Fitness Distance Correlation in Structural Mutation Genetic Programming

Leonardo Vanneschi, Marco Tomassini, Philippe Collard, Manuel Clergue

Fitness Distance Correlation in Structural Mutation Genetic Programming*

Leonardo Vanneschi[1], Marco Tomassini[1],
Philippe Collard[2], and Manuel Clergue[2]

[1] Computer Science Institute, University of Lausanne, Lausanne, Switzerland
{Leonardo.Vanneschi, Marco.Tomassini}@iis.unil.ch
[2] I3S Laboratory, University of Nice, Sophia Antipolis, France
{pc, clerguem}@i3s.unice.fr

Abstract. A new kind of mutation for genetic programming based on the structural distance operators for trees is presented in this paper. We firstly describe a new genetic programming process based on these operators (we call it structural mutation genetic programming). Then we use structural distance to calculate the fitness distance correlation coefficient and we show that this coefficient is a reasonable measure to express problem difficulty for structural mutation genetic programming for the considered set of problems, i.e. unimodal trap functions, royal trees and MAX problem.

1 Introduction

Studies of fitness distance correlation (*fdc*) as a tool for measuring problem difficulty in genetic algorithms (GAs) and genetic programming (GP) have lead to controversial results: even though some counterexamples has been found for GAs ([1], [17]), *fdc* has been proven an useful measure on a large number of GA (see for example [7] or [11]) and GP functions (see [5], [18]). In particular, Clergue *et al.* ([5]) have shown *fdc* to be a reasonable way of quantifying problem difficulty for GP for a set of functions. To calculate *fdc* they defined a genotypic distance for trees based on recursion. No particular relationship between this distance and the genetic operator they used (standard GP crossover) was evident (as is the case in GAs with Hamming distance and standard crossover). The distance metric should instead be defined with regard to the actual neighborhood produced by the genetic operators, so to assure the conservation of the genetic material between neighbors. In this paper we want to overcome this limitation, establishing a strong relationship between distance and genetic operators. Thus, we use structural distance (see [9, 12]) to calculate *fdc* and we use the tranformations on which structural distance is based to define two new genetic operators.

* Published in: "Genetic Programming - 6th European Conference, EuroGP 2003" Essex, UK, April 2003, Proceedings, pages 455-464, C. Ryan et al. Eds., Lecture Notes in Computer Science, LNCS 2610, Springer Verlag.

The new resulting evolutionary process will be called *structural mutation genetic programming* (SMGP), to distinguish it from GP based on the standard Koza's crossover (that will be referred to as standard GP).

This paper deeply differs from [5] for the following main reasons: a different distance measure is used, for the first time here, to calculate *fdc*; an original evolutionary process with new genetic operators is used; the coherence between these operators and the distance used is not only hint at, but it is formally proven. Finally, a larger set of test functions is used to validate our hypothesis, including two MAX problems, in which fitness is also a function of the semantics of the programs and not only of their syntactical structure.

This paper is structured as follows: in section 2 we describe one version of the structural distance. In section 3 we define two new mutations based on the structural operators and we describe SMGP. In section 4, *fdc* is tested as a measure of problem difficulty for SMGP on trap functions, royal trees and two MAX problems. Finally, in section 5 we offer our conclusions.

2 Distance Measure for Genetic Programs

In genetic algorithms (GAs) individuals are represented as strings of digits and typical distance measures are Hamming distance or alternation (see for instance [6]). Defining a distance between genotypes in GP is much more difficult, given the tree structure of the individuals. In [9] a version of the structural distance for trees has been proposed. According to this measure, given the sets $\mathcal{F}$ and $\mathcal{T}$ of functions and terminals, a coding function c must be defined such that $c : \{\mathcal{T} \cup \mathcal{F}\} \rightarrow \mathbb{N}$. One can think of many guidelines for the specification of c, for example the "complexity" of the functions or their arity. The distance between two trees T_1 and T_2 is calculated in three steps: (1) T_1 and T_2 are overlapped at the root node and the process is applied recursively starting from the leftmost subtrees (see [3] for a description of the overlapping algorithm and figure 1 for a visual intuition of it). (2) For each pair of nodes at matching positions, the difference of their codes (eventually elevated to an exponent) is computed. (3) The differences computed in the previous step are combined in a weighted sum.

