On the Locality of Grammatical Evolution

Franz Rothlauf and Marie Oetzel

Department of Business Administration and Information Systems
University of Mannheim, 68131 Mannheim/Germany
rothlauf@uni-mannheim.de

Abstract. This paper investigates the locality of the genotype-phenotype mapping (representation) used in grammatical evolution (GE). The results show that the representation used in GE has problems with locality as many neighboring genotypes do not correspond to neighboring phenotypes. Experiments with a simple local search strategy reveal that the GE representation leads to lower performance for mutation-based search approaches in comparison to standard GP representations. The results suggest that locality issues should be considered for further development of the representation used in GE.

1 Introduction

Grammatical Evolution (GE) [1] is a variant of Genetic Programming (GP) [2] that can evolve complete programs in an arbitrary language using a variable-length binary string. In GE, phenotypic expressions are created from binary genotypes by using a complex representation (genotype-phenotype mapping). The representation selects production rules in a Backus-Naur form grammar and thereby creates a phenotype. GE approaches have been applied to test problems and real-world applications and good performance has been reported [1, 3, 4].

The locality of a genotype-phenotype mapping describes how well genotypic neighbors correspond to phenotypic neighbors. Previous work has indicated that a high locality of representations is necessary for efficient evolutionary search [5–9]. Until now locality has mainly been used in the context of standard genetic algorithms to explain performance differences.

The purpose of this paper is to investigate the locality of the genotype-phenotype mapping used in GE. The design of high-locality genotype-phenotype encodings is important to ensure high GE performance. We present experiments for standard GE test problems that show that the mapping used in GE has low locality leading to low performance of standard mutation operators. The study at hand is an example of how basic GA design principles can be applied to explain performance differences between different GP approaches and demonstrates current challenges in the design of GE-based systems.

2 Representations, Locality and Mutation Operators

When using a representation, every optimization problem f can be decomposed into a genotype-phenotype mapping f_g (representation), and a phenotype-fitness

mapping f_p (problem) [10]. Φ_g is the genotypic search space where the search operators are applied and Φ_p is the phenotypic search space. Consequently, we distinguish between phenotypes $x^p \in \Phi_p$ and genotypes $x^g \in \Phi_g$.

2.1 Metrics

When using search algorithms, a metric has to be defined on the search space Φ. Based on the metric, the distance $d_{\boldsymbol{x}_a,\boldsymbol{x}_b}$ between two individuals $\boldsymbol{x}_a \in \Phi$ and $\boldsymbol{x}_b \in \Phi$ describes how different the two individuals are. The larger the distance, the more different two individuals are. Two individuals are neighbors if the distance between them is minimal.

If we use a representation f_g there are two different search spaces, Φ_g and Φ_p. Therefore, different metrics can be used for Φ_g and Φ_p. In general, the metric used on Φ_p is determined by the specific problem that should be solved. For GP approaches, common phenotypes are tree structures that describe programs or expressions and possible distances are tree edit distances. In contrast, the metric defined on Φ_g is not given a priori. Different GP variants use different types of genotypes. For example, GE uses linear bitstrings and standard GP [2] uses tree structures and applies search operators directly to trees.

2.2 Locality

The locality [5, 6, 10] of a representation describes how well neighboring genotypes correspond to neighboring phenotypes. The locality of a representation is high if all neighboring genotypes correspond to neighboring phenotypes. In contrast, the locality of a representation is low if some neighboring genotypes do not correspond to neighboring phenotypes.

We want to emphasize that the locality of a representation depends on the representation f_g and the metrics that are defined on Φ_g and Φ_p. f_g only determines which phenotypes are represented by which genotypes and says nothing about the similarity between solutions. To describe or measure the locality of a representation, a metric must be defined on Φ_g and Φ_p.

2.3 Locality and Mutation-based Search

The metric defined on Φ_g and the functionality of the search operators depend on each other. In most search heuristics, mutation usually creates offspring that have a small or sometimes even minimal distance to their parents. As the metric used on Φ_g defines which genotypes are similar to each other, the used genotypic metric directly determines the mutation operator.

In mutation-based search approaches, mutation steps must be small and should result in similar solutions as larger search steps would result in a randomization of the search. Then, guided search around good solutions would become impossible as the mutation-based search algorithm would jump randomly around the search space. However, low-locality representations show exactly this behavior, as small changes in a genotype do not result in small changes of a phenotype. Therefore, for low-locality representations, guided search is no longer possible as local search steps in Φ_g result into random (large) search steps in Φ_p. This leads to a low performance of EA approaches when using low-locality encodings.

