The Importance of Neutral Mutations in GP

Conference Paper *in* Lecture Notes in Computer Science · January 2006

DOI: 10.1007/11844297_88 · Source: DBLP

2 authors, including:

Katya Rodriguez
Universidad Nacional Autónoma de México
71 PUBLICATIONS **479** CITATIONS

The Importance of Neutral Mutations in GP

Edgar Galván-López[1] and Katya Rodríguez-Vázquez[2]

[1] University of Essex, Colchester, CO4 3SQ, UK
egalva@essex.ac.uk
[2] IIMAS-UNAM, Circuito Escolar s/n, Ciudad Universitaria
Del. Coyoacán, México, D.F. 04510, MEXICO
katya@uxdea4.iimas.unam.mx

Abstract. Understanding how neutrality works in EC systems has drawn increasing attention. However, some researchers have found neutrality to be beneficial for the evolutionary process while others have found it either useless or worse. We believe there are various reasons for these contradictory results: (a) many studies have based their conclusions using crossover and mutation as main operators rather than using only mutation (Kimura's studies were done analysing only mutations) and, (b) studies often consider problems and representation with larger complexity. The aim of this paper is to analyse how neutral mutations tend to behave in GP and establish how important they are. For this purpose we introduce an approach which has two advantages: (a) it allows us to specify neutrality and, (b) this makes possible to understand how neutrality affects the evolutionary search process.

1 Introduction

In late 1960s, Motoo Kimura observed that mutations were present more often than previously thought. Evolutionary Computation (EC) systems are mostly inspired from the theories of genetic inheritance and natural selection. However, Neutral theory [8] has interested some researchers who want to understand it so that they can incorporate it in their EC systems to solve complex problems. This theory suggests that a mutation from a gene to another is neutral if this modification does not affect the phenotype.

Kimura's theory seems to contradict the Darwinian theory but this is not right. The Darwinian theory judges genes by their phenotypic expression whereas Kimura's theory argues that mutations occurring during evolution are neither advantageous nor disadvantageous to the survival and reproduction of individuals. Such random genetic drift should be considered into the study of the evolutionary process which is an issue neglected by the EC community.

Some researchers have made effort to understand how neutrality works in EC systems in order to add elements to the evolutionary process to evolve complex problem solutions. In Genetic Programming (GP) [9], neutrality if often identified with redundancy and introns. Both have being widely studied in the EC community [1, 10, 12–14].

Functional redundancy refers to the fact that many different individuals, at the phenotype level, represent the same function. For instance the following two genotypes represent the XOR function:

(NOR (AND (NOT (NOT A)) B) (NOT (OR A B)))
(NOR ((NAND (NAND A B) (OR A B)) (NOT (OR A B))))

Introns refers to code that is part of an individual but that semantically does not affect the program's behaviour. A good example of an intron could be found in a typical individual generated for the artificial ant problem [6]. Suppose that in the root node there is an IF instruction, which means that either the left or right subtree will not be executed, so, any change in the subtree which is not executed will have no effect on the behaviour of such individual.

The problem with functional redundancy and introns is that both emerge and vary during the evolutionary process and for this reason it is very difficult to measure and study the effects of neutrality.

The aim of this paper is to analyse how neutral mutations behave in GP and establish how important they are. For this reason, we introduce a new approach to study the effects of neutrality. This method has two advantages: (a) it allows us to specify neutrality, (b) this makes possible to understand how explicit neutrality[3] affects the evolutionary search process.

The paper is organized as follows. In Section 2, previous work on neutrality is presented. Section 3, the approach used to carry out our research is described. Section 4 provides details on the experimental setup used and results are presented. In Section 5 conclusions are drawn.

2 Previous Work

As we will see in the next paragraphs, neutrality theory has been explored in Genetic Algorithms. However, neutrality could be easier to find in GP due to its representations.