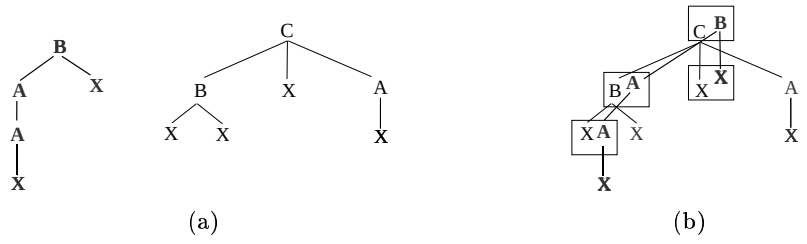

Fig. 1. (a): Two trees T_1 and T_2 (b): The trees T_1 and T_2 overlapped at the root node. Overlapped nodes are included inside a rectangle.

Formally, the distance of two trees T_1 and T_2 with roots R_1 and R_2 is defined as follows:

$$dist(T_1, T_2) = d(R_1, R_2) + k\sum_{i=1}^{m} dist(child_i(R_1), child_i(R_2)) \quad (1)$$

where: $d(R_1, R_2) = (|c(R_1) - c(R_2)|)^z$, $child_i(Y)$ is the i^{th} of the m possible children of a generical node Y, if $i \leq m$, or the empty tree otherwise, and c evaluated on the root of an empty tree is 0. Constant k is used to give different weights to nodes belonging to different levels and z is a constant usually chosen in such a way that $z \in \mathbb{N}$.

In most of this paper, except the MAX function, individuals will be coded using the same syntax as in [5] and [16], i.e. considering a set of functions A, B, C, etc. with increasing arity (i.e. $arity(A) = 1$, $arity(B) = 2$, and so on) and a single terminal X (i.e. $arity(X) = 0$) as follows: $\mathcal{F} = \{A, B, C, D, \ldots\}, \mathcal{T} = \{X\}$ and the c function will be defined as follows: $\forall x \in \{\mathcal{F} \cup \mathcal{T}\} \quad c(x) = arity(x) + 1$. In our experiments we will always set $k = \frac{1}{2}$ and $z = 2$. By keeping $0 < k < 1$, the differences near the root have higher weight. This is convenient for GP as it has been noted that programs converge quickly to a fixed root portion [14].

3 Structural Mutation Operators

Given the sets $\mathcal{F}$ and $\mathcal{T}$ and the coding function c defined in section 2, we define c_{max} (respectively, c_{min}) as the maximum (respectively, the minimum) value assumed by c on the domain $\{\mathcal{F} \cup \mathcal{T}\}$. Moreover, given a symbol n such that $n \in \{\mathcal{F} \cup \mathcal{T}\}$ and $c(n) < c_{max}$ and a symbol m such that $m \in \{\mathcal{F} \cup \mathcal{T}\}$ and $c(m) > c_{min}$, we define: $succ(n)$ as a node such that $c(succ(n)) = c(n) + 1$ and $pred(m)$ as a node such that $c(pred(m)) = c(m) - 1$. Then we can define the following operators on a generic tree T:

- **grow mutation**. A node labelled with a symbol n such that $c(n) < c_{max}$ is selected in T and replaced by $succ(n)$. A new random terminal node is added to this new node in a random position (i.e. the new terminal becomes the i^{th} son of $succ(n)$, where i is comprised between 0 and $arity(n)$).
- **shrink mutation**. A node labelled with a symbol m such that $c(m) > c_{min}$, and such that at least one of his sons is a leaf, is selected in T and replaced by $pred(m)$. A random leaf, between the sons of this node, is deleted from T.

