Analysing the Effects of Diverse Operators in a Genetic Programming System

MinHyeok Kim[1], Bob (RI) McKay[1], Kangil Kim[1], and Nguyen Xuan Hoai[2]

[1] Seoul National University, Korea
[2] Hanoi University, Vietnam
{rniritz,rimsnucse,kangil.kim.01,nxhoai}@gmail.com
http://sc.snu.ac.kr

Abstract. Some Genetic Programming (GP) systems have fewer structural constraints than expression tree GP, permitting a wider range of operators. Using one such system, TAG3P, we compared the effects of such new operators with more standard ones on individual fitness, size and depth, comparing them on a number of symbolic regression and tree structuring problems. The operator effects were diverse, as the originators had claimed. The results confirm the overall primacy of crossover, but strongly suggest that new operators can usefully supplement, or even replace, subtree mutation. They give a better understanding of the features of each operator, and the contexts where it is likely to be useful. They illuminate the diverse effects of different operators, and provide justification for adaptive use of a range of operators.

Keywords: Evolutionary Operator, Tree Adjoining Grammar, Genetic Programming, TAG3P, Fitness, Tree Size, Tree Depth

1 Introduction

Many diverse genetic operators are used in evolutionary computation. In classical Genetic Programming (GP) [11], and most other tree-based GP systems (e.g. Context Free Grammar (CFG) GP [19]) there is less flexibility: constraints on tree structures restrict the available operators, and most of those since defined (e.g. [12,17]) are sub-operators of Koza's. However a number of GP systems do permit more varied operators: for example, linear GP representations such as Grammatical Evolution (GE) and Gene Expression Programming (GEP), which has led to claims that these more varied operators support better search, with some level of supporting analysis [8,3]. Similarly, Tree Adjoining Grammar-Guided GP (TAG3P) [6] permits greater structural flexibility than other tree representations. The resulting range of operators were claimed in [6] to be both diverse and beneficial, but beyond some tailoring of operators to specific problems, there has been little analysis. One is entitled to scepticism. Perhaps the new operators merely overlap each other in functionality, without greatly changing the search. We aim to characterise the effects of these operators, determining the extent of their problem specificity. This forms part of a wider stream of work,

C.A. Coello Coello et al. (Eds.): PPSN 2012, Part I, LNCS 7491, pp. 387–396, 2012.

investigating the combination of a diverse range of operators with operator rate self-adaptation. It focuses on TAG3P [10] as an example of a much wider range of such algorithms.

There is already extensive research in Genetic Algorithms (GA) into how genetic operators affect individuals and move them in the solution space [14]. Related research in GP has been more limited [5], and restricted to a fairly narrow range of operators. Thus the (potentially) more diverse operators of TAG3P form an interesting field of study in its own right. In GA, the issues to consider are relatively limited – because the complexity of individuals does not change, the main interest lies in how fitness distributions change under the effect of operators. While this is also important in GP, there are other important dimensions of change, notably the change in complexity [13], as measured for example by individual size and/or depth. Our objective is to study these effects.

In section 2 we provide additional background on previous studies of operator effects in GP, and also on genetic operators supported by TAG3P. Section 3 details the methods we used, including the experimental regime and parameters. Section 4 presents the results of the experiments. In section 5 we discuss their implications, concluding in section 6 with the assumptions and limitations of our study, a summary of the general conclusions, and directions for further work.

2 Background

2.1 Genetic Operators and the Solution Space

Evolutionary algorithms operate on a search space, moving individuals toward optima by applying genetic operators such as crossover and mutation. GA researchers have analysed the effect of operators in the search space (e.g. [20,14]). The flexible chromosome structure makes this more complex in GP; nevertheless a schema theory has been derived (e.g. [18,16] and others [5] have presented a new operator-based distance measure for GP, implicitly analysing the effects of genetic operators on individuals in the search space.