Harvey and Thompson [5] studied some effects of neutral networks in an evolvable hardware problem. In their work, they defined the concept of potentially useful junk that refers to loci in a genotype that are functionless within the current context, but with different values elsewhere in the genotype they may become functional. Harvey and Thompson argued that it is possible to reach a global optimum without worrying about premature convergence if one uses neutrality in the evolutionary process.

Banzhaf [2] proposed an approach where a genotype-phenotype mapping was used in the context of constrained optimisation problems. Banzhaf argued that, very often, constraining the solution space leads to local optima which are difficult to escape from with traditional methods. He used high variability of neutral variants to escape from local optima on saddle surfaces. Keller and Banzhaf extended this work in [7].

[3] Term coined by Yu and Miller [19], which means that neutrality can be added to the evolutionary process.

Shipman *et al.* [15] explored the benefits of neutrality in the context of a mapping based on an abstraction of a genetic regulatory network — a random Boolean network. The mapping used in their experiments provided a very large degree of neutrality. From the experimental results they concluded that neutral drift allowed the discovery of many more phenotypes than would be the case with a direct encoding without redundancy. In [16] they proposed four different redundant mappings to study their effect in the evolutionary process and see how neutrality influences the search. They argued that redundancy was useful in three of their mappings. They concluded that some kind of redundancy (neutrality) is crucial.

Smith *et al.* [17] analysed the effects of the presence of neutral networks on the evolutionary process. They observed how evolvability was affected by the presence of such neutral networks. For this purpose they used a system with an extremely complex genotype-to-fitness mapping. They concluded that the existence of neutral networks in the search space, which allows the evolutionary process to escape from local optima, does not necessarily provide any advantage. This is because the population does not evolve any faster due to inherent neutrality. In [18] they focused their research on looking at the dynamics of the population rather than looking at just the fitness, and argued that neutrality did not perform a useful role in an evolutionary robotic task.

Yu and Miller [19] showed that neutrality improves the evolutionary search process for a Boolean benchmark problem. They used Miller's Cartesian GP [11] to measure explicit neutrality in the evolutionary process. They have explained that mutation on a genotype that has part of its genes active and others inactive may produce different effects: mutation on active genes is adaptive because it exploits accumulated beneficial mutations, while mutation on inactive genes has a neutral effect on a genotype's fitness, yet it provides exploratory power by maintaining genetic diversity. Yu and Miller extended this work in [20] showing that neutrality was helpful and that there is a relationship between neutral mutations and success rate in a Boolean function induction problem. However, Collins [4] claimed that the conclusion that, in this problem, neutrality is beneficial is flawed.

Yu and Miller [21] also investigated neutrality using the simple OneMax problem. They attempted a theoretical approach in this work. With their experiments, they showed that neutrality is advantageous because it provides a buffer to absorb destructive mutations.

Chow [3] proposed a method that uses individuals which contains multiple chromosomes instead of a single chromosome. The idea of his approach was to apply genetic operators which do not maintain a one-to-one mapping between a genotypic bit and a phenotypic bit. Chow tested his approach in well known deceptive problems with good results.

As it can be seen from the brief summaries provided above, some researchers have found neutrality to be beneficial to evolutionary process while others have found it either useless or worse. We believe there are various reasons of why contradictory results on neutrality have been reported. In the next section we will describe in detail the proposed approach.

3 Graph-GP Representation

We believe that contradictory results regarding neutrality have several reasons:

- many studies have based their conclusions using crossover and mutation as main operators rather that using only mutation,
- studies often consider problems, representations and search algorithms that are relatively complex and so results represent the composition of multiple effects (e.g., bloat or spurious attractors in GP).

In this paper, we make an effort to clarify these problems. That is,

- We use the traditional representation as suggested by Koza, with the difference to allow explicit neutrality,
- We analyse performance with and without the presence of neutrality, using only the mutation operator.

The inspiration of using GP to study some effects of neutrality in the evolutionary process comes from many facets of their properties.