The *grow* and *shrink* mutations defined above should not be confused with the well known mutation operators with the same name that have already been proposed in GP. Figure 2 gives an example of the application of the above new operators. Given these definitions, we can prove the following property:

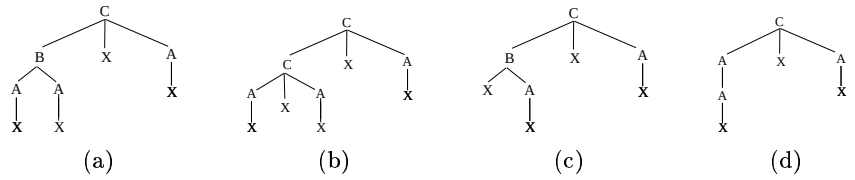

Fig. 2. The tree in (b) can be obtained from the tree in (a) in one step with the grow mutation. The tree in (d) can be obtained from the tree in (c) in one step with the shrink mutation.

Property 1. Distance/Operator Consistency.
Let's consider the sets $\mathcal{F}$ and $\mathcal{T}$ and the coding function c defined in section 2. Let T_1 and T_2 be two trees composed by symbols belonging to $\{\mathcal{F} \cup \mathcal{T}\}$ and let's consider the k and z constants of definition (1) to be both equal to 1. If $dist(T_1, T_2) = D$, then T_2 can be obtained from T_1 by a sequence of $\frac{D}{2}$ structural operations, where a structural operation can be a grow mutation or a shrink mutation.

The proof of this property can be found in appendix A (note that, given the sets $\mathcal{F}$ and $\mathcal{T}$ and the coding function c defined in section 2, and having set $k = z = 1$ in definition (1), $dist(T_1, T_2)$ is an even natural number for every couple of trees T_1 and T_2. The proof of this property can be done by recursion on the depths of T_1 and T_2 and is omitted for reasons of space). From property 1 we deduce that, at least for the language used to code trees in most of this paper (except the MAX function), the operators of grow mutation and shrink mutation are completely coherent with the notion of structural distance defined in section 2: an application of these operators allow us to move on the research space from a tree to its neighbors according to the structural distance. Thus we are interested in defining a new GP process based on these operators. We call this process structural mutation genetic programming (SMGP).

4 Experimental Results

4.1 Fitness Distance Correlation

An approach proposed for GAs [11] states that an indication of problem hardness is given by the relationship between fitness and distance of the genotypes from known optima. Given a sample $F = \{f_1, f_2, ..., f_n\}$ of n individual fitnesses and a corresponding sample $D = \{d_1, d_2, ..., d_n\}$ of the n distances to the nearest global optimum, *fdc* is defined as:

$$fdc = \frac{C_{FD}}{\sigma_F \sigma_D}$$

where:

$$C_{FD} = \frac{1}{n} \sum_{i=1}^{n} (f_i - \overline{f})(d_i - \overline{d})$$

is the covariance of F and D and σ_F, σ_D, $\overline{f}$ and $\overline{d}$ are the standard deviations and means of F and D. As shown in [11], GA problems can be classified in three classes, depending on the value of the *fdc* coefficient: **misleading** ($fdc \geq 0.15$), in which fitness increases with distance, **difficult** ($-0.15 < fdc < 0.15$) in which there is virtually no correlation between fitness and distance and **straightforward** ($fdc \leq -0.15$) in which fitness increases as the global optimum approaches. The second class corresponds to problems for which the difficulty can't be estimated, because *fdc* doesn't bring any information. In this case, examination of the fitness-distance scatterplot may give information on problem difficulty (see [11]).

4.2 Trap Functions

Trap functions [8] allow to define the fitness of the individuals as a function of their distance from the optimum. A function $f : distance \rightarrow fitness$ is an unimodal trap function if it is defined in the following way:

$$f(d) = \begin{cases} 1 - \dfrac{d}{B} & \text{if } d \leq B \\ \dfrac{R \cdot (d - B)}{1 - B} & \text{elsewhere} \end{cases}$$

These functions have a number of different optima and d is the distance of the current individual from the unique global one, while B and R are constants $\in [0, 1]$. B allows to set the width of the attractive basin for each of the two optima and R sets their relative importance. By construction, the difficulty of trap functions decreases as the value of B increases, while it increases as the value of R increases.

