Proceedings of the International Multiconference on
Computer Science and Information Technology pp. 409–419
ISSN 1896-7094

An investigation of the mutation operator using different representations in Grammatical Evolution

Jonatan Hugosson, Erik Hemberg, Anthony Brabazon & Michael O'Neill

Natural Computing Research & Applications Group
University College Dublin, Ireland
jhugosson@gmail.com, erik.hemberg@ucd.ie, anthony.brabazon@ucd.ie,
m.oneill@ucd.ie

Abstract. Grammatical evolution (GE) is a form of grammar-based genetic programming. A particular feature of GE is that it adopts a distinction between the genotype and phenotype similar to that which exists in nature by using a grammar to map between the genotype and phenotype. This study seeks to extend our understanding of GE by examining the impact of different genotypic representations in order to determine whether certain representations, and associated diversity-generation operators, improve GE's efficiency and effectiveness. Four mutation operators using two different representations, binary and gray code representation respectively, are investigated. The differing combinations of representation and mutation operator are tested on three benchmark problems. The results provides support for the continued use of the standard genotypic integer representation as the alternative representations do not exhibit higher locality nor better GE performance. The results raise the question as to whether higher locality in GE actually improves GE performance.

1 Introduction

Grammatical evolution (GE) [15, 14, 20] is a form of grammar-based genetic programming. A special feature of GE is that, unlike genetic programming, it has a clear distinction between the genotype and phenotype. The mapping of the genotype and phenotype is governed by a grammar and this grammar can contain domain knowledge to bias the form a phenotypic solution can take. By separating the search and solution spaces, GE allows the implementation of generic search algorithms without a requirement to tailor the diversity-generating operators to the nature of the phenotype. A substantial literature has emerged on GE and its applications [15, 1, 19, 17]. Some of the more recent developments of GE are focused on the various components of the GE approach including the use of alternative search engines [10, 13], the use of alternative grammar constructs [11, 3, 9, 16], and the examination of different mapping processes in GE [12]. One aspect of GE which has seen less research is the examination of the impact of the choice of genotypic representation, and associated diversity-generation operators, on GE's efficiency and effectiveness. A recent paper by Oetzel and

Rothlauf [19] examined the locality properties of a binary representation in GE and found that a genotypic bit-mutation operator produced non-local changes in the phenotype. The authors of this study proposed that further research be undertaken in order to find other representations and associated mutation operators which would produce higher locality, suggesting that this would increase the performance and effectiveness of GE. This study addresses this research issue by investigating the impact of four mutation operators using two different representations, binary and Gray code representation respectively, on the performance of GE. The combinations are tested using three standard benchmark problems, symbolic regression, the Santa Fe ant trail and the even-5-parity problem.

The remainder of the paper is structured as follows. Section 2 describes GE and provides background on earlier work on representations. Section 3 details the experimental approach adopted and results, and finally section 4 details conclusions and future work.

2 Background

This section provides an introduction to GE and to some prior work on the importance of representation in evolutionary algorithms. GE is an evolutionary algorithm which has some similarities to genetic programming (GP) [7]. Rather than representing the programs as parse trees, as in GP, a linear genome representation is used. A genotype-phenotype mapping is employed such that each individual's variable length binary string, contains in its codons (groups of 8 bits) the information to select production rules from a Backus Naur Form (BNF) grammar, see Fig. 1. Consequently, the genetic operators such as crossover and mutation are applied to the linear genotype in a typical genetic algorithm (GA) [5] manner, unlike in a tree-based GP approach where they are applied directly to the phenotypic parse trees. The grammar allows the generation of programs in an arbitrary language that are guaranteed to be syntactically correct. The user can tailor the grammar to produce solutions that are purely syntactically constrained, or they may incorporate domain knowledge by biasing the grammar. The mapping process creates a clear distinction between the search and solution space.

