Maximum Homologous Crossover for Linear Genetic Programming

Michael Defoin Platel, Manuel Clergue, Philippe Collard

Maximum Homologous Crossover for Linear Genetic Programming

Michael Defoin Platel[1,2], Manuel Clergue[1] and Philippe Collard[1]

[1] Laboratoire I3S, CNRS-Université de Nice Sophia Antipolis
[2] ACRI-ST

Abstract. We introduce a new recombination operator, the Maximum Homologous Crossover for Linear Genetic Programming. In contrast to standard crossover, it attempts to preserve similar structures from parents, by aligning them according to their homology, thanks to an algorithm used in Bio-Informatics. To highlight disruptive effects of crossover operators, we introduce the Royal Road landscapes and the Homology Driven Fitness problem, for Linear Genetic Programming. Two variants of the new crossover operator are described and tested on this landscapes. Results show a reduction in the bloat phenomenon and in the frequency of deleterious crossovers.

1 Introduction

The role played by crossover in the Genetic Programming (GP) evolutionary process is a much debated question. Traditionally, individuals are encoded using a tree-based representation, and crossover consists in swapping subtrees. According to Koza [9], crossover is the central operator in the GP search process, where useful subtrees tend to spread as they are swapped. However, Banzhaf et al. [17] argue that crossover behaves more like a macro mutation operator, so GP can be viewed as a population based hill-climber. In the same way, some authors [2] have obtained worse results for crossover compared to mutation based system. Moreover, standard crossover exchanges subtrees without taking context into account ; this is a brutal operation that may prevent emergence of structured solutions [6]. Altenberg [1] notes that crossover may cause the program growth phenomenon, called bloat, which arises during evolution as the population attempts to protect useful subtrees. Finally, Poli and Langdon [15] point out the fact that standard GP crossover is a local and biased operator, which can not explore search space properly.

Some new operators have been designed to overcome the drawbacks of the standard GP crossover. The main idea behind all those recombination mechanisms is *homology*. This notion comes directly from the properties of the crossover in nature which does not exchange genes randomly. Indeed, during the second stage of the prophase of meiosis (called *zygoten*), the homologous chromosomes are first aligned according to their similarity before crossover takes place. This implies that genes are swapped with others that represent similar features. We

note several previous attempts to improve the effectiveness of crossover which, either implicitly or explicitly, try to better preserve homology, see [6][10][12][15].

The way individuals are represented in Evolutionary Computation is always crucial, this is also the case in GP. The emergence of GP in the scientific community arose with the use, *inter alia*, of a tree-based representation, in particular with the use of Lisp in the work of Koza [9]. However, GP systems manipulating linear structures exist, like in [3][14], which have shown experimental performances equivalent to Tree GP (TGP). In contrast to TGP, Linear GP (LGP) programs are sequence of instructions of an imperative language (C, machine code, ...). In this paper, we focus on LGP mainly because it allows direct access to instructions and so it provides easier way to perform recombination. Moreover, in LGP all possible sequences of instructions are valid programs, so there are no syntactical constraints on sequences swapped during recombination and classical genetic crossover operators in use could be chosen.

In Section 2, we introduce a new biologically inspired crossover for LGP, called Maximum Homologous Crossover (MHC). In order to study the way MHC works in Section 3 we introduce Royal Road Landscapes for LGP and report experimental results in Section 4.

2 Maximum Homologous Crossover

In this section, we present a new recombination mechanism mimicking natural crossover by preserving homology. In biology, homology indicates genetic relationship, i.e. the structural relatedness of genomes due to descent from common form. Indeed, reproduction in nature is a smooth process which ensures that offspring will not be so different from ancestors, allowing the structural stability that defines species.

2.1 Edit distance

The Maximum Homologous Crossover (MHC) preserves structural and lexical homology by computing an alignment that minimises a metric of dissimilarity between parents. As a metric, we use string edit distance, like *Levenshtein distance* [11] which has been already used in GP to compute or control diversity [4], or to study the influence of genetic operators [13]. By definition, the edit distance between two programs corresponds to the minimal number of elementary operations (deletion, insertion and substitution) required to change one program into the other.

