Heuristic-Based Multi-Agent Monte Carlo Tree Search

Edgar Galván-López*, Ruohua Li*, Constantinos Patsakis†, Siobhán Clarke* and Vinny Cahill*
*Distributed Systems Group, School of Computer Science and Statistics, Trinity College Dublin
{edgar.galvan, liru, siobhan.clarke, vinny.cahill}scss.tcd.ie
†Software Engineering Lab, Department of informatics, University of Piraeus. kpatsak@unipi.gr

***Abstract*—Monte Carlo Tree Search (MCTS) is a relatively new sampling best-first method to search for optimal decisions. The MCTS' popularity is based on its extraordinary results in the challenging two-player based game Go, a game considered much harder than Chess and that until very recently was considered unfeasible for Artificial Intelligence methods. Different MCTS variants have been proposed, mainly to enhance its capabilities. Perhaps, one of the main limitations of this approach is its applicability in scenarios where multiple agents (more than two) are required. Some works have made an attempt to overcome this limitation by using a vector of reward values for each agent and allowing the algorithm to find an optimal equilibrium strategy. Inspired by these approaches, in this work we make an effort to explore a new proposal for handling multiple agents in MCTS by using a vector of values of what the agents need to do (defined tasks) instead of a vector of rewards for each agent. To achieve this we use a rather simple, but powerful heuristic that estimates the desired values of this vector. That is, a set of values that could lead to the optimal completion of the task. We tested this idea in a real-world scenario rather than using it in games as traditionally done. The results achieved by our proposed approach, named Heuristic-Based Multi-Agent Monte Carlo Tree Search, indicate the feasibility of using heuristics in the MCTS algorithm in situations where more than two agents are required.**

***Index Terms*—Monte Carlo Tree Search, Heuristics, Demand-Side Management Systems.**

I. Introduction

Monte Carlo Tree Search (MCTS) is a relatively new sampling method for finding *optimal decisions* by performing random samples in the decision space and building a tree according to partial results.

MCTS has started to attract the attention of researchers as shown by the increasing number of publications over recent years. Perhaps, the main reason for this is due to its major success in the two-player game Go [7], [12], a game that until very recently was considered to be highly difficult for Artificial Intelligence (AI) techniques.

Different variants of the MCTS algorithm have been proposed to use it in different contexts or to improve results on different benchmark problems. In terms of benchmark problems and based on its widely used application (board games), a natural extension of its applicability is in video games [22]. Other applications of MCTS include combinatorial optimisation (e.g., traveling salesman problem [20], [21]), constraint satisfaction (e.g., mathematical expression [3], constraint problems [1]). See [2] for a detailed survey of applications using MCTS.

In terms of MCTS algorithm variants, different approaches have been proposed. For example, a Parallel MCTS [4] has been proposed to exploit today's computer capabilities in terms of using multiple processors and tested with Go. A single-player based MCTS has also been proposed in board games where no opponent is necessary (e.g., SameGame) to estimate potential winning positions (actions) [23]. Other variants of the algorithm include the modification of some of the core components of MCTS (these components are presented in Section II) specifically in trying to improve its selection mechanism [6], [14]. Multi-agent approaches in MCTS have been briefly explored before, but they are very different compared to our approach. For example, in [17] the authors used "multiple" agents to record the outcomes of multiple Monte Carlo simulations in the computer game of Go. This idea is similar to Parallel MCTS where the goal is to get better estimates by performing multiple simulations rather than handling a real multi-agent approach. More closely related works to our approach are those based on the notion of $\max^n$ [16]. The idea is to develop an algorithm capable of being used in multi-player games using a matrix of values. The main difference compared to our approach, is that the $\max^n$ algorithm searches a game tree and finds a strategy which is in equilibrium. In our case, we use a simple, but powerful heuristic that calculates some desirable values for each agent in terms of their goals. That is, the heuristic builds a vector of actions, rather than trying to build a vector of rewards, as carried out by methods based on the $\max^n$ algorithm.