Programs are represented in GP as parse tree, rather than as lines of code. For instance, the expression `AND(a AND (b OR b), a OR a)` could be expressed as shown in Figure 1(a). The set of internal nodes used in GP parse trees is called *function set*, $F = \{f_1, \cdots, f_{NF}\}$. The function set could include almost any kind of programming construct: arithmetic operators, Boolean functions, looping instructions, etc. The set of terminal nodes is called *terminal set* $T = \{t_1, \cdots, t_{NT}\}$. This set can include variables, constants, random constants, etc.

For our experiments, $F = \{AND, OR, NOT\}$, while $T = \{a, b, c, \cdots\}$ representing input wires. Moreover, we have added an extra element in the function set, p. This p symbol works as follows: (a) Once an individual has been created as usual, we use a probability to replace a function with a p symbol which is a function of arity 2, which means that only functions of arity 2 can be replaced by p symbol. However, this is not a restriction because p can be defined of any arity, (b) If an individual contains this p symbol, this will point to code somewhere in the program, so when p is executed, the subtree rooted at that node is ignored, (c) If p symbol points to a function symbol, the p symbol effectively represents the sub-tree rooted at that function, and, (d) If p symbol points to a terminal symbol, the p symbol simply represents that node.

As can be seen, when p symbol is executed, the subtree rooted at that node is ignored, and so plays the role of inactive code[4]. When the mutation is applied in the inactive code, the individual will change at the genotype level but it will not change at the phenotype level. Figure 1(b) illustrates the concept.

The mutation operator is applied as usual on a per node basis. The only difference is that when a mutation is applied to the p symbol, we reassign the position to which it is pointing to. The fitness function that we used for circuit design is the raw fitness, where we assign the fitness to the individual according to the number of correct output bits.

[4] However, it is worth pointing out that this inactive code can be activated if p point to this code.

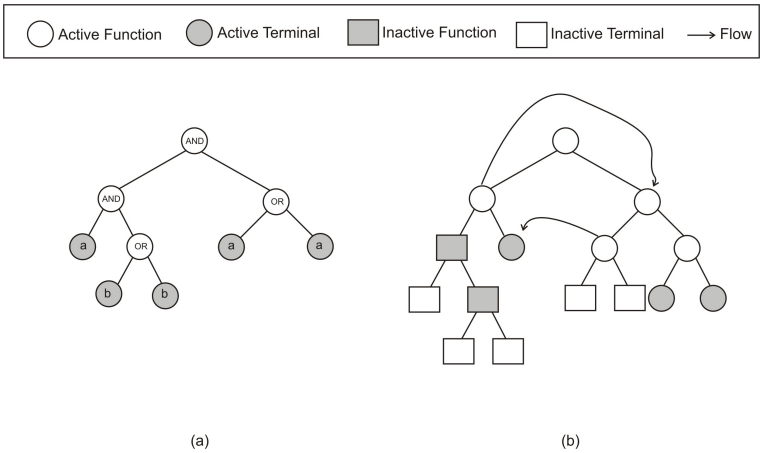

Fig. 1. (a) Parse-tree representation, and, (b) Graph-GP representation created with our approach.

Table 1. Summary of Parameters.

Parameter	*Value*
Depth	6, 7, 8, 9, 10
Population Size	200
Generations	400
Mutation Rate per node	0.02, 0.03, 0.04, 0.05, 0.06
P Rate per node	0.00, 0.01, 0.02, 0.03, 0.04

4 Experiments

4.1 Experimental Setup

In this section we use the 6-bit Multiplexer Boolean function to evaluate the proposed approach. To obtain meaningful and conclusive results, we performed 20 independent runs for each of different mutation rates and different p rates. We will show the results and analyse them in the following section. Moreover, runs were stopped when the maximum number of generations was reached. The parameters we have used are shown in Table 1 (these parameters were defined after a series of preliminary experiments). Initialization of the population was made with the full method. Crossover operator was not used in any of our experiments.