More formally, let us consider $P_x \in P_\Sigma$ a program of size m such that $P_x = x_0 x_1 \dots x_{m-1}$, with $x_i \in \Sigma \ \forall i \in [0, m-1]$, where Σ is a finite set of available instructions [3]. Let P_x and P_y be two programs of size m and n respectively

[3] In LGP, the traditional distinction between the set of terminals and the set of functions is not relevant.

and ε be an empty instruction. An alignment $(\overline{P}_x, \overline{P}_y)$ of size p with $\overline{P}_x$ and $\overline{P}_y \in P_\Sigma \cup \{\varepsilon\}$ is :

$$(\overline{P}_x, \overline{P}_y) = \begin{pmatrix} \overline{x}_0 & \overline{x}_1 \dots & \overline{x}_{p-1} \\ \overline{y}_0 & \overline{y}_1 \dots & \overline{y}_{p-1} \end{pmatrix}$$

where :

- $p \in [max(n, m), n + m]$
- $\overline{x}_i = x_j$ or $\overline{x}_i = \varepsilon$ for $i \in [0, p-1]$ and $j \in [0, m-1]$
- $\overline{y}_i = y_j$ or $\overline{y}_i = \varepsilon$ for $i \in [0, p-1]$ and $j \in [0, n-1]$
- $\nexists\ i \in [0, p-1]$ such that $\overline{x}_i = \overline{y}_i = \varepsilon$

An aligned pair of instructions $\binom{\overline{x}_i}{\overline{y}_i}$ indicates either a substitution of x_j by y_j, or a deletion of x_j (if $\overline{y}_i = \varepsilon$), or an insertion of y_j (if $\overline{x}_i = \varepsilon$). So, an alignment $(\overline{P}_x, \overline{P}_y)$ may also be viewed as a sequence of operations (insertion, deletion and substitution) that transforms P_x into P_y.

We define the cost χ of an alignment such as $\chi(\overline{P}_x, \overline{P}_y) = \sum_{i=0}^{p-1} cost(\overline{x}_i, \overline{y}_i)$ with :

$$cost(\overline{x}_i, \overline{y}_i) = \begin{cases} C_1 \text{ (insertion or deletion cost)} & \text{if } \overline{x}_i = \varepsilon \text{ or } \overline{y}_i = \varepsilon \\ C_2 \text{ (substitution cost)} & \text{else if } \overline{x}_i \neq \overline{y}_i \\ 0 & \text{else} \end{cases}$$

and $A(P_x, P_y)$ the set of all alignments of P_x and P_y, then the distance between P_x and P_y is :

$$\mathcal{D}(P_x, P_y) = min\big\{\chi(\overline{P}_x, \overline{P}_y) | (\overline{P}_x, \overline{P}_y) \in A(P_x, P_y)\big\}$$

As an example, in Figure 1, the distance between P_x and P_y is 7, i.e. 7 operations are required to transform P_x into P_y (5 insertions, 1 deletion and 1 substitution). Each column of the alignment $(\overline{P}_x, \overline{P}_y)$ refers to a program and stores a sequence of instructions with gaps inserted (corresponding to ε). Note that, during the alignment process, numerical constants of P_x and P_y are viewed as a same type of instruction.

2.2 Best Alignment and Recombination

A best alignment $(\overline{P}_x, \overline{P}_y)$ between P_x and P_y is that for which $\chi(\overline{P}_x, \overline{P}_y) = \mathcal{D}(P_x, P_y)$ holds. We denote $A^*(P_x, P_y)$ the set of all best alignments between P_x and P_y. Computation of $A^*(P_x, P_y)$ can be reasonably performed, using dynamic programming in $O(nm)$ time complexity. Such an algorithm [8] has also been used to align DNA strings in Bio-Informatics.