More specifically, the main research contribution of this paper is to extend the MCTS to a multi-agent approach via a matrix-based representation that is created with desired values that can solve the problem. To do so, we use a heuristic method to determine some useful combination of these values and a reward is determined for all the agents. We tested this idea in a real-world problem borrowed from the area of Demand-Side Management Systems[1] (a research area highly popular in smart grids as shown by the increasing number of publications, e.g., [9], [10], [13], [24], [25]): we try to automatically charge the battery of six electric vehicles (EVs) with two, potentially, conflicting goals: (a) that each of the EVs is as fully charged

[1]Demand-Side Management System is normally considered as a mechanism or program, implemented by utility companies, to control the energy consumption at the customer side [18].

as possible at the time of departure, and (b) that the energy usage is intelligently distributed so that the energy load at the transformer level is reduced. We describe in more detail the test bed scenario used in this work in Section II.

This paper is organised as follows. In the following section we present in detail how the MCTS works and present our proposed approach. Section III presents the experimental setup used to test our approach. In Section IV, we present and discuss the results obtained by our approach, and finally, Section V draws some conclusions and discusses future work.

II. Monte Carlo Tree Search

A. The Mechanics Behind MCTS

MCTS relies on two key elements: (a) that the true value of an action can be approximated using simulations, and (b) that these values can be used to adjust the policy towards a best-first strategy. The algorithm builds a partial tree, guided by the results of previous exploration of that tree. Thus, the algorithm iteratively builds a tree until a condition is reached or satisfied (e.g., number of simulations, time given to perform Monte Carlo simulations), then the search is halted and the best performing action is executed. In the tree, each node represents a state, and directed links to child nodes represents actions leading to subsequent states.

Like many AI techniques, MCTS has several variants. Perhaps, the most accepted steps involved in MCTS are those described in [2] and are the following: (a) *Selection:* a selection policy is recursively applied to descend through the built tree until an expandable (a node is classified as expandable if it represents a non-terminal state and also, if it has unvisited child nodes) node has been reached, (b) *Expansion:* normally one child is added to expand the tree subject to available actions, (c) *Simulation:* from the new added nodes, a simulation is run to get an outcome (e.g., reward value), and (d) *Back-propagation:* the outcome obtained from the simulation step is back-propagated through the selected nodes to update their corresponding statistics.

Simulations in MCTS start from the root state (e.g., actual position) and are divided in two stages: when the state is added in the tree, a *tree policy* is used to select the actions (the selection step is a key element and it is discussed in detail later in this section). A *default policy* is used to roll out simulations to completion, otherwise. This is depicted in Figure 1.

We now proceed to define the most popular MCTS algorithm, named Upper Confidence Bounds for Trees, which has been used as a foundation in our work.

B. Upper Confidence Bounds for Trees

The success of MCTS depends heavily on how the tree is build and the selection process plays a fundamental role on this. One particular selection mechanism that has proved to be very reliable is UCB1 (tree policy) [15]. Formally, UCB_j is defined as:

$$UCB_j = \bar{X}_j + 2K\sqrt{\frac{2 \cdot \ln n}{n_j}} \tag{1}$$

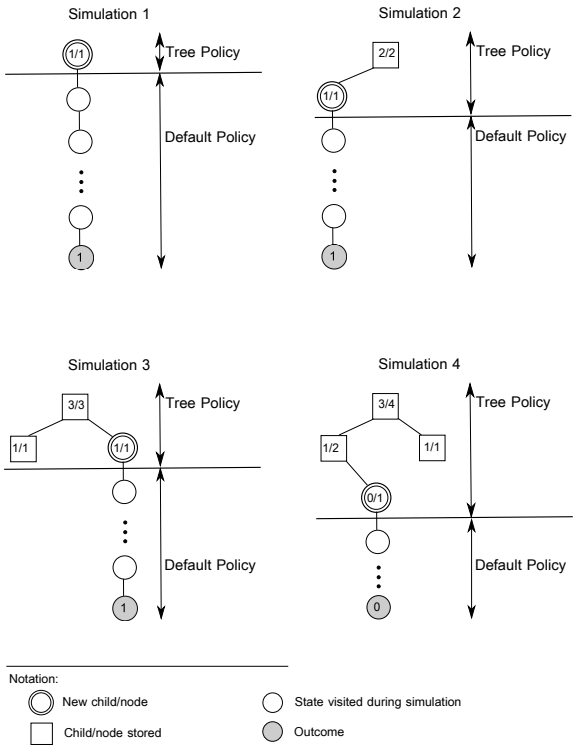

Fig. 1. Four simulations. In Simulation 1, a new node is added to the tree search and its statistics are updated (e.g., outcome and number of visits). Simulation 2, stores the first node and adds a new one. Simulation 3 adds a new node according to the other action (in this case there are only two actions, as indicated by the number of children). Simulation 4 selects (in this case it is a tie, so a random selection is used) and beneath the selected node, a new one is added. Figure adopted from [11].