Figures 3 and 4 show values of the performance p (defined as the number of executions for which the global optimum has been found in less than 500 generations divided by the total number of executions, i.e. 100 in our experiments) and of *fdc* for various trap functions obtained by changing the values of the constants B and R.
Two trees of different shapes are considered as optimum in the different experiments (see (c) parts of the figures). In all cases *fdc* is confirmed to be a reasonable measure to quantify problem difficulty. The same experiments have been repeated using two other differently shaped trees as global optimum and the results (not shown here to save space) are qualitatively analogous to the ones shown in figures 3 and 4. In all experiments *fdc* has been calculated via a sample of 40000 randomly chosen individuals. All the experiments have been done with generational SMGP, a total population size of 100 individuals, tournament selection of size 10 and E as the node with maximum arity allowed.

4.3 Royal Trees

The next functions we take into account are the royal trees proposed by Punch *et al.* [16]. The language used is the same as in section 2, and the fitness of a tree (or

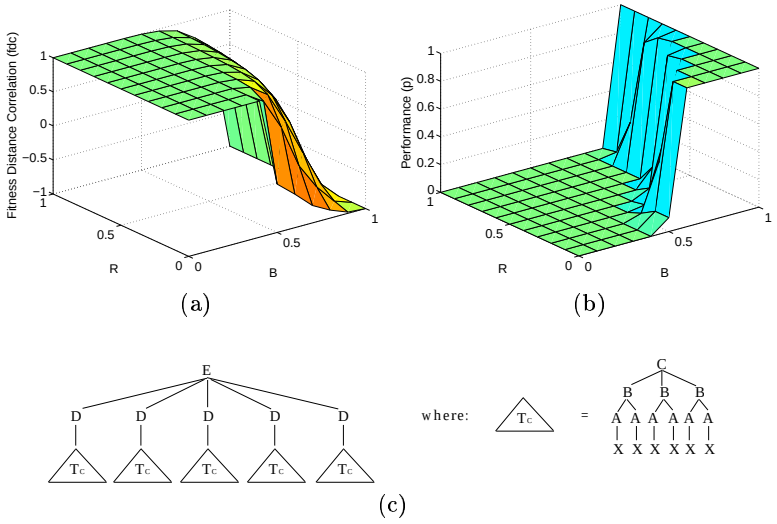

Fig. 3. (a): *fdc* values with structural distance for some trap functions obtained by changing the values of the constants B and R. (b): Performance values of SMGP with structural distance for traps. (c): Stucture of the tree used as optimum in the experiments reported in (a) and (b).

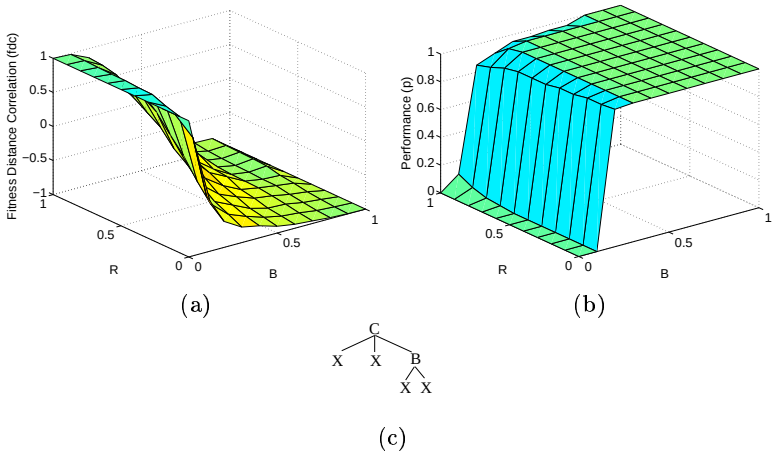

Fig. 4. (a): *fdc* values with structural distance for some trap functions obtained by changing the values of the constants B and R. (b): Performance values of SMGP with structural distance for traps. (c): Stucture of the tree used as optimum in the experiments reported in (a) and (b).

