANN}_n^{\lambda,d}$ in ℓ_∞-norm has (s, w, t)-certificates for a $w = d^{O(1)}$, then $t = \Omega(d / \log \frac{sd}{n})$.*

Proof. We have the following model-independent reduction. For each query pattern x of partial match, we make the following transformation to each coordinate: $0 \to -\frac{1}{2}; * \to \frac{1}{2}; 1 \to \frac{3}{2}$. For a string in database the ℓ_∞-distance is $\frac{1}{2}$ if it matches pattern x and $\frac{3}{2}$ if otherwise.

5 Lower Bounds Implied by Lopsided Set Disjointness

It is observed in [13] that a variety of cell-probe lower bounds can be deduced from the communication complexity of one problem, the lopsided set disjointness (LSD). In [16], the communication complexity of LSD is also used to prove the cell-probe lower bound for approximate distance oracle.

In this section, we modify these communication-based reductions to make them model-independent. A consequence of this is a list of certificate lower bounds as shown in Table 1 for: 2-Blocked-LSD, reachability oracle, 2D stabbing, 2D range counting, 4D range reporting, and approximate distance oracle.

The rest of this section can be found in the full version [17].

Acknowledgment. We are deeply grateful to Kasper Green Larsen for helpful discussions about lower bound techniques in cell-probe model.

References

1. Andoni, A., Indyk, P., Pătraşcu, M.: On the optimality of the dimensionality reduction method. In: Proc. 47th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 449–458 (2006)

2. Barkol, O., Rabani, Y.: Tighter lower bounds for nearest neighbor search and related problems in the cell probe model. Journal of Computer and System Sciences 64(4), 873–896 (2002)
3. Borodin, A., Ostrovsky, R., Rabani, Y.: Lower bounds for high dimensional nearest neighbor search and related problems. In: Proc. 31st ACM Symposium on Theory of Computing (STOC), pp. 312–321 (1999)
4. Husfeldt, T., Rauhe, T.: Hardness results for dynamic problems by extensions of fredman and saks' chronogram method. In: Larsen, K.G., Skyum, S., Winskel, G. (eds.) ICALP 1998. LNCS, vol. 1443, pp. 67–78. Springer, Heidelberg (1998)
5. Indyk, P.: On approximate nearest neighbors under ℓ_∞ norm. Journal of Computer and System Sciences 63(4), 627–638 (2001)
6. Jayram, T.S., Khot, S., Kumar, R., Rabani, Y.: Cell-probe lower bounds for the partial match problem. In: Proc. 35th ACM Symposium on Theory of Computing (STOC), pp. 667–672 (2003)
7. Larsen, K.G.: Higher cell probe lower bounds for evaluating polynomials. In: Proc. 53rd IEEE Symposium on Foundations of Computer Science (FOCS), pp. 293–301 (2012)
8. Liu, D.: A strong lower bound for approximate nearest neighbor searching. Information Processing Letters 92(1), 23–29 (2004)
9. Miltersen, P.B., Nisan, N., Safra, S., Wigderson, A.: On data structures and asymmetric communication complexity. Journal of Computer and System Sciences 57(1), 37–49 (1998)
10. Panigrahy, R., Talwar, K., Wieder, U.: A geometric approach to lower bounds for approximate near-neighbor search and partial match. In: Proc. 49th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 414–423 (2008)
11. Panigrahy, R., Talwar, K., Wieder, U.: Lower bounds on near neighbor search via metric expansion. In: Proc. 51th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 805–814 (2010)
12. Pătraşcu, M.: Lower bounds for 2-dimensional range counting. In: Proc. 30th ACM Symposium on Theory of Computing (STOC), pp. 40–46 (2007)
13. Pătraşcu, M.: Unifying the landscape of cell-probe lower bounds. SIAM Journal on Computing 40(3), 827–847 (2011), See also FOCS 2008
14. Pătraşcu, M., Thorup, M.: Time-space trade-offs for predecessor search. In: Proc. 38th ACM Symposium on Theory of Computing (STOC), pp. 232–240 (2006)
15. Pătraşcu, M., Thorup, M.: Higher lower bounds for near-neighbor and further rich problems. SIAM Journal on Computing 39(2), 730–741 (2010); See also FOCS 2006
16. Sommer, C., Verbin, E., Yu, W.: Distance oracles for sparse graphs. In: Proc. 50th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 703–712 (2009)
17. Wang, Y., Yin, Y.: Certificates in data structures, `http://arxiv.org/abs/1404.5743`
18. Yin, Y.: Cell-probe proofs. ACM Transactions on Computation Theory (TOCT) 2(1), 1 (2010)

Optimal Query Complexity for Estimating the Trace of a Matrix

Karl Wimmer[1,*], Yi Wu[2,**], and Peng Zhang[2]

[1] Duquesne University
[2] Purdue University

Abstract. Given an implicit $n \times n$ matrix A with oracle access $x^T Ax$ for any $x \in \mathbb{R}^n$, we study the query complexity of randomized algorithms for estimating the *trace* of the matrix. This problem has many applications in quantum physics, machine learning, and pattern matching. Two metrics are commonly used for evaluating the estimators: i) variance; ii) a high probability multiplicative-approximation guarantee. Almost all the known estimators are of the form $\frac{1}{k}\sum_{i=1}^{k} x_i^T Ax_i$ for $x_i \in \mathbb{R}^n$ being i.i.d. for some special distribution.

Our main results are summarized as follows:

1. We give an *exact* characterization of the minimum variance unbiased estimator in the broad class of *linear nonadaptive* estimators (which subsumes all the existing known estimators).
2. We also consider the query complexity lower bounds for *any* (possibly nonlinear and adaptive) estimators:
   (a) We show that *any* estimator requires $\Omega(1/\epsilon)$ queries to have a guarantee of variance at most ϵ.
   (b) We show that *any* estimator requires $\Omega(\frac{1}{\epsilon^2}\log\frac{1}{\delta})$ to achieve a $(1 \pm \epsilon)$-multiplicative approximation guarantee with probability at least $1 - \delta$.

   Both above lower bounds are asymptotically tight.

As a corollary, we also resolve a conjecture in the seminal work of Avron and Toledo (Journal of the ACM 2011) regarding the sample complexity of the Gaussian Estimator.

1 Introduction

Given an $n \times n$ matrix $A = \{A_{ij}\}_{1\leq i\leq n, 1\leq j\leq n}$, we study the problem of estimating its trace

$$trace(A) = \sum_{i=1}^{n} A_{ii}$$

[*] Most of the work is done when the author is visiting the Simons Institute for the Theory of Computing, University of California-Berkeley. Supported in part by NSF grant CCF-1117079.

[**] Most of the work is done when the author is visiting the Simons Institute for the Theory of Computing, University of California-Berkeley.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 1051–1062, 2014.

with a randomized algorithm that can query $f_A(x) = x^T Ax$ for any $x \in \mathbb{R}^n$. The goal is to minimize the number of queries used to achieve certain type of accuracy guarantee, such as the variance of the estimate or a multiplicative approximation (which holds with high probability). Finding an estimator that achieves such an accuracy guarantee with few queries has several applications. For example, this problem is well studied in the subject of lattice quantum chromodynamics, since such queries are physically feasible and can be used to efficiently estimate the trace of a function of a large matrix $f(A)$. Such an estimator can also be used as a building block for many other applications including solving least-squares problems [Hut89], computing the number of triangles in a graph [Avr10, Tso08], and string pattern matching [ACD01, AGW13].

This problem has been well studied in the literature. All of the previously analyzed estimators are of the form $\frac{1}{k}\sum_{i=1}^{k} x_i^T Ax_i$ for $x_1, x_2, \ldots, x_k \in \mathbb{R}^n$; nearly all take $x_1, x_2, \ldots, x_k$ to be independent and identically distributed (i.i.d.) from some well designed distribution. For example, in [Hut89], the author just takes each query to be a random vector whose entries are i.i.d. Rademacher random variables (i.e., each coordinate is a uniformly random sample from $\{-1, 1\}$); we call this the Rademacher estimator. There are also several other alternative distributions on $x_1, x_2, \ldots, x_k$, such as drawing each query from a multivariate normal distribution [SR97]; we call this the Gaussian estimator. Here, the coordinates of each vectors are i.i.d. Gaussian random variables. The work of [IE04] considers the case where only one query is allowed, but that query can be a unit vector in $\mathbb{C}^n$. Other estimators occur in [DS93, Wan94]. Recent work by [AT11], the authors propose several new estimators such as the unit vector estimator, normalized Rayleigh-quotient trace estimator, and the mixed unit vector estimator. One estimator that does not use i.i.d. queries is due to [RKA13]; in that work, the authors propose querying random standard basis vectors without replacement.

To characterize the performance of an estimator, perhaps the most natural metric is the variance of the estimator. It is known that the Gaussian estimator has variance $2\|A\|_F^2$ and the random Rademacher vector estimator has variance $2(\|A\|_F^2 - \sum_{i=1}^{n} A_{ii}^2)$, where $\|A\|_F = \sqrt{trace(A^T A)}$ is the Frobenius norm. In recent work by Avron and Toledo [AT11], it is suggested that the notion of a multiplicative approximation guarantee might be a better success metric of an estimator than the variance. Formally, we say an estimator is an (ϵ, δ)-estimator if it outputs an estimate in the interval $((1-\epsilon)trace(A), (1+\epsilon)trace(A))$ with probability at least $1 - \delta$. It should be noted that some assumptions on the matrices need to be made to have a valid (ϵ, δ)-estimator, as it is impossible to achieve any multiplicative approximation when the matrix could have a trace of 0. A natural choice is to assume that A comes from the class of symmetric positive semidefinite (SPD) matrices. For a SPD matrix, the authors in [AT11] prove that the Gaussian estimator with $k = O(\frac{1}{\epsilon^2}\log(\frac{1}{\delta}))$ queries to the oracle is an (ϵ, δ)-estimator. It was recently shown in [RKA13] that the random Rademacher vector estimator is also an (ϵ, δ)-estimator with the same sample complexity.

An open problem asked in [AT11] is the following: does the Gaussian estimator require $\Omega(\frac{1}{\epsilon^2}\log(\frac{1}{\delta}))$ in order to be an (ϵ, δ)-estimator? The authors showed that this number of queries suffices and conjectured that their analysis of the Gaussian estimator is tight with supporting evidence from empirical experiments. The paper gives some intuition on how to show an $\Omega(\frac{1}{\epsilon^2})$ lower bound. The authors suggested that the difficulty of turning this argument into a formal proof is that "current bounds [on the χ^2 cumulative distribution function] are too complex to provide a useful lower bound". Regarding lower bounds for trace estimators, we note the related work of [LNW14], which considers the problem of sketching the nuclear norms of A using bilinear sketches (which can be viewed as nonadaptive queries of the form $x^T Ay$). The problem is similar to estimating trace when the underlying matrix is positive semidefinite.

All of the above mentioned estimators (with one exception in [RKA13]) use independent identically distributed queries from some special distributions, and the output is a linear combination of the query results. On the other hand, we view an estimator as a randomized algorithm, so we can choose any distribution over the queries, and the output can be any (possibly randomized) function of the results of the queries. Given the success of the previously mentioned estimators, it is natural to ask whether these extensions are helpful. For example, can we get a significantly better estimator with a non i.i.d. distribution? Can we do better with adaptive queries? Can we do better with a nonlinear combination of the query results?

In this paper, we make progress on answering above questions and understanding the optimal query complexity for randomized trace estimators. Below is an informal summary of our results.

1. Among all the linear nonadaptive trace estimators (which subsumes all the existing trace estimators), we prove that the "random k orthogonal vector" estimator is the minimum variance estimator. The distribution on the queries is *not* i.i.d., and we are unable to find an occurrence of this estimator in the literature regarding trace estimators.
2. We also prove two asymptotically optimal lower bounds for any (possibly adaptive and possibly nonlinear) estimator.
   (a) We show that *every* trace estimator requires $\Omega(1/\epsilon)$ queries to have a guarantee that the variance of the estimator is at most ϵ.
   (b) We show that *every* (ϵ, δ)-estimator requires $\Omega(\frac{1}{\epsilon^2}\log\frac{1}{\delta})$ queries.

As a simple corollary, our result also confirms the above mentioned conjecture in [AT11] (as well as the tightness of the analysis of the Rademacher estimator in [RKA13]). Notice our result is a much stronger statement: the original conjectured lower bound is only for an estimator that returns a linear combination of i.i.d. Gaussian queries; we prove the lower bound holds for any estimator. Our lower bound also suggests that adaptiveness as well as nonlinearity will not help asymptotically as all these lower bounds are matched by the nonadaptive Gaussian estimator. On the other hand, our upper bound suggests that the exact minimum variance estimator might not use i.i.d. queries.

1.1 Problem Definitions

Definition 1 (Estimator for the Trace). *A trace estimator is a possibly randomized algorithm that, given query access to an oracle $f_A(\cdot)$ for an unknown $n \times n$ matrix A, makes a sequence of k queries $x_1, x_2, \ldots, x_k \in \mathbb{R}^n$ to the oracle and receives $f_A(x_1), f_A(x_2), \ldots, f_A(x_k)$. The output of the estimator is a real number $h(A)$ determined by the queries and the answers to the queries (and possibly uses randomness).*

Definition 2 (Nonadaptive Linear Unbiased Trace Estimator). *We say a trace estimator is* nonadaptive *if the distribution of x_i is not dependent on $f_A(x_1), f_A(x_2), \ldots, f_A(x_{i-1})$. A trace estimator is* linear *if we sample from a distribution over k queries as well as their weights: $(x_1, x_2, \ldots, x_k)$, and $(w_1, w_2, \ldots, w_k)$, and output $\sum w_i f_A(x_i)$. In addition, a linear trace estimator is* unbiased *if*

$$\mathop{\mathbf{E}}_{w_1,w_2,\ldots,w_n,x_1,x_2,\ldots,x_n}[\sum_{i=1}^{n} w_i f_A(x_i)] = \mathit{trace}(A)$$

Without loss of generality, we can assume that all the queries in a linear estimator are of unit length, where the actual lengths of the queries are absorbed by the weights.

The most natural measure of quality of an estimator is its variance. There is a large body of work on the existence of and finding a *minimum variance unbiased estimator*. Such an estimator has a strong guarantee; it is the estimator for which the variance is minimized for all possible values of the parameter to estimate. In general, finding such an estimator is quite difficult. It is easy to see that the variance depends on the scale of the matrix. To normalize, we assume that the Frobenius norm of the matrix is fixed.

Definition 3. *We define the variance of a trace estimator as the worst case of variance over all matrices with Frobenius norm* 1*. To be specific, given a matrix A let us define $\mathit{Var}(A,h) = \mathbf{E}[(h(A) - \mathit{trace}(A))^2]$, then*

$$\mathit{Var}(h) = \sup_{\|A\|_F^2 = 1} \mathit{Var}(A,h).$$

If the variance of an estimator h is at most δ, we say that h is a δ-variance estimator.

Given an unbiased estimator class, the *minimum variance unbiased estimator* has the minimum variance among all the (unbiased) estimators in the class.

Another natural accuracy guarantee for a trace estimator is the notion of (ϵ, δ)-estimator that is introduced in [AT11].

Definition 4 ((ϵ, δ)-estimator). *A trace estimator h is said to be an (ϵ, δ)-estimator of the trace if, for every matrix A, we have that $|\mathit{trace}(A) - h(A)| \leq \epsilon \cdot \mathit{trace}(A)$ with probability at least $1 - \delta$.*

We stress that both Definitions 3 and 4 involve worst case estimates over the choice of the matrix, and the randomness only comes from the internal randomness of the estimator.