where n is the number of times the parent node has been visited, n_j the number of times child j has been visited and $K > 0$ is a constant. In case of a tie for selecting a child node, a random selection is normally used.This selection mechanism works due to its emphasis in balancing both exploitation (first part of Eq. 1) and exploration (second part of Eq. 1).

We now proceed to present the multi-agent problem used in this research that will help us to explain, in Section II-C, how we have extended the MCTS algorithm to tackle this type of problems (e.g., multi-agent).

As mentioned before, MCTS has had a profound positive impact in a variety of challenging problems, predominantly in two-player board games. We are interested in exploring the possibility of extending its use in a multi-agent approach for the scenario of charging six electric vehicles with two goals: (a) to guarantee that each EV can complete a journey (e.g., each EV should be charged as much as possible at the time of departure), and (b) to intelligently use energy consumption so that the energy load at the transformer level is reduced.

In our considered benchmark problem there are six users, each with an EV. Each EV can only be charged at home. Moreover, all EVs can only be charged in the same period of time (i.e., t_i = 18:00, t_f = 7:00).

In each time slot a decision has to be made (i.e., every 30

minutes) for each EV to start charging or not. If a decision for an EV is to start charging it, the amount of power consumption can be $1/3$, $2/3$ or 100% of the allowed power consumption. Thus, we have that for each EV, we have four options (actions) at each time slot, including not charging it.

With the problem defined, we now proceed to explain our proposed Heuristic-Based Multi-Agent MCTS.

C. Heuristic-Based Multi-Agent MCTS

With the problem defined in the previous section, we are now in position of explaining how we have extended the MCTS algorithm to support a multi-agent scenario (e.g., charge of six EVs). To do so, we used two main elements in our approach:

1) a vector of desired values (i.e., amount of electricity consumption for each of the EVs), and
2) an heuristic approach to get some "ideal" values for each of the EVs at each time slot.

More specifically, we know that each of the six EVs has four different options (i.e., 0, $1/3$, $2/3$, 100% of the allowed power consumption) at each time step. We use this information to build a vector of values for each node (vector of values) of the tree (Figure 2 depicts this idea). We use a simple heuristic that generates a set of values for the vector such that the summation of these is within a range that could lead to accomplish the (full) charge of each EV at the time of departure, fulfilling the first goal of the problem. Thus, the heuristic aims at generating values given by the following function:

$$C * T_d/T_s < \sum_{n=1}^{n=6} d_{EV_n} < T_d/T_s \tag{2}$$

where C is a constant used to set the lower limit ($k = 0.9$) of what was considered an "ideal value", T_d is the total demand of all the EVs that can be calculated based on the current time slot t_i and the final time slot (t_f), T_s is the total number of slots given by the absolute difference of t_i and t_f.

Notice how this heuristic, in combination with the MCTS algorithm, could eventually lead to a combination of values where the load of the transformer can be minimised while fulfilling the second goal of the problem. We limit our heuristic to generate 100 different combinations of values every time a new node is added to the decision tree. By narrowing this, we can run more simulations and, potentially, find better results. The downside of limiting the number of combinations is that good values for each agent might not be generated. We further discuss this in Section V.

The reward function for all the agents is defined as the difference of the total demand needed by all the EVs and the current demand given by the current time slot, normalised in the region of $[0, 1]$.