An important element in a successful application of evolutionary methodologies is a careful co-selection of a representation and associated diversity-generating operators which are well-suited for a specific problem landscape. These choices can radically change the performance of an algorithm. Easy problems in one representation can be hard in another. A substantial literature exists on the importance of representation choice and readers are referred to [18] for a detailed discussion of this issue. In that study three different types of representations are analysed for a GA under influence of the crossover operator. The study focuses on redundancy, scaling of building blocks and the modification of distance between individuals when mapping the genotype to the corresponding phenotype. It suggests that redundancy in a representation typically has a neutral or negative effect on algorithmic performance. In contrast, GE always maps

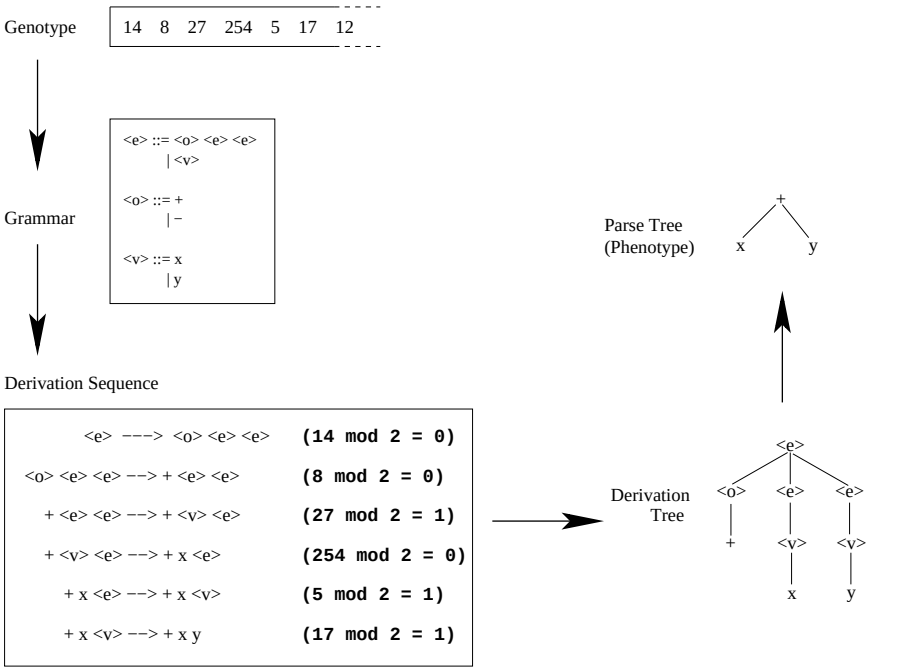

Fig. 1. An example GE genotype-phenotype mapping, where the genotype is used to select production rules from a grammar to produce a derivation sequence. The derivation sequence represents the development of a program from the embryonic non-terminal start symbol (<e>). The derivation sequence can be represented as a derivation tree, which can then be simplified to correspond to the parse tree adopted in standard tree-based GP.

via the BNF grammar to the solution space, producing a much more complex genotype-phenotype mapping than typically exists in a GA. The locality of a genotype-phenotype mapping describes how well genotypic neighbors correspond to phenotypic neighbors. Rothlauf and Oetzel [19] investigate the locality of this mapping in GE and suggest that it has low locality as neighboring genotypes do not necessarily correspond to neighboring phenotypes. Based on this finding, they suggest that further work is needed in order to develop representations with higher locality in order to maximise the efficiency of GE.

In prior investigations on GAs, Hollstien [6] claimed that gray code works slightly better than the binary representation. Gray code has some advantages compared to binary code [18] as it is not affected by scaling and it has perfect locality concerning small changes. Therefore the difficulty of a problem remains unchanged when using mutation-based search. However gray coding may not produce the same effect in GE since GE uses a complex many to one mapping (arising from the BNF grammar) causing neutral mutations whose effects also must be considered. The neutral mutations are believed to cause higher diversity

and thus higher fitness [2, 22, 4, 19]. If neutral mutations are important mutation operators with high neutrality should be advantageous. Yu and Miller, [21], state a hypothesis about the importance of the ratio between adaptive/neutral mutations. The mapping in GE gives it a non uniform representation and thus may take longer to converge. GE is also subject to a *ripple effect* when a standard GA is applied. As the function of a gene depends on the genes that proceeds it, a small genotypic change can lead to a big phenotypic change. Evidence suggests that this effect can promote a useful exchange of derivation sub-sequences during crossover [9].

3 Experimental setup & Results

This section outlines the experimental setup used in this study, details the results of these experiments, and provides a discussion of the key findings.