Definition 5 (Random Gaussian Matrix and Random Orthogonal Matrix).

- *We call a vector $g \in \mathbb{R}^n$ a random Gaussian vector if each coordinate is sampled independently from $N(0,1)$.*
- *We call an $n \times n$ matrix G a random Gaussian matrix if its entries are sampled independently from $N(0,1)$.*
- *We call an $n \times n$ matrix U a random orthogonal matrix if it is drawn from the distribution whose probability measure is the Haar measure on the group of orthogonal matrices; specifically, it is the unique probability measure that is invariant under orthogonal transformations.*
- *We call k vectors $x_1, x_2, \ldots, x_k \in \mathbb{R}^n$ k random orthogonal unit vectors if they are chosen as k row vectors of a random orthogonal matrix.*

We note that one way to generate a random orthogonal matrix is to generate a random Gaussian matrix and perform Gram-Schmidt orthonormalization on its rows.

1.2 Main Results

Our main results are the following:

Theorem 1. *Among all linear nonadaptive unbiased trace estimators, the minimum variance unbiased estimator that makes k queries is achieved by sampling k random orthogonal unit vectors (see Definition 5) $x_1, x_2, \ldots, x_k$ and outputting $\frac{n}{k}\sum_i f_A(x_i)$.*

Theorem 2. *Any trace estimator with variance ϵ requires $\Omega(1/\epsilon)$ queries.*

Theorem 3. *Any (ϵ, δ)-estimator for the trace requires $\Omega(\frac{1}{\epsilon^2}\log(\frac{1}{\delta}))$ queries, even if the unknown matrix is known to be positive semidefinite.*

The bounds in Theorem 2 and 3 are tight: both bounds can be asymptotically matched by the Gaussian estimator and the uniform Rademacher vector estimator.

1.3 Proof Techniques Overview

All of our results crucially use a powerful yet simple trick, which we call *symmetrization.* The heart of this trick lies in the fact that the trace of a matrix is unchanged under similarity transformations; $\mathit{trace}(A) = \mathit{trace}(U^T AU)$ for every A and orthogonal U. If we have a nonadaptive estimator with query distribution $(x_1, x_2, \ldots, x_k) \sim P$ and an orthogonal matrix U, using the queries distributed as $(Ux_1, Ux_2, \ldots, Ux_k)$ should not be too different in terms of worst-case behavior. (We have to be more careful with adaptive estimators, which we discuss in Section 2.) Thus, applying symmetrization to a nonadaptive estimator yields a nonadaptive estimator where it draws queries as in the original estimator, but transforms the queries using a random orthogonal transformation. This "symmetrizes" the estimator. We prove that the performance of the estimator never decreases when symmetrization is applied, so we can exclusively consider symmetrized estimators.

In order to characterize the minimum variance linear nonadaptive unbiased estimator, we notice that after the symmetrization, the distribution over queries for any such estimator is defined by a distribution over the pairwise angles of the k queries. We then show that the queries should be taken to be orthogonal with certainty in order to minimize variance.

As for the lower bounds for adaptive and nonlinear estimators, the symmetrization also plays an important role. Consider the problem of proving a query lower bound for (ϵ, δ)-approximation: the most common approach of proving such a lower bound is to use Yao's minimax principle. To apply this principle, we would need to construct two distributions of matrices such that the distributions cannot be distinguished after making a number of queries, even though the traces of the matrices are very different in the two distributions. There are several technical difficulties in applying the minimax principle directly here. First of all, the query space is $\mathbb{R}^n$, so it is unclear whether one can assert that there exists a sufficiently generalized minimax principle to handle this case. Second, even if one can apply a suitable version of minimax principle, we do not have general techniques of analyzing the distribution of k adaptive queries, especially when the queries involve real numbers and thus the algorithm might have infinitely many branches.

We overcome the above two barriers and avoid using a minimax principle entirely by applying symmetrization. One nice property of the symmetrization process is that a symmetrized estimator outputs the same distribution of results on all matrices with the same diagonalization. In the proof we carefully construct two distributions of matrices with the same diagonalization in each distribution, while the traces are different for different distributions. Each distribution is simply the "orbit" of a single diagonal matrix D; the support consists of all matrices similar to D. Using the symmetrization, it suffices to show that we can not distinguish these two distributions of matrices by k adaptive queries, as it is equivalent to distinguish two diagonal matrices for symmetrized trace estimators. The argument for achieving a lower bound for adaptive estimators is more subtle. Roughly speaking, we show that due to the structure of symmetrized estimators, we define a stronger query model such that adaptive estimators behave the same as the nonadaptive estimators while we achieve the same lower bound, even with the stronger query model.

1.4 Organization

In section 2, we introduce the idea of symmetrization. We prove Theorem 1 in section 3. Due to the space constraint, proofs of Theorem 2 and Theorem 3 are omitted, and they are included in our full version.

2 Symmetrization of an Estimator

In this section, we introduce the idea of symmetrization of an estimator which is a crucial element of all our remaining proofs. We first define the *rotation* of an estimator, which we will denote h^U for an $n \times n$ orthogonal matrix U. Intuitively, the construction of h^U is such that h^U emulates the behavior of h on a rotated version of the matrix A. More specifically, h^U makes queries in the following way:

- Letting q_1 be a random variable whose distribution is the same as the first query of h, the distribution of the first query of h^U is the same as the random variable Uq_1.
- Given queries $Uq_1, Uq_2, \ldots, Uq_{j-1}$ made by h^U so far with responses $t_1, t_2, \ldots, t_{j-1}$, the distribution of the jth query of h^U has the same distribution as Uq_j, where q_j is distributed the same as the jth query that h makes, given queries $q_1, q_2, \ldots, q_{j-1}$ with responses $t_1, t_2, \ldots, t_{j-1}$.

In the case that h is a nonadaptive estimator, the queries of h^U are just $Ux_1, Ux_2, \ldots, Ux_k$, where $x_1, x_2, \ldots, x_k$ is a set of queries from the distribution of queries that h makes.

Lemma 1. *For any estimator h and orthogonal matrix U,*

- *$Var(h^U) = Var(h)$.*
- *h is an (ϵ, δ)-approximation estimator if and only if h^U is also an (ϵ, δ)-estimator.*

Proof. We know that given a matrix A, the behavior of h^U is the same as h on estimating $U^T AU$. On the other hand, we know that $trace(U^T AU) = trace(A)$ and $\|A\|_F = \|U^T AU\|_F$. Therefore, the variance of h^U on A is the same as the variance of h on $U^T AU$. Now suppose h is an (ϵ, δ)-estimator. We know that the approximation guarantee of h^U on A is the same as h on $U^T AU$. Therefore, we know that with probability at least $(1 - \delta)$, the estimator h^U's output is within

$$\big((1-\epsilon) trace(U^T AU), (1+\epsilon) trace(U^T AU)\big) = \left((1-\epsilon) trace(A), (1+\epsilon) trace(A)\right).$$

Definition 6 (Averaging Estimators over a Distribution). *Suppose we have a collection of estimators H, for any probability distribution P on H, we define $h_{H,P}$ as the following estimator:*

1. *Randomly sample an estimator $h \sim P$.*
2. *Output according to the estimation of h.*

Lemma 2. *Averaging a collection of estimators cannot increase variance or weaken an (ϵ, δ)-guarantee. Specifically:*

- *If all the estimators H are unbiased and have variance at most c, then $h_{H,P}$'s variance is also at most c.*
- *If all the estimators in H are (ϵ, δ)-estimators, then $h_{H,P}$ is also an (ϵ, δ)-estimator.*

Proof. For the first, we apply the law of total variance conditioned on the draw of $h \sim P$:

$$Var[h_{H,P}] = \mathop{\mathbf{E}}_{h \sim P}[Var[h]] + \mathop{Var}_{h \sim P}[\mathbf{E}[h]]$$

The second term above is 0, since all estimators in H are unbiased. Since $Var[h] \leq c$ for every $h \in H$, $\mathbf{E}_{h \sim P}[Var[h]] \leq c$ as well.

For the second claim, assuming that

$$\mathbf{Pr}[h(A) \in ((1-\epsilon)trace(A), (1+\epsilon)trace(A))] \geq 1 - \delta$$

for each $h \in H$, we have

$$\mathbf{Pr}[h_{H,P}(A) \in ((1-\epsilon)trace(A), (1+\epsilon)trace(A))] \geq$$
$$\inf_{h \in H} \mathbf{Pr}[h(A) \in ((1-\epsilon)trace(A), (1+\epsilon)trace(A))] \geq 1 - \delta.$$

Definition 7 (Symmetrization of a Trace Estimator). *Given an estimator h, we define the symmetrization h^{sym} of h to be the estimator where we*

1. *sample a random orthogonal matrix U (see definition 5), and*
2. *use h^U to estimate the trace.*

We say an estimator is symmetric *if it is equivalent to the symmetrization of some estimator.*

By Lemma 1 and Lemma 2, we know that h^{sym}'s variance as well as its (ϵ, δ)-approximation is always no worse than h. Therefore, without loss of generality, we can always assume that the optimal estimator is symmetric.

One nice property of the symmetric estimator is that it has the same performance on all matrices with the same diagonalization.

Lemma 3. *Given a symmetrized estimator h^{sym}, its variance and approximation guarantee is the same for any matrix A and $U^T A U$ for any orthogonal matrix U.*

Proof. Given any matrix A, we know that the variance of h^{sym} is $\mathbf{E}_{U_1}[Var(h, U_1^T A U_1)]$ and the variance of h^{sym} on the matrix $U^T A U$ is $\mathbf{E}_{U_1}[Var(h, (U_1 U)^T A U_1 U)]$. We know that $U_1 U$ is also is a “uniformly” random orthogonal matrix. Therefore, h^{sym} has the same estimation variance on A and $U^T A U$.

Similarly, for the approximation guarantee, suppose h^{sym} is an (ϵ, δ)-estimator, which means that

$$\mathop{\mathbf{E}}_{U_1}\left[\mathbf{Pr}\left(h(U_1^T A U_1) \in ((1-\epsilon)\mathit{trace}(A), (1+\epsilon)\mathit{trace}(A))\right)\right] \geq 1-\delta.$$

We know that for the matrix $A' = UAU$, $U_1^T A' U_1 = U_1^T U^T A U U_1$ has the same distribution as $U_1^T A U_1$. Therefore,

$$\begin{aligned}&\mathop{\mathbf{E}}_{U_1}\left[\mathbf{Pr}\left(h(U_1^T A U_1) \in ((1-\epsilon)\mathit{trace}(A), (1+\epsilon)\mathit{trace}(A))\right)\right]\\ &\qquad = \mathop{\mathbf{E}}_{U_1}\left[\mathbf{Pr}\left(h(U_1^T A' U_1) \in ((1-\epsilon)\mathit{trace}(A'), (1+\epsilon)\mathit{trace}(A'))\right)\right]\end{aligned}$$

3 Optimal Linear Nonadaptive Estimator

Without loss of generality, we can assume that the optimal estimator is symmetric. For a symmetric nonadaptive estimator, we can think of $x_1, x_2, \ldots, x_k$ as generated by the following process.

- Sample a configuration $\theta = \{\theta_{ij}\}_{1 \leq i < j \leq k}$ from some distribution P_Θ . For each configuration θ, there is a corresponding weight vector $w^\theta = (w_1^\theta, w_2^\theta, \ldots, w_k^\theta)$.
- Generate $x_1, x_2, \ldots, x_k$ by drawing k random unit vectors conditioned on the angle between x_i, x_j being θ_{ij} for all $i < j$. (This can be done efficiently.)
- Output $\sum_{i=1}^k w_i^\theta f_A(x_i)$.

The proof of Theorem 1 consists of two steps. First we will show that we can set all of the angles (deterministically) to be $\frac{\pi}{2}$ without increasing the variance, so we can assume that the queries are orthogonal. In the second step, we will then show that the optimal way of assigning weight is to (deterministically) set each weight to be $\frac{n}{k}$.

We first prove that we can replace the queries $x_1, x_2, \ldots, x_k$ by k random orthogonal unit vectors without increasing the variance.

Lemma 4. *Let $y_1, y_2, \ldots, y_k$ be k randomly orthogonal unit vectors. We have that*

$$\mathit{Var}\left(\sum_{i=1}^k w_i^\theta f_A(y_i)\right) \leq \mathit{Var}\left(\sum_{i=1}^k w_i^\theta f_A(x_i)\right) \tag{1}$$

Proof. It is easy to see that the marginal distribution on each x_i is the same as the marginal distribution on y_i. Therefore, we have that

$$\mathop{\mathbf{E}}_{\theta, y_1, \ldots, y_k}\left[\sum_{i=1}^k w_i^\theta f_A(y_i)\right] = \mathop{\mathbf{E}}_{x_1, \ldots, x_k, \theta}\left[\sum_{i=1}^k w_i^\theta f_A(x_i)\right] = \mathit{trace}(A)$$

This implies that $\sum_{i=1}^k w_i^\theta f_A(y_i)$ is also an unbiased estimator.

Since both estimators have the same expectation, in order to show (1), it suffices to prove that

$$\mathop{\mathbf{E}}_{\theta,x_1,\ldots,x_n}\left[\left(\sum_{i=1}^{k} w_i^\theta f_A(x_i)\right)^2\right] \geq \mathop{\mathbf{E}}_{\theta,y_1,\ldots,y_n}\left[\left(\sum_{i=1}^{k} w_i^\theta f_A(y_i)\right)^2\right] \tag{2}$$

By the process of generating $x_1, x_2, \ldots, x_k$, we know that the marginal distribution of x_i is independent of θ and equal to the marginal distribution of y_i. Therefore, in order to prove (2), it suffices to prove that for any i and j, we have

$$\mathop{\mathbf{E}}_{\theta,x_i,x_j}[w_i^\theta w_j^\theta f_A(x_i) f_A(x_j)] \geq \mathop{\mathbf{E}}_{\theta}[w_i^\theta w_j^\theta] \mathop{\mathbf{E}}_{y_i,y_j}[f_A(y_i) f_A(y_j)] \tag{3}$$

To compare $\mathbf{E}_{\theta,x_i,x_j}[w_i^\theta w_j^\theta f_A(x_i) f_A(x_j)]$ and $\mathbf{E}_\theta[w_i^\theta w_j^\theta]\,\mathbf{E}_{y_i,y_j}[f_A(y_i) f_A(y_j)]$, we note that the marginal distribution on the pair (x_i, x_j) is equivalent to drawing x_i, x_j from the following process:

1. Draw $\theta \sim P_\Theta$.
2. Set $x_i = y_i$ and $x_j = y_i \cos\theta_{ij} + y_j \sin\theta_{ij}$

It is easy to check that the joint distribution on x_i and x_j has the same distribution as two random unit vectors with angle θ_{ij}.