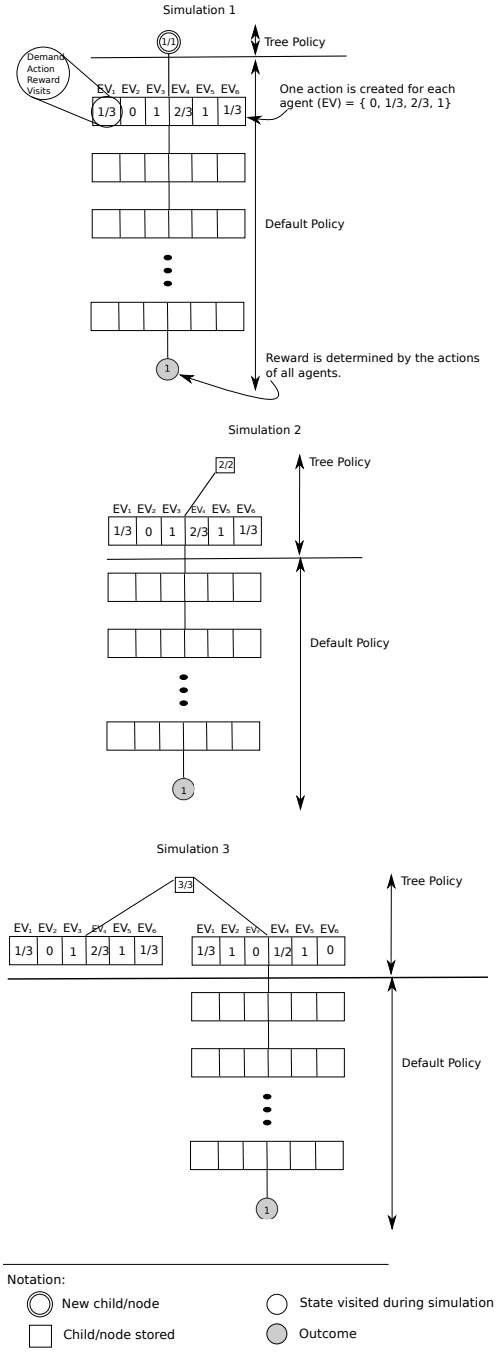

Fig. 2. A schematic view of our proposed Heuristic-Based Multi-Agent Tree Search. Each node contains a set of values that represent the actions that should be executed at the end of the simulations. These values are generated via an heuristic as described in Section II-C.

III. Experimental Setup

A. Demand-Side Management System Scenario

As mentioned previously, the goals of our Heuristic-Based Multi-Agent MCTS in the Demand-Side Management System scenario used in this research are: (a) to guarantee that each EV can complete a journey (e.g., each EV to be charged as much as possible at the time of departure), and (b) that the energy usage is intelligently distributed so that the energy load at the transformer level is reduced. In the next section, we show how both are met by our proposed approach by showing that (a) the state of charge (SoC) at the time of departure is high, and (b) by showing that the averaged transformer load for the period of

TABLE I
SUMMARY OF PARAMETERS USED FOR OUR SMART GRID SYSTEM

Parameter	*Value*
Number of EVs (N)	6
Initial and latest time to charge	t_i =18:00, t_f =7:00
Frequency of making a decision	30 minutes
Number of times slots T	27
State of Charge (SoC) at t_i	Variable (read text)
Type of Charging	0, 1/3, 2/3, 100%
Number of Simulated Days	28

TABLE II
SUMMARY OF PARAMETERS USED FOR OUR APPROACH

Parameter	*Value/Description*
Number of Simulations	10,000
Number of Combinations	100
Number of Actions	$N = 6$ (see Table I)
Constant Value Used in Eq. 1	K=1.0
Constant Value Used in Eq. 2	C=0.9
Reward Value	Variable (see Section II-C)
Selection of Winning Action	Max Child

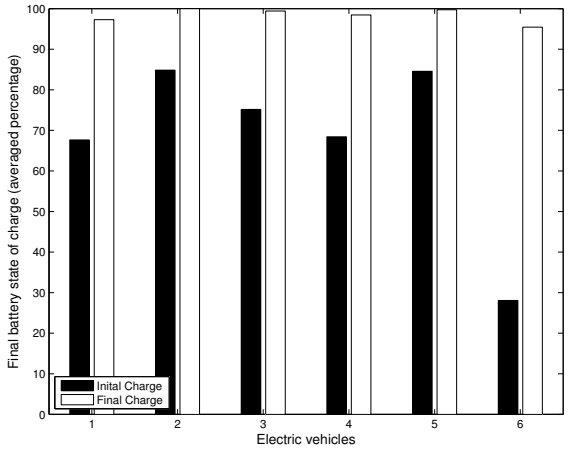

Fig. 3. State of charge (SoC), averaged over a period of 28 days, at the time of arrival for each EV (indicated in black bars) and their corresponding SoC at the time of departure (indicated in white bars) achieved by our proposed Heuristic-Based Multi-Agent Monte Carlo Tree Search.

time where the EVs can be charged and that the peak-of-ratio (PAR)[2] in load demand are reduced, for the goals discussed previously.