3 Grammatical Evolution

Grammatical evolution is a form of linear GP that employs linear genomes, uses a grammar in Backus-Naur form (BNF) to define the phenotypic structures, and performs an ontogenetic mapping from the genotype to the phenotype.

3.1 Functionality

GE is an EA variant that can evolve computer programs defined in BNF. In contrast to standard GP [2], the genotypes are not parse trees but bitstrings of a variable length. A genotype consists of groups of eight bits (denoted as codons) that select production rules from a BNF grammar. For the construction of the phenotype from the genotype, see Sect. 3.3.

The functionality of GE follows standard EA approaches using binary genotypes. As simple binary strings are used as genotypes, no specific crossover or mutation operators are necessary. Therefore, standard crossover operators like one-point or uniform crossover and standard mutation operators like bit-flipping mutation can be used. A common metric for measuring the similarity of binary strings (compare Sect. 2.1) is the Hamming distance. Therefore, the application of bit-flipping mutation creates a new solution with genotypic distance $d^g = 1$. For selection, standard operators like tournament selection or roulette-wheel selection can be used. Some GE implementations use steady state replacement mechanisms and duplication operators that duplicate a random number of codons and insert these after the last codon position. As usual, selection decisions are performed based on the fitness of the phenotypes.

GE has been successfully applied to a number of diverse problem domains such as symbolic regression [1, 3], trigonometric identities [4], symbolic integration [3], the Santa Fe trail [1], and others. The results indicate that GE can be applied to a wide range of problems and validates the ability of GE to generate multi-line functions in any language following BNF notation.

3.2 Backus-Naur-Form

In GE, the Backus-Naur form (BNF) grammar is used to define the grammar of a language as production rules. Based on the information stored in the genotypes, BNF-production rules are selected and form the phenotype. In BNF, it can be distinguished between terminals, which are equivalent to leaf nodes in trees, and non-terminals, which can be interpreted as interior nodes in a tree and can be expanded. A grammar in BNF is defined by the quadruple $\{N, T, P, S\}$, where N is the set of non-terminals, T is the set of terminals, P is a set of production rules that maps N to a set of elements of T and N, and $S \in N$ is a start symbol.

To apply GE to a problem, it is necessary to define the BNF grammar for the problem. The BNF grammar must be defined such that the optimal solution for a specific problem can be created from the elements defined by the grammar.

3.3 Genotype-phenotype Mapping of Grammatical Evolution

In GE, a phenotype is created from binary genotypes in two steps. In a first step, integer values are calculated from codons of eight bits. Therefore, from a

binary genotype $x^{g,bin}$ of length $8l$ we get an integer genotype $x^{g,int}$ of length l, where each integer $x_i^{g,int} \in \{0, \ldots, 255\}$, for $i \in \{0, \ldots, l-1\}$. Beginning with the start symbol $S \in N$, the integer value $x_i^{g,int}$ is used to select production rules from the BNF grammar. We denote with n_P the number of production rules in P. To select a rule, we calculate the number of the used rule as $x_i^{g,int} mod\ n_P$, where *mod* denotes the modulo operation. In this manner, the mapping process traverses the genome beginning from the left hand side ($x_0^{g,int}$) until one of the following situations arises:

- The mapping is complete. All non-terminals are transformed into terminals and a complete phenotype x^p is generated.
- The end of the genome is reached ($i = l - 1$) but the mapping process is not yet finished. The individual is wrapped, the alleles are reused, and the reading of codons continues. As genotypic alleles are used several times with different meaning, wrapping can have a negative effect on locality. However, without mapping a larger number of individuals is incomplete and invalid.
- An upper threshold on the number of wrapping events is reached and the mapping is not yet complete. The mapping process is halted and the individual is assigned the lowest possible fitness value.

The mapping is deterministic, as the same genotype always results in the same phenotype. However, the interpretation of $x_i^{g,int}$ can be different if the genotype is wrapped and a different type of rule is selected. A more detailed description of the mapping process including illustrative examples can be found in [1, 3].

4 Test Problems

We investigate the locality and performance of GE for the Santa Fe Ant trail and symbolic regression problem. Both problems are standard for GP and GE.