Therefore,

$$\begin{aligned}
&\mathop{\mathbf{E}}_{\theta,x_i,x_j}[w_i^\theta w_j^\theta f_A(x_i) f_A(x_j)] \\
&= \mathop{\mathbf{E}}_{\theta,y_i,y_j}[w_i^\theta w_j^\theta y_i^T A y_i (\cos\theta_{ij} \cdot y_i + \sin\theta_{ij} \cdot y_j)^T A (\cos\theta_{ij} \cdot y_i + \sin\theta_{ij} \cdot y_j)] \\
&= \mathop{\mathbf{E}}_{\theta}[w_i^\theta w_j^\theta \cos^2\theta_{ij}] \mathop{\mathbf{E}}_{y_i}[y_i^T A y_i \cdot y_i^T A y_i] + \mathop{\mathbf{E}}_{\theta}[w_i^\theta w_j^\theta \sin^2\theta_{ij}] \mathop{\mathbf{E}}_{y_i,y_j}[y_i^T A y_i y_j^T A y_j] \\
&+ \mathop{\mathbf{E}}_{\theta}[w_i^\theta w_j^\theta \sin\theta_{ij}\cos\theta_{ij}] \mathop{\mathbf{E}}_{y_i,y_j}[y_i^T A y_i y_i^T A y_j + y_i^T A y_i y_j^T A y_i]
\end{aligned} \tag{4}$$

In order to simplify the above expression, we first claim that

$$\mathop{\mathbf{E}}_{y_i,y_j}[y_i^T A y_i y_i^T A y_j + y_i^T A y_i y_j^T A y_i] = 0.$$

To see this, note that y_j is a random unit vector orthogonal to y_i. Although y_i, y_j are dependent, conditioned on any fixed realization of y_i, the distribution on y_j is symmetric about $\mathbf{0}$; y_j has the same distribution as $-y_j$.

In addition, using Cauchy-Schwarz and the fact that y_i and y_j have the same distribution, we have that

$$\mathop{\mathbf{E}}_{y_i}[(y_i^T A y_i)^2] = \sqrt{\mathop{\mathbf{E}}_{y_i}[(y_i^T A y_i)^2] \mathop{\mathbf{E}}_{y_j}[(y_j^T A y_j)^2]} \geq \mathop{\mathbf{E}}_{y_i,y_j}[y_j^T A y_j \cdot y_i^T A y_i].$$

Therefore, we have that

$$\begin{aligned}
(4) &\geq \mathop{\mathbf{E}}_{\theta}[w_i^\theta w_j^\theta \cos^2\theta] \mathop{\mathbf{E}}_{y_i,y_j}[y_j^T A y_j \cdot y_i^T A y_i] + \mathop{\mathbf{E}}_{\theta}[w_i^\theta w_j^\theta \sin^2\theta] \mathop{\mathbf{E}}_{y_i,y_j}[y_i^T A y_i y_j^T A y_j] \\
&= \mathop{\mathbf{E}}_{\theta}[w_i^\theta w_j^\theta] \cdot \mathop{\mathbf{E}}_{y_i,y_j}[y_j^T A y_j \cdot y_i^T A y_i]
\end{aligned}$$

which proves (3), completing the proof of Lemma 4.

Now that we can assume that the queries are mutually orthogonal, we can view this as an estimator with randomized weights w_i^θ for $\theta \sim P_\Theta$. Below we will use the random variable w_i to denote w_i^θ as θ is independent from $y_1, y_2, \ldots, y_k$.

Lemma 5. *Let $(y_1, \ldots, y_k)$ be k random orthogonal unit vectors. Then the estimator $h = \sum_{i=1}^k w_i f_A(y_i)$ has minimum variance when $w_1 = w_2 = \cdots = w_k = n/k$.*

Proof. First we must have $\mathbf{E}[\sum_{i=1}^k w_i] = n$ to make the estimator unbiased, since $\mathbf{E}[f_A(y_i)] = \frac{1}{n} trace(A)$. Also,

$$\mathbf{E}\left[\left(\sum_{i=1}^k w_i\right)^2\right] = \mathbf{E}\left[\sum_{i=1}^k w_i^2\right] + 2 \cdot \mathbf{E}\left[\sum_{1 \le i < j \le k} w_i w_j\right] \ge n^2$$

Minimizing the variance is equivalent to minimizing

$$\begin{aligned}
&\mathop{\mathbf{E}}_{w,y}\left[\left(\sum_{i=1}^k w_i f_A(y_i)\right)^2\right] \\
&= \sum_{i=1}^k \mathbf{E}[f_A(y_i)^2]\,\mathbf{E}[w_i^2] + 2 \cdot \sum_{1 \le i < j \le k} \mathbf{E}[f_A(y_i) f_A(y_j)] \cdot \mathbf{E}[w_i w_j] \\
&= \mathbf{E}\left[\sum_{i=1}^k w_i^2\right] \mathbf{E}[f_A^2(y_1)] + \left(\mathbf{E}\left[\left(\sum_{i=1}^k w_i\right)^2\right] - \mathbf{E}\left[\sum_{i=1}^k w_i^2\right]\right) \mathbf{E}[f_A(y_1) f_A(y_2)] \\
&\ge \frac{n^2}{k}\,\mathbf{E}[f_A(y_1)^2] + \left(n^2 - \frac{n^2}{k}\right) \mathbf{E}[f_A(y_1) f_A(y_2)] \\
&= \mathbf{E}\left[\left(\sum_{i=1}^k \frac{n}{k} f_A(y_i)\right)^2\right]
\end{aligned}$$

Equality holds for $w_1 = \cdots = w_k = n/k$, completing the proof.

Combining Lemma 4 and Lemma 5, the minimum variance linear nonadaptive unbiased estimator making k queries is $\sum_{i=1}^k \frac{n}{k} f_A(y_i)$, where $y_1, y_2, \ldots, y_k$ is a collection of random orthogonal unit vectors. This completes the proof of Theorem 1.

Acknowledgement. The second author is grateful for Yi Li and Siu-On Chan for helpful discussions.

References

[ACD01] Atallah, M.J., Chyzak, F., Dumas, P.: A randomized algorithm for approximate string matching. Algorithmica 29(3), 468–486 (2001)

[AGW13] Atallah, M.J., Grigorescu, E., Wu, Y.: A lower-variance randomized algorithm for approximate string matching. Information Processing Letters 113(18), 690–692 (2013)

[AT11] Avron, H., Toledo, S.: Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. Journal of the ACM (JACM) 58(2), 8 (2011)

[Avr10] Avron, H.: Counting triangles in large graphs using randomized matrix trace estimation. In: Proceedings of KDD-LDMTA, vol. 10 (2010)

[DS93] Drabold, D.A., Sankey, O.F.: Maximum entropy approach for linear scaling in the electronic structure problem. Physical Review Letters 70(23), 3631 (1993)

[Hut89] Hutchinson, M.F.: A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. Communications in Statistics-Simulation and Computation 18(3), 1059–1076 (1989)

[IE04] Iitaka, T., Ebisuzaki, T.: Random phase vector for calculating the trace of a large matrix. arXiv preprint cond-mat/0401202 (2004)

[LNW14] Li, Y., Nguyên, H.L., Woodruff, D.P.: On sketching matrix norms and the top singular vector. In: Proceedings of the Twenty Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (2014)

[RKA13] Roosta-Khorasani, F., Ascher, U.: Improved bounds on sample size for implicit matrix trace estimators (2013)

[SR97] Silver, R.N., Röder, H.: Calculation of densities of states and spectral functions by chebyshev recursion and maximum entropy. Physical Review E 56(4), 4822 (1997)

[Tso08] Tsourakakis, C.E.: Fast counting of triangles in large real networks without counting: Algorithms and laws. In: Eighth IEEE International Conference on Data Mining, ICDM 2008, pp. 608–617. IEEE (2008)

[Wan94] Wang, L.-W.: Calculating the density of states and optical-absorption spectra of large quantum systems by the plane-wave moments method. Physical Review B 49(15), 10154 (1994)

Faster Separators for Shallow Minor-Free Graphs via Dynamic Approximate Distance Oracles

Christian Wulff-Nilsen

Department of Computer Science, University of Copenhagen
koolooz@di.ku.dk

Abstract. Plotkin, Rao, and Smith (SODA'97) showed that any graph with m edges and n vertices that excludes K_h as a depth $O(\ell \log n)$-minor has a separator of size $O(n/\ell + \ell h^2 \log n)$ and that such a separator can be found in $O(mn/\ell)$ time. A time bound of $O(m + n^{2+\epsilon}/\ell)$ for any constant $\epsilon > 0$ was later given (W., FOCS'11) which is an improvement for non-sparse graphs. We give three new algorithms. The first two have the same separator size (the second having a slightly larger dependency on h) and running time $O(\mathrm{poly}(h)\ell n^{1+\epsilon})$ and $O(\mathrm{poly}(h)(\sqrt{\ell} n^{1+\epsilon} + n^{2+\epsilon}/\ell^{3/2}))$, respectively. The former is significantly faster than previous bounds for small h and ℓ. Our third algorithm has running time $O(\mathrm{poly}(h)\sqrt{\ell} n^{1+\epsilon})$. It finds a separator of size $O(n/\ell) + \tilde{O}(\mathrm{poly}(h)\ell\sqrt{n})$[1] which is no worse than previous bounds when h is fixed and $\ell = \tilde{O}(n^{1/4})$. A main tool in obtaining our results is a decremental approximate distance oracle of Roditty and Zwick.

1 Introduction

Given an undirected graph with a non-negative vertex weight function, a separator of G is a subset of vertices whose removal partitions G into connected components none of which contain more than a c-fraction of the total vertex weight of the graph, where $c < 1$ is a constant. Clearly, any graph contains a separator (the entire vertex set). The goal is to find separators of small size.

A celebrated theorem of Lipton and Tarjan [4] states that every planar graph of size n has an $O(\sqrt{n})$-size separator which can be found in linear time. Alon, Seymour, and Thomas [1] generalized this to minor-free graphs.

In this paper we study the bigger class of shallow minor-free graphs. A depth ℓ-minor of a graph is a minor where every vertex corresponds to a contracted subgraph of radius at most ℓ. This class was introduced by Plotkin, Rao, and Smith [6]. They gave applications of these graphs to, e.g., geometry. Shallow minors are used to distinguish between somewhere and nowhere dense graphs [5].

Plotkin, Rao, and Smith showed that an n-vertex graph excluding K_h as a depth $O(\ell \log n)$-minor has a separator of size $O(n/\ell + \ell h^2 \log n)$. They gave an algorithm with $O(mn/\ell)$ running time which either outputs a depth $O(\ell \log n)$-minor or a separator of size $O(n/\ell + h^2 \ell \log n)$. For non-sparse graphs, running

[1] We use $\tilde{O}$-notation instead of O-notation when suppressing polylogarithmic factors.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 1063–1074, 2014.

time was improved to $O(n^{2+\epsilon}/\ell)$ for an arbitrarily small constant $\epsilon > 0$ with only a constant-factor increase (depending on ϵ) in the separator size [11].

We give three new algorithms to find separators in shallow minor-free graphs. The first achieves the same separator size as in [6] but with a running time of $O(h\ell m^{1+\epsilon} + h^2 n \log n)$ for an arbitrarily small constant $\epsilon > 0$. For small h and ℓ, this is near-linear time and a significant improvement over the near-quadratic time in [11]. It is possible to replace m by n in the time bound if we allow the algorithm to reject if there exists a shallow minor without actually outputting such a minor. Our second algorithm achieves essentially the same separator size in time $O(\text{poly}(h)(\sqrt{\ell}n^{1+\epsilon} + n^{2+\epsilon}/\ell^{3/2}))$. The special case $\ell = \sqrt{n}$ gives the same time bound for minor-free graphs as in [11]. Our third algorithm is the fastest, achieving a running time of $O(\text{poly}(h)\sqrt{\ell}n^{1+\epsilon})$. It finds a separator of size $O(n/\ell) + \tilde{O}(\text{poly}(h)\ell\sqrt{n})$ which is no worse than the size achieved in [6] when $\ell = \tilde{O}(n^{1/4})$. A main tool to achieve these new time bounds is a novel application of a dynamic approximate distance oracle of Roditty and Zwick [8]. We believe this connection is interesting in itself and should further motivate the study of dynamic distance oracles, an area which has only recently received attention from the research community.

The organization of the paper is as follows. We first give basic definitions, notation, and results in Section 2. Then we give a generic algorithm in Section 3 which is quite similar to that of Plotkin, Rao, and Smith. All our algorithms are implementations of this generic algorithms. Then our three algorithms are given in Sections 4, 5, and 6, respectively. Finally, we conclude in Section 7. Some details are omitted due to space constraints and will appear in a full version.

2 Preliminaries

We consider undirected graphs only. For a graph G, $V(G)$ resp. $E(G)$ denotes the vertex set resp. edge set of G and for a subset X of V, $G[X]$ is the subgraph of G induced by X. Connected components are referred to simply as *components*. For vertices $u, v \in V$, $d_G(u, v)$ denotes the shortest path distance between u and v in G. This definition is extended to edge-weighted graps.

Consider a graph $G = (V, E, w : V \to \mathbb{R})$ with a non-negative vertex weight function w. For any subset X of V, let $w(X) = \sum_{v \in X} w(v)$. A *separator* of G is a subset S of V such that for each component C of $G[V \setminus S]$ we have $w(C) \leq cw(V)$, for some constant $c < 1$.

Given graphs G and H, H is a *minor* of G if H can be obtained from a subgraph of G by edge contractions. Otherwise, we say that G is *H-minor-free* or that G *excludes H as a minor*. If H is a minor of G such that the subgraphs of G corresponding to vertices of H have radius at most ℓ, H is a *minor of depth ℓ*. If not, G *excludes H as a minor of depth ℓ*. We only consider complete excluded minors, i.e., H is of the form K_h. For our application, this is without loss of generality since if G excludes a minor H (of some depth), it also excludes K_h as a minor (of some depth), where $h = |V(H)|$; furthermore, our bounds only have small polynomial dependency on the size of the excluded minor.

For integers $h \geq 1$ and $\ell \geq 0$, let $\mathcal{G}_{h,\ell}$ denote the set of graphs that exclude K_h as a minor of depth ℓ. We have the following simple lemma

Lemma 1. *For any h, $\mathcal{G}_{h,0} \supseteq \mathcal{G}_{h,1} \supseteq \mathcal{G}_{h,2} \supseteq \ldots$.*

An essential tool in our algorithms is a decremental approximate distance oracle of Roditty and Zwick which can report approximate distances for close vertex pairs [8]. The following lemma states the performance of this oracle.

Lemma 2. *Let $k, d \in \mathbb{N}$ and let G be a given graph with integer edge weights and with m edges and n vertices. There is a data structure of size $O(m+n^{1+1/k})$ which can be maintained under edge deletions in G in $O(dmn^{1/k})$ total expected time such that after each edge deletion, for any vertices u, v in G, an estimate $\tilde{d}_G(u,v)$ of the shortest path distance $d_G(u,v)$ can be reported in $O(k)$ time satisfying the following: If $d_G(u,v) \leq d$ then $d_G(u,v) \leq \tilde{d}_G(u,v) \leq (2k-1)d_G(u,v)$ and if $d_G(u,v) > d$ then $d_G(u,v) \leq \tilde{d}_G(u,v)$. A uv-path achieving this distance estimate can be reported in constant time per vertex, starting from either u or v.*

The last part of the lemma is not stated in [8] but it follows immediately from the observation that the oracle maintains cluster structures (see definition in [10]) and paths can be traversed in constant time per vertex using these.