In our considered benchmark Demand-Side Management system there are six EVs. Each EV can only be charged at home. Furthermore, all EVs can be charged in the same period of time (i.e., t_i = 18:00, t_f = 7:00).

We simulated a dynamic scenario, where every day, the SoC of each EV at time of arrival (t_i) at home varies as a consequence of having some variations during their driving time (recall that EVs can only be charged once they are at home). Table I summarises the parameters used to simulate our scenario. The MCTS makes a decision for all the EVs, via the use of the heuristic-based approach, every 30 minutes of simuated time. We ran our simulations for 28 days, using GridLab-D (version 2.3) [5]. This is an electrical, open source, grid simulator developed by the US Department of Energy.

B. Heuristic-Based Multi-Agent MCTS

The generation of automatic decisions (actions) were obtained using the proposed approach, as introduced and explained in Section II-C, with 10,000 maximum simulations. Monte Carlo simulations were stopped when the maximum number of simulations was reached. The rest of the parameters used for our proposed Heuristic-Based Multi-Agent MCTS are shown in Table II.

To compare the results achieved by our proposed approach, we used a deterministic approach, denominated in this work as "greedy", where all the EVs start charging as soon as they reach home. To make a fair comparison of both approaches, each EV in these two methods, used exactly the same initial SoC at time of arrival t_i =18:00 at home.

[2]The PAR in load demand is calculated as the highest load across all users for a period of time (e.g., time available to charge EVs) over the average load for the same period of time. Details of this calculation can be found in [19]. Thus, a reduction of PAR is generally preferred.

It is worth notice that the search space is rather large. More precisely, it extends $4^{27} = 2^{54}$ possible states per day for one EV, and $4^{(27*6)}$ for 6 EVs. Therefore, a brute force approach cannot be considered a viable solution.

IV. RESULTS

A. Final Battery State of Charge

Let us start by analysing the results on the first objective: trying to maximise the SoC of all the EVs at the time of departure. Figure 3 shows the average SoC over a period of 28 days, at the time of arrival at home (initial charge is indicated with black bars) and their corresponding SoC at the time of departure (final charge is indicated with white bars). As indicated in Section III, the initial SoC for each EV every day is different as a consequence of different driving patterns. This is clearly visible in Figure 3, where the initial SoC for each EV varies from as little as 29% of the initial SoC (i.e., EV_6) to 85% initial SoC (i.e., EV_2, EV_5).

Our proposed approach behaves fairly well finding, almost, an optimal solution for all the EVs (i.e., $SoC > 98\%$). It should be noticed that the final SoC achieved by the greedy approach (i.e., EVs charging as soon as they reach home) is not reported because all the EVs are able to being fully charged at the time of departure.

We now take a more fine-grained view by analysing how the charge happens for an EV for the 28 days used in this study and using the proposed approach, explained in Section II. Due to space limitations, we only report the results on EV_1 shown in Figure 4. This shows that regardless the initial state of charge, our proposed approach is able to significantly charge the EV. For example, see how for days 13 and 23 the initial SoC is considerable low (less than 55%) and our proposed approach was able to fully charge the battery of EV_1. There are, however, other cases where the results are not that impressive. For instance, see how for day 12, where the initial SoC is around 70% the final SoC is around 82%