4.2 Results and Analysis

Due to space limitations we will focus our attention on results generated with depth 8. However, it is worth pointing out that results with depths 6, 7, 9 and 10 are very similar to those found with depth 8.

Table 2. Results found with our approach in the 6-bit Multiplexer. P and Mutation rate values are shown in the first and second column, respectively. Average of Fitness, Standard Deviation (σ) and Median are shown in the third, fourth and fifth column, respectively. Finally, Feasible Circuits (Success Rate) and Average of Generations (this refers to the average number of generations that are necessary to reach the feasible zone) are shown in the last two columns.

% P	*% Mutation*	*Avr. of Fit.*	*Standard Deviation*	*Median*	*F. Circuits*	*Avr. of Gen.*
0.01	0.02	64	0	64	100%	80.6
0.01	0.03	63.6	1.23	64	90%	155.66
0.01	0.04	62.65	2.98	64	70%	138.62
0.01	0.05	62.4	1.7	62.5	55%	146.65
0.01	0.06	62.25	1.37	62	30%	203.5
0.02	0.02	63.15	1.76	64	75%	91.46
0.02	0.03	62.5	2.42	64	65%	136.66
0.02	0.04	62.4	2.01	63	50%	141.5
0.02	0.05	60.95	3.24	61	35%	163.71
0.02	0.06	60.35	3.5	60	35%	243.14
0.03	0.02	63.5	2.86	64	85%	122.7
0.03	0.03	63.6	1.05	64	85%	109.64
0.03	0.04	62.3	2	62.5	45%	179.5
0.03	0.05	62	2.68	63	50%	237.5
0.03	0.06	60.7	3.51	61.5	30%	157.56
0.04	0.02	61.5	4.05	64	65%	109.65
0.04	0.03	61.3	4.17	64	60%	117
0.04	0.04	62.4	3.27	64	65%	132.45
0.04	0.05	61.9	2.95	63	45%	220.54
0.04	0.06	60.2	3.29	60	30%	164

From Figure 2 we can see the success rates found by our approach with different p and mutation rate values. The highest success rate found was 100% with $p = 0.01$ and mutation rate = 0.02. Keeping constant this p value and increasing the mutation rate values, the success rate tends to decrease.

Similar behaviour can be observed with different p rate values. Therefore, we can conclude that regardless the value of p is, the higher the mutation rate is, the lower the success rate will be.

At this point, one question arises: what happen if we do not allow the presence of the p element in our individuals? To answer this question we need to take a look to Figure 2. In no case the system was able to reach a success rate of 100% in the absence of the p symbol. Moreover, the performance of the GP system without the presence of p is poor comparing when it is present. Actually, the performance of the GP system when p is not present in the individuals is, the worst for all mutation rates, except when mutation rate is 0.03.

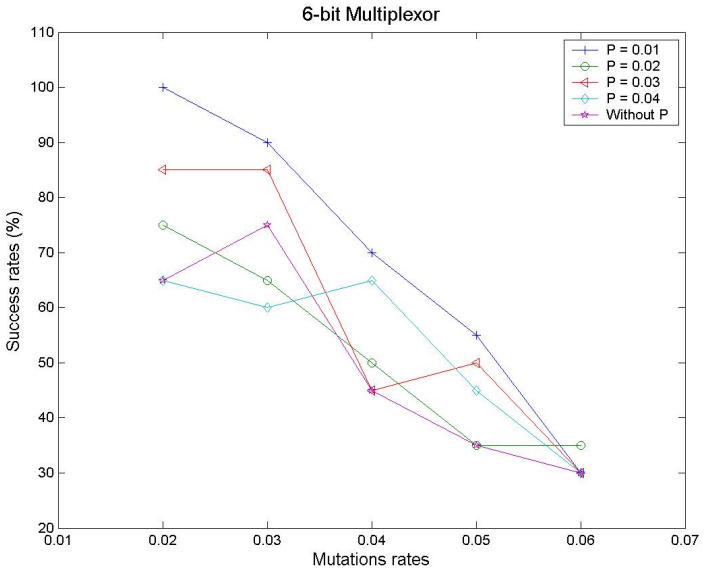

Fig. 2. Success rate results. P refers to the p rates used in our experiments.