3 A Generic Algorithm

All of our algorithms are implementations of the same generic algorithm (except the algorithm in Section 5 which implements a slight variant) and we refer to it as `genericalg`$(G = (V, E, w : V \to \mathbb{R}), h, \ell)$; see pseudocode in Figure 1. It is similar to that of Plotkin, Rao, and Smith [6] but with some subtle differences that we come back to later. We refer to [6] for a less compact description. Subsets V', M, V_r, B, and A of V are maintained where initially $V' = V$ and $M = V_r = B = A = \emptyset$ [2]. The algorithm attempts to form a separator of G in a number of iterations of the while-loop in lines 2–17. Set M is the union of vertex sets of trees and $\mathcal{M}$ denotes the set of these trees. At any given time, the size p of $\mathcal{M}$ is at most h and the trees of $\mathcal{M}$ form the minor K_p of depth $O(\ell \log n)$ in G. At the beginning of each iteration, trees that are no longer incident to V' in G are removed from $\mathcal{M}$ (lines 4–5). Set A is monotonically growing and consists at any given time of the vertices that previously belonged to M but have since been removed from this set. Set V' is a monotonically shrinking set and consists of vertices yet to be processed. The algorithm terminates if each component of $G[V']$ has vertex weight at most $\frac{2}{3}w(V)$.

In the following, let $\rho = 2\lceil \ell \ln n \rceil$. In each iteration, the algorithm attempts to find a tree T in $G[V \setminus A]$ which is rooted at an arbitrary $u \in V'$, has radius at most $C\rho$ for a constant $C \geq 1$, and is incident in G to all trees in $\mathcal{M}$. If such a tree is found, the trees in $\mathcal{M} \cup \{T\}$ form the minor K_{p+1} of depth $O(\ell \log n)$ in G. In this case, T is added to $\mathcal{M}$. If $p + 1 = h$, the algorithm outputs the

[2] Here we use a similar naming convention as in [6].

```
Algorithm genericalg($G = (V, E, w : V \to \mathbb{R}), h, \ell$)

1.  initialize $V' = V$, $M = \mathcal{M} = V_r = B = A = \emptyset$, and $\rho = 2\lceil \ell \ln n \rceil$
2.  while there is a component $X$ of $G[V']$ with $w(X) > \frac{2}{3}w(V)$ and $|\mathcal{M}| < h$
3.    update $V'$ to $X$ and move any remaining components to $V_r$
4.    for each $T \in \mathcal{M}$ that is not incident to $V'$ in $G$
5.      $\mathcal{M} \leftarrow \mathcal{M} \setminus \{T\}$, $M \leftarrow M \setminus V(T)$, $V_r \leftarrow V_r \cup V(T)$, $A \leftarrow A \cup V(T)$
6.    let $u$ be any vertex in $V'$
7.    if a tree $T$ is identified in $G[V \setminus (M \cup A)]$ rooted at $u$, with radius $\leq C\rho$,
      (for some constant $C \geq 1$) and incident in $G$ to all trees in $\mathcal{M}$
8.      update $\mathcal{M} \leftarrow \mathcal{M} \cup \{T\}$, $M \leftarrow M \cup V(T)$, and $V' \leftarrow V' \setminus V(T)$
9.    else // it is assumed that $v$ can be found in line 10
10.     pick a $v \in V'$ which is incident in $G$ to $M$ such that $d_{G[V']}(u, v) > \rho$
11.     letting $w$ be any of $u$ and $v$, start growing a BFS tree in $G[V']$ from $w$
12.     for each BFS layer $N$ explored
13.       let $S$ be the set of vertices explored in all layers so far
14.       let $S'$ be the set of smaller vertex weight among $S$ and $V' \setminus (S \setminus N)$
15.       if $|N| \leq |S'|/\ell$
16          update $B \leftarrow B \cup N$, $V_r \leftarrow V_r \cup S' \setminus N$, and $V' \leftarrow V' \setminus S'$
17          terminate the BFS
18. if $|\mathcal{M}| = h$ then output $\mathcal{M}$ as a $K_h$-minor of $G$ of depth $O(\ell \log n)$
19. else output $M \cup B$ as a separator of $G$
```

Fig. 1. A generic algorithm for either finding a separator or reporting K_h as a depth $O(\ell \log n)$-minor in a vertex-weighted graph $G = (V, E, w)$ where $n = |V|$

$p + 1$ trees found as they constitute a certificate that G contains K_h as a minor of depth $O(\ell \log n)$.

Suppose such a tree T could not be found. Then it is assumed in line 10 that a vertex $v \in V'$ incident in G to M exists such that $d_{G[V']}(u, v) > \rho$. (this assumption holds for the algorithm in [6] with $C = 1$ since in this case there is a tree $T' \in \mathcal{M}$ such that for any $v \in V'$ incident to T' in G, $d_{G[V \setminus A]}(u, v) > C\rho$.) Then either u or v is picked; call it w. A BFS tree is grown from w in $G[V']$ until some layer N is small compared to S and $V' \setminus (S \setminus N)$, where S is the set of vertices explored so far. More precisely, the process stops if $|N| \leq |S'|/\ell$, where S' is the set of smaller vertex weight among S and $V' \setminus S$. Using the same analysis as in [6], such a layer N will eventually be found. For assume otherwise. Since $d_{G[V']}(u, v) > \rho$, more than ρ layers are explored by the BFS. For each layer explored, either $|S|$ grows by at least a factor of $1 + 1/\ell$ or $|V' \setminus (S \setminus N)|$ is reduced by at least a factor of $1 + 1/\ell$. Since $(1 + 1/\ell)^{\rho/2} \geq n$, after ρ layers, either S contains at least n vertices or $V' \setminus (S \setminus N)$ contains less than 1 vertex, which is a contradiction.

Correctness: To prove correctness of the generic algorithm (under the assumption stated in line 9), note that the output in line 18 is indeed a K_h-minor of depth $O(\ell \log n)$. We need to show that in line 19, $M \cup B$ is a separator. It is easy to see that V' is disjoint from $V_r \cup M \cup B$ during the algorithm. Consider the final iteration of the while-loop. At the beginning of line 7, $w(V') > \frac{2}{3}w(V)$ so $w(V_r) < w(V)/3$. If line 8 is executed then all vertices that are removed from V' are added to the set output in line 19. If lines 10–17 are executed then let $W' > \frac{2}{3}w(V)$ be the vertex weight of V' just before executing line 10 and let W_r be the vertex weight of V_r just after executing line 17. By definition of S', $W_r \leq (w(V) - W') + W'/2 = w(V) - W'/2 < \frac{2}{3}w(V)$. Since this is the final iteration, we also have that each component of $G[V']$ has vertex weight less than $\frac{2}{3}w(V)$ at the end of the iteration. Hence, $M \cup B$ is a separator of G in line 19.

The main difference between the above generic algorithm and that in [6] is that the former searches for a tree in $G[V \setminus (M \cup A)]$ and allows M to overlap with B and V_r whereas the latter does not maintain A and instead searches for a tree in $G[V']$, ensuring that (V', M, V_r, B) is a partition of V. This difference will be important for our algorithm in Section 6. Unlike [6], we also allow constant C in line 7 in order to facilitate the use of approximate distances.

It is easy to see that if each tree added to $\mathcal{M}$ in line 5 of genericalg$(G = (V, E), h, \ell)$ contains at most s vertices and if a separator is output in line 19, its size is $O(hs + n/\ell)$. Note that if a tree T exists in line 7 of genericalg$(G = (V, E), h, \ell)$, we may pick it such that its size is $O(h\ell \log n)$. This then gives a separator of size $O(h^2\ell \log n + n/\ell)$, matching that in [6]. For our algorithm in Section 6, we will need to allow a bigger size of each tree T. This will increase the size of the separator but only for large values of ℓ.

4 A Fast Separator Theorem via Distance Oracles

For the description of our first algorithm, it will prove useful to regard $\mathcal{M}$ as consisting of exactly $h-1$ trees at any given time; this is done by permitting trees with empty vertex sets. Each tree of $\mathcal{M}$ is assigned a unique index between 1 and $h-1$ and we let T_i denote the tree with index i. From the moment a tree is added to $\mathcal{M}$ until it leaves $\mathcal{M}$ or the algorithm terminates, the index of that tree is fixed. We refer to the trees of $\mathcal{M}$ with non-empty vertex sets as *proper* trees. Each of them will have diameter $O(\ell \log n)$ and size $O(\ell h \log n)$. Thus, the size of the separator output in line 19 is $O(h^2\ell \log n + n/\ell)$.

For $1 \leq i \leq h-1$, we maintain a decremental approximate distance oracle $\mathcal{D}_i$ satisfying the conditions in Lemma 2 with $k = \lceil 1/\epsilon \rceil$ and $d = 4k\rho$. Denote by V'_i the set $(V \setminus (M \cup A)) \cup V(T_i)$. At any point, $\mathcal{D}_i$ is an approximate distance oracle for the graph with vertex set V and containing all edges of $G[V'_i]$ except those with both endpoints in T_i that do not belong to T_i. We denote this graph by $G'_i = (V, E'_i)$. Note that the vertex set is fixed to V since $\mathcal{D}_i$ does not support vertex deletions.

4.1 Identifying a Tree of Small Diameter

Now, consider an iteration of the while-loop of genericalg. To determine whether to execute line 8 or lines 10–17, we do as follows. Assume that at least one tree in $\mathcal{M}$ is proper (otherwise, a tree satisfying the conditions in line 7 can easily be found) and let $i_{\min}$ be the smallest index of such a tree. Pick u as any vertex of V' incident in G to $T_{i_{\min}}$. For each proper $T_i \in \mathcal{M}$ with $i > i_{\min}$ and for each $v \in V(T_i)$, we query $\mathcal{D}_i$ for a distance estimate from u to v in G'_i. Let $v_i \in V(T_i)$ be a vertex v achieving a minimum estimate and denote this estimate by $\tilde{\delta}_i$. In the following, we consider the following two possible cases:

1. $\tilde{\delta}_i \leq d$ for all $i > i_{\min}$ for which T_i is proper,
2. $\tilde{\delta}_i > d$ for some $i > i_{\min}$ for which T_i is proper.

We will show how a tree satisfying the conditions in line 7 can be found if we are in case 1 and that a vertex v satisfying the conditions in line 10 can be found if we are in case 2.

Assume case 1 first. We traverse the approximate uv_i-path from u until reaching the first vertex which is incident in G to T_i. Note that the subpath traversed is contained in $G[V \setminus (M \cup A)]$. By Lemma 2, each traversal takes $O(d)$ time and since each approximate distance query can be done in $O(1)$ time, v_i is found in $O(|V(T_i)|) = O(hd)$ time. Summing over all i, we obtain in $O(h^2 d)$ time a tree T in $G[V \setminus (M \cup A)]$ rooted at u of radius at most $d = 4k\rho$ and of size $O(hd)$ which is incident in G to each proper tree of $\mathcal{M}$ (T is in the union of the paths traversed). This tree satisfies the conditions in line 7 with $C = 4k$. We may assume that T contains at least d vertices; if not, grow it by adding additional incident vertices to it from $V \setminus (M \cup A)$ while keeping the radius at most d.

Having found T, we check if there is an index available for T in $\mathcal{M}$, i.e., an index i between 1 and $h-1$ such that T_i is not proper. If there is no such index, G has K_h as a minor of depth $O(\ell \log n)$ and the algorithm outputs T and the collection of proper trees in $\mathcal{M}$ as a certificate of this. Otherwise, T_i is set to T, thereby adding it to $\mathcal{M}$ and making it proper, and $V(T)$ is removed from V' and added to M. This update may cause some other proper trees $T_j \in \mathcal{M}$ to no longer be incident to V' in G and each of these trees need to be moved from $\mathcal{M}$ to V_r in the next iteration (if any). For this to happen for such a tree T_j, each edge of G'_j incident to T_j must either belong to T_j or be incident to the set V'' of vertices that will be removed from V' in line 3 of the next iteration. Note that since V' is a monotonically decreasing set, we can afford to visit the set E'' of edges of G which are incident to V'' and mark them. Now for each proper T_j, $j \neq i$, we visit all its vertices and check if it has any incident non-tree edges in E'_j which are not marked; this takes $O(|T_j| + |E''|) = O(hd + |E''|)$ time. If there is no such vertex, $V(T_j)$ is moved from M to V_r and T_j is updated to be an empty tree in $\mathcal{M}$. Over all j, this takes $O(h^2 d + h|E''|)$ time. The latter term is paid for by initially putting h credits on each edge of E.

To update the approximate distance oracles accordingly, consider first moving $V(T) = V(T_i)$ from V' to M. This amounts to deleting from G'_j all edges incident to $V(T_i)$, for each $j \neq i$. From $\mathcal{D}_i$, we need to delete edges with both endpoints in T_i that do not belong to T_i. We identify these by simply visiting all edges incident to T_i since this can be paid for by the removal of $V(T_i)$ from V'. For a tree T_j whose vertex set needs to be moved from M to V_r, we update $\mathcal{D}_j$ by deleting all edges incident to $V(T_j)$.

By Lemma 2, total time to maintain all $h-1$ distance oracles is $O(hdmn^{1/k}) = O(h\ell mn^{\epsilon}\log n)$. Since each tree removed from V' has size $\Omega(d)$, total additional time for the above is $O(hm + (n/d)h^2d) = O(hm + h^2n)$.

4.2 Identifying a Small Separator for Distant Vertex Pairs

The following lemma considers case 2 above.

Lemma 3. *With the above definitions, suppose $\tilde{\delta}_i > d$ for some $i > i_{\min}$ where T_i is proper. Then the shortest path distance in $G[V']$ from u to any vertex of V' incident in G to T_i is at least $d/(2k-1)-1$.*

Proof. Let w_i be a vertex of V' incident in G to a vertex v_i in T_i and assume for contradiction that $d_{G[V']}(u, w_i) < d/(2k-1) - 1$. We have

$$d_{G'_i}(u, v_i) \leq d_{G'_i}(u, w_i) + 1 \leq d_{G[V']}(u, w_i) + 1 < d/(2k-1)$$

so $\tilde{\delta}_i > d > (2k-1)d_{G'_i}(u, v_i)$. By Lemma 2, $d_{G'_i}(u, v_i) > d$, contradicting that $d_{G'_i}(u, v_i) < d/(2k-1) \leq d$.

Let i be an index satisfying Lemma 3. The lemma shows that if we are in case 2 above, the condition in line 10 holds since for any vertex $v \in V'$ incident in G to T_i, $d_{G[V']}(u, v) > d/(2k-1) - 1 = 4k\rho/(2k-1) - 1 > \rho$. Such a vertex v can be found in $O(|T_i|) = O(hd)$ time. We now describe how to efficiently execute lines 10–17. Since we are free to perform a BFS from u or from v, we perform both of these searches in parallel. More precisely, one edge is visited by the BFS from u, then one edge by the BFS from v, and so on. When the condition in line 15 is satisfied for either of them, both searches terminate.

To analyze the running time, observe that when the BFS procedures terminate, each has visited at most $\rho - 1 = d/(4k) - 1$ layers. By Lemma 3, $E_u \cap E_v = \emptyset$ where E_u resp. E_v denotes the set of edges explored by the BFS from u resp. v. Let U be the vertex set moved from V' to $V_r \cup B$ after the BFS procedures from u and v terminate and let t denote the BFS time for exploring E_u and E_v. Since $|E_u| = |E_v| \pm 1$, we have that $t = O(|E_u|)$ as well as $t = O(|E_v|)$. Since either $E_u \subseteq E(G[U])$ or $E_v \subseteq E(G[U])$, we get $t = O(|E(G[U])|)$. Hence, the removal of U from V' can pay for t so total BFS time over all iterations is $O(m)$.