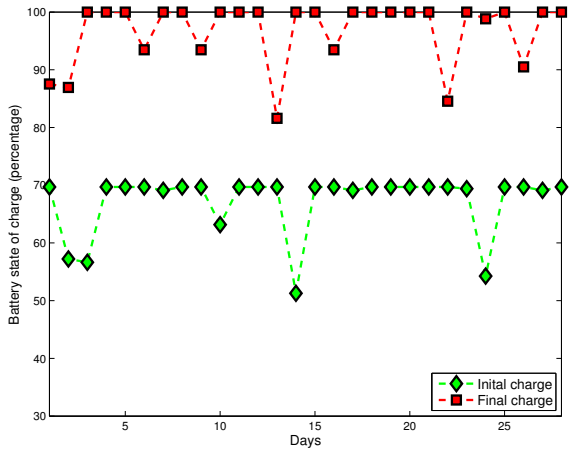

Fig. 4. Battery state of charge (SoC) represented in terms of percentage for EV_1. The initial SoC, indicated in green diamond marks, for each of the 28 days, starts within the range $[50, 70]$ (see Table I for details). The proposed approach is able to find the optimal solution, for the first objective (i.e., final SoC, indicated in red square marks, $SoC = 100$) most of time.

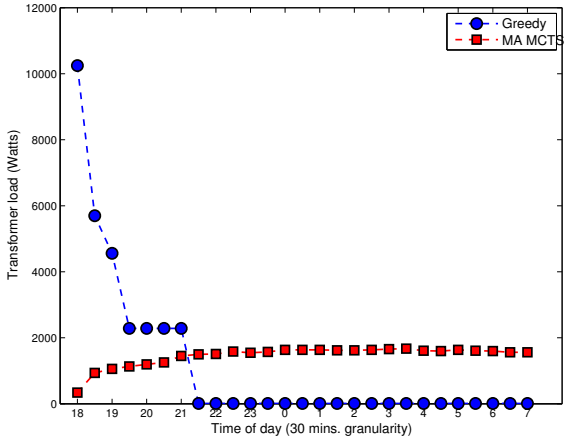

Fig. 5. Transformer load, averaged for 28 days, for 6 EVs with different initial state of charge SoC. All 6 EVs could be charged from 18:00 until 7:00, as indicated in the 'x-axis' of the figure. Blue circles show the transformer load for the greedy approach (i.e., EVs start charging as soon as the reach home) whereas the red squares show the transformer load using our proposed approach.

and a similar trend can be observed for day 22 with 70% and 83% for initial and final SoC, respectively. Overall, the final SoC of EV_1 is quite high, where in the majority of the days, the proposed approach is able to fully charge the battery (e.g., SoC =100%).

B. Transformer Load and Peak-To-Average Ratio

To fully appreciate the results achieved by our approach, we need to analyse its impact at the transformer load. Figure 5 shows the transformer load when the EVs are being charged. Due to the nature of the greedy approach, indicated with blue circles, in Figure 5 (i.e., EVs starting charging as soon as they arrive at home), it is natural to imagine how this approach will achieve the highest peak in electricity load at the initial time of charging (t_i =18:00) and decreases linearly thereafter.

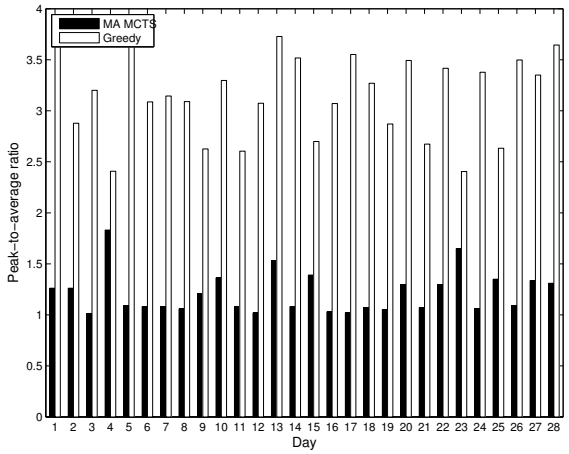

Fig. 6. Peak-to-average ratio (PAR) load demand achieved by both our proposed approach (black bars) vs. the greedy approach (white bars). Our approach is able to reduce the PAR for every day of the simulated defined period of time (i.e., 28 days). A lower PAR is preferred.