2.2 Genetic Operators in Tree Adjoining Grammar Guided GP

Tree Adjoining Grammar-Guided GP (TAG3P) is a grammar-guided GP [6], based on Tree Adjoining Grammars (TAG) [9]. It is based on the adjunction operator, which models the way elements such as adjectives ('big', 'black') and phrases ('preening its fur') – β trees in TAG terminology – may be inserted into basic sentences ('The cat sat on the mat') – α trees – so as to generate new, more complex sentences ('The big black cat sat on the mat preening its fur'). TAG3P's key property is flexibility: the representation is less constrained than other tree-based GP systems.[1] Thus it is possible to extend typical GP operators to TAG3P,

[1] Specifically, it is always possible to delete any subtree from a TAG3P tree while retaining its feasibility, and it is always possible to adjoin to any unoccupied adjunction site; this property is not shared by other GP tree representations.

but also to define a range of further genetic operators, including some based more directly on classical GA operators, and others modelling features of biological genetic phenomena. In this respect, it resembles linear GP representations more than it does other tree-based GP systems.

Typical TAG3P Genetic Operators include:

1. Size-Fair Crossover (X) is directly analogous to that in other tree-based GP, permitting exchange of random subtrees with roots having matching nonterminals, and (in size-fair form [12]) of the same size.
2. Subtree Mutation (M) selects a random point in the tree, deletes the subtree below it, and replaces it with a subtree built with the initialisation algorithm.
3. Duplication and Truncation (D/T) randomly choose a node. In duplication, the subtree below is copied to a matching location in the individual; in truncation, it is removed. These operators have opposite size and depth biases, so they are used paired: Duplication or truncation is randomly chosen with 0.5 probability. They are useful for coarse adjustment.
4. (Point) Insertion and Deletion (I/D) randomly choose a node. Insertion selects an open node (a location that has not been adjoined), randomly chooses a matching β tree, and adjoins it. Deletion chooses a closed node and deletes its child. As with the duplication and truncation operators, they are used paired. They are useful for fine-tuning the size of an individual.
5. Relocation (R) disconnects a random subtree from the tree, and randomly re-adjoins it at another open location with the same label. By design, it is a deterministically size-fair operator.
6. Replication copies a parent to its child, preserving it for the next generation.

3 Methods

This work aims to characterise the effect on fitness, size or depth of the various evolutionary operators. The change depends on the state of the system, hence we wanted to see how that change itself varied over the course of an evolutionary run. We did this by conducting typical GP runs. At each generation, in addition to the normally-created children which were actually used in the evolutionary run, we generated extra children simply to evaluate the effects of the different operators, but not otherwise used in the run.

In each generation, we took 200 additional samples for each operator (in addition to those used for evolution) – of the same order as the number of real trials of each operator in a generation. We selected the parents for these trials using the selection mechanism. Thus we were examining the children actually reachable after selection.

3.1 Test Problems

We used a family of symbolic regression problems [11] and *Lid* problems [4,15,1]. In the symbolic regression problems, the target was a polynomial $F_n = \Sigma_{i=0}^{n} x^i$,

$n \in \{3, 6, 9, 12, 15\}$. We evaluated candidates f on 20 random points $X \subset [-1, 1]$; f is a *hit* if, for some predefined $\epsilon, \forall x \in X : |F_n(x) - f(x)| < \epsilon$. The objective was, using $\{+, -, \times, \div, \sin, \cos, \exp, \log\}$, to construct a hit, with fitness $(f) = \sum_{x \in X} |F_n(x) - f(x)|$. Among *Lid* problems, we used the Majority and Order problems. The target for Majority is a tree in which the number of nodes P_i is larger than of nodes N_i for all i; the target for Order is a tree in which there is a P_i before each N_i, in preorder traversal, for all i. All trees in these problems are binary trees, using only one function $\{JOIN\}$ and $2n$ terminals $\{P_i, N_i : i = 1 \ldots n\}$. The fitness function is {the number of i satisfying the condition}. We used $n \in \{25, 30\}$ for both problems.

3.2 Experimental Settings

Figure 1 shows the elementary trees defining the TAG grammar used by TAG3P [7,6]; we ran 100 trials for each problem. Table 1 shows the detailed parameter settings. Typical GP systems use high rates of crossover and lower rates of other operators for best performance. But our aim was to examine the behaviour of the system; a high crossover rate would imply low rates for other operators. We used a compromise rate of 0.5 for crossover, other operators 0.1 in creating the 'normal' children that were actually used in evolution.