any subtree) is defined as the score of its root. Each function calculates its score by summing the weighted scores of its direct children. If the child is a perfect

tree of the appropriate level (for instance, a complete level-C tree beneath a D node), then the score of that subtree, times a *FullBonus* weight, is added to the score of the root. If the child has a correct root but is not a perfect tree, then the weight is *PartialBonus*. If the child's root is incorrect, then the weight is *Penalty*. After scoring the root, if the function is itself the root of a perfect tree, the final sum is multiplied by *CompleteBonus* (see [16] for a more detailed explanation). Values used here are as in [16] i.e. *FullBonus* = 2, *PartialBonus* = 1, *Penalty* = $\frac{1}{3}$, *CompleteBonus* = 2. Results on the study of *fdc* are shown in table 1.

Root	*fdc*	*fdc* prediction	p (SMGP)
B	-0.31	straightf.	1
C	-0.25	straightf.	1
D	-0.20	straightf.	0.76
E	0.059	difficult	0
F	0.44	misleading	0
G	0.73	misleading	0

Table 1. Results of *fdc* for the Royal Trees using SMGP.

Fdc correctly classifies the difficulty of level-B, level-C and level-D functions. Level-E function is "difficult" to be predicted by the *fdc* (i.e. no correlation between fitness and distance is observed). Finally, level-F and level-G functions are predicted to be "misleading" (in accord with [16]) and they really are, since the global optimum is never found before generation 500. In conclusion, it appears that royal trees problem spans all the classes of difficulty as described by the *fdc*.

4.4 MAX Problem

The task of the MAX problem for GP, defined in [10] and [13], is "to find the program which returns the largest value for a given terminal and function set with a depth limit d, where the root node counts as depth 0". We set d equal to 8 and we use the set of functions $\mathcal{F} = \{+\}$ and the set of terminals $\mathcal{T}_1 = \{1\}$ or $\mathcal{T}_2 = \{1, 2\}$. When using $\mathcal{T}_1$, we specify the coding function c as: $c(1) = 1$, $c(+) = 2$, when using $\mathcal{T}_2$, we pose: $c(1) = 1$, $c(2) = 2, c(+) = 3$. The grow and shrink mutations, this time, are defined in such a way that, when using $\mathcal{T}_1$, a terminal symbol 1 can be transformed in the subtree $T_1 = +(1, 1)$ by one step of grow mutation and the vice-versa can be done by the shrink mutation. When using $\mathcal{T}_2$, the grow mutation can transform a 1 node into a 2 node and a 2 node into the subtrees $T_2 = +(2, 1)$ or $T_3 = +(1, 2)$ (with a uniform probability). On the other hand, the shrink mutation can tranform T_1 or T_2 into a leaf labelled by 2, and a 2 node into a 1 node. Table 2 shows the *fdc* and p values for these test cases.

MAX problem	*fdc*	*fdc* prediction	p (SMGP)
{+} {1}	-0.87	straightf.	1
{+} {1,2}	-0.86	straightf.	1

Table 2. Results of *fdc* for the MAX problem using SMGP with structural distance. The first column shows the sets of functions and terminals used in the experiments.

5 Conclusions and Future Work

Two new tree mutations corresponding to the structural distance operators are defined in this paper. Fitness distance correlation (*fdc*) calculated using a version of the structural distance is shown to be a reasonable measure to quantify the difficulty of trap functions, royal trees and two MAX functions for GP using these mutations as genetic operators (SMGP). Experiments using GP based on standard crossover (and no mutation) have also been performed on all the test cases considered here. Results (not shown for reasons of space) confirm *fdc* as a reasonable measure of problem difficulty also for GP with standard crossover. This seems to point in the same direction as in [2], [4] and [15], where the standard GP crossover is defined as a "macro-mutation". In view of some counterexamples that have been mentioned in the text, it remains to be checked whether the use of *fdc* extends to other classes of functions, such as typical GP benchmarks. In the future we also plan to investigate the use of *fdc* in cases of fitness landscapes containing multiple optima and to look for a measure for GP difficulty that can be calculated without prior knowledge of the global optima, thus eliminating the strongest limitations of *fdc*. Moreover, we plan to build a counterexample for *fdc* in GP. Another open problem consists in taking into account in the distance definition the phenomenon of introns (whereby two different genotypes can lead to the same phenotypic behavior). Finally, we intend to look for a better measure than performance to identify the success rate of functions, possibly independent from the maximum number of generations chosen.

