Investigating the Influence of Depth and Degree of Genotypic Change on Fitness in Genetic Programming

Christian Igel
Institut für Neuroinformatik
Ruhr-Universität Bochum
44780 Bochum, Germany
igel@neuroinformatik.ruhr-uni-bochum.de

Kumar Chellapilla
Center for Information Engineering
University of California at San Diego
9500 Gilman Drive, La Jolla, CA 92093-4007, USA
kchellap@ece.ucsd.edu

Abstract

In this paper we investigate the influence of (a) the amount of variation generated in the genotype and (b) the depth of application of variation operators on the offspring fitness in genetic programming. Simulation results on three common test problems indicate that for certain features of the fitness distribution the location of the variation may play as important a role as the choice of the applied operators.

1 INTRODUCTION

In genetic programming (GP), computer programs are represented as parse trees whose structure and elements are to be evolved.

It is a common point of view that, on average, the larger the amount of genotypic change produced by a variation operator, the larger the associated change in the phenotype, i.e., the behavior of the program. It has also been hypothesized that the farther away the genotypic change occurs from the root node, the smaller is the corresponding change in the phenotype (Rosca and Ballard, 1995).

In the current study, these two hypotheses are investigated using a set of fitness distribution features. Five different variation operators are analyzed based on the depth of the node, to which they are applied, and the amount of structural change.

The depth of a node n in a tree is defined as the distance from the root node to n (Aho et al., 1983; Banzhaf et al., 1998) and the height of a tree as the maximum depth of its nodes (Aho et al., 1983). Let the depth of a subtree be defined to be the depth of its root node.

In most GP systems, the node for the application of a variation operator is chosen independently of how much genetic material (i.e., the number of nodes and the complexity of the associated subtree) will be altered from the tree or the depth of the node. The node is chosen uniformly at random or with a bias towards internal (function) nodes (Koza, 1992). Alternative operators, that take into account the position of the variation point, have been successfully applied: O'Reilly and Oppacher (1994) proposed a crossover operator that selected the pivot points based on the height of the subtrees and Harris and Smith (1997) and Ito et. al. (1998) investigated various depth-based crossover operators. In this investigation, we analyze the effect of the depth of the application point of crossover and four different mutation operators.

In order to measure structural differences between two parse trees a metric is needed. One such generic distance measure between trees is the tree edit distance (Sankhoff and Kruskal, 1983), which provides a suitable measure on the GP search space (O'Reilly, 1997). It is a well observed fact that changing even a single instruction of a program can drastically change its fitness. However, it seems intuitive that the more the number of instructions that are modified, the higher the probability of a large fitness change:

Hypothesis 1. *On average, the larger the change in the genotype* (*measured by a distance measure between parent and offspring*) *the larger is the change in fitness.*

Such a relation between distances in genotype-space and fitness-space is important for using a metric for measuring structural differences between parse trees as a means for analyzing GP algorithms. A distance measure on the genotype space that is independent of fitness looses much of its usefulness (Sendhoff, Kreutz, and von Seelen, 1997; Igel, 1998).

The second hypothesis of what makes a structural change a small one can be found in the work of Rosca and Ballard (1995):

Hypothesis 2. *The longer the path from a selected subtree to the root node, the higher is the probability that the subtree plays a less important role in the overall evaluation.*

The size of a subtree is partially determined by its depth. Consider a tree where all the leaves have the same depth: In this case, subtrees with small depth values contain more nodes than subtrees at larger depths and both of the above hypotheses are equivalent.

Of course, the validity of both statements depends not only on the problem at hand but also on the function and terminal sets used. As a result, it may not be possible to derive general results for all GP applications. However, one can hope to identify principles that hold for certain problem classes.

2 METHOD

2.1 FITNESS DISTRIBUTIONS

Fitness distributions have been proposed as tools for understanding the effects of variation operators in evolutionary computation (Grefenstette, 1995; Fogel and Ghozeil, 1996). Recent efforts in trying to understand GP through fitness distribution features include Nordin and Banzhaf (1995) and Igel and Chellapilla (1999).