As for case 1 above, we move a tree T_j from M to V_r if T_j is no longer incident in G to V' in line 3 of the next iteration. Letting V'' and E'' be defined as above, we mark all edges of G incident to V'' and then for each proper T_j check in $O(|T_j|)$ time whether G'_j has any non-marked edges incident to T_j that do not belong to T_j. This takes $O(|E''| + h^2d)$ time over all j. Since V'' has size $\Omega(\ell)$, this is $O(m + (n/\ell)h^2d) = O(m + n\log nh^2)$ over all iterations.

We can now conclude this section with our first main result.

Theorem 1. *Given a graph G with m edges and n vertices and given $h, \ell \in \mathbb{N}$ and $\epsilon > 0$, there is an algorithm with $O(h\ell m^{1+\epsilon} + h^2n\log n)$ running time which either gives a certificate of K_h as a depth $O(\ell\log n)$-minor in G or outputs a separator of G of size $O(h^2\ell\log n + n/\ell)$.*

In the next section, we show that when $\ell = \Omega(n^\epsilon)$ then $m = O(\text{poly}(h)n)$. Hence, in this case, our algorithm can be modified to initially check if G exceeds this edge bound and if so reject and halt. This combined with Theorem 1 gives the first result in the abstract.

5 Speed-Up Using Bootstrapping and Spanners

In this section, we speed up the result in Theorem 1 for large ℓ using a bootstrapping technique presented in [11]. Note that the running time in Theorem 1 grows with ℓ. The idea for the speed-up is to apply the theorem for a smaller depth than ℓ (which is possible by Lemma 1) to obtain large separators fast and to use these separators to build a compact representation of certain subgraphs of G which can be used to give a fast implementation of `genericalg` for the actual depth ℓ. To simplify our bounds, we assume here and in the next section that $h = O(1)$ and $\ell = \Omega(n^\epsilon)$ for an arbitrarily small $\epsilon > 0$. The actual dependency on h is a low-degree polynomial and for smaller values of ℓ, we can apply our first algorithm. We have omitted the details for our second algorithm due to space constraints and since the technique is basically the same as in [11]. The performance of our second algorithm is stated in the following theorem.

Theorem 2. *Given a graph G with n vertices and given $h, \ell \in \mathbb{N}$ with $h = O(1)$ and $\ell = \Omega(n^\epsilon)$ for constant $\epsilon > 0$, there is an algorithm with $O(n^{1+\epsilon}\sqrt{\ell} + n^{2+\epsilon}/\ell^{3/2})$ running time which either correctly reports that G contains K_h as a depth $O(\ell \log n)$-minor or outputs a separator of G of size $O(\ell \log n + n/\ell)$.*

6 Combining Spanners and Distance Oracles

Our third and final algorithm combines techniques from the other two. The overall idea is to maintain decremental distance oracles as in the first algorithm but for a graph of sublinear size consisting of spanners as in the second algorithm. However, there are several obstacles that we need to overcome to handle this, most notably that maintaining a sublinear-size graph representation require edge insertions in addition to deletions, something that the oracles do not support.

First, we give some definitions and results that we needed for the second algorithm (more details to appear in the full version). For a connected graph $G = (V, E)$, a *cluster* (of G) is a connected subgraph of G. A *clustering* (of G) is a collection $\mathcal{C}$ of clusters of G whose edge sets form a partition of E. A *boundary vertex* of a cluster $C \in \mathcal{C}$ is a vertex that C shares with other clusters in $\mathcal{C}$. All other vertices of C are *interior vertices* of C. For a subgraph C' of C, we let $\delta C'$ denote the set of boundary vertices of C contained in C' and we refer to them as the boundary vertices of C' (w.r.t. C).

Let n be the number of vertices of G. For a parameter $r > 0$, an *r-clustering* (of G) is a clustering with clusters having a total of $\tilde{O}(n/\sqrt{r})$ boundary vertices (counted with multiplicity) and each containing at most r vertices and $\tilde{O}(\sqrt{r})$ boundary vertices. The total vertex size of clusters in an r-clustering is $n + \tilde{O}(n/\sqrt{r})$ and the number of clusters is $\tilde{O}(n/\sqrt{r})$.

Lemma 4. *Let G be a vertex-weighted graph with m edges and n vertices, let $h, \ell \in \mathbb{N}$ and $0 < \epsilon < 1$ where h and ϵ are constants and where $\ell = O(\sqrt{n})$ and $\ell = \Omega(n^\epsilon)$. For any parameter $r \in (C \log n, \ell]$ for a sufficiently large constant C, there is an algorithm with $O(\sqrt{r} m^{1+\epsilon})$ running time that either gives a certificate that G contains K_h as a depth $O(\ell \log n)$-minor or outputs an r-clustering of G.*

Corollary 1. *Let $\epsilon > 0$ be a constant. For any $h = O(1)$ and $\ell = \Omega(n^\epsilon)$, any n-vertex graph excluding K_h as a depth ℓ-minor has only $O(n)$ edges.*

Lemma 5. *For an ℓ-clustering $\mathcal{C}$, $\sum_{C \in \mathcal{C}} |C||\delta C|, \sum_{C \in \mathcal{C}} |\delta C|^3 = \tilde{O}(n\sqrt{\ell})$.*

For a cluster C and a subset B of δC, consider the complete undirected graph $D_B(C)$ with vertex set B. Each edge (u, v) in $D_B(C)$ has weight equal to the weight of a shortest path in C between u and v that does not contain any other vertices of δC. We call $D_B(C)$ the *dense distance graph of C (w.r.t. B)*. We have $d_{D_B(C)}(u, v) = d_C(u, v)$ for all $u, v \in B$. Our algorithm maintains a compact approximate representation of dense distance graphs using a spanner construction for general graphs; For *stretch* $\delta \geq 1$, a *δ-spanner* of a graph is a subgraph that preserves all shortest path distances up to a factor of δ.

Lemma 6. *Let H be an undirected graph with nonnegative edge weights. For any constant $0 < \epsilon \leq 1$, a $(1/\epsilon)$-spanner of H of size $O(|V(H)|^{1+2\epsilon})$ can be constructed in linear time.*

Now, we give our third and final algorithm. It starts by checking whether G is sparse. If not, it rejects and halts (Corollary 1). Otherwise, the set V_Δ of vertices of degree larger than some value Δ are removed. This set has size $O(n/\Delta)$ and will be added as separator vertices in the end. As we show below, the separator found for the remaining graph will have size $O(n/l) + \tilde{O}(\ell^2 \Delta)$ so we pick $\Delta = \sqrt{n}/\ell$ to get a separator for the full graph of size $O(n/\ell) + \tilde{O}(\sqrt{n}\ell)$. Note that this is not an asymptotic increase over the separator size in [6] when $\ell = \tilde{O}(n^{1/4})$. We will give an efficient implementation of algorithm `genericalg` to find such a separator in time $O(n^{1+\epsilon}\sqrt{\ell})$.

6.1 Mini Clusters

In the following, we let G denote the graph after the removal of high-degree vertices. Then the degree of G is bounded by $\Delta = \sqrt{n}/\ell$. We start by computing an ℓ-clustering $\mathcal{C}'$ of G. From this, we will obtain a more refined set of *mini clusters*. Let C be a cluster of $\mathcal{C}'$. In the following, we describe how mini clusters associated with C are formed.

For each interior vertex w of C, let C_w be the subgraph of C reachable from w using paths in which no interior vertices belong to δC. Let $\mathcal{C}_1'$ be the set of subgraphs obtained this way over all $C \in \mathcal{C}'$. Notice that they are connected and intersect only in boundary vertices of C. We form the subset $\mathcal{C}_1$ of $\mathcal{C}_1'$ consisting of those subgraphs C_w such that there is a pair of distinct vertices $b_1, b_2 \in \delta C_w$ where $d_{C_w}(b_1, b_2)$ is smallest among all subgraphs of $\mathcal{C}_1'$ that have a path between

b_1 and b_2; here we we resolve ties by regarding the path with smaller minimum index of an interior vertex (for some arbitrary fixed assignment of indices to V) as the shortest one. We say that (b_1, b_2) is *associated* with C_w. Note that any pair of boundary vertices is associated with at most one subgraph of $\mathcal{C}_1$.

Let $\mathcal{C}_2$ be the set of edges e such that e has both endpoints in δC for some $C \in \mathcal{C}'$. We regard each edge of $\mathcal{C}_2$ as a graph. Let $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$ be the set of mini clusters. Note that $\mathcal{C}$ is a clustering.

Lemma 7. *With the above definitions, if G excludes K_h as a depth $O(\ell \log n)$-minor, the total number of boundary vertices (counted with multiplicity) of mini clusters of a cluster $C \in \mathcal{C}'$ is $O(|\delta C| \log |\delta C|)$.*

Proof. First, note that each subgraph of each cluster is sparse. If not, it would mean that it does not exclude K_h as a minor [3,9] and hence (since its vertex size is at most ℓ) that G does not exclude K_h as a minor of depth $\ell - 1 = O(\ell \log n)$.

Let $\mathcal{C}_1(C)$ resp. $\mathcal{C}_2(C)$ be the set of mini clusters of $\mathcal{C}_1$ resp. $\mathcal{C}_2$ belonging to C. Note that the union of mini clusters of $\mathcal{C}_2(C)$ is a subgraph of C with vertex set contained in δC. Since any subgraph of C is sparse, so is this union so the total number of boundary vertices of mini clusters in $\mathcal{C}_2(C)$ is $2|\mathcal{C}_2(C)| = O(|\delta C|)$.

It remains to bound the total number of boundary vertices of mini clusters in $\mathcal{C}_1(C)$. Consider such mini cluster C' and let w be an interior vertex of C'. Growing a BFS tree from w in C', backtracking once a boundary vertex is reached, we get a spanning tree of C' where each boundary vertex is a leaf. The star with center w and with $\delta C'$ as the set of leaves is a minor of C' as it can be obtained from this spanning tree using edge contractions. For $i = 1, \ldots, \lceil \log |\delta C| \rceil$, let $\mathcal{S}_i$ be the set of stars with between 2^{i-1} and 2^i leaves over all mini clusters of $\mathcal{C}_1(C)$ and let $s_i = |\mathcal{S}_i|$. The lemma will follow if we can show that $\sum_{1 \leq i \leq \lceil \log |\delta C| \rceil} s_i 2^i = O(|\delta C| \log |\delta C|)$.

Since mini clusters of $\mathcal{C}_1(C)$ intersect only in boundary vertices, the union of stars in any set $\mathcal{S}_i$ is a minor of C and hence excludes K_h as a minor. In particular, this union is sparse so $s_i 2^i \leq c(s_i + |\delta C|)$, for some constant c independent of i. Hence, there is a constant I such that $s_i 2^i \leq 2c|\delta C|$ for all $i \geq I$, implying that $\sum_{i \geq I} s_i 2^i = O(|\delta C| \log |\delta C|)$. For $i < I$, consider a star $S \in \mathcal{S}_i$ and let C_S be the mini cluster that yielded S in the procedure above. There is a pair of distinct leaves b_1 and b_2 of S such that (b_1, b_2) is associated with C_S. Note that there is a path from b_1 to b_2 in C_S such that no interior vertex of this path belongs to any other mini cluster. Thus, picking such a pair for each $S \in \mathcal{S}_i$ and regarding it as an edge, we obtain a minor H of C with vertex set contained in δC. Since H excludes K_h as a minor (otherwise C would not), we get $s_i 2^i \leq s_i 2^I = |E(H)| 2^I = O(|\delta C|)$.

Combining this lemma with Lemma 5 and the definition of ℓ-clustering, we get the following corollary.

Corollary 2. *With the set $\mathcal{C}$ of mini clusters defined above, if G excludes K_h as a depth $O(\ell \log n)$-minor, $\sum_{C \in \mathcal{C}} |\delta C| = \tilde{O}(n/\sqrt{\ell})$ and $\sum_{C \in \mathcal{C}} |C||\delta C| = \tilde{O}(n\sqrt{\ell})$.*

Our algorithm will reject and halt if the bounds of Corollary do not hold as then G contains K_h as a depth $O(\ell \log n)$-minor.

6.2 Implementing the Generic Algorithm

The implementation of `genericalg` is in many ways similar to our first algorithm so we only highlight the differences here. Let S be the graph consisting of the union of spanners $S(C)$ over all mini clusters $C \in \mathcal{C}$, where the stretch is chosen to be $6/\epsilon$. We maintain data structures $\mathcal{D}_i$ for $1 \leq i \leq h-1$ with d of order $\ell \ln n$ in Lemma 2 and stretch $6/\epsilon$ and each structure is initialized with graph S. By Lemma 2 and Corollary 2, this takes $O(|E(S)|^{1+\epsilon/3}\ell) = \tilde{O}((n/\sqrt{\ell})^{(1+\epsilon/3)(1+\epsilon/3)}\ell) = O(n^{1+\epsilon}\sqrt{\ell})$ time over all deletions. We will use these structures to identify trees in $G[V \setminus (M \cup A)]$. Note that the stretch of the paths obtained from the structures is $36/\epsilon^2 = O(1)$.

A problem with this approach is that $G[V \setminus (M \cup A)]$ may contain parts of mini clusters of $\mathcal{C}$. Approximate paths in a part of a mini cluster C might not be represented by a subgraph of $S(C)$ but each structure $\mathcal{D}_i$ only supports edge deletions, not insertions. We handle this by requiring the following invariant: for any mini cluster C, if $V(C) \cap V \setminus (M \cup A) \neq \emptyset$ then $V(C) \subseteq V \setminus (M \cup A)$. If we can maintain this invariant, at any point, distances in $G[V \setminus (M \cup A)]$ between boundary vertices of $\mathcal{C}$ can be approximated in a union of spanners of a subset of the mini clusters in $\mathcal{C}$. Hence these distances can be approximated by the structures $\mathcal{D}_i$ if we delete all edges of $S(C)$ from them whenever a mini cluster C leaves $G[V \setminus (M \cup A)]$.

In a given iteration, if a tree T is found (line 7) by a data structure $\mathcal{D}_i$, it consists of edges from spanners $S(C)$. By precomputing shortest path trees from each boundary vertex of each mini cluster (which can be done in $\tilde{O}(n\sqrt{\ell})$ time by Corollary 2), we can identify the edges of G that are contained in T in time proportional to their number. Now, to ensure the invariant, we expand T into the mini clusters it intersects. More precisely, for each mini cluster $C \in \mathcal{C}$ for which $V(T) \cap \delta C \neq \emptyset$, we expand T to include all interior vertices of C but no vertices in δC that do not already belong to T. This is possible since by definition of mini clusters, there is a spanning tree of C in which every boundary vertex of C is a leaf. This tree expansion ensures the invariant but it comes at a cost of a worse separator size guarantee since trees of the final set $\mathcal{M}$ may be larger than in our first algorithm. Consider one such tree. It was obtained by expanding a tree T of size $O(\ell \log n)$ into the mini clusters of $\mathcal{C}$ it intersects. By our degree bound, each vertex of T is contained in at most Δ mini clusters each having size $O(\ell)$. Hence, each tree in $\mathcal{M}$ has size $O(\ell^2 \Delta \log n) = O(\ell\sqrt{n}\log n)$.

We have shown how trees are efficiently found in our implementation of `genericalg`. Lines 10–17 are implemented exactly as for the first algorithm, where we do BFS in parallel in $G[V']$ from two vertices that are far apart. Note that when these lines are executed, no updates are made to $M \cup A$ (vertices may be removed from M if they are no longer incident to V' but in that case they are added to A) so our data structures $\mathcal{D}_i$ do not require updates here. The total time for lines 10–17 is $O(m + n\log n)$ as for our first algorithm. We can now conclude with our final main result.