3.1 Hypothesis and Setup

Given the best fitness value after 50 generations for each mutation operator: μ_0, best fitness using normal integer mutation. μ_1, plus minus mutation. μ_2, binary mutation and μ_3, gray code mutation. The following hypothesis is stated:

H_0**:** None of the representations and mutations proposed gains significant performance to GE in any of the experiments, i.e. $\mu_0 = \mu_1, \mu_2, \mu_3$.
H_1**:** At least one of the mutations gains significant performance for at least one experiment, i.e. $\mu_0 > \mu_1$ or $\mu_0 > \mu_2$ or $\mu_0 > \mu_3$.
α**:** The significance level of the test is 0.05.[1]

Since the mutation operator is believed to be very problem dependent all three experiments are run with the four different mutation operators over 30 runs. Ramped Half-and-half initialization with a maximum tree depth of 8 is used. The mutation probability was 0.01, crossover probability 0.9 and the number of wraps was 30. To raise the diversity within the population, unique children regarding phenotypic difference from its parents are more likely to be produced. Two parents are picked using tournament selection with tournament size ten (population size 500). Steady state replacement is used. The solution quality is measured by cumulative frequency (rate of success) and best fitness value for the last generation (50th generation). To analyze GE's performance a t-test is performed on the best fitness after 50 generations. The data is assumed to come from normal distribution with unknown, but equal, variance. See Fig. 2 for the grammars.

[1] α is the probability of making a type 1 error, i.e. The probability of rejecting H_0 given that H_0 is in fact true.

```
EVEN-5-PARITY
<prog> ::= <expr>
<expr> ::= <expr> <op> <expr> | ( <expr> <op> <expr> ) | <var>
<op> ::= "|" | & | ^
<var> ::= d0 | d1 | d2 | d3 | d4
SYMBOLIC REGRESSION
<prog> ::= <expr>
<expr> ::= <expr> <op> <expr> | ( <expr> <op> <expr> ) | <pre-op> ( <expr> ) | <var>
<op> ::= + | * | -
<pre-op> ::= sin | cos | exp | log | inv
<var> ::= X | 1.0
SANTA FE ANT TRAIL
<prog> ::=<code>
<code> ::=<line>|<code><line>
<line> ::=<condition>|<op>
<condition> ::= if (food_ahead()==1) { <line>} else {<line> }
<op> ::=  left(): | right(): | move():
```

Fig. 2. BNF grammars used in the experiments.

3.2 Description of different mutations

Mutations that change the genotype but not the phenotype are called neutral and can arise due to functional redundancy, implicit neutrality, or mutation on inactive genes [22, 15]. Adaptive mutations are mutations that do change the phenotype. Neutral mutations cause diversity within equally fit individuals in the search space, while adaptive mutations explore the solution space.

In canonical GE, a 32 bit integer representation is used, where each codon is defined by an integer. In a *integer mutation*, the current integer value in a codon is replaced by a new randomly-generated integer. All points in the search space can be reached with the mutation operator. A *plus minus mutation* adds or subtract one from the current integer value of the codon in question. This type of mutation extinguishes the neutral mutations, and narrows the search space given some local optimum. The *binary mutation* used here flips one bit in the codon. Using the Gray code representation of the bitstring a *gray code mutation* will change the codons in a different pattern compared to the binary representation, notice that the gray code changes one bit every time adding or subtracting one. The choice of grammar affects the result of the different operators, for example a smooth grammar might be advantageous for the gray code and plus minus mutation while a grammar with an equal number of production rules for each non-terminal would make the neutral mutations vanish.

3.3 Mutation With Crossover

The amount of statistical analysis for these experiments is quite large, the tables for statistical analysis are omitted, instead significant difference is mentioned in the caption of the figures. Results for mutation with crossover can be seen in Fig. 3.

The only experiments that has a significant difference in the GE performance (best fitness) is the even-five-parity experiment, see Fig. 4. In this experiment

Symbolic regression $f(x) = x^4 + x^3 + x^2 + x$			
Mutation operator	Mean best fit.	Cum. frequency	Standard deviation
Integer mutation	1.1	9	1.4
Binary mutation	1.3	2	0.8
Gray code mutation	1.5	4	1.2
Plus Minus mutation	1.6	1	1.4
No mutation	4.1	4	3.4
Even-five-parity			
Mutation operator	Mean best fit.	Cum. frequency	Standard deviation
Integer mutation	3.2	0	1.9
Binary mutation	4.0	0	2.1
Gray code mutation	2.8	0	2.8
Plus Minus mutation	3.7	0	1.7
No mutation	6.5	0	2.2
Santa Fe ant trail			
Mutation operator	Mean best fit.	Cum. frequency	Standard deviation
Integer mutation	15.6	14	17.4
Binary mutation	13.2	15	15.7
Gray code mutation	12.5	15	15.1
Plus Minus mutation	8.9	19	13.8
No mutation	33	2	12