In order to perform MHC between P_x and P_y, only one alignment $(\overline{P}_x, \overline{P}_y)$ is randomly chosen in A^*. Recombination between $\overline{P}_x$ and $\overline{P}_y$ can then take place. Since $\overline{P}_x$ and $\overline{P}_y$ have the same length, classical crossovers existing in Genetic Algorithms (GA) can be performed (1-point, 2-point, uniform, ...). Finally, to get valid children, the ε symbols are removed.

By choosing the costs of operations, C_1 and C_2, we can define two sets A_1^*, A_2^* of best alignments :

– A_1^*, with C_2=C_1=1, where a substitution is always preferred to a pair of insertion and deletion. This setting is used to compute *Levenshtein distance*.
– A_2^*, with C_2=2 and C_1=1, where a substitution is chosen as often as a pair of insertion and deletion.

We denote MHC_1 and MHC_2, the corresponding variants of MHC.

Figure 1 gives a 1-point MHC_1 with the recombination site at position 5 between P_x and P_y in stack-based representation, producing offspring P'_x and P'_y.

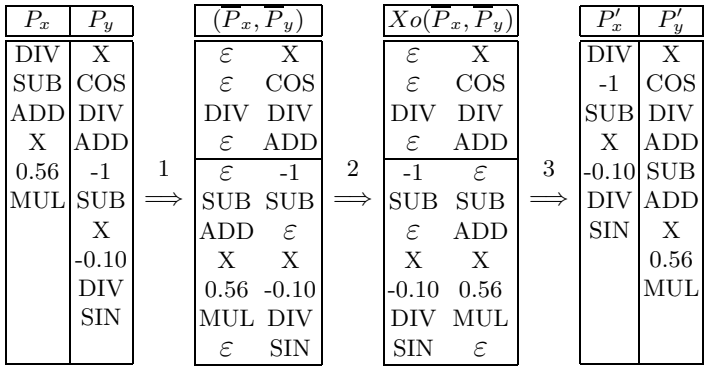

Fig. 1. 1-point MHC_1 of P_x and P_y in stack based representation : Step 1, alignment and Xover site selection (here 5); Step 2, swapping sequences ; Step 3, deletion of gaps.

2.3 Features of Maximum Homologous Crossover

We note that offspring produced with MHC could also be obtained using standard LGP crossover. Indeed, MHC only restricts the choice of possible crossover sites in both parents : for example in Figure 1, numerical constants 0.56 from P_x and -0.10 from P_y corresponds to the same crossover site in alignment $(\overline{P}_x, \overline{P}_y)$, then they could not appear together in children. Moreover, MHC modifies the probability of sites selection according to the local homology of parents. In previous example, the alignment $(\overline{P}_x, \overline{P}_y)$ gives a probability 3/11 to the sequence 'DIV, SUB' in P_x to be broken. Without alignment, this probability was only 1/6. This particular behaviour increases disruption rate of the less homologous regions (like 'DIV, SUB'), where many gaps are present, since they are more involved in sites selection. On the other hand, most homologous regions are more rarely disrupted.

An interesting property of MHC, is that the more similar the parents are to each other, the more similar the offspring are to their parents ; in other words, the distance between parents and offspring are always smaller than distance between parents. We have found experimentally, by performing MHC between

randomly generated programs that $\mathcal{D}(P_z, P_x) + \mathcal{D}(P_z, P_y) = \mathcal{D}(P_x, P_y)$, with P_x, $P_y \in P_\Sigma$ and $P_z \in \{MHC(P_x, P_y)\}$. Thus, we can assert that MHC performs more like GA crossover than standard LGP crossover, since it is a global search operator at the beginning of the evolutionary process, and becomes more local as the population diversity falls (decrease of distance).

3 Royal Road landscapes for LGP

In GA, Royal Road landscapes (RR) were originally designed to describe how building blocks are combined to produce fitter and fitter solutions and to investigate how the schemata evolution actually takes place [7]. Little work is related to RR in GP ; e.g. the Royal Tree Problem [16] which is an attempt to develop a benchmark for GP and which has been used in Clergue et al. [5] to study problem difficulty. Moreover there is nothing relevant for LGP.