Theorem 3. *Given a graph G with n vertices and given $h, \ell \in \mathbb{N}$ with $h = O(1)$, there is an algorithm with $O(n^{1+\epsilon}\sqrt{\ell})$ running time which either correctly reports* *that G contains K_h as a depth $O(\ell \log n)$-minor or outputs a separator of G of size $O(n/\ell) + \tilde{O}(\ell\sqrt{n})$.*

7 Concluding Remarks

We gave three new algorithms to find separators of shallow-minor free graphs. For small-depth minors, a speed-up in running time of almost a linear factor is achieved compared to previous algorithms while still giving separators of the same size. A main idea was an application of a dynamic approximate distance oracle of Roditty and Zwick for general graphs. We believe our speed-ups should give improved algorithms for static graph problems such as shortest paths and maximum matching since one of the bottlenecks for many separator-based algorithms is finding a good separator. For instance, a faster separator theorem for minor-free graphs led to several improved static graph algorithms [11].

Can our bounds be improved further? Nothing suggests that the dependencies on ℓ in our time bounds are natural. The bottleneck is the dynamic distance oracle of Roditty and Zwick. Any improvement of that result would give a speed-up of our algorithms as well.

References

1. Alon, N., Seymour, P.D., Thomas, R.: A separator theorem for nonplanar graphs. J. Amer. Math. Soc. 3(4), 801–808 (1990)
2. Djidjev, H.N., Gilbert, J.R.: Separators in Graphs with Negative and Multiple Vertex Weights. Algorithmica 23, 57–71 (1999)
3. Kostochka, A.V.: Lower bound of the Hadwiger number of graphs by their average degree. Combinatorica 4(4), 307–316 (1984)
4. Lipton, R.J., Tarjan, R.E.: A separator theorem for planar graphs. SIAM Journal on Applied Mathematics 36, 177–189 (1979)
5. Nesetril, J., de Mendez, P.O.: Sparsity – Graphs, Structures, and Algorithms. In: Algorithms and Combinatorics vol. 28, pp. I-XXIII, 1–457 (2012) ISBN 978-3-642-27874-7
6. Plotkin, S., Rao, S., Smith, W.D.: Shallow excluded minors and improved graph decompositions. In: Proc. of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 462–470 (1994)
7. Roditty, L., Thorup, M., Zwick, U.: Deterministic Constructions of Approximate Distance Oracles and Spanners. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 261–272. Springer, Heidelberg (2005)
8. Roditty, L., Zwick, U.: Dynamic Approximate All-Pairs Shortest Paths in Undirected Graphs. SIAM J. Comput. 41(3), 670–683 (2012); See also FOCS 2004
9. Thomason, A.: An extremal function for contractions of graphs. Math. Proc. Cambridge Philos. Soc. 95(2), 261–265 (1984)
10. Thorup, M., Zwick, U.: Approximate Distance Oracles. J. Assoc. Comput. Mach. 52, 1–24 (2005)
11. Wulff-Nilsen, C.: Separator Theorems for Minor-Free and Shallow Minor-Free Graphs with Applications. In: Proc. FOCS, pp. 37–46 (2011); See extended version on arXiv

Spatial Mixing of Coloring Random Graphs[*]

Yitong Yin[**]

State Key Laboratory for Novel Software Technology, Nanjing University, China
yinyt@nju.edu.cn

Abstract. We study the strong spatial mixing (decay of correlation) property of proper q-colorings of random graph $G(n, d/n)$ with a fixed d. The strong spatial mixing of coloring and related models have been extensively studied on graphs with bounded maximum degree. However, for typical classes of graphs with bounded average degree, such as $G(n, d/n)$, an easy counterexample shows that colorings do not exhibit strong spatial mixing with high probability. Nevertheless, we show that for $q \geq \alpha d + \beta$ with $\alpha > 2$ and sufficiently large $\beta = O(1)$, with high probability proper q-colorings of random graph $G(n, d/n)$ exhibit strong spatial mixing *with respect to an arbitrarily fixed vertex*. This is the first strong spatial mixing result for colorings of graphs with unbounded maximum degree. Our analysis of strong spatial mixing establishes a block-wise correlation decay instead of the standard point-wise decay, which may be of interest by itself, especially for graphs with unbounded degree.

1 Introduction

A proper q-coloring of a graph G is an assignment of q colors $\{1, 2, \ldots, q\}$ to the vertices so that adjacent vertices receive different colors. Each coloring corresponds to a configuration in the q-state zero-temperature antiferromagnetic Potts model. The uniform probability space, known as the Gibbs measure, of proper q-colorings of the graph, receives extensive studies from both Theoretical Computer Science and Statistical Physics.

An important question concerned with the Gibbs measure is about the mixing rate of Glauber dynamics, usually formulated as: on graphs with maximum degree d, assuming $q \geq \alpha d + \beta$, the lower bounds for α and β to guarantee rapidly mixing of the Glauber dynamics over proper q-colorings. (See [9] for a survey.)

Recently, much attention has been focused on the spatial mixing (correlation decay) aspect of the Gibbs measure, which is concerned with the case where the site-to-boundary correlations in the Gibbs measure decay exponentially to zero with distance. In Statistical Physics, spatial mixing implies the uniqueness of infinite-volume Gibbs measure. The notion of *strong spatial mixing* was introduced in Theoretical Computer Science by Weitz [18]. Here, the exponential decay of site-to-boundary correlations is required to hold even conditioning on an arbitrarily fixed boundary. Strong spatial mixing is interesting to Computer

[*] The full version of this paper is available at http://arxiv.org/abs/1402.4556

[**] Supported by NSFC grants 61272081 and 61321491.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 1075–1086, 2014.

Science because it may imply efficient approximation algorithms for counting and sampling. This implication was fully understood for two-state spin systems. For multi-state spin systems such as coloring, this algorithmic implication of strong spatial mixing is only known for special classes of graphs, such as neighborhood-amenable (slow-growing) graphs [13]. Strong spatial mixing of proper q-coloring has been proved for classes of degree-bounded graphs, including regular trees [12], lattices graphs [13], and finally the general degree-bounded triangle-free graphs [11], all with the same $\alpha > \alpha^*$ bound where $\alpha^* = 1.763...$ is the unique solution to $x^x = \mathrm{e}$.

All these temporal and spatial mixing results are established for graphs with bounded *maximum degree*. It is then natural to ask what happens for classes of graphs with bounded *average degree*. A natural model for the "typical" graphs with bounded average degree d is the Erdös-Rényi random graph $G(n, d/n)$. In this model, the Gibbs measure of proper q-colorings becomes more complicated because the maximum degree is unbounded and the decision of colorability is nontrivial. Nevertheless, it was discovered in [5] that for $G(n, d/n)$ the rapid mixing of (block) Glauber dynamics over the proper q-colorings can be guaranteed by a $q = O(\log\log n/\log\log\log n)$, much smaller than the maximum degree of $G(n, d/n)$. This upper bound on the number of colors was later reduced to a constant $q = \mathrm{poly}(d)$ in [8] and independently in [15, 16], and very recently to a linear $q \geq \alpha d + \beta$ with $\alpha = 5.5$ in [7].

On the spatial mixing side, the strong spatial mixing of the models which are simpler than coloring has been studied on random graph $G(n, d/n)$, or other classes of graphs with bounded average degree. Recently in [17], such average-degree based strong spatial mixing is established for the independent sets of graphs with bounded *connective constant*. Since $G(n, d/n)$ has connective constant $\approx d$ with high probability, this result is naturally translated to $G(n, d/n)$.

It is then an important open question to ask about the conditions for the spatial mixing of colorings of graphs with bounded average degree. The following simple example shows that this can be very hard to achieve: Consider a long path of ℓ vertices, each adjacent to $q-2$ isolated vertices, where q is the number of colors. When the path is sufficiently long, the connective constant of this graph can be arbitrarily close to 1. However, colors of those isolated vertices can be properly fixed to make the remaining path effectively a 2-coloring instance, which certainly has long-range correlation, refuting the existence of strong spatial mixing.

More devastatingly, it is easy to see that for any constant q, with high probability the random graph $G(n, d/n)$ contains a path of length $\ell = \Theta(\log n)$ in which every vertex has degree $q-2$. As in the above example, even in a weaker sense of site-to-site correlation which was considered in [13], this forbids the strong spatial mixing up to a distance $\Theta(\log n)$. Meanwhile, it is well known that the diameter of $G(n, d/n)$ is $O(\log n)$ with high probability. So the strong spatial mixing of colorings of random graph $G(n, d/n)$ cannot hold except for a narrow range of distances in $\Theta(\log n)$.

In this case, inspired by the studies of spatial mixing in rooted trees, where only the decay of correlation to the root is considered, we propose to study the strong spatial mixing *with respect to a fixed vertex*, instead of all vertices.

Assumption 1. *We make following assumptions:*

- *$d > 1$ is fixed, and $q \geq \alpha d + \beta$ for $\alpha > 2$ and sufficiently large $\beta = O(1)$ ($\beta \geq 23$ is fine);*
- *$v \in V$ is arbitrarily fixed and $G = (V, E)$ is a random graph drawn from $G(n, d/n)$, where n is sufficiently large.*

Note that vertex v is fixed independently of the sampling of random graph. With these assumptions we prove the following theorem.

Theorem 2. *Let q, v and G satisfy Assumption 1, and $t(n) = \omega(1)$ an arbitrary super-constant function. With high probability, G is q-colorable and the following holds: for any region $R \subset V$ containing v, whose vertex boundary is ∂R, for any feasible colorings $\sigma, \tau \in [q]^{\partial R}$ partially specified on ∂R which differ only at vertices that are at least $t(n)$ distance away from v in G, for some constants $C_1, C_2 > 0$ depending only on d and q, it holds that*

$$|\Pr[c(v) = x \mid \sigma] - \Pr[c(v) = x \mid \tau]| \leq C_1 \exp(-C_2 \cdot \mathrm{dist}(v, \varDelta)),$$

for a uniform random proper q-coloring c of G and any $x \in [q]$, where $\varDelta \subset \partial R$ is the vertex set on which σ and τ differ, and $\mathrm{dist}(v, \varDelta)$ denotes the shortest distance in G between v and any vertex in $\varDelta$.

This is the first strong spatial mixing result for colorings of graphs with unbounded maximum degree. Our technique is developed upon the error function method introduced in [11], which uses a cleverly designed error function to measure the discrepancy of marginal distributions, and the strong spatial mixing is implied by an exponential decay of errors measured by this function.

In all existing techniques for strong spatial mixing of colorings, when the degree of a vertex is unbounded, a multiplicative factor of ∞ is contributed to the decay of correlation, which unavoidably ruins the decay. However, in the real case for colorings of graphs with unbounded degree, a large-degree vertex may at most locally "freeze" the coloring, rather than nullify the existing decay of correlation. This limitation on the effect of large-degree vertex has not been addressed by any existing techniques for spatial mixing.

We address this issue by considering a block-wise correlation decay, so that within a block the coloring might be "frozen", but between blocks, the decay of correlation is as in that between vertices in the degree-bounded case. This analysis of block-wise correlation decay can be seen as a spatial analog to the block dynamics over colorings of random graphs, and is the first time that such an idea is used in the analysis of spatial mixing.

Related Work. As one of the most important random CSP, the decision problem of coloring sparse random graphs has been extensively studied, e.g. in [1, 3].

Monte Carlo algorithms for sampling random coloring in sparse random graphs were studied in [4,7,8,15,16], and in [6], a non-Monte-Carlo algorithm was given for the same problem which uses less colors but has worse error dependency than the Monte-Carlo algorithms. In [10,14] the correlation decay on computation tree for coloring was studied which implies FPTAS for counting coloring.

2 Preliminaries

Graph coloring. Let $G = (V, E)$ be an undirected graph. For each vertex $v \in V$, let $d_G(v)$ denote the degree of v. For any $u, v \in V$, let $\mathrm{dist}_G(u, v)$ denote the distance between u and v in G; and for any vertex sets $S, T \subseteq V$, let $\mathrm{dist}_G(u, S) = \min_{v\in S} \mathrm{dist}_G(u, v)$ and $\mathrm{dist}_G(S, T) = \min_{u\in S, v\in T} \mathrm{dist}_G(u, v)$. The subscripts can be omitted if graph G is assumed in context. For any vertex set $S \subset V$, we use $\partial S = \{v \notin S \mid uv \in E, u \in S\}$ to denote the *vertex boundary* of S, and use $\delta S = \{uv \in E \mid u \in S, v \notin S\}$ to denote the *edge boundary* of S.

We consider the *list-coloring problem*, which is a generalization of q-coloring problem. Let $q > 0$ be a finite integer, a pair $(G, \mathcal{L})$ is called a *list-coloring instance* if $G = (V, E)$ is an undirected graph, and $\mathcal{L} = (L(v) : v \in V)$ is a sequence of lists where for each vertex $v \in V$, $L(v) \subseteq [q]$ is a list of colors from $[q] = \{1, 2, \ldots, q\}$ associated with vertex v. A $\sigma \in [q]^V$ is a *proper coloring* of $(G, \mathcal{L})$ if $\sigma(v) \in L(v)$ for every vertex $v \in V$ and no two adjacent vertices in G are assigned with the same color by σ. A list-coloring instance $(G, \mathcal{L})$ is said to be *feasible* or *colorable* if there exists a proper coloring of $(G, \mathcal{L})$. A coloring can also be partially specified on a subset of vertices in G. For $S \subseteq V$, let $L(S) = \{\sigma \in [q]^S \mid \forall v \in V, \sigma(v) \in L(v)\}$ denote the set of all possible colorings (not necessarily proper) of the vertices in S. A coloring $\sigma \in L(S)$ partially specified on a subset $S \subseteq V$ of vertices is said to be *feasible* if there is a proper coloring τ of $(G, \mathcal{L})$ such that σ and τ are consistent over set S. A coloring $\sigma \in L(S)$ partially specified on a subset $S \subseteq V$ of vertices is said to be *proper* or *locally feasible* if σ is a proper coloring of $(G[S], \mathcal{L}_S)$ where $G[S]$ is the subgraph of G induced by S and $\mathcal{L}_S = (L(v) : v \in S)$ denotes the sequence $\mathcal{L}$ of lists restricted on set S of vertices. For any $S \subseteq V$, we use $L^*(S)$ to denote the set of proper colorings of S.

When $L(v) = [q]$ for all vertices $v \in V$, a list-coloring instance $(G, \mathcal{L})$ becomes an instance for q-coloring, which we denote as $(G, [q])$.

Self-avoiding Walk (SAW) Tree. Given a graph $G(V, E)$ and a vertex $v \in V$, a tree T rooted by v can be naturally constructed from all self-avoiding walks starting from v so that each walk corresponds to a vertex in T, and each walk p is the parent of walks (p, u) where $u \in V$ is a vertex. We use $T_{\mathrm{SAW}}(G, v) = T$ to denote this tree constructed as above, and call it a *self-avoiding walk tree (SAW)* of graph G.