Fig. 3. Results for experiments using mutation and crossover for 30 runs. The mean best fitness (minimizing), cummulative frequency of success and the standard deviation.

integer mutation outperforms binary mutation while no other significant differences exist. Thus the hypothesis H_0, that none of the proposed operators produces any significant performance advantage, cannot be rejected.

Regarding the mean fitness value, see Fig. 4 and 5, we can draw the conclusion that in all the experiments except even-five-parity, the mutation operator causing the most neutral mutations also has the lowest mean fitness value. Neutral mutations cause phenotypic diversity to decrease and thus produce reduced exploration. The adaptive mutation operators in all experiments have the highest cumulative frequency. Adaptive mutations therefore seems to be advantageous for finding the optimal solution, however maybe neutral mutation with higher mutation rate would perform as well or better using both good exploration and neutral features.

For each experiment no wrapping and no mutations is also tested. Since wrapping is turned off the length of the genotypes are increased to an average of 400 codons. As can be seen in Fig. 3 the canonical GE settings of using mutation and wrapping do produce higher performance than when these mechanisms are 'turned-off'.

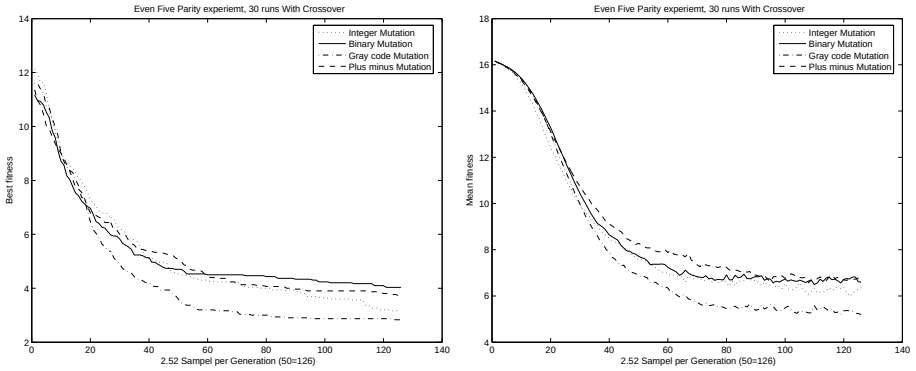

Fig. 4. *Left*: Even-five-parity GE performance, best fitness after 50 generations, for the four mutation operators. There are significant difference between the binary mutation and the integer mutation. *Right*: Even-five-parity mean of the best fitness after 50 generations for the four mutation operators. There are significant difference between the Plus minus mutation and the three others.

3.4 Mutation Without Crossover

In these experiments the crossover probability is set to zero, other settings are as in the previous experiments. The results are shown in Fig. 6.

No conclusions about GE performance and the mutation operators should be drawn since the only factor that seems to matter is the actual rate of exploration. This can be altered by increasing the mutation probability. In the Santa Fe ant trail experiment, it can clearly be seen that the binary and integer mutation are outperformed by plus minus and gray code mutation. Without crossover, neutral mutations cause worse exploration and thus lower variance, diversity, and performance. Since the algorithm has a phenotypic diversity implemented this does not properly justify the way neutral mutations have been shown to work in previous work, see [21, 22]. The best fitness for the plus minus operator is not significantly better than that produced by any other operator. There is no bias induced by the grammar since the Santa Fe ant trail experiment always chooses among three or fewer production rules. The plus minus operator can reach all the points of the search space, but unlike the integer mutation no mutations are neutral and thus the plus minus mutation has a higher rate of exploration. The results do not indicate that any firm conclusions can be drawn regarding locality and mutation. None of the proposed operators appear to produce higher locality, the only factor that seems to be of importance is the exploration rate caused by fewer neutral mutations. It is likely that these result emerge because the GE grammar mapping is so complex that no matter what representation is used, the effect caused by a mutation causes a substantial change in the phenotypic structure.