Our aim is to examine in depth MHC's behaviour during evolution in order to quantify how it preserves homology and under what conditions. To achieve this goal, we need experiments able to highlight the destructive (or constructive) effects of crossover on building blocks. So we propose a new kind of fitness landscapes, called Royal Road landscapes for LGP, which have to be seen as preliminary steps in MHC understanding.

To define RR, we have choose a family of optimal programs and we break them into a set of small building blocks. The set of optima is :

$$O_{RR} = \{P_x \in P_\Sigma \mid \forall s \in \Sigma,\, B_K(P_x, s)\}$$

with :

$$B_K(P_x, s) = \begin{cases} true & \text{if } \exists i \in [0, m-K] \mid \forall k \in [0, K-1],\, x_{i+k} = s \\ false & \text{else} \end{cases}$$

and N the size of Σ, K the size of blocks, and x_{i+k} the $i+k$ instructions of P_x, a program of size m. The following program $P \in P_\Sigma$ is an example of RR optimum, with $N = 4$ and $K = 3$:

$$P = \mathbf{DDD}BCA\mathbf{BBB}BDD\mathbf{AAACCC}BCCC$$

with $\Sigma = \{A, B, C, D\}$. Thus, a block is a contiguous sequence of a single instruction, and only the presence of a block is taken into account in fitness evaluation (neither the position or repetition). The number of blocks corresponds to the number of instructions $s \in \Sigma$ for which the predicate $B_K(P_x, s)$ is true. In P, we only boldfaced sequences that contribute to fitness [4]. We have arbitrarily fixed the worst possible fitness [5] to F_w for programs having no blocks. In RR the contribution of each block is simply F_w/N and so, the standard fitness $\mathcal{F}(P_x)$ with P_x having n blocks is $\mathcal{F}(P_x) = F_w - n \times F_w/N$.

[4] Altough the last sequence of 'C' instruction in P is a valid block, it doesn't contribute to fitness since it is only a repetition.

[5] Actual fitness value does not matter since we use tournament selection.

To efficiently reach an optimum in RR landscapes, a GP system has to create and combine blocks without breaking existing structures. RR were designed so that fitness degradation due to crossover occurs only when recombination sites are chosen inside blocks but never in case of blocks translocations or concatenations. In other words, there is no inter blocks epistasy in RR.

4 Experimental Results

4.1 Setup

We want to compare effectiveness of two different recombination operators, the standard LGP 1-point crossover (SC) and the 1-point MHC, which have very distinct behaviours, that is why the search for the best tuning of evolutionary parameters is necessary. We have performed, on our own LGP system, 35 independent runs with various mutation and crossover rates. Let us notice that a mutation rate of 0.9 means that each program involved in reproduction has a 0.9 probability to undergo one insertion, one deletion and one substitution. The population of 1000 individuals was randomly created according to a maximum creation size of 50 and a set of N (depending on problem definition) available instructions. The evolution, with elitism, maximum program size of 100, 10-tournament selection, and steady-state replacement, took place during 400 generations. We used a T-test with 95% confidence to determine if results were significantly different.

4.2 Homology Driven Fitness Problem

We introduce the Homology Driven Fitness problem (HDF), where the fitness of a candidate program is given by its distance (homology) to a randomly chosen optimum of a given size. HDF matches the unitation problem or One-Max problem, which is known to be easier to handle using a GA than using a steepest ascent hill-climber.

We perform experiments on HDF with N=16 and size of optimum fixed to 80. Figure 2 shows that the increase of crossover rate improves the average fitness of best program found on HDF with MHC. In contrast, the use of SC decreases performance even with a small application rate. We find for both crossover operators that the best mutation rate is 0.3. Figure 3 gives evolution of the average fitness of the best program found using the best tuning of parameters found. Using MHC, the optimal program is found at generation 50 in more than 90% of runs, whereas with SC the percentage falls to 28%. This empirical results confirm that, in terms of its dynamics, MHC performs more like crossover in GA than SC.