4.1 Santa Fe Ant Trail

In the Santa Fe Ant trail problem, 89 Pieces of food are located on a discontinuous trail which is embedded in a 32 by 32 toroidal grid. The goal is to determine rules that guide the movements of an artificial ant and allows the ant to collect a maximum number of pieces of food in t_{max} search steps. In each search step, exactly one action can be performed. The ant can turn left (`left()`), turn right (`right()`), move one square forward (`move()`), or look ahead one square in the direction it is facing (`food_ahead()`). The BNF grammar for the Santa Fe ant trail problem is shown in Fig. 1(a).

4.2 Symbolic Regression

In this example [2], a mathematical expression in symbolic form must be found that approximates a given set of 20 data points (x_i, y_i). The function that should be approximated is

$$f(x) = x^4 + x^3 + x^2 + x, \tag{1}$$

where $x \in [-1; 1]$. The used BNF grammar is shown in Fig. 1(b).

```
 N= {code,line,expr,if-stat,op},
 T= {left(), right(), move(),
    food_ahead(), else, if, {, },
    (,), ;},
 S= code.
Production rules P:
<code>    ::= <line>
          |   <code><line>
<line>    ::= <expr>
<expr>    ::= <if-stat>
          |   <op>
<if-stat> ::= if(food_ahead())
              {<expr>} else
              {<expr>}
<op>      ::= left();
          |   right();
          |   move();
```

(a) Santa Fe Ant trail

```
 N= {expr, op, pre_op}
 T= {sin,cos,exp,log,+,-,/,*,x,1,(,)}
 S= <expr>
Production rules P:
<expr>   ::= <expr><op><expr>
         |   (<expr><op><expr>)
         |   <pre-op>(<expr>)
         |   <var>
<op>     ::= +
         |   -
         |   /
         |   *
<pre-op> ::= sin
         |   cos
         |   exp
         |   log
<op>     ::= x
         |   1
```

(b) symbolic regression

Fig. 1. BNF grammars for test problems

5 Locality of Grammatical Evolution

To measure the locality of a representation, we have to define a metric for Φ_g and Φ_p. For binary genotypes, usually the Hamming distance is used. It measures the number of different alleles in two genotypes x^g and y^g and is calculated as $d^g_{x^g,y^g} = \sum_i |x^g_i - y^g_i|$. A mutation (bit-flipping) of an individual x results in a neighboring solution y with distance $d^g_{x,y} = 1$.

5.1 Tree Edit Distance

It is more difficult to define appropriate metrics for phenotypes that are programs or expressions. In GE and GP, phenotypes can be described as expression trees. Therefore, edit distances can be used for measuring differences/similarities between different phenotypes. In general, the edit distance between two trees (phenotypes) is defined as the minimum cost sequence of elemental edit operations that transform one tree into the other. There are the following three elemental operations:

1. deletion: A node is removed from the tree. The children of this node become children of their parent.
2. insertion: A single node is added.
3. replacement: The label of a node is changed.

To every operation a cost is assigned (usually the same for the different operations). [11] presented an algorithm to calculate an edit distance where the operations insertion and deletion may only be applied to the leaves. [12] introduced an unrestricted edit distance and [13] developed a dynamic programming algorithm to compute tree edit distances.

In the context of GP, tree edit distances have been used as a measurement for the similarity of trees [14–16]. [17, 18] used tree edit distances for analyzing the causality of GP approaches.

5.2 Results

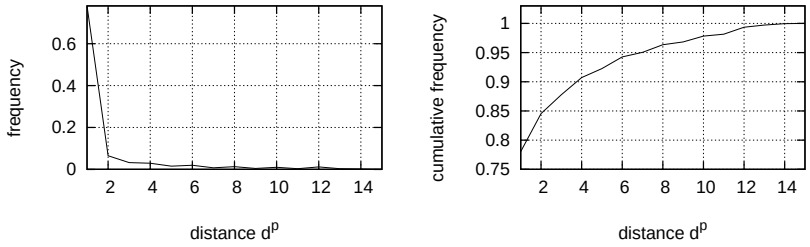
For investigating the locality of the genotype-phenotype mapping used in GE, we created 1,000 random genotypes. For the genotypes, we used standard parameter settings. The length of an individual is 160 bits, the codon size is 8, the wrapping operator is used, the upper bound for wrapping events is 10, and the maximum number of elements in the phenotype is 1,000. For each individual x, we created all 160 neighbors y, where $d^g_{x,y} = 1$. The neighbors differ in exactly one bit from the original solution. The locality of the genotype-phenotype mapping can be determined by measuring the distance $d^p_{x,y}$ between the phenotypes that correspond to the neighboring genotypes x and y. The phenotypic distance $d^p_{x,y}$ is measured as the edit distance between x^p and y^p.