In contrast, our proposed approach indicated with red square in Figure 5, is able to balance energy usage during the period of time where the EVs can be charged (t_i =18:00, t_f = 7:00), as expressed in the 'x-axis' of Figure 5) while at the same time all the EVs are being charged as much as possible as discussed in the previous paragraphs. An interesting trend that can be observed by our Heuristic-Based Multi-Agent Monte Carlo Tree Search approach is how the energy consumption starts relatively low and increases slightly after some time to remain fairly constant after 23:00. This indicates that even when the heuristic produces high values for each of the EVs to be charged as much as possible, the MCTS algorithm prefers to pick those combinations with low values (hence the low consumption at the beginning of the charging period) and as time progresses those values generated by the heuristic are slightly higher to complete the first goal (each EV being charged as much as possible, as discussed previously - Figure 3 summaries the final state of charge for each of the EVs).

Finally, let us analyse the impact of our proposed approach *vs.* the greedy approach in the PAR load demand, shown in Figure 6. As indicated in Section III, the PAR is calculated by the maximum load demand for a period of time over the average load demand. Thus, a reduction in PAR is preferred. We can see how our proposed Heuristic-Based Multi-Agent MCTS approach (indicated in white bars in Figure 6) is able to reduce the PAR load demand compared to the greedy approach (indicated with white bars in Figure 6) all the time. This is to be expected. That is, we know that the PAR load demand is calculated by considering the maximum load demand of a period of time. We know, from Figure 5, that the maximum load is considerable reduced by our proposed approach hence we should expect to see a reduction of PAR, in general.

V. Conclusions and Future Work

MCTS has attracted the attention of researchers thanks to its impressive performance in the challenging computer two-

player based game Go.

Perhaps, as a consequence of this, the MCTS algorithm has been used primarily in two-player based board games, although it has been used in other applications such as combinatorial optimisation, constraint satisfaction, to mention a few. This intuitively means that different variants of the MCTS algorithm have been proposed (e.g., single-based, parallel-based approaches). Multi-agent based approaches have been also proposed in MCTS based on the notion of the $\max^n$ algorithm [16], where the idea is to use a vector of reward values for each agent and let the algorithm to find an optimal equilibrium strategy.

Inspired by the latter, we proposed a rather different approach, named Heuristic-Based Multi-Agent Monte Carlo Tree Search, where the idea is to use a vector of values that indicates what the agents should do to complete a defined task rather than using a vector of rewards. To this end, we use a rather simple heuristic that builds this vector with some "desired" values that can lead towards the completion of the task. To test this idea, we decided to use, as a bechmark problem, an scenario where the goal is to charge six electric vehicles as much as possible at the time of departure while at the same trying to balance the energy consumption at the transformer level. The outstanding results obtained by our proposed approach indicate the feasibility of using heuristics to extend the basic MCTS algorithm to a multi-agent approach.

We are planning in extending the proposed approach by considering different requirements (e.g., use of different appliances [8]). It would be also interesting to compare our approach with a similar approach to get a better estimate of how competitive this approach really is (e.g., use of the well-known $\max^n$ approach [16]).

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers and Leonardo Trujillo for their thoughtful comments on the paper. This research was supported by Science Foundation Ireland (SFI) under the Principal Investigator research program 10/IN.1/I2980 "Self-organizing Architectures for Autonomic Management of Smart Cities" and by SFI grant 10/CE/I1855 to Lero.

REFERENCES

[1] S. Baba, Y. Joe, A. Iwasaki, and M. Yokoo. Real-time solving of quantified csps based on monte-carlo game tree search. In *IJCAI*, pages 655–661, 2011.

[2] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *Computational Intelligence and AI in Games, IEEE Transactions on*, 4(1):1 –43, March 2012.

[3] T. Cazenave. Nested monte-carlo expression discovery. In *Proc. of the 2010 conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, pages 1057–1058, The Netherlands, 2010. IOS Press.

[4] G.-B. Chaslot, M. Winands, and H. Herik. Parallel monte-carlo tree search. In H. Herik, X. Xu, Z. Ma, and M. Winands, editors, *Computers and Games*, volume 5131 of *Lecture Notes in Computer Science*, pages 60–71. Springer Berlin Heidelberg, 2008.

[5] D. Chassin, K. Schneider, and C. Gerkensmeyer. Gridlab-d: An open-source power systems modeling and simulation environment. In *Transmission and Distribution Conference and Exposition, 2008. IEEE/PES*, pages 1–5, 2008.

[6] P.-A. Coquelin and R. Munos. Bandit algorithms for tree search. In R. Parr and L. C. van der Gaag, editors, *UAI*, pages 67–74. AUAI Press, 2007.