Table 1. Experimental Settings (Left: Symbolic Regression; Right: Lid)

Target Function	$F_3, F_6, F_9, F_{12}, F_{15}$	$M_{25}, M_{30}, O_{25}, O_{30}$
Fitness Cases	20 Random Points from $[-1, 1]$	
Fitness Function	Sum of MAE of cases	# of satisfied i
Success Predicate	Error <0.01 on all fitness cases	n fitness value
Function Set	$+, -, \times, \div, \sin, \cos, \exp, \log$	JOIN
Terminal Set	X	$P_1, ...P_n, N_1, ...N_n$

Generations	50	Population Size	500	Tournament Size	3

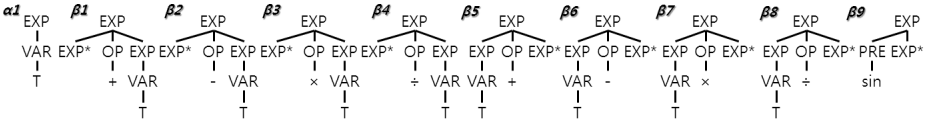

Fig. 1. TAG ElementaryTrees for Experiments for Symbolic Regression

Table 2. Overall Problem Performance

Problem	Success	Hit Time	Problem	Success	Hit Time	Problem	Success	Hit Time
F_3	100%	1.45	F_{12}	7%	48.84	M_{30}	12%	48.15
F_6	57%	30.96	F_{15}	6%	48.72	O_{25}	97%	23.11
F_9	23%	45.03	M_{25}	28%	42.98	O_{30}	80%	33.72

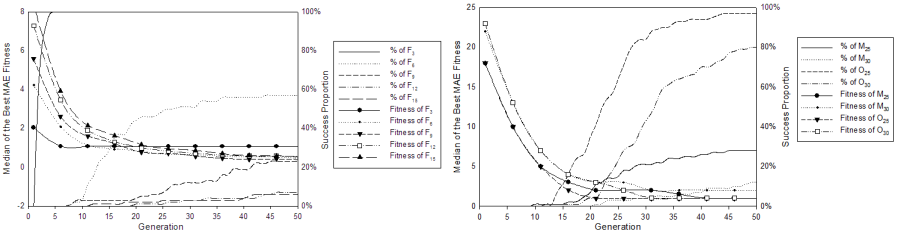

Fig. 2. General Performance

4 Results

4.1 Overall Performance

Before analysing operator effects, we first present an overview of the system performance. Figures 2 and Table 2 illustrate the performance (hit time is the first generation that a hit occurs). Figures 2 shows both the median (over all runs) of the best fitness (left axis), and the cumulative success rate (right axis). System performance on all problems decreases as n increases: the problems become tougher with n, but at a decreasing rate. The fitness curves are typical for evolutionary processes – an initial steep fall to about generation 10, then gradual convergence. The cumulative success curves are also typical, with little success at first, an increasing rate in the middle region, and a final tailing off. Based on this, to condense the immense amount of data generated, we divided the generations into three stages: begin (1-10), middle (11-25), end (26-50).

4.2 Detailed Analyses

We conducted detailed analyses on all experiments, but can only show F_9 and O_{30} due to space. F_9 is intermediate in difficulty and typical of both extremes, while O and M problems behaved similarly to each other. F_9 and O_{30} are sufficient to summarise the general trends, though we will mention some more detailed observations when appropriate. For brevity, we denote a plot for function X_m calculated from the fittest $n\%$ of children as $X_m^{n\%}$, with $n \in \{10, 30, 50, 70, 90\}$ and $X \in \{F, M, O\}$. The figures show how the genetic operators change the properties of individuals in each learning stage. The horizontal bars indicate means over 100 runs, while the vertical lines show their standard deviations.

All plots show how each operator changes the specific property for individuals (the difference between child and parent values – for fitness, negative values indicate improvement). Replication is omitted because it deterministically has no effect.

Fitness Analysis: The results in Fig. 3 overall reflect our understanding of evolutionary behaviour: the operators have a larger range of effect in early search (they are more exploratory), whereas later on, elite children resemble their parents much more.