For the GE genotype-phenotype mapping, we use the version 1.01 written by Michael O'Neill. The GE representation also contains the BNF Parser Gramma, version 0.63 implemented by Miguel Nicolau. For calculating the tree edit distance, we used a dynamic programming approach implemented by [13].

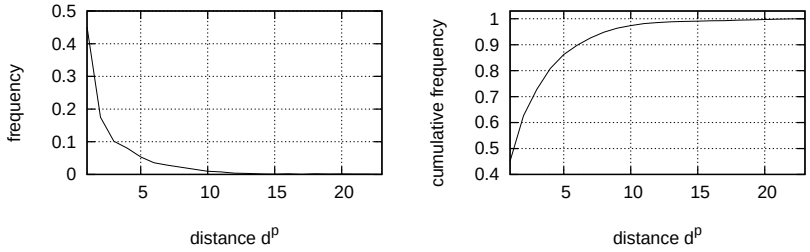
As the representation used in GE is redundant, some changes of the genotypes may not affect the corresponding phenotypes. We performed experiments for the Santa Fe Ant trail problem and the symbolic regression problem and found that either 81.98% (Santa Fe) or 94.01% of all genotypic neighbors are phenotypically identical ($d^p_{x,y} = 0$). Therefore, in about 90 % of cases a mutation of a genotype (resulting in a neighboring genotype) does not change the corresponding phenotype.

What is important for the locality of GE are the remaining neighbors that result in different phenotypes. The locality is high if the corresponding phenotypes are similar to each other. Figure 2 shows the frequency and cumulative frequency over the distance $d^p_{x,y}$ between expression trees for the two different test problems. We only consider the case where $d^p_{x,y} > 0$. The results show that for the Santa Fe Ant trail problem, many genotypic neighbors are also phenotypic neighbors (about 78%). However, there are also a significant amount of genotypic neighbors where the corresponding phenotypes are completely different. For example, more than 8% of all genotypic neighbors have a tree edit distance $d^p_{x,y} \geq 5$. The situation is worse for symbolic regression. Only about 45% of all genotypic neighbors correspond to phenotypic neighbors and about 14% of all genotypic neighbors correspond to phenotypes where $d^p_{x,y} \geq 5$.

We see that the locality of the genotype-phenotype mapping used in GE is not perfect. For the two test problems, a substantial percentage of neighboring genotypes do not correspond to neighboring phenotypes. Therefore, we expect some problems with the performance of mutation-based GE search approaches in comparison to other approaches that use a high-locality encoding.



(a) Santa Fe Ant trail



(b) symbolic regression

Fig. 2. Distribution of tree edit distances $d^p_{x,y}$ for neighboring genotypes x and y, where $d^g_{x,y} = 1$. We show the frequency (left) and cumulative frequency (right) over $d^p_{x,y}$ for the Santa Fe Ant trail problem and the symbolic regression problem.

6 Influence of Locality on GE Performance

The previous results indicate some problems of GE with low locality. Therefore, we investigate how strong the low locality of the genotype-phenotype mapping influences the performance of GE. We focus the study on mutation only. However, we assume that the results for mutation are also relevant for crossover operators (compare [8, 6, 10]).

6.1 Experimental Setting

For the experiments, we want to make sure that we only examine the impact of locality on GE performance and that no other factors blur the results. Therefore, we implemented a simple local (1+1)-EA using only mutation as a search operator. The search strategy starts with a randomly created genotype and iteratively applies bit-flipping mutations to the genotypes. If the offspring has a higher fitness than the parent it replaces it. Otherwise the parent remains the actual solution. The (1+1)-EA behaves like a simple local search.

We perform experiments for both test problems and compare an encoding with high locality with the representation used in GE. In the runs, we randomly generate a GE-encoded initial solution and use this solution as the initial solution for both types of representations. For GE, a search step is the mutation of one bit of the genotype, and the phenotype is created from the genotype using the GE genotype-phenotype mapping process. Due to the low locality of the representation, we expect problems when focusing the search on areas of the search space where solutions with high fitness can be found. However, the low locality increases the evolvability of GE what often makes it easier to escape

local optima. Furthermore, we should bear in mind that many genotypic search steps do not result in a different phenotype.