4.3 Royal Road Landscape

Table 1 gives results on RR with N=8, 10, 16 for SC, MHC_1, and MHC_2. The average number of blocks found shows that the problem difficulty increases with

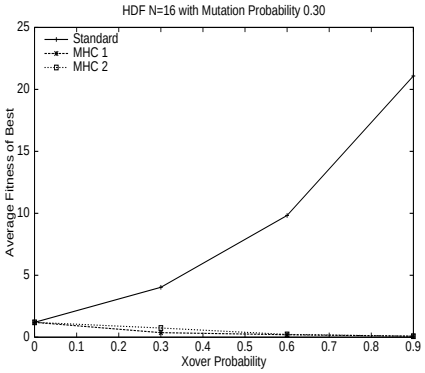

Fig. 2. Average fitness of best as a function of the crossover rate on HDF N=16 and size of optimum 80 at generation 50.

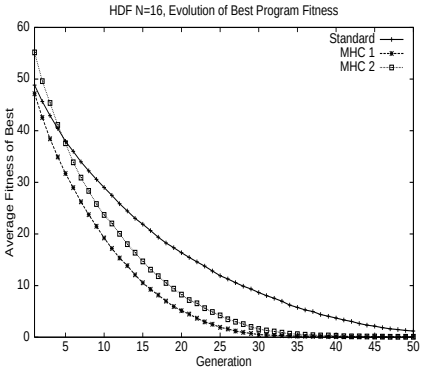

Fig. 3. Evolution of average fitness of best on HDF N=16 and size of optimum 80 with the best crossover rate found and mutation rate 0.3.

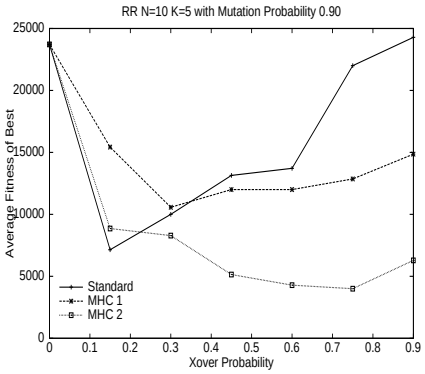

Fig. 4. Average fitness of best as a function of the crossover rate on RR N=16 and K=5.

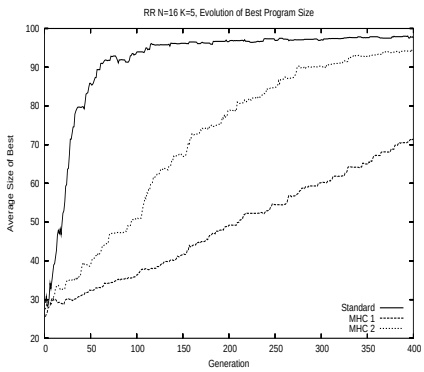

Fig. 5. Evolution of average size of best on RR N=16 and K=5 with the best crossover rate found and mutation rate 0.9.

N, which is to be expected, because of a smaller probability of block discovery and a stronger constraint due to size limitation (for N=16 optimum length is 80% of maximum allowed size). In this case, the best tuning of mutation rate is 0.9. For all N, MHC_2 is the best of the three crossovers, whereas MHC_1 gives surprisingly poor results even whith N=8 (convergence speed). Then, we see that using MHC_2, LGP performs better with high recombination rates, see also Figure 4. Figure 5 reports evolution of average size of the best. Let us notice that the bloat phenomenon is strongly reduced with MHC. In the case of MHC_1, the size of the best increases too slowly to reach size of optimum (at least 80 instructions) in 400 generations.