Gibbs Measure and Strong Spatial Mixing. A feasible list-coloring instance $(G, \mathcal{L})$ gives rise to a natural probability distribution $\mu = \mu_{G,\mathcal{L}}$, which is the uniform distribution over all proper list-colorings. This distribution μ is also called the *Gibbs* *measure* of list-colorings. We also a notation of $\mathrm{P}_{G,\mathcal{L}}(\text{event}(c)) = \Pr[\text{event}(c)]$ to evaluate probability of an event defined on a uniform random proper coloring c of $(G, \mathcal{L})$. Let $B \subset V$ and $\Lambda \subset V$. For any feasible coloring $\sigma \in L(\Lambda)$ partially specified on vertex set Λ, we use $\mu_B^\sigma = \mu_{G,\mathcal{L},B}^\sigma$ to denote the marginal distribution over colorings of vertices in B conditioning on that the coloring of vertices in Λ is as specified by σ. And when $B = \{v\}$, we write $\mu_v^\sigma = \mu_{G,\mathcal{L},v}^\sigma = \mu_{G,\mathcal{L},\{v\}}^\sigma$. The list-coloring instance $(G, \mathcal{L})$ in the subscripts can be omitted if it is assumed in context. Formally, for a uniformly random proper coloring c of $(G, \mathcal{L})$, we have

$$\begin{aligned} \forall x \in L(v), \quad & \mu_v^\sigma(x) = \mathrm{P}_{G,\mathcal{L}}(c(v) = x \mid \sigma), \\ \forall \pi \in L(B), \quad & \mu_B^\sigma(\pi) = \mathrm{P}_{G,\mathcal{L}}(c(B) = \pi \mid \sigma). \end{aligned}$$

The notion strong spatial mixing is introduced in [18,19] for independent sets and extended to colorings in [11,13].

Definition 3 (Strong Spatial Mixing). *The Gibbs measure on proper q-colorings of a family $\mathcal{G}$ of finite graphs exhibits* strong spatial mixing (SSM) *if there exist constants $C_1, C_2 > 0$ such that for any graph $G(V, E) \in \mathcal{G}$, any $v \in V, \Lambda \subseteq V$, and any two feasible q-colorings $\sigma, \tau \in [q]^\Lambda$, we have*

$$\|\mu_v^\sigma - \mu_v^\tau\|_{\mathrm{TV}} \le C_1 \exp(-C_2 \mathrm{dist}(v, \Delta)),$$

where $\Delta \subseteq \Lambda$ is the subset on which σ and τ differ, and $\|\cdot\|_{\mathrm{TV}}$ is the total variation distance.

When the exponential bound relies on $\mathrm{dist}(v, \Lambda)$ instead of $\mathrm{dist}(v, \Delta)$, the definition becomes *weak spatial mixing (WSM).* The difference is SSM requires the exponential correlation decay continues to hold even conditioning on the coloring of a subset $\Lambda \setminus \Delta$ of vertices being arbitrarily (but feasibly) specified.

Random graph model. The Erdös-Rényi random graph $G(n, p)$ is the graph with n vertices V and random edges E where for each pair $\{u, v\}$, the edge uv is chosen independently with probability p. We consider $G(n, d/n)$ with fixed $d > 1$.

We say an event occurs *with high probability (w.h.p.)* if the probability of the event is $1 - o(1)$.

3 Correlation Decay along Self-avoiding Walks

In this section, we analyze the propagation of errors between marginal distributions measured by a special norm introduced in [11] in general degree-unbounded graphs. Throughout this section, we assume $(G, \mathcal{L})$ to be a list-coloring instance with $G = (V, E)$ and $\mathcal{L} = (L(v) : v \in V)$ where each $L(v) \subseteq [q]$.

The following error function is introduced in [11].

Definition 4 (Error Function). *Let $\mu_1 : \Omega \to [0, 1]$ and $\mu_2 : \Omega \to [0, 1]$ be two probability measures over the same sample space Ω. We define*

$$\mathcal{E}(\mu_1, \mu_2) = \max_{x,y \in \Omega} \left(\log \left(\frac{\mu_1(x)}{\mu_2(x)} \right) - \log \left(\frac{\mu_1(y)}{\mu_2(y)} \right) \right),$$

with the convention that $0/0 = 1$ *and* $\infty - \infty = 0$.

We assume $(G, \mathcal{L})$ to be feasible so that for vertex set $B \subset V$ and feasible colorings $\sigma, \tau \in L(\Lambda)$ of vertex set $\Lambda \subset V$, the marginal probabilities μ_B^σ and μ_B^τ are well-defined. The strong spatial mixing is proved by establishing a propagation of errors $\mathcal{E}(\mu_B^\sigma, \mu_B^\tau)$. Note that unlike in bounded-degree graphs, in general the value of $\mathcal{E}(\mu_B^\sigma, \mu_B^\tau)$ can be infinite, which occurs when the possibility of a particular coloring of B is changed by conditioning on σ and τ. This is avoided when a vertex cut with certain "permissive" property separating B from the boundary. The following proposition is proved in the full version.

Proposition 5. *If there is a $S \subset V \setminus (B \cup \Lambda)$ such that $|L(v)| > d(v) + 1$ for every $v \in S$ and removing S disconnects B and Λ, then $\mathcal{E}(\mu_B^\sigma, \mu_B^\tau)$ is finite for any feasible colorings $\sigma, \tau \in L(\Lambda)$.*

This motivates the following definition of permissive vertex and vertex set.

Definition 6. *Given a list-coloring instance $(G, \mathcal{L})$, a vertex v is said to be* permissive *in $(G, \mathcal{L})$ if for all neighbors u of v and $u = v$, it holds that $|L(u)| > d(u) + 1$. A set S of vertices is said to be permissive if all vertices in S are permissive.*

Let $T = T_{\mathrm{SAW}}(G, v)$ be the self-avoiding walk tree of graph G expanded from vertex v. Recall that every vertex u in T can be naturally identified (many-to-one) with the vertex in G at which the corresponding self-avoiding walk ends (which we also denote by the same letter u).

Definition 7. *Given a list-coloring instance $(G, \mathcal{L})$, let $v \in V$, $T = T_{\mathrm{SAW}}(G, v)$, and S a set of vertices in T. Suppose that the root v has m children $v_1, v_2, \dots, v_m$ in T and for $i = 1, 2 \dots, m$, let T_i denote the subtree rooted by v_i. The quantity $\mathcal{E}_{T,\mathcal{L},S}$ is recursively defined as follows*

$$\mathcal{E}_{T,\mathcal{L},S} = \begin{cases} \sum_{i=1}^{m} \delta(v_i) \cdot \mathcal{E}_{T_i,\mathcal{L},S} & \text{if } v \notin S, \\ 3q & \text{if } v \in S, \end{cases}$$

where $\delta(u)$ is a piecewise function defined as that for any vertex u in T,

$$\delta(u) = \begin{cases} \frac{1}{|L(u)|) - d_G(u) - 1} & \text{if } |L(u)| > d_G(u) + 1, \\ 1 & \text{otherwise,} \end{cases}$$

where $d_G(v)$ is the degree in the original graph G instead of the degree in SAW-tree T.

In particular, when $(G, \mathcal{L})$ is a q-coloring instance $(G, [q])$, we denote this quantity as $\mathcal{E}_{T,[q],S}$.

To state the main theorem of this section, we need one more definition.

Definition 8. *Let $G = (V, E)$, $v \in V$, $\Delta \subset V$, and $T = T_{\mathrm{SAW}}(G, v)$. A set S of vertices in T is a* cutset *in T for v and Δ if: (1) no vertex in S is identified to*

v or any vertex u with $\mathrm{dist}(u, \Delta) < 2$ *by* $T_{\mathrm{SAW}}(G, v)$*; and (2) any self-avoiding walk from v to a vertex in Δ must intersect S in T. A cutset S in T for v and Δ is said to be permissive in $(G, \mathcal{L})$ if every vertex in S is identified with a permissive vertex in $(G, \mathcal{L})$ by $T_{\mathrm{SAW}}(G, v)$.*

The following theorem is the main theorem of this section, which bounds the error function $\mathcal{E}(\mu_v^\sigma, \mu_v^\tau)$ by the $\mathcal{E}_{T,\mathcal{L},S}$ defined in Definition 7 when there is a good cutset in the SAW tree.

Theorem 9. *Let $(G, \mathcal{L})$ be a feasible list-coloring instance where $G = (V, E)$ and $\mathcal{L} = (L(v) \subseteq [q] : v \in V)$. Let $v \in V$, $\Lambda \subset V$ and $\Delta \subseteq \Lambda$ be arbitrary, and $T = T_{\mathrm{SAW}}(G, v)$. If there is a permissive cutset S in T for v and Δ, then for any feasible colorings $\sigma, \tau \in L(\Lambda)$ which differ only on Δ, it holds that*

$$\mathcal{E}(\mu_v^\sigma, \mu_v^\tau) \leq \mathcal{E}_{T,\mathcal{L},S}.$$

This theorem is implied by the following weak spatial mixing version of the theorem.

Lemma 10. *Let $(G, \mathcal{L})$ be a feasible list-coloring instance where $G = (V, E)$ and $\mathcal{L} = (L(v) \subseteq [q] : v \in V)$. Let $v \in V$ and $\Delta \subseteq \Lambda$ be arbitrary, and $T = T_{\mathrm{SAW}}(G, v)$. If there is a permissive cutset S in T for v and Δ, then for any feasible colorings $\sigma, \tau \in L(\Delta)$, it holds that*

$$\mathcal{E}(\mu_v^\sigma, \mu_v^\tau) \leq \mathcal{E}_{T,\mathcal{L},S}.$$

The implication from Lemma 10 to Theorem 9 is quite standard, whose proof is in the full version. It now remains to prove Lemma 10.

3.1 The Block-Wise Correlation Decay

Now our task is to prove Lemma 10. This is done by establishing the decay of $\mathcal{E}(\mu_B^\sigma, \mu_B^\tau)$ along walks among blocks B with the following good property.

Definition 11. *Given a list-coloring instance $(G, \mathcal{L})$, a vertex set $B \subseteq V$ is a* permissive block around *v in $(G, \mathcal{L})$ if $v \in B$ and $|L(u)| > d_G(u) + 1$ for every vertex u in the vertex boundary ∂B.*

For permissive blocks B, a coloring of B is globally feasible if and only if it is locally feasible (i.e. proper on B).

Lemma 12. *Let $\Delta \subset V$ and $B \subset V$ a permissive block such that* $\mathrm{dist}(B, \Delta) \geq 2$. *Then for any feasible coloring $\sigma \in L(\Delta)$, for any coloring $\pi \in L(B)$, it holds that $\mu_B^\sigma(\pi) > 0$ if and only if π is proper on B.*

Proof. Let $S = \partial B$. Note that with $\mathrm{dist}(B, \Delta) \geq 2$ and S must be a vertex cut separating B and Δ. Then the lemma can be proved by the same argument as in the proof of Proposition 5.

Notations. We now define some notations which are used throughout this section. Let $B \subset V$ be a permissive block in a feasible list-coloring instance $(G, \mathcal{L})$. Let $\delta B = \{uw \in E \mid u \in B \text{ and } w \notin B\}$ be the edge boundary of B. We enumerate these boundary edges as $\delta B = \{e_1, e_2, \ldots, e_m\}$. For $i = 1, 2, \ldots, m$, we assume $e_i = u_i v_i$ where $u_i \in B$ and $v_i \notin B$. Note that in this notation more than one u_i or v_i may refer to the same vertex in G. Let $G_B = G[V \setminus B]$ be the subgraph of G induced by vertex set $V \setminus B$. For a coloring $\pi \in L(B)$ and $1 \leq i \leq m$, we denote $\pi_i = \pi(u_i)$. For $1 \leq i \leq m$ and $\pi, \rho \in L(B)$, let $\mathcal{L}_{i,j,\pi,\rho} = (L'(v) : v \in V \setminus B)$ be obtained from $\mathcal{L}$ by removing the color π_k from the list $L(v_k)$ for all $k < i$ and removing the color ρ_k from the list $L(v_k)$ for all $k > i$ (if any of these lists do not contain the respective color then no change is made to them).

With this notation, the following lemma (proved in the full version) generalizes a recursion introduced in [11] for bounded-degree graphs to general graphs.

Lemma 13. *Let $(G, \mathcal{L})$ be a feasible list-coloring instance, $B \subset V$ a permissive block with edge boundary $\delta B = \{e_1, e_2, \ldots, e_m\}$ where $e_i = u_i v_i$ for each $i = 1, 2, \ldots, m$, and $\pi, \rho \in L^*(B)$ any two proper colorings of B. For every $1 \leq i \leq m$,*

- *if a vertex $u \notin B$ is permissive in $(G, \mathcal{L})$, then it is permissive in the new instance $(G_B, \mathcal{L}_{i,\pi,\rho})$;*
- *the new instance $(G_B, \mathcal{L}_{i,\pi,\rho})$ is feasible.*

For any feasible coloring $\sigma \in L(\Delta)$ of a vertex set $\Delta \subset V$ with $\operatorname{dist}(B, \Delta) \geq 2$, we have

$$\frac{\mathrm{P}_{G,\mathcal{L}}(c(B) = \pi \mid \sigma)}{\mathrm{P}_{G,\mathcal{L}}(c(B) = \rho \mid \sigma)} = \prod_{i=1}^{m} \frac{1 - \mathrm{P}_{G_B,\mathcal{L}_{i,\pi,\rho}}(c(v_i) = \pi_i \mid \sigma)}{1 - \mathrm{P}_{G_B,\mathcal{L}_{i,\pi,\rho}}(c(v_i) = \rho_i \mid \sigma)}.$$

The following marginal bounds are standard and are proved in full version.

Lemma 14. *Given a feasible list-coloring instance $(G, \mathcal{L})$, if vertex v has $|L(v)| > d(v) + 1$ and $v \notin \Delta$, then for any feasible coloring $\sigma \in L(\Delta)$ and any $x \in L(v)$, we have*

$$\mathrm{P}_{G,\mathcal{L}}(c(v) = x \mid \sigma) \leq \frac{1}{|L(v)| - d(v)}.$$

If vertex v is permissive in $(G, \mathcal{L})$ and $\operatorname{dist}(v, \Delta) \geq 2$, then for any feasible coloring $\sigma \in L(\Delta)$ and any $x \in L(v)$, we have

$$\mathrm{P}_{G,\mathcal{L}}(c(v) = x \mid \sigma) \geq \frac{1}{|L(v)| 2^{d(v)}}.$$

The recursion in Lemma 13 can imply the following bound for the block-wise decay of error function $\mathcal{E}(\mu_v^\sigma, \mu_v^\tau)$. The proof generalizes the analysis of the point-wise decay in degree-bounded graphs in [11], and is put to the full version.

Lemma 15. *Let $(G, \mathcal{L})$ be a feasible list-coloring instance, $v \in V$ and $B \subset V$ a permissive block around v with edge boundary $\delta B = \{e_1, e_2, \ldots, e_m\}$ where*

$e_i = u_iv_i$ for each $i = 1, 2, \ldots, m$. Let $\varDelta \subset V$ be a vertex set with $\mathrm{dist}(B, \varDelta) \geq 2$, and $\sigma, \tau \in L(\varDelta)$ any two feasible colorings of $\varDelta$. Assume $\pi, \rho \in L^*(B)$ to be two proper colorings of B achieving the maximum in the error function:

$$\mathcal{E}(\mu_B^\sigma, \mu_B^\tau) = \max_{\pi,\rho \in L^*(B)} \left(\log \left(\frac{\mu_B^\sigma(\pi)}{\mu_B^\tau(\pi)} \right) - \log \left(\frac{\mu_B^\sigma(\rho)}{\mu_B^\tau(\rho)} \right) \right).$$

It holds that

$$\mathcal{E}(\mu_v^\sigma, \mu_v^\tau) \leq \sum_{i=1}^{m} \frac{1}{|L(v_i)| - d(v_i) - 1} \cdot \mathcal{E}(\mu_i^\sigma, \mu_i^\tau),$$

where $\mu_i^\sigma = \mu_{G_B,\mathcal{L}_{i,\pi,\rho},v_i}^\sigma$ and $\mu_i^\tau = \mu_{G_B,\mathcal{L}_{i,\pi,\rho},v_i}^\tau$ are the respective marginal distributions of coloring of vertex v_i conditioning on σ and τ in the new list-coloring instance $(G_B, \mathcal{L}_{i,\pi,\rho})$.