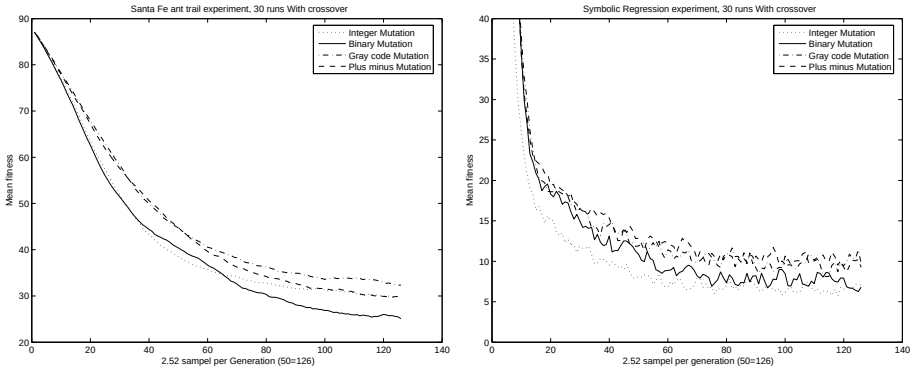

Fig. 5. *Left*: Santa Fe ant trail mean of the best fitness after 50 generations for the four mutation operators. There are significant difference between the Gray code and binary mutation. *Right*: Symbolic regression mean of the best fitness after 50 generations for the four mutation operators. There are significant difference between the binary and Gray code and plus minus mutation respectively.

4 Conclusions & Future Work

The object of this study was to examine the impact of different genotypic representations in GE in order to determine whether certain representations, and associated diversity-generation operators, improve GE's efficiency and effectiveness. Four mutation operators using two different representations, binary and Gray code representation respectively, are investigated. The different combinations of representation and mutation operator were tested on three benchmark problems. The following conclusions are made regarding the mutation operators.

- None of the proposed mutations gain any significant GE performance. Thus it is not possible to reject the hypothesis that at least one of the proposed mutation operators is significantly better than the integer mutation for at least one experiment.
- Using a mutation rate at 0.99, adaptive mutation operators seem to be preferable since they result in higher cumulative frequency. This result could arise because the mutation probability, for optimal performance, should be higher. Then, having the correct rate of adaptive mutations, neutral mutations could be of importance. However it is clear that adaptive mutations are of greater significance than neutral mutations.

Even though no clear improvements can be seen through new mutation operators there is no doubt that mutation increases GE's performance (see Fig. 3). Since GE has a complex mapping from genotype to fitness value, the results do not suggest that changing the representation of the genotype has a significant impact on GE performance. This does not imply that the representation is irrelevant, rather it suggests that the examination of the utility of a specific representation

Symbolic regression $f(x) = x^4 + x^3 + x^2 + x$			
Mutation operator	Mean best fit.	Cum. frequency	Standard deviation
Integer mutation	8.4	0	3.7
Binary mutation	7.9	0	3.8
Gray code mutation	7.4	0	3.8
Plus Minus mutation	7.2	0	3.5
Even-five-parity			
Mutation operator	Mean best fit.	Cum. frequency	Standard deviation
Integer mutation	5.0	0	2.5
Binary mutation	5.5	0	2.5
Gray code mutation	5.0	0	2.3
Plus Minus mutation	5.8	0	2.4
Santa Fe ant trail			
Mutation operator	Mean best fit.	Cum. frequency	Standard deviation
Integer mutation	33.2	2	15.7
Binary mutation	38.3	1	9.1
Gray code mutation	29.8	1	13.7
Plus Minus mutation	26.2	6	16.4

Fig. 6. Results for experiments using only mutation for 30 runs. The mean best fitness (minimizing), cummulative frequency of success and the standard deviation.

cannot be isolated from an examination of the mapping process embedded in the grammar. One way of investigating locality further would be to create a grammar that changes gradually between the properties of each production rules, this mutation could explore the search space more smoothly. Investigation of different mapping methods and how to create operators that perform a more local search, e.g. context sensitive mutations might improve the understanding of GE.