The fitness distribution (FD) of a variation operator v that produces one offspring o from a single parent p can be defined as the conditional probability

$$\mathrm{FD}_v(F_{\{p\}}) =_{def} P(F_o | F_{\{p\}}) \ , \qquad (1)$$

where the random variables F_o and $F_{\{p\}}$ denote the fitnesses of the offspring and parent, respectively. If multiple parents are involved in generating the offspring, $F_{\{p\}}$ depends on the fitnesses of all parents used.

The FD_v is generally quite complex and difficult to compute. However, it is usually sufficient to focus on estimating important features of the FD. In this investigation, two fitness distribution features are used. Firstly, the expected absolute fitness change

$$\mathrm{AC}_{op} =_{def} \mathcal{E}(|F_o - F_{\{p\}}|) \ , \qquad (2)$$

where $\mathcal{E}(.)$ denotes the expectation. Secondly, the probability of silent variations, i.e., the fraction of applications of an operator that do not lead to a fitness change

$$\mathrm{SVP}_{op} =_{def} P(F_o = F_{\{p\}}) \ . \qquad (3)$$

In this paper, AC and SVP are viewed as functions of (a) the depth of the modified subtree and (b) the structural differences between parent and offspring. The AC can be used to distinguish between local (or exploitative) and global (or explorative) search operators: Typically, small AC values indicate a more local search while large AC values indicate a more global search.

2.2 EDIT DISTANCE

The tree edit distance (Sankhoff and Kruskal, 1983) has been proposed as a metric for program spaces (O'Reilly, 1997). In view of its direct applicability to tree spaces, the tree edit distance was adopted to measure structural differences in the experiments for testing Hypothesis 1.

Parse trees representing computer programs can be uniquely defined by their preorder traversal. Therefore, the distance between two GP trees may also be computed as the *Levenshtein* distance between their preorder traversals instead of using the "standard" tree edit distance that is defined for general trees (Igel, 1998). The Levenshtein distance is a common distance measure between strings (Sankhoff and Kruskal, 1983). It is the minimum cost of a sequence of insertion, deletion and substitution operations that transform one string into the other. In this investigation, each of these elemental operations was assigned a cost of one, except the replacement of a numerical constant with another numerical constant which was assigned a cost of zero.

The Levenshtein distance measure was used, since it can be computed faster[1] (and is easier to implement) than the standard tree edit distance (Zhang and Shasha, 1989). This definition of the distance between parse trees coincides with the standard tree edit metric in most cases and the rare differences were usually very small (Igel, 1998).

3 EXPERIMENTS

3.1 TEST PROBLEMS

Three common GP test problems were used to test the hypotheses. The first two were the artificial ant problem on the Santa Fe trail, using the function set {`is-food-ahead`, `prog2`, `prog3`} and the terminal set {`left`, `right`, `move`}, and the 6-multiplexer problem

[1]The tree edit distance can be computed in time complexity $\mathcal{O}(|T_1| \cdot |T_2| \cdot \min\{\text{leaves}(T_1), \text{depth}(T_1)\} \cdot \min\{\text{leaves}(T_2), \text{depth}(T_2)\})$ and the Levenshtein distance in $\mathcal{O}(|T_1| \cdot |T_2|)$.

with the function set {`and`, `or`, `not`, `if`} (Koza, 1992). The third task was the sunspot time series forecasting problem (Angeline, 1997). The goal was to predict the average number of sunspots observed in a year. Average sunspot data from the years 1700 to 1989 containing 290 samples was used as the training set. The function set was $\{+, -, *, \%, \texttt{sin}\}$ and the terminal set was $\{x_{t-1},\ x_{t-2},\ x_{t-4},\ x_{t-8},$ numerical constants$\}$. The fitness of an individual was determined by the normalized mean square error on the training set.