[7] R. Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *Proceedings of the 5th international conference on Computers and games*, CG'06, pages 72–83, Berlin, Heidelberg, 2007. Springer-Verlag.

[8] E. Galvan, C. Harris, I. Dusparic, S. Clarke, and V. Cahill. Reducing electricity costs in a dynamic pricing environment. In *Proc. Third IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 169 – 174, Tainan, Taiwan, november 2012. IEEE Press.

[9] E. Galván-López, C. Harris, L. Trujillo, K. Rodriguez-Vazquez, S. Clarke, and V. Cahill. Autonomous Demand-Side Management System Based on Monte Carlo Tree Search. In *IEEE International Energy Conference*, Dubrovnik, Croatia, May 2014. IEEE.

[10] E. Galván-López, A. Taylor, S. Clarke, and V. Cahill. Design of an Automatic Demand-Side Management System Based on Evolutionary Algorithms. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, Gyeongju, Korea, March 2014. ACM.

[11] S. Gelly and D. Silver. Monte-carlo tree search and rapid action value estimation in computer go. *Artif. Intell.*, 175(11):1856–1875, July 2011.

[12] S. Gelly, Y. Wang, R. Munos, and O. Teytaud. Modification of UCT with patterns in Monte-Carlo Go. Technical Report 6062, INRIA, France, Nov. 2006.

[13] C. Harris, I. Dusparic, E. Galván-López, A. Marinescu, V. Cahill, and S. Clarke. Set Point Control for Charging of Electric Vehicles on the Distribution Network. In *IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, Washington, D.C., USA, Feb 2014. IEEE.

[14] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In *Proceedings of the 17th European conference on Machine Learning*, ECML'06, pages 282–293, Berlin, Heidelberg, 2006. Springer-Verlag.

[15] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In *ECML*, pages 282–293, 2006.

[16] C. Luckhart and K. B. Irani. An algorithmic solution of n-person games. In T. Kehler, editor, *AAAI*, pages 158–162. Morgan Kaufmann, 1986.

[17] L. S. Marcolino and H. Matsubara. Multi-agent monte carlo go. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '11, pages 21–28, Richland, SC, 2011. International Foundation for Autonomous Agents and Multiagent Systems.

[18] G. M. Masters. *Renewable and Efficient Electric Power Systems*. Wiley-Interscience, 2004.

[19] A. Mohsenian-Rad, V. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia. Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid. *Smart Grid, IEEE Transactions on*, 1(3):320 –331, dec. 2010.

[20] D. Perez, P. Rohlfshagen, and S. M. Lucas. The physical travelling salesman problem: Wcci 2012 competition. In *IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2012.

[21] A. Rimmel, F. Teytaud, and T. Cazenave. Optimization of the nested monte-carlo algorithm on the traveling salesman problem with time windows. In *Proceedings of the 2011 international conference on Applications of evolutionary computation - Volume Part II*, EvoApplications'11, pages 501–510, Berlin, Heidelberg, 2011. Springer-Verlag.

[22] S. Samothrakis, D. Robles, and S. Lucas. Fast approximate max-n monte-carlo tree search for ms pac-man. *Computational Intelligence and AI in Games, IEEE Transactions on*, PP(99):1, 2011.

[23] M. Schadd, M. Winands, H. Herik, G.-B. Chaslot, and J. Uiterwijk. Single-player monte-carlo tree search. In H. Herik, X. Xu, Z. Ma, and M. Winands, editors, *Computers and Games*, volume 5131 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 2008.

[24] A. Taylor, I. Dusparic, E. Galván-López, S. Clarke, and V. Cahill. Transfer Learning in Multi-Agent Systems Through Parallel Transfer. In *Workshop on Theoretically Grounded Transfer Learning at the 30th International Conference on Machine Learning (Poster)*, volume 28, Atlanta, USA, 2013.

[25] A. Taylor, E. Galván-López, S. Clarke, and V. Cahill. Accelerating Learning in Multi-Objective Systems through Transfer Learning. In *In a Special Session on Learning and Optimization in Multi-Criteria Dynamic and Uncertain Environments at the International Joint Conference on Neural Network 2014 (IEEE IJCNN)*, Beijing, China, 2014. IEEE.