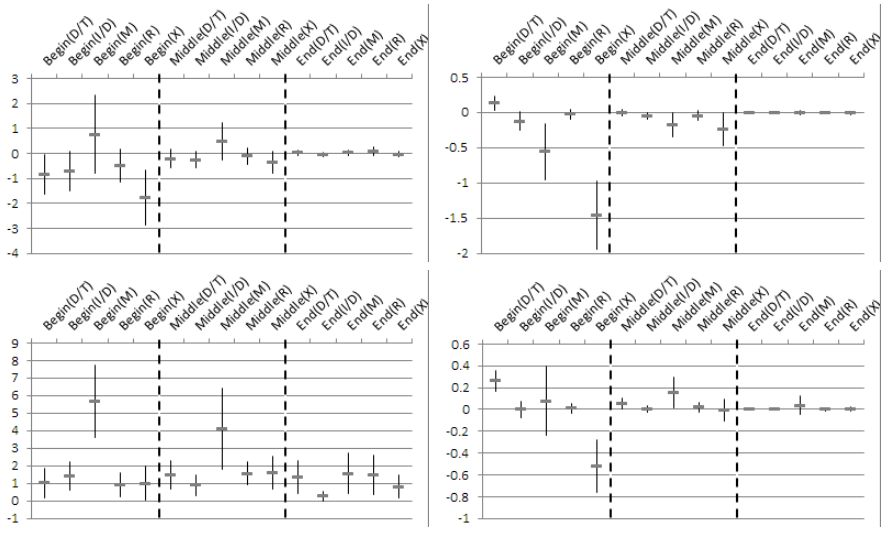

Fig. 3. Fitness Change for Selected Parents
Top: 30% Elite; Bottom: 70% Elite; Left: F_9; Right: O_{30}

The most notable differential effect in Fig. 3 is the much larger range of effect of the traditional M and X operators: the new TAG3P operators have a much smaller overall range of effect, suggesting that they are much less exploratory. In the early stages, X is on average much more beneficial than mutation – for F_9, most of the 30% elite children are an improvement on their parents, while much fewer M children are; any benefit from M comes from rarer positive mutations. While M is overall constructive for problem O_{30}, it is still substantially less so than X. However the effect of X rapidly diminishes, especially for O_{30}; M remains effective longer.

I/D are generally beneficial in early stages (the 30% elite see some worthwhile improvement on their parents. I/D retains small but very slightly beneficial effect until the end stages, befitting its proposed role as a fine-tuning operator.

D/T behave similarly to I/D on F_9, though any beneficial effect disappears by the end stages. Their effect on O_{30} is rather different, being slightly damaging in the early stages of search, very slightly beneficial in the mid stages, and losing all effect at the end.

R throughout has a relatively small effect, disappearing almost entirely by the end stages (deterministically, it had no effect in the majority problem, since it cannot change the fitness).

Size Analysis. While we saw different trends between 30% and 70% elite children in the fitness plots, there was no such difference for size – size effects were independent of child fitness; we display the results for the 50% elite. R and X do not change size at all, so we omit them from discussion.

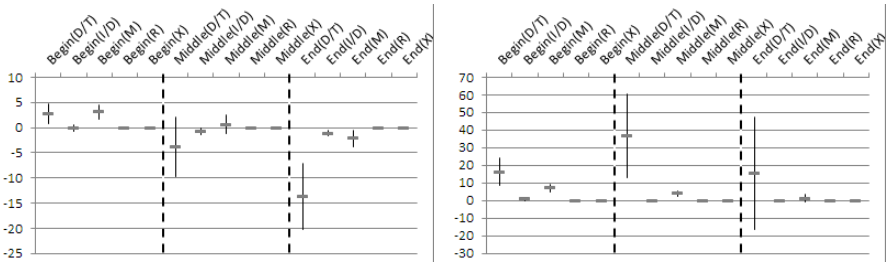

Fig. 4. Size Change, Left: $F_9^{50\%}$; Right: $O_{30}^{50\%}$

D/T generally causes a size change over the run (Fig. 4), with the scale increasing gradually. However the effect is reversed between the problems: D/T decreases size for F_9 but increases it for O_{30} (similar, but less pronounced, effects were seen with other operators). The difference may be because most individuals were near the size bound in F_9, so that many larger duplications would fail, while most truncations would succeed, introducing a bias.

M began by slightly increasing the size of individuals, but the scale decreased to zero for O_{30}, and M eventually became reducing for F_9. I/D (by design) made only very small size changes throughout.