The artificial ant problem was chosen, because it is one of the best investigated tasks in GP. The regression problems were chosen since they are instances of two important GP problem classes, i.e., boolean and symbolic regression problems. Results on the sunspot problem may be expected to carry over to similar symbolic regression tasks and the results for the multiplexer may also be expected to hold for other Boolean regression problems[2].

3.2 VARIATION OPERATORS & GP ALGORITHM

An evolutionary programming procedure was used to evolve computer programs (cf. Chellapilla 1997, 1998).

Instead of using a depth-based tree generation method like *ramped half-and-half* an algorithm based on the length of the trees was employed to create the initial population: The number of nodes of each program was chosen at random from $\{3, 4, ..., 50\}$ and a program with approximately that length was generated (Chellapilla, 1998).

Several different variation operators were employed to generate offspring. The location at which an operator was applied to a tree T was chosen in a depth-dependent manner (except for the *OneC* mutation, see below):

1. First, the depth of application d was chosen uniformly at random in $\{0, 1, \ldots, h\}$ where h was the height of the tree T.
2. Then a node of the tree T at depth d was selected at random.

This was done in order to get a more uniform sampling over all depths. In comparison, non depth-dependent selection methods typically favor deeper nodes. The employed variation operators were:

[2]We compared the 6-multiplexer with another, quite different Boolean regression task, the 3-parity problem (Koza, 1992), and the results of our analysis were qualitatively similar.

OneNode: The *OneNode* operator replaced the randomly selected node with a different node of the same arity. An application of *OneNode* corresponds to a smallest structural change of the parse tree. In the ant and multiplexer problem, *OneNode* always produces an offspring with edit distance one from its parent. In the symbolic regression problem, a cost of zero was assigned to the replacement of a numerical constant with another numerical constant, so an application of *OneNode* leads to either an edit distance of one or zero.

Trunc: The *Trunc* mutation randomly selected a *function* node using the depth-dependent selection method described above (the method was simply repeated until a function node was found, i.e., $d \in \{1, \ldots, h-1\}$), and replaced it with a random terminal (Chellapilla, 1997).

Crossover: The crossover operator used in this investigation chose a mate uniformly at random from the population and selected crossover points in the parent and mate trees independently according to the depth fair selection method described above. It always returned the offspring that shared the same root node as the parent before the crossover operation.

RandomMateCrossover: A mate tree was chosen uniformly at random from the population. This *pattern tree* was randomized by replacing every function and terminal node by a randomly selected node of the same arity. Crossover was applied between the selected parent and the content-randomized tree. Once again, the offspring that shared the same root node as the parent before the crossover operation was returned. This mutation operator was inspired by the strong headless chicken crossover introduced by Angeline (1997).

OneC: The *OneC* operator was used only in the sunspot problem. It performed a zero mean Gaussian mutation with standard deviation 0.1 on a randomly chosen (in a non-depth-dependent manner) numerical constant.

In each generation, for every tree p_i in the population one of these operators was chosen at random with equal probability and applied to p_i to produce a single offspring. This operation was repeated if the application of the selected operator did not lead to an offspring with a genotype which was different from the parent. In this paper, $F_{\{p\}}$ was set equal to F_{p_i} for the calculation of the fitness distribution features described in Sec. 2.1. Previous investigations have shown

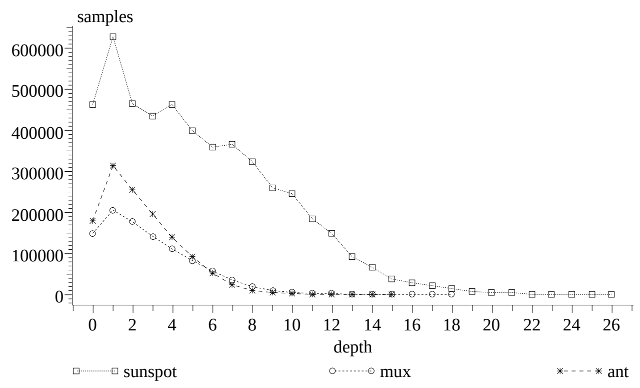

Figure 1: Number of samples used in this study per depth.