We compare the representation used in GE with a standard representation used in GP. We define the search operators in such a way that a mutation always results in a neighboring phenotype ($d^p_{x,y} = 1$). Therefore, the mutation operators are directly applied to the trees x^p. We use the following mutation operators:

- Santa Fe Ant trail
  - Deletion: A leaf node from the set of terminals T is deleted.
  - Insertion: A new leaf node from T is inserted.
  - Replacement: A leaf node (from T) is replaced by another leaf node.
- symbolic regression
  - Deletion: Either two nodes (a leaf node that contains x and a preceding node that contains sin, cos, exp, or log) or three nodes (two leaf nodes x or 1 and the common preceding node that contains +, -, *, or /) are replaced by a leaf node x.
  - Insertion: Either sin, cos, exp, or log or +, -, *, or / (plus an additional leaf node x or 1) are inserted at a leaf that contains x.
  - Replacement: +, -, *, and / are replaced by each other; sin, cos, exp, and log are replaced by each other; x and 1 are replaced by each other.

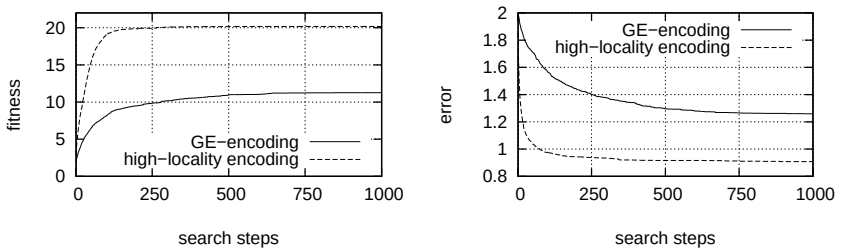
A mutation step (in the EA, the type of mutation operator is chosen randomly) always results in a neighboring phenotype and we do not need an additional genotype-phenotype mapping like in GE as we apply the search operators directly to the phenotypes.

Comparing these two different approaches, in GE, a mutation of a genotype results in most cases in the same phenotype, sometimes in a neighboring phenotype, but also sometimes in phenotypes that are completely different (compare the plots presented in Fig. 2). The standard GP representation is a high-locality representation as a mutation always results in a neighboring phenotype. Therefore, the search can be focused on promising areas of the search space but the search can never escape the local optima.

6.2 Performance Results

For the GE approach, we use the same parameter setting as described in Sect. 5.2. For both problems, we perform 1,000 runs of the (1+1)-EA using randomly created initial solutions. Each EA run is stopped after 1,000 search steps. Figure 3 compares the performance for the Santa Fe Ant trail (Fig. 3(a)) and the symbolic regression problem (Fig. 3(b)) over the number of search steps. Figure 3(a) shows the mean fitness of the found solution and Fig. 3(b) shows the mean error $1/20 \sum_{i=0}^{19} |f_j(x_i) - f(x_i)|$), where f is defined in (1) and f_j ($j \in \{0, \ldots, 1000\}$) denotes the function found by the search in search step j. The results are averaged over all 1,000 runs.

The results show that the (1+1)-EA using a high-locality representation outperforms a (1+1)-EA using the GE representation. Therefore, the low-locality of the encoding illustrated in Sect. 5 has a negative effect on the performance



(a) Santa Fe Ant trail (b) symbolic regression

Fig. 3. Performance of a mutation-based (1+1)-EA using either the GE encoding or a high-locality encoding for the Santa Fe Ant trail problem and the symbolic regression problem

of evolutionary search. Although the low locality of the GE encodings allows a local search strategy to escape local optima, EAs using the GE encoding show lower performance than a high-locality encoding.

The presented results show that using the GE encoding prolongs search as more search steps are necessary to converge. This increase is expected as for the GE encoding a search step often does not change the corresponding phenotypes. However, the plots show that allowing the (1+1)-EA using the GE encoding to run for a higher number of search steps does not increase its performance.

7 Conclusions

Previous work has shown that the locality of the genotype-phenotype mapping (representation) is important for the success of EAs. This study analyzes the locality of the representation used in grammatical evolution (GE). GE differs from other GP approaches by using binary genotypes and constructing phenotypes by choosing construction rules in Backus-Naur form grammar.

The results show that the GE representation has some problems with locality as neighboring genotypes often do not correspond to neighboring phenotypes. Therefore, a guided search around high-quality solutions can be difficult. However, due to the lower locality of the representation, it is easier to escape from local optima. Comparing a simple (1+1)-EA using either the GE representation with a standard GP encoding with high-locality reveals that the low locality of the GE representation reduces the performance of local search.