References

1. Brabazon Anthony and O'Neill Michael (2006), *Biologically Inspired Algorithms for financial Modelling*, Springer.
2. Burke E., Gustafson S. and Kendall G. (2004), *Evolutionary Optimization in uncertain Environments—A Survey*, IEEE Transactions on Evolutionary Computation 9(3) 2005, pp. 303–317.
3. Dempsey I., O'Neill M., and Brabazon A. (2005). *Meta-grammar constant creation*, Proc. of GECCO 2005, pp. 1665–1672, ACM Press.
4. Galvan-Lopez E. and Rodriguez-Vasques K. (2002), *The Importance of Neutral Mutations in GP*, PPSN IX, LNCS 4193, pp. 870–879, 2006.
5. Goldberg David E. (1989), *Genetic Algorithms,* Addison Wesley Longman.
6. Hollstein R. B. (1971), *Artificical Genetic Adaption in Computer Control Systems,* PhD thesis, University of Michigan.
7. Koza J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection,* MIT Press.

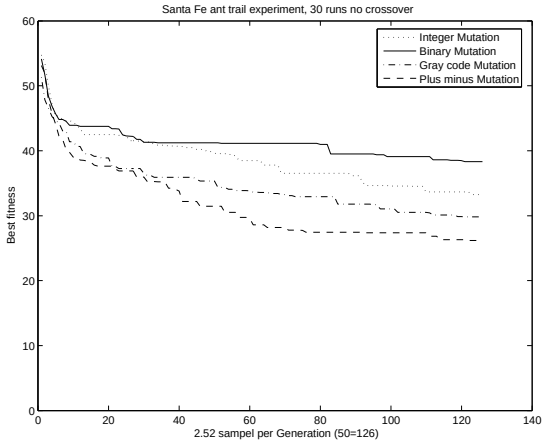

Fig. 7. Santa Fe ant trail GE performance for the four mutation operators without the crossover operator. The Binary mutation is significantly different from the plus minus and the Gray code mutation.

8. Langdon William B. and Poli Riccardo (1998), *Foundations of Genetic Programming,* Springer.
9. Nicolau M. and Dempsey I. (2006), *Introducing Grammar Based Extensions for Grammatical Evolution,* CEC 2006, pp. 648–655.
10. O'Neill M. and Brabazon A. (2004). *Grammatical swarm,* Proc. of GECCO 2004 LNCS 3120, Part 1, pp. 163–174, Springer-Verlag.
11. O'Neill M. and Brabazon A. (2005). *mGGA: The meta-Grammar genetic algorithm,* EuroGP 2005 LNCS 3447, pp. 311–320, Springer.
12. O'Neill M., Brabazon A., Nicolau M., McGarraghy S., Keenan P. (2004). *πGrammatical Evolution,* Proc. of GECCO 2004, LNCS 3103, Part 2, pp. 617–629, Springer-Verlag.
13. O'Neill M. and Brabazon A. (2006). *Grammatical Differential Evolution,* Proc. of ICAI '06, pp. 231–236, CSEA Press.
14. O'Neill M., Ryan C. (2001). *Grammatical Evolution,* IEEE Trans. Evolutionary Computation 2001.
15. O'Neill, Michael and Ryan, Conor (2003), *Grammatical Evolution,* Kluwer.
16. O'Neill M. and Ryan C. (2004). *Grammatical evolution by grammatical evolution. The evolution of grammar and genetic code,* EuroGP 2004 LNCS 3003, pp. 138–149 Springer.
17. O'Reilly, U. M. and Hemberg M. (2007), *Integrating generative growth and evolutionary computation for form exploration,* Genetic Programming and Evolvable Machines, Vol. 8, pp. 163–186, Springer.
18. Rothlauf F. (2002), *Representations for Genetic and Evolutionary Algorithms,* Physica-Verlag, Heidelberg.
19. Rothlauf F. and Oetzel M. (2005), *On the Locality of Grammatical Evolution,* EuroGP 2005 LNCS 3905, pp. 320–330, Springer.
20. Ryan C., Collins J. J., O'Neill M. (1998). *Grammatical Evolution: Evolving Programs for an Arbitrary Language,* EuroGP 1998 pp. 83–95, Springer-Verlag.

21. Yu T. and Miller J. (2002), *Finding Needles in Haystack is Not Hard with Neutrality,* EuroGP 2002, pp. 13–25.
22. Yu T. and Miller J. (2002), *Neutrality and the Evolvability of Boolean Function Landscape,* EuroGP 2001, Vol. 2038, pp. 204–217.