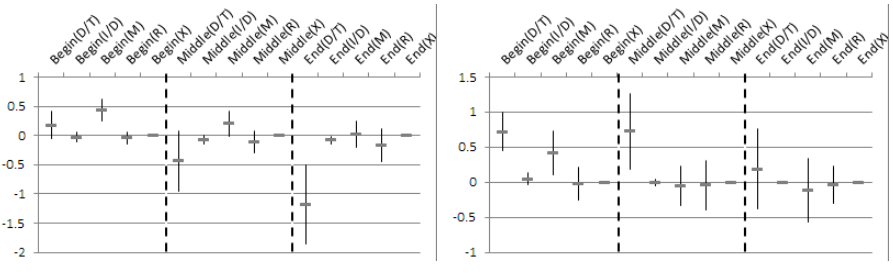

Fig. 5. Depth Change, Left: $F_9^{50\%}$; Right: $O_{30}^{50\%}$

Depth Analysis. We omit analysis of X because, as with size, most operator applications result in no change in depth, so there is little to see.

The general trends are similar to size (Fig. 5), but on a reduced scale (because of the logarithmic relationship between depth and size). The shapes of the plots are generally very similar. The only exception is with operator R, which shows a slight bias toward depth reduction, increasing in scale over time.

5 Discussion

From the perspective of fitness change, crossover appears to be the most effective operator at the start of a run. However, insertion/deletion may be preferable at

the end because of its ability to fine-tune results. On the other hand, subtree mutation is not particularly effective when we consider fitness change. It has very low probability of improving individuals; but it does cause the biggest changes in fitness. It appears from these results that other operators (not available in most tree-based GP systems) may be more effective. Duplication/truncation does theoretically similar work to insertion/deletion: it adds or deletes a sub-tree in the individual. Because insertion/deletion uses just one elementary tree, but duplication/truncation uses a sub-tree, duplication/truncation seems to work similarly but with a larger step size. On the other hand, there are no obviously strong points for relocation in terms of fitness: It may not have much effect on learning, at least in these problems.

Size and depth effects appear to be largely independent of fitness.This may have some implications for theories of the cause of GP bloat. There is little difference between the operators in their effects on size and depth (except for operators specifically designed not to affect them). At first, all operators that are free to do so increase size and depth. At the end of a run, however, the reverse occurs, and the operators decrease size and depth in the symbolic regression problem. Our hypothesis at this point is, in interacting with the size bound, the genetic operators and selection reach an equilibrium – selection increasing size and depth, with the genetic operators decreasing them.

While mutation caused the greatest change in fitness, duplication/truncation led to the biggest changes in both size and depth, while relocation was able to change depth without affecting size. Thus when a problem requires structural change, duplication/truncation and relocation may be useful, but they can be correspondingly wasteful on problems such as Order.

6 Conclusions

6.1 Summary

We investigated the roles genetic operators play and what they are useful for. We confirmed that crossover is an effective operator in the early stages of GP, but it is not effective throughout a run. Subtree mutation, another well known operator, causes large changes in fitness, even in the middle of a run, but the changes are generally negative. Insertion/deletion may be a useful alternative, leading to smoother fitness search – It is effective for fine-tuning, but at the risk of getting stuck in local optima. Duplication/truncation and relocation may be useful when structural change is needed, but can also have negative effects on poorly-matched problems.

More generally, we may conclude that there is value in having a diverse range of operators: they really do perform different tasks, either in different problems, or at different times in the evolution of solutions for the same problem. Since we will not, in general, have a priori knowledge of which operator is most suitable a any specific time, this motivates and justifies research into operator adaptation in evolutionary algorithms in general, and in GP in particular.

6.2 Assumptions and Limitations

Our study was limited to TAG3P; conclusions are likely to extend to other more flexible GP representations, but have limited relevance to standard expression tree or CFG-based GP. While the problems considered are very different, yet generally yielded similar results, they may extend to other problem domains, but a wider sample would be desirable in future. Although we saw something of the gross structural effect of operators (on size and depth), we did not investigate their effect on tree shape in detail.

6.3 Future Directions

Our extensions will have two main directions. We will look more closely at the shapes of solutions, to better understand operator effects, including working with structure-only problems such as Daida's *Lid* problem [2]. We also aim to extend our analysis to a range of other GP test problems.

Acknowledgements. Seoul National University Institute for Computer Technology provided research facilities for this study, which was supported by the Basic Science Research Program of the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (Project No. 2012-004841), and the BK21-IT program of MEST. The fourth author was partly funded by The Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant 102.01-2011.08 for doing this work.