Table 2 shows additional details on our results. The last column reports the average number of generations that are necessary to reach the feasible region[5]. We can see that the lower the value of p is, the smaller the number of generations that are required to solve the problem.

The experimental results also show how the individuals in the population tends to behave in the presence of p in their structures. Figure 3 summarizes such a behaviour on 4 different p rates. The highest success rates were found when mutation rates were set with the lowest value (0.02), regardless of the p rates.

At the beginning of the evolutionary process, the number of individuals affected by neutral mutations is high but it tend to decrease after few generations. In other words, individuals with p elements in their structures tend to disappear at the beginning of the process. We think this happened because at the beginning of the evolutionary process the solution needs to be protected by allowing the presence of p in their structures. However, further analysis need to be done to know why this happen.

Around generation 50 - 60 it is when the number of individuals affected by neutral mutations becomes stable. As can be observed in all plots in Figure 3, the best performance is achieved when the number of individuals affected by neutral mutations is in the range of 90 - 100 (notice that this range is close to

[5] The feasible region is the area of the search space containing circuits that match all the outputs of the problem's truth table

half of the population). On the other hand, the worst performance was found when the number of individuals affected by neutral mutations is below 80.

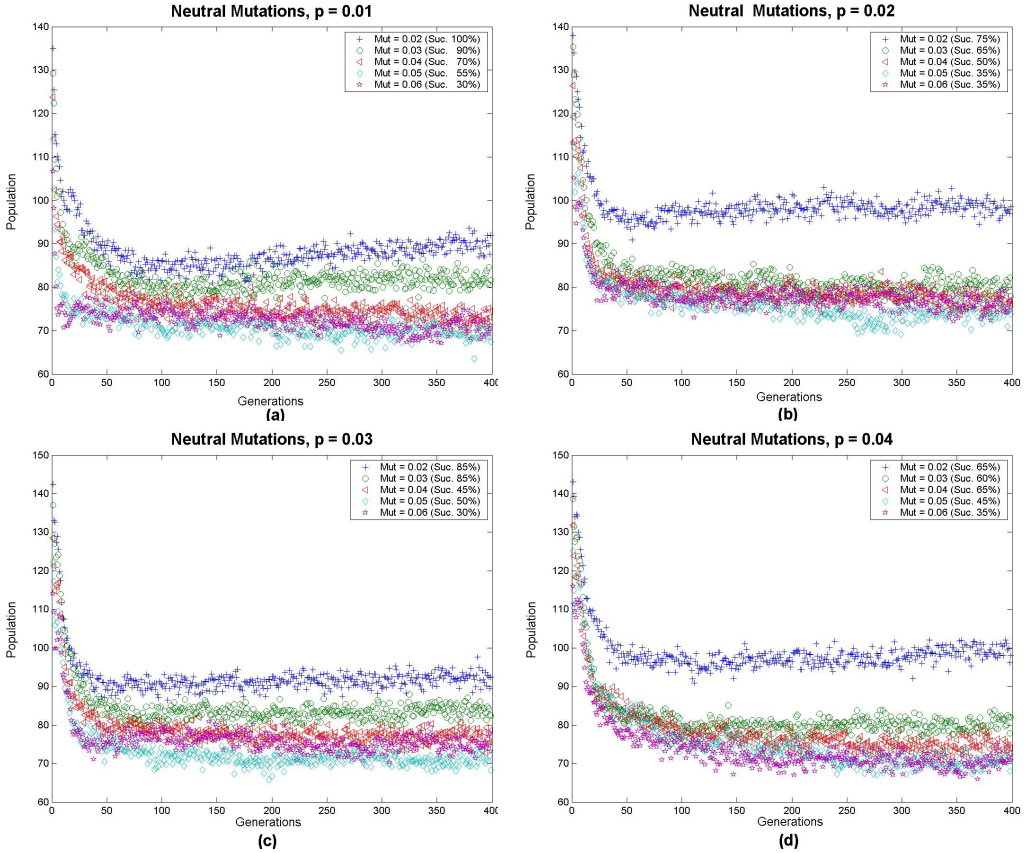

Fig. 3. Number of individuals affected by neutral mutations. $p = 0.01$ (a), $p = 0.02$ (b), $p = 0.03$ (c), $p = 0.04$ (d).

From this analysis, it is clear that the presence of p in the individuals can make the solution avoid get stuck in local optimum. However, a balance between p rate and mutation rate is needed in order to improve the exploration of the search space.

5 Conclusions

In late 1960s, Motoo Kimura observed that mutations were present more often than previously thought. He explained this phenomenon with the concept of Neutrality which established that the majority of mutations that are present

during the evolutionary process do not have impact at the phenotype level. This paper makes an analysis of some effects of neutrality in GP. Moreover, we have shown that neutrality is an important research area to be considered in the evolutionary process. With the approach described in this paper, we have been able to analyse some effects of neutrality.