In what follow, we say that a crossover event is selectively Advantageous (noted A), when at least one child outperforms both parents. We say that a crossover event is selectively Deleterious (noted D), when both parents outperforms children. Other crossover events are said selectively Neutral (noted N). Table 2 reports the frequency of such crossover events. We observe, for N=8 and N=16, that D events prevail with SC and that their frequency is dramatically reduced using MHC. Moreover, we notice a significant increase in the number of A events, when MHC is used. In Figure 6, we have plotted these frequencies as a function of the average number of blocks in parents. We see that with SC, the frequency of D events increases linearly with the number of blocks so that the SC becomes massively disruptive at the end of the evolutionary process. On the other hand, the frequency of the various events is approximately constant ; it is a nearly neutral operator. The Building Blocks Hypothesis (BBH) exists in GP and states that good building blocks in individuals can be combined into even larger and better building blocks to form better individuals. However, SC works against the hypothesis since it tends to break high order blocks. As for MHC, it seems to be able to preserve and combine blocks, even when they represent the main part of the genome, as in RR with N=16 and K=5, where 80 instructions (over 100 potential) of an individual are useful.

5 Conclusion and Perspectives

A better understanding of the role of crossover, together with the improvement of its contribution to the overall performances, is a necessary search for GP. Standard crossover, that blindly swaps parts of parents, should be considered more like a macro mutation operator.

Considering the HDF problem, which should be viewed as a One-Max problem for LGP, we show that the standard crossover prevents the formation of fitter individuals : as increasing its rate, the performances of the algorithm decrease. Blocks disruption can be studied with Royal Road landscapes. Indeed, to be efficiently treated, this problem imposes the preservation of the blocks during evolution. An insight into the disruption rate indicates that it is not the case, since children have less blocks than their worst parent in around 80% of crossover applications.

Table 1. Best results found on RR.

Xover Type	Best Xover Rate	Best Mut. Rate	Avg. Best Fitness	Succes Rate	Avg. Generations	Avg. Number of Blocks
$N = 8$ and $K = 5$						
SC	0.45	0.9	$0_{(\sigma=0)}$	1.0	$146.6_{(\sigma=82.7)}$	8.00
MHC 1	0.75	0.9	$0_{(\sigma=0)}$	1.0	$206.0_{(\sigma=94.4)}$	8.00
MHC 2	1.0	0.9	$0_{(\sigma=0)}$	1.0	$126.2_{(\sigma=42.4)}$	8.00
$N = 10$ and $K = 5$						
SC	0.15	0.9	$7142_{(\sigma=7928)}$	0.31	$297.6_{(\sigma=73.4)}$	9.28
MHC 1	0.3	0.9	$10571_{(\sigma=7537)}$	0.22	$318.3_{(\sigma=68.4)}$	8.94
MHC 2	0.75	0.9	$4000_{(\sigma=5993)}$	0.62	$283.0_{(\sigma=71.8)}$	9.60
$N = 16$ and $K = 5$						
SC	0.15	0.9	$54462_{(\sigma=6705)}$	0	-	7.25
MHC 1	0.3	0.9	$58214_{(\sigma=6971)}$	0	-	6.68
MHC 2	0.45	0.9	$48928_{(\sigma=6423)}$	0	-	8.11

Table 2. Frequency of Deleterious, Neutral and Advantageous crossovers on RR.

Xover Type	$N = 8$ and $K = 5$			$N = 16$ and $K = 5$		
	D	N	A	D	N	A
SC	79.47	20.43	0.09	76.12	23.78	0.08
MHC 1	9.21	90.68	0.09	7.44	93.33	0.21
MHC 2	3.22	96.61	0.15	2.62	96.95	0.41

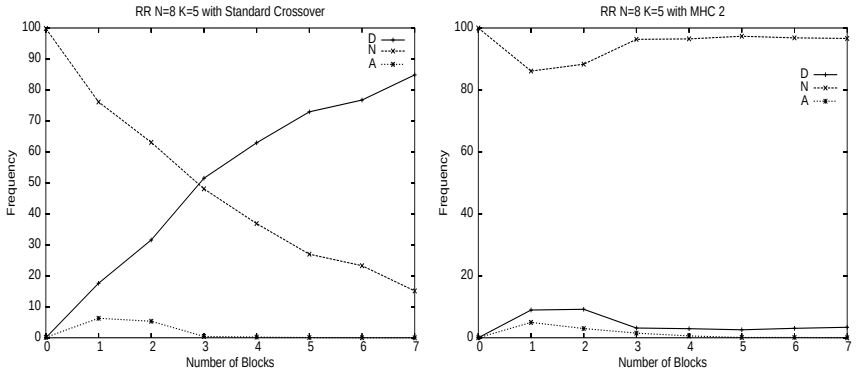

Fig. 6. RR with N=8 and K=5 : Frequency of Deleterious, Neutral and Advantageous crossovers according to the number of blocks.