With the above block-wise decay, we are now ready to prove Lemma 10, which implies Theorem 9.

Proof (Proof of Lemma 10). Given a feasible list-coloring instance $(G, \mathcal{L})$ and a vertex v, let $T = T_{\mathrm{SAW}}(G, v)$ and S a permissive cutset in T separating v and $\varDelta$. We consider the following procedure:

1. Let B be the *minimal* permissive block around v with edge boundary $\delta B = \{e_1, e_2, \ldots, e_m\}$, where $e_i = u_iv_i$ for $i = 1, 2, \ldots, m$ (note that more than one u_i or v_i may refer to the same vertex). By Lemma 15, we have

$$\mathcal{E}(\mu_v^\sigma, \mu_v^\tau) \leq \sum_{i=1}^{m} \frac{1}{|L(v_i)| - d(v_i) - 1} \cdot \mathcal{E}(\mu_i^\sigma, \mu_i^\tau), \tag{1}$$

   where $\mu_i^\sigma = \mu_{G_B,\mathcal{L}_{i,\pi,\rho},v_i}^\sigma$ and $\mu_i^\tau = \mu_{G_B,\mathcal{L}_{i,\pi,\rho},v_i}^\tau$ are the respective marginal distributions at v_i in the new list-coloring instance $(G_B, \mathcal{L}_{i,\pi,\rho})$ for the $\pi, \rho \in L^*(B)$ defined in Lemma 15. By Lemma 13, all these new list-coloring instances are feasible.
2. We identify each v_i with a distinct self-avoiding walk in G from v to v_i through only vertices in B and approaching v_i via the edge $e_i = u_iv_i$. Such self-avoiding walk must exist or otherwise B is not minimal. If there are more than one such self-avoiding walk for a v_i, choose an arbitrary one to identify v_i with. We use w_i to denote this walk to v_i.
   Note that along every such self-avoiding walk w_i from v to v_i, all vertices u except v and v_i must have $|L(u)| \leq d(u) + 1$ in $(G, \mathcal{L})$ or otherwise B is not minimal. Thus by Definition 7, in quantity $\mathcal{E}_{T,\mathcal{L},S}$, along every walk w_i from v to v_i, at each intermediate vertex $u \notin \{v, v_i\}$, only a factor of $\delta(u) = 1$ is multiplied in $\mathcal{E}_{T,\mathcal{L},S}$, so we have

$$\mathcal{E}_{T,\mathcal{L},S} \geq \sum_{i=1}^{m} \frac{1}{|L(v_i)| - d(v_i) - 1} \mathcal{E}_{T_{w_i},\mathcal{L},S}, \tag{2}$$

   where T_{w_i} denotes the subtree of the SAW tree T rooted by the self-avoiding walk w_i.

3. For each $1 \leq i \leq m$, if the self-avoiding walk w_i corresponds to a vertex in the permissive cutset S in the SAW tree T, then v_i itself must be permissive in $(G, \mathcal{L})$ and $\mathrm{dist}(v_i, \Delta) \geq 2$, both of which continue to hold in the new instance $(G_B, \mathcal{L}_{i,\pi,\rho})$. By Lemma 14, we have $\mu_i^\sigma(x), \mu_i^\tau(x) \in \left[\frac{1}{q2^{q-2}}, \frac{1}{2}\right]$ for any $x \in L(v_i)$, thus
$$\mathcal{E}\left(\mu_i^\sigma, \mu_i^\tau\right) \leq 2\left(\ln q + q \ln 2\right) \leq 3q; \tag{3}$$
and if otherwise, w_i is not in S in the SAW tree T, we repeat from the first step for vertex v_i in the new instance $(G_B, \mathcal{L}_{i,\pi,\rho})$.

We can then apply an induction to prove that $\mathcal{E}(\mu_v^\sigma, \mu_v^\tau) \leq \mathcal{E}_{T,\mathcal{L},S}$, with (3) as basis, and (1) and (2) as induction step. We only need to clarify that each application of (1) creates new instances $(G_B, \mathcal{L}_{i,\pi,\rho})$, while $\mathcal{E}_{T,\mathcal{L},S}$ is defined using only the original instance $(G, \mathcal{L})$. This will not cause any issue because by Lemma 13, every new instance $(G_B, \mathcal{L}_{i,\pi,\rho})$ created during this procedure must be feasible. Moreover, the operation the new instance $(G_B, \mathcal{L}_{i,\pi,\rho})$ applying on $(G, \mathcal{L})$ never makes any vertex less permissive, and never increases the multiplicative factor $\frac{1}{|L(v_i)|-d(v_i)-1}$ in the recursion.

4 Strong Spatial Mixing on Random Graphs

In this section, we prove Theorem 2, the strong spatial mixing of q-coloring of random graph $G(n, d/n)$ with respect to a fixed vertex. The theorem is proved by applying Theorem 9 to random graph $G(n, d/n)$. The following lemma states the existing with high probability of a good permissive cutset in the self-avoiding walk tree of a random graph $G(n, d/n)$. The proof is in the full version.

Lemma 16. *Let $d > 1$, $q \geq \alpha d + \beta$ for $\alpha > 2$ and $\beta \geq 23$, and $t(n) = \omega(1)$ an arbitrary super-constant function. Let $v \in V$ be arbitrarily fixed and $G = (V, E)$ a random graph draw from $G(n, d/n)$. The following event holds with high probability: for any $t(n) \leq t \leq \frac{\ln n}{\ln d}$ and any vertex set $\Delta \subset V$ satisfying $\mathrm{dist}_G(v, \Delta) > 2t$, there exists a permissive cutset S in $T = T_{\mathrm{SAW}}(G, v)$ for v and Δ such that $t \leq \mathrm{dist}_T(v, u) < 2t$ for all vertices $u \in S$.*

We then observe that the quantity $\mathcal{E}_{T,\mathcal{L},S}$ decays fast on average. The proof is also in the full version.

Lemma 17. *Let $f_q(x)$ be a piecewise function defined as*
$$f_q(x) = \begin{cases} \frac{1}{q-x-1} & \text{if } x \leq q-2, \\ 1 & \text{otherwise.} \end{cases}$$
Let X be a random variable distributed according to binomial distribution $B(n, \frac{d}{n})$ where $d = o(n)$. For $q \geq 2d + 4$, it holds that $\mathbb{E}\left[f_q(X)\right] < \frac{1}{d}$.

We then prove a strong spatial mixing theorem with the norm of error function $\mathcal{E}(\mu_v^\sigma, \mu_v^\tau)$.

Lemma 18. *Let $d > 1$, $q \geq \alpha d + \beta$ for $\alpha > 2$ and $\beta \geq 23$, and $t(n) = \omega(1)$ an arbitrary super-constant function. Let $v \in V$ be arbitrarily fixed and $G = (V, E)$ a random graph draw from $G(n, d/n)$. There exist constants $C_1, C_2 > 0$ depending only on d and q such that with high probability G is q-colorable and*

$$\mathcal{E}(\mu_v^\sigma, \mu_v^\tau) \leq C_1 \exp(-C_2 \mathrm{dist}(v, \Delta))$$

for any feasible q-colorings $\sigma, \tau \in [q]^\Lambda$ partially specified on a subset $\Lambda \subset V$ of vertices, such that σ and τ differ only on a subset $\Delta \subseteq \Lambda$ with $\mathrm{dist}(v, \Delta) \geq t(n)$.

Proof (Sketch of Proof). We only give a sketch of the proof. The detailed proof is given in the full version.

Fix $v \in V$. Let $T = T_{\mathrm{SAW}}(G, v)$ be the self-avoiding walk tree of G. Fix an arbitrary $t(n) \leq t \leq \frac{\ln n}{\ln d}$. Consider $\mathcal{E}_t = \max_S \mathcal{E}_{T,[q],S}$ where the maximum is taken over all vertex set S in T satisfying $t \leq \mathrm{dist}_T(v, u) < 2t$ for all $u \in S$. By enumerating all self-avoiding walks $P = (v, v_1, \ldots, v_k)$ from v to a vertex $v_k \in S$, we have

$$\mathbb{E}\left[\mathcal{E}_t\right] \leq 3q \sum_{k=t}^{2t-1} d^k \cdot \mathbb{E}\left[\prod_{i=1}^{k} f_q(d_G(v_i)) \,\middle|\, P = (v, v_1, \ldots, v_k) \text{ is a path}\right],$$

where the function $f_q(x)$ is as defined in Lemma 17. We then calculate the expectations. Fix a tuple $P = (v, v_1, \ldots, v_k)$. We construct an independent sequence whose product dominates the $\prod_{i=1}^{k} f_q(d_G(v_i))$.

Conditioning on $P = (v, v_1, \ldots, v_k)$ being a path in G. Let $X_1, X_2, \ldots, X_k$ be such that each X_i is the number of edges between v_i and vertices in $V \setminus \{v_1, \ldots, v_k\}$; and let Y be the number of edges between vertices in $\{v_1, \ldots, v_k\}$ except for the edges in the path $P = (v, v_1, \ldots, v_k)$. Then $X_1, X_2, \ldots, X_k, Y$ are mutually independent binomial random variables, and for each v_i in the path we have $d_G(v_i) = X_i + 2 + Y_i$ for some $Y_1 + Y_2 + \cdots + Y_k = 2Y$.

Due to the property of function $f_q(x)$ we can bound that

$$\prod_{i=1}^{k} f_q(d_G(v_i)) = \prod_{i=1}^{k} f_q(X_i + 2 + Y_i) \leq 4^Y \prod_{i=1}^{k} f_{q-2}(X_i).$$

Since $X_1, X_2, \ldots, X_k, Y$ are mutually independent conditioning on P is a path,

$$\mathbb{E}\left[\prod_{i=1}^{k} f_q(d_G(v_i)) \,\middle|\, P \text{ is a path}\right] \leq \mathbb{E}\left[4^Y \prod_{i=1}^{k} f_{q-2}(X_i)\right] \leq \mathbb{E}\left[4^Y\right] \mathbb{E}\left[f_{q-2}(X)\right]^k,$$

where $\mathbb{E}\left[f_{q-2}(X)\right]$ can be upper bounded by Lemma 17, and $\mathbb{E}\left[4^Y\right]$ by the binomial theorem. Then a calculation gives

$$\mathbb{E}\left[\mathcal{E}_t\right] \leq 3q \sum_{k=t}^{2t-1} \mathbb{E}\left[\prod_{i=1}^{k} f_q(d_G(v_i)) \,\middle|\, P \text{ is a path}\right] \leq \exp\left(-\Omega(t)\right).$$

By Markov's inequality and union bound, with high probability we have $\mathcal{E}_t \leq \exp\left(-\Omega(t)\right)$ for all $t(n) \leq t \leq \frac{\ln n}{\ln d}$.

By [1], w.h.p. G is q-colorable. By Lemma 16, w.h.p. we have good permissive cutset S satisfying the conditions in Lemma 16, which by Theorem 9, implies that $\mathcal{E}(\mu_v^\sigma, \mu_v^\tau) \leq \mathcal{E}_{T,[q],S} \leq \mathcal{E}_t \leq \exp(-\Omega(t))$. By [2], w.h.p. the diameter of G is in $O(\frac{\ln n}{\ln d})$, thus we can choose $t = \Theta(\mathrm{dist}(v, \Delta))$ with a $t(n) \leq t \leq \frac{\ln n}{\ln d}$, which gives us $\mathcal{E}(\mu_v^\sigma, \mu_v^\tau) \leq \exp(-\Omega(\mathrm{dist}(v, \Delta)))$.

With Lemma 18, the proof of Theorem 2 is immediate, which is in the full version.

References

1. Achlioptas, D., Naor, A.: The two possible values of the chromatic number of a random graph. Annals of Mathematics 162(3), 1335–1351 (2005)
2. Chung, F., Lu, L.: The diameter of sparse random graphs. Advances in Applied Mathematics 26(4), 257–279 (2001)
3. Coja-Oghlan, A., Vilenchik, D.: Chasing the k-colorability threshold. In: FOCS, pp. 380–389 (2013)
4. Dyer, M., Frieze, A.: Randomly coloring random graphs. Random Structures & Algorithms 36(3), 251–272 (2010)
5. Dyer, M.E., Flaxman, A.D., Frieze, A.M., Vigoda, E.: Randomly coloring sparse random graphs with fewer colors than the maximum degree. Random Struct. Algorithms 29(4), 450–465 (2006)
6. Efthymiou, C.: A simple algorithm for random colouring $G(n, d/n)$ using $(2 + \varepsilon)d$ colours. In: SODA, pp. 272–280. SIAM (2012)
7. Efthymiou, C.: Mcmc sampling colourings and independent sets of G (n, d/n) near the uniqueness threshold (2014)
8. Efthymiou, C., Spirakis, P.G.: Randomly colouring sparse random graphs using a constant number of colours. Technical report (2007)
9. Frieze, A., Vigoda, E.: A survey on the use of markov chains to randomly sample colourings. Oxford Lecture Series in Mathematics and its Applications 34, 53 (2007)
10. Gamarnik, D., Katz, D.: Correlation decay and deterministic FPTAS for counting colorings of a graph. Journal of Discrete Algorithms 12, 29–47 (2012)
11. Gamarnik, D., Katz, D., Misra, S.: Strong spatial mixing for list coloring of graphs. arXiv preprint arXiv:1207.1223 (2012)
12. Ge, Q., Stefankovic, D.: Strong spatial mixing of q -colorings on bethe lattices. arXiv preprint arXiv:1102.2886 (2011)
13. Goldberg, L., Martin, R., Paterson, M.: Strong spatial mixing with fewer colors for lattice graphs. SIAM Journal on Computing 35(2), 486 (2005)
14. Lu, P., Yin, Y.: Improved FPTAS for multi-spin systems. In: Raghavendra, P., Raskhodnikova, S., Jansen, K., Rolim, J.D.P. (eds.) RANDOM 2013 and APPROX 2013. LNCS, vol. 8096, pp. 639–654. Springer, Heidelberg (2013)
15. Mossel, E., Sly, A.: Rapid mixing of gibbs sampling on graphs that are sparse on average. In: SODA, pp. 238–247 (2008)
16. Mossel, E., Sly, A.: Gibbs rapidly samples colorings of $G(n, d/n)$. Probability Theory and Related Fields 148(1-2), 37–69 (2010)
17. Sinclair, A., Srivastava, P., Yin, Y.: Spatial mixing and approximation algorithms for graphs with bounded connective constant. In: FOCS (2013)
18. Weitz, D.: Mixing in time and space for discrete spin systems. PhD thesis (2004)
19. Weitz, D.: Counting independent sets up to the tree threshold. In: STOC, pp. 140–149 (2006)

Author Index