References

1. Daida, J., Li, H., Tang, R., Hilss, A.: What Makes a Problem GP-hard? Validating a Hypothesis of Structural Causes. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A., Dowsland, K.A., Jonoska, N., Miller, J., Standish, R.K. (eds.) GECCO 2003. LNCS, vol. 2724, pp. 1665–1677. Springer, Heidelberg (2003)
2. Daida, J.M., Bertram, R.R., Stanhope, S.A., Khoo, J.C., Chaudhary, S.A., Chaudhri, O.A., Polito, J.A.I.: What makes a problem gp-hard? analysis of a tunably difficult problem in genetic programming. Genetic Programming and Evolvable Machines 2, 165–191 (2001), doi:10.1023/A:1011504414730
3. Ferreira, C.: Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence. Springer (2002)
4. Goldberg, D.E., O'Reilly, U.-M.: Where Does the Good Stuff Go, and Why? How Contextual Semantics Influences Program Structure in Simple Genetic Programming. In: Banzhaf, W., Poli, R., Schoenauer, M., Fogarty, T.C. (eds.) EuroGP 1998. LNCS, vol. 1391, pp. 16–36. Springer, Heidelberg (1998)
5. Gustafson, S., Vanneschi, L.: Operator-Based Distance for Genetic Programming: Subtree Crossover Distance. In: Keijzer, M., Tettamanzi, A.G.B., Collet, P., van Hemert, J., Tomassini, M. (eds.) EuroGP 2005. LNCS, vol. 3447, pp. 178–189. Springer, Heidelberg (2005)

6. Hoai, N.: A Flexible Representation for Genetic Programming: Lessons from Natural Language Processing. Ph.D. thesis, University of New South Wales, Australian Defence Force Academy, Australia (2004)
7. Hoai, N., McKay, R.: A framework for tree adjunct grammar guided genetic programming. In: Proc. Postgraduate Conf. on Computer Science, Canberra, Australia, pp. 93–99 (2001)
8. Hugosson, J., Hemberg, E., Brabazon, A., O'Neill, M.: Genotype representations in grammatical evolution. Applied Soft Computing 10(1), 36–43 (2010)
9. Joshi, A.K., Levy, L.S., Takahashi, M.: Tree adjunct grammars. Journal of Computer and System Sciences 10(1), 136–163 (1975)
10. Kim, M., McKay, R.I.B., Kim, D.K., Nguyen, X.H.: Evolutionary Operator Self-adaptation with Diverse Operators. In: Moraglio, A., Silva, S., Krawiec, K., Machado, P., Cotta, C. (eds.) EuroGP 2012. LNCS, vol. 7244, pp. 230–241. Springer, Heidelberg (2012)
11. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
12. Langdon, W.B.: Size fair and homologous tree crossovers for tree genetic programming. Genetic Programming and Evolvable Machines 1(1-2), 95–119 (2000)
13. Langdon, W., Soule, T., Poli, R., Foster, J.: The evolution of size and shape. In: Advances in Genetic Programming, vol. 3, ch.8, pp. 163–190. The MIT Press, Cambridge (1999)
14. Moraglio, A., Poli, R.: Topological Interpretation of Crossover. In: Deb, K., Tari, Z. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 1377–1388. Springer, Heidelberg (2004)
15. O' Reilly, U., Goldberg, D.: How fitness structure affects subsolution acquisition in genetic programming. Genetic Programming 98, 269–277 (1998)
16. Poli, R., McPhee, N.: General schema theory for genetic programming with subtree-swapping crossover: Part i. Evolutionary Computation 11(1), 53–66 (2003)
17. Tackett, W., Carmi, A.: The unique implications of brood selection for genetic programming. In: World Congress on Computational Intelligence, Orlando, FL, USA, vol. 1, pp. 160–165 (June 1994)
18. Whigham, P.A.: A schema theorem for context-free grammars. In: IEEE Conf. on Evolutionary Computation, vol. 1, pp. 178–181. IEEE Press (November 1995)
19. Whigham, P., et al.: Grammatically-based genetic programming. In: Proc. Workshop on Genetic Programming: From Theory to Real-World Applications, July 9, pp. 33–41. Ueniversity of Rochester (1995)
20. Wineberg, M., Oppacher, F.: Distance between Populations. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A., Dowsland, K.A., Jonoska, N., Miller, J., Standish, R.K. (eds.) GECCO 2003. LNCS, vol. 2723, pp. 1481–1492. Springer, Heidelberg (2003)