From results found for the benchmark Boolean problem, we conclude that: (a) Neutral Mutations help to the evolutionary process to reach feasible regions, (b) Regardless the value of p, with higher mutation rates the success rates are low, (c) Neutral Mutations do not allow to get stuck in local optima, and (d) With low probability of p and low probability of mutation, the evolutionary process tends to behave in a consistent way and shows better overall performance.

Further work need to be done about the effects of p symbol in tree's structures. The amount of neutrality and the mutation rate present in the evolutionary process play an important role during the evolutionary process. In the future we will investigate the reasons behind the fine balance between these two elements that is required to aid evolution. We also would like to know why the presence of p in individuals' structure tends to decrease in the evolutionary process and with the use of family trees we can be able to clarify this point. In a mutation based algorithm each individual has only one parent. This makes it possible to track the origin of a sample point, and, in fact, the full evolutionary path of an individual within its family tree.

Acknowledgments. The first author thanks to CONACyT for support to pursue graduate studies at University of Essex. The second author gratefully acknowledges support from CONACyT through project 40602-A. The authors would like to thank the anonymous reviewers for their valuable comments.

References

1. P. Angeline. Genetic Programming and Emergent Intelligence. In K. E. Kinnear, Jr., editor, *Advances in Genetic Programming*, pages 75–98. MIT Press, 1994.
2. W. Banzhaf. Genotype-phenotype-mapping and neutral variation – A case study in genetic programming. In Y. D. *et al.*, editor, *Parallel Problem Solving from Nature III*, volume 866 of *LNCS*, pages 322–332. Springer-Verlag, 9-14 Oct. 1994.
3. R. Chow. Evolving genotype to phenotype mappings with a multiple-chromosome genetic algorithm. In K. D. *et al.*, editor, *GECCO 2004: Proceedings of the 2004 Conference on Genetic and Evolutionary Computation*, volume 1, pages 1006–1017, Seattle WA, USA, 26-30 June 2004. Springer-Verlag.
4. M. Collins. Finding needles in haystacks is harder with neutrality. In H.-G. B. *et al.*, editor, *GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation*, volume 2, pages 1613–1618, Washington DC, USA, 25-29 June 2005. ACM Press.
5. I. Harvey and A. Thompson. Through the labyrinth evolution finds a way: A silicon ridge. In *Proceedings of the First International Conference on Evolvable Systems: From Biology to Hardware (ICES)*, pages 406–422. Springer-Verlag, 1996.