The results of this study allow a better understanding of the functionality of GE and can deliver some explanations for problems of GE that have been observed in literature. We want to encourage GE researchers to consider locality issues for further developments of the genotype-phenotype mapping. We believe that increasing the locality of the GE representation can also increase the performance and effectiveness of GE.

References

1. O'Neill, M., Ryan, C.: Grammatical evolution. IEEE Transactions on Evolutionary Computation **5** (2001) 349–358
2. Koza, J.R.: Genetic programming: On the programming of computers by natural selection. MIT Press, Cambridge, Mass. (1992)

3. O'Neill, M., Ryan, C.: Grammatical Evolution: Evolutionary Automatic Programming in a Arbitrary Language. Volume 4 of Genetic Programming. Kluwer Academic Publishers (2003)
4. Ryan, C., O'Neill, M.: Grammatical evolution: A steady state approach. In: Late Breaking Papers, Genetic Programming 1998. (1998) 180–185
5. Rothlauf, F., Goldberg, D.E.: Tree network design with genetic algorithms - an investigation in the locality of the prüfernumber encoding. In Brave, S., Wu, A.S., eds.: Late Breaking Papers at the Genetic and Evolutionary Computation Conference 1999, Orlando, Florida, USA, Omni Press (1999) 238–244
6. Gottlieb, J., Raidl, G.: Characterizing locality in decoder-based eas for the multidimensional knapsack problem. In Fonlupt, C., Hao, J.K., Lutton, E., Ronald, E., Schoenauer, M., eds.: Proceedings of Artificial Evolution. Volume 1829 of Lecture Notes in Computer Science., Springer (1999) 38–52
7. Gottlieb, J., Raidl, G.R.: The effects of locality on the dynamics of decoder-based evolutionary search. In Whitley, D., Goldberg, D.E., Cantú-Paz, E., Spector, L., Parmee, L., Beyer, H.G., eds.: Proceedings of the Genetic and Evolutionary Computation Conference 2000, San Francisco, CA, Morgan Kaufmann Publishers (2000) 283–290
8. Gottlieb, J., Julstrom, B.A., Raidl, G.R., Rothlauf, F.: Prüfer numbers: A poor representation of spanning trees for evolutionary search. IlliGAL Report No. 2001001, University of Illinois at Urbana-Champaign, Urbana (2001)
9. Rothlauf, F., Goldberg, D.E.: Prüfernumbers and genetic algorithms: A lesson on how the low locality of an encoding can harm the performance of GAs. In Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J.J., Schwefel, H.P., eds.: Parallel Problem Solving from Nature, PPSN VI, Berlin, Springer-Verlag (2000) 395–404
10. Rothlauf, F.: Representations for Genetic and Evolutionary Algorithms. 2 edn. Springer, Heidelberg (2006)
11. Selkow, S.M.: The tree-to-tree editing problem. Information Processing Letters **6** (1977) 184–186
12. Tai, K.C.: The tree-to-tree correction problem. Journal of the ACM **26** (1979) 422–433
13. Zhang, K., Shasha, D.: Simple fast algorithms for the editing distance between trees and related problems. SIAM Journal on Computing **18** (1989) 1245–1262
14. Keller, Banzhaf: Genetic programming using genotype-phenotype mapping from linear genomes into linear phenotypes. In Koza, J.e.a., ed.: Proceedings of First Annual Conference on Genetic Programming, MIT Press (1996) 116–122
15. O'Reilly, U.M.: Using a distance metric on genetic programs to understand genetic operators. In: Late breaking papers at the 1997 Genetic Programming Conference, Stanford University, CA (1997) 199–206
16. Brameier, Banzhaf: Explicit control of diversity and effective variation distance in linear genetic programming. In Tettamanzi, A.e.a., ed.: Genetic Programming, Proceedings of the 5th European Conference. Volume 2278 of LNCS., Springer (2002) 162–171
17. Igel, C.: Causality of hierarchical variable length representations. Proceedings of 1998 IEEE International Conference on Evolutionary Computation (1998) 324–329
18. Igel, C., Chellapilla, K.: Investigating the influence of depth and degree of genotypic change on fitness in genetic programming. In Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E., eds.: Proceedings of the Genetic and Evolutionary Computation Conference. Volume 2., Orlando, Florida, USA, Morgan Kaufmann (1999) 1061–1068