In our point of view, efficiency requires that crossover, as with GA, behaves like a recombination operator. That is the reason why we introduce the Maximum Homologous Crossover. This operator tends to keep safe similar regions of the parents, in order to favour a kind of "respect" property (the common features of the parents are present in children). MHC_2 operator gives expected results on Royal Road landscapes.

The Royal Road problem is far from real GP problems, at least for two reasons : firstly, there is no inter blocks epistasy and secondly, the relative position of blocks does not matter in the evaluation process. That is why we consider this contribution as a preliminary step to study MHC behaviour. Future work should introduce epistasy and block location, before addressing classical benchmarks and real problems, like Symbolic Regression and the Even-Parity problems.

References

1. Lee Altenberg. The evolution of evolvability in genetic programming. In *Advances in Genetic Programming*. MIT Press, 1994.
2. P. J. Angeline. Subtree crossover: Building block engine or macromutation ? In *Genetic Programming 1997: Proceedings of the Second Annual Conference*. Morgan Kaufmann, July 1997.
3. Markus Brameier and Wolfgang Banzhaf. A comparison of linear genetic programming and neural networks in medical data mining. *IEEE Transactions on Evolutionary Computation*, 5(1):17–26, 2001.
4. Markus Brameier and Wolfgang Bhanzhaf. Explicit control of diversity and effective variation distance in linear genetic programming, 2001.
5. Manuel Clergue, Philippe Collard, Marco Tomassini, and Leonardo Vanneschi. Fitness distance correlation and problem difficulty for genetic programming. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 724–732, New York, 9-13 July 2002. Morgan Kaufmann Publishers.
6. Patrik D'haeseleer. Context preserving crossover in genetic programming. In *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, volume 1, pages 256–261, Orlando, Florida, USA, 27-29 1994. IEEE Press.
7. Stephanie Forrest and Melanie Mitchell. Relative building-block fitness and the building-block hypothesis. In *Foundation of Genetic Algorithms 2*, pages 109–126. Morgan Kaufman, 1993.
8. D. Gusfield. *Algorithms on Strings, Tree and Sequences*. Cambridge University Press, 1997.
9. J. Koza. Genetic programming - on the programming of computers by means of natural selection. Nature, 1993.
10. W. B. Langdon. Size fair and homologous tree genetic programming crossovers. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1092–1097, Orlando, Florida, USA, 13-17 July 1999. Morgan Kaufmann.
11. V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics-Doklandy, 1966.
12. Peter Nordin, Wolfgang Banzhaf, and Frank D. Francone. Efficient evolution of machine code for CISC architectures using instruction blocks and homologous crossover. In *Advances in Genetic Programming 3*, chapter 12, pages 275–299. MIT Press, Cambridge, MA, USA, June 1999.

13. U. O'Reilly. Using a distance metric on genetic programs to understand genetic operators, 1997.
14. Tim Perkis. Stack-based genetic programming. In *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, volume 1, pages 148–153, Orlando, Florida, USA, 27-29 1994. IEEE Press.
15. Riccardo Poli and W. B. Langdon. Genetic programming with one-point crossover. In *Soft Computing in Engineering Design and Manufacturing*, pages 180–189. Springer-Verlag London, 23-27 June 1997.
16. William F. Punch, Douglas Zongker, and Erik D. Goodman. The royal tree problem, a benchmark for single and multiple population genetic programming. In *Advances in Genetic Programming 2*, chapter 15, pages 299–316. MIT Press, Cambridge, MA, USA, 1996.
17. R.E. Keller W. Banzhaf, P. Nordin and F.D. Francone. *Genetic Programming - An Introduction.* Morgan Kaufmann, 1998.