6. D. Jefferson, R. Collins, C. Cooper, M. Dyer, M. Flowers, R. Korf, C. Taylor, and A. Wang. Evolution as a Theme in Artificial Life: The Genesys/Tracker System. In C. G. L. *et al.*, editor, *Artificial Life II*, volume X of *Santa Fe Institute Studies in the Sciences of Complexity*, pages 549–578. Addison-Wesley, Febrary 1992.
7. R. E. Keller and W. Banzhaf. Genetic programming using genotype-phenotype mapping from linear genomes into linear phenotypes. In J. R. K. *et al.*, editor, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 116–122, Stanford University, CA, USA, 28–31 July 1996. MIT Press.
8. M. Kimura. Evolutionary rate at the molecular level. In *Nature*, volume 217, pages 624–626, 1968.
9. W. Langdon and R. Poli. *Foundations of Genetic Programming*. Springer, 2002.
10. S. Luke. Code Growth is Not Caused by Introns. In D. Whitley, editor, *Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference*, pages 228–235, Las Vegas, Nevada, USA, 8 July 2000.
11. J. F. Miller and P. Thomson. Cartesian genetic programming. In R. P. *et al.*, editor, *Genetic Programming, Proceedings of EuroGP'2000*, volume 1802 of *LNCS*, pages 121–132, Edinburgh, 15-16 Apr. 2000. Springer-Verlag.
12. P. Nordin and W. Banzhaf. Complexity Compression and Evolution. In L. Eshelman, editor, *Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95)*, pages 310–317, Pittsburgh, PA, USA, 1995. Morgan Kaufmann.
13. P. Nordin, W. Banzhaf, and F. Francone. Introns in Nature and in Simulated Structure evolution. In D. L. *et al.*, editor, *Bio-Computation and Emergent Computation*, Skovde, Sweden, 1-2 Sept. 1997. World Scientific Publishing.
14. P. Nordin, F. Francone, and W. Banzhaf. Explicitly Defined Introns and Destructive Crossover in Genetic Programming. In J. P. Rosca, editor, *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, pages 6–22, Tahoe City, California, USA, 9 July 1995.
15. R. Shipman, M. Schackleton, and I. Harvey. The use of neutral genotype-phenotype mappings for improved evolutionary search. *BT. Technology Journal*, 18(4):103–111, October. ISSSN 2000.
16. R. Shipman, M. Schakleton, M. Ebner, and R. Watson. Neutral search spaces for artificial evolution: A lesson from life. In M. Bedau, S. Rasmussen, J. McCaskill, and N. Packard, editors, *Artificial Life: Proceedings of the Seventh International Conference on Artificial Evolution*, pages 162–169. MIT Press, 2000.
17. T. Smith, P. Husbands, and M. O'Shea. Neutral networks and evolvability with complex genotype-phenotype mapping. *Lecture Notes in Computer Science*, 2159:272–282, 2001.
18. T. Smith, P. Husbands, and M. O'Shea. Neutral networks in an evolutionary robotics search space. In *Congress on Evolutionary Computation: CEC 2001*, pages 136–145. IEEE Press, 2001.
19. T. Yu and J. Miller. Neutrality and the evolvability of boolean function landscape. In *Fourth European Conference on GP*, pages 204–211. Springer-Verlag, 2001.
20. T. Yu and J. F. Miller. Needles in haystacks are not hard to find with neutrality. In J. A. F. *et al.*, editor, *Genetic Programming, Proceedings of the 5th European Conference, EuroGP 2002*, volume 2278 of *LNCS*, pages 13–25, Kinsale, Ireland, 3-5 Apr. 2002. Springer-Verlag.
21. T. Yu and J. F. Miller. The role of neutral and adaptive mutation in an evolutionary search on the onemax problem. In E. Cantú-Paz, editor, *Late Breaking Papers at the Genetic and Evolutionary Computation Conference (GECCO-2002)*, pages 512–519, New York, NY, July 2002. AAAI.