A Proof of the Distance/Operator Consistency Property

Let's prove property 1 (see section 3) by recursion on the depths of T_1 and T_2. Let p_1 and p_2 be the respective depths of T_1 and T_2. If $p_1, p_2 \leq 0$ (i.e. $p_1, p_2 = 0$), then T_1 and T_2 are both composed of a single node X. In this case, their distance is equal to zero and the number of structural mutations to transform T_1 into T_2 is obviously equal to zero. So the property holds. Let's now suppose the property true $\forall p_1, p_2 \leq p$ (with $p \geq 0$) and, starting from this hypothesis, let's prove it $\forall p_1, p_2 \leq p+1$. Let's consider two trees T_1 and T_2 of depths p_1 and p_2 with $p_1, p_2 \leq p+1$. Let R_1 be the root of T_1 and R_2 be the root of T_2. Let the difference between the codes of R_1 and R_2 be equal to y (i.e. $d(R_1, R_2) = y$) and, without lost of generality, let's consider $c(R_1) \leq c(R_2)$. Now, to transform R_1 into R_2, we have to perform y grow mutations on R_1, which lead to y increments of R_1's label cost and to the creation of y new X nodes. Let's call T_1' the tree generated from T_1 after these y operations. Then, we have: $dist(T_1, T_2) = 2y + dist(T_1', T_2)$, since T_1's root has been incremented y times and y

new X nodes have been created. Moreover, since T_1' and T_2 have the same root, we can write:

$$dist(T_1, T_2) = 2y + \sum_{i=1}^{m} dist(child_i(R_1), child_i(R_2)) \quad (2)$$

where m is T_2's arity. On the other hand, it is clearly possible to transform T_1 into T_2 with the same number of operations that can be used to transform T_1' into T_2, plus y operations (used to transform T_1 into T_1'). Let's express this property with the notation: $op(T_1, T_2) = y + op(T_1', T_2)$ and, since T_1' and T_2 have the same root:

$$op(T_1, T_2) = y + \sum_{i=1}^{m} op(child_i(R_1), child_i(R_2)) \quad (3)$$

Now, $child_i(R_1)$ and $child_i(R_2)$ have respective depths p_1^i and p_2^i where $p_1^i, p_2^i \leq p$ and so, by recursive hypothesis, if we call D_i the distance between $child_i(R_1)$ and $child_i(R_2)$, it is possible to tranform $child_i(R_1)$ into $child_i(R_2)$ with $\frac{D_i}{2}$ structural operations. We can express it with the following notation:

$$dist(child_i(R_1), child_i(R_2)) = D_i \quad (4)$$

and

$$op(child_i(R_1), child_i(R_2)) = \frac{D_i}{2} \quad (5)$$

As a consequence, by replacing (4) in (2), we obtain: $dist(T_1, T_2) = 2y + \sum_{i=1}^{m} D_i$ and by replacing (5) in (3) we obtain: $op(T_1, T_2) = y + \sum_{i=1}^{m} \frac{D_i}{2} = \frac{1}{2} dist(T_1, T_2)$, i.e. it is possible to transform T_1 into T_2 with a number of structural operations equal to the distance between T_1 and T_2 divided by 2. This proves the property. □

Acknowledgment. We would like to thank Leslie Luthi for his help with the proof of property 1.

References

1. L. Altenberg. Fitness distance correlation analysis: an instructive counterexample. In T. Back, editor, *Seventh International Conference on Genetic Algorithms*, pages 57–64. Morgan Kaufmann, 1997.
2. P. J. Angeline. Subtree crossover: Building block engine or macromutation? In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 9–17, Stanford University, CA, USA, 1997. Morgan Kaufmann.
3. E. Burke, S. Gustafson, G. Kendall, and N. Krasnogor. Advanced population diversity measures in genetic programming. In J. J. Merelo, P. Adamidis, H. G. Beyer, J.-L. Fernández-Villacanas, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VII*, volume 2439 of *Lecture Notes in Computer Science*, pages 341–350. Springer-Verlag, Heidelberg, 2002.
4. K. Chellapilla. Evolutionary programming with tree mutations: Evolving computer programs without crossover. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 431–438, Stanford University, CA, USA, 1997. Morgan Kaufmann.

5. M. Clergue, P. Collard, M. Tomassini, and L. Vanneschi. Fitness distance correlation and problem difficulty for genetic programming. In *Proceedings of the genetic and evolutionary computation conference GECCO'02*, pages 724–732, San Francisco, CA, 2002. Morgan Kaufmann.
6. P. Collard, M. Clergue, and F. Bonnin. Misleading functions designed from alternation. In *Congress on Evolutionary Computation (CEC'2000)*, pages 1056–1063. IEEE Press, Piscataway, NJ, 2000.
7. P. Collard, A. Gaspar, M. Clergue, and C. Escazut. Fitness distance correlation as statistical measure of genetic algorithms difficulty, revisited. In *European Conference on Artificial Intelligence (ECAI'98)*, pages 650–654, Brighton, 1998. John Witley & Sons, Ltd.
8. K. Deb and D. E. Goldberg. Analyzing deception in trap functions. In D. Whitley, editor, *Foundations of Genetic Algorithms, 2*, pages 93–108. Morgan Kaufmann, 1993.
9. A. Ekárt and S. Z. Németh. Maintaining the diversity of genetic programs. In J. A. Foster, E. Lutton, J. Miller, C. Ryan, and A. G. B. Tettamanzi, editors, *Genetic Programming, Proceedings of the 5th European Conference, EuroGP 2002*, volume 2278 of *LNCS*, pages 162–171, Kinsale, Ireland, 3-5 April 2002. Springer-Verlag.
10. C. Gathercole and P. Ross. An adverse interaction between crossover and restricted tree depth in genetic programming. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 291–296, Stanford University, CA, USA, 28–31 July 1996. MIT Press.
11. T. Jones. *Evolutionary Algorithms, Fitness Landscapes and Search.* PhD thesis, University of New Mexico, Albuquerque, 1995.
12. R. Keller and W. Banzhaf. Explicit maintenance of genotypic diversity on genospaces. Unpublished, 1994. http://ls11-www.informatik.uni-dortmund.de/people/banzhaf/gp.html.
13. W. B. Langdon and R. Poli. An analysis of the max problem in genetic programming. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference on Genetic Programming*, pages 222–230, San Francisco, CA, 1997. Morgan Kaufmann.
14. N.F. McPhee and N.J. Hopper. Analysis of genetic diversity through population history. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1112–1120, Orlando, Florida, USA, 13-17 July 1999. Morgan Kaufmann.
15. U.-M. O'Reilly. *An Analysis of Genetic Programming.* PhD thesis, Carleton University, Ottawa, Ontario, Canada, 1995.
16. B. Punch, D. Zongker, and E. Goodman. The royal tree problem, a benchmark for single and multiple population genetic programming. In P. Angeline and K. Kinnear, editors, *Advances in Genetic Programming 2*, pages 299–316, Cambridge, MA, 1996. The MIT Press.
17. R.J. Quick, V.J. Rayward-Smith, and G.D. Smith. Fitness distance correlation and ridge functions. In *Fifth Conference on Parallel Problems Solving from Nature (PPSN'98)*, pages 77–86. Springer-Verlag, Heidelberg, 1998.
18. V. Slavov and N. I. Nikolaev. Fitness landscapes and inductive genetic programming. In *Proceedings of International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA97)*, University of East Anglia, Norwich, UK, 1997. Springer-Verlag KG, Vienna.