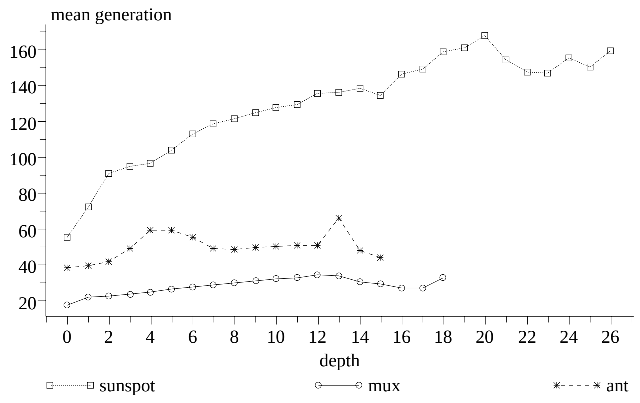

Figure 2: The average generation at which a depth was sampled. Smaller depths are sampled more often in early generations.

that the fitness distribution features of the employed five operators differ significantly from each other for the three test problems (Igel and Chellapilla, 1999).

Evolutionary programming style tournament selection with a tournament size of 50 was applied to form the next generation (Chellapilla 1997, 1998). The population size was 200 and the length of the individuals was limited to 50 nodes in all the experiments. For each test problem 50 independent trials were performed. The trials were terminated either upon discovering an optimal solution or when the number of generations exceeded 200.

4 RESULTS & DISCUSSION

All multiplexer trials and all but three artificial ant trials found an optimal solution before generation 200.

The results presented in the figures were averaged over all generations and trials. Figure 1 shows the number of samples obtained as a function of the depth of the node selected during variation. Due to the size constraint of 50 nodes, nodes at depths larger than 15 were rarely selected. This led to lower accuracy of the statistics at larger depths, i.e., to noisy curves at larger depth in Figs. 3 to 12. As evolved programs tend to grow with each generation during evolution (an effect commonly known as *bloat* (Banzhaf et al., 1998)), larger depths were sampled less often in the early generations. This effect can be observed in Figure 2, which shows the mean generation at which a depth was sampled.

On average, the multiplexer problems finished first, followed by the artificial ant and sunspot trials (the sunspot trials were not terminated until generation 200). This ranking explains the quantitative differences between the curves for the different operators in Figs. 1 and 2.

4.1 DEPENDENCE ON LEVENSHTEIN DISTANCE

Figures 3 through 5 show the AC curves as a function of the Levenshtein distance of the corresponding variation generated by the operators for the artificial ant, 6-bit multiplexer, and sunspot problem.

The AC curves presented in Figs. 3 and 4 show a high correlation between the AC values and the associated Levenshtein distance between parent and offspring. On the sunspot problem the curves were very noisy and Hypothesis 1 does not appear to hold. However, for small distances in the genotype-space (< 8) the AC curves were increasing which supports Hypothesis 1. In other words, the relation between distance in genotype-space and fitness-space holds *locally* (in a certain neighborhood in the genotype-space), which has been identified as being a very important property, (c.f. Rechenberg, 1994; Sendhoff et al., 1997; Igel, 1998).

The AC values for the *OneNode* operator (which always generates a change equivalent to producing a Levenshtein distance of at most one between parent and offspring) were larger than the corresponding AC values for the other operators when they produced structural changes with edit distance one. The other operators generated edit distance one changes usually only when applied near the leaf nodes, i.e., at larger depths. For example, crossover between two trees will lead to a structural change larger than one in most cases except when leaf nodes were selected as crossover points. Thus, of the operators used in the experiments, only *OneNode* frequently generates modifications of edit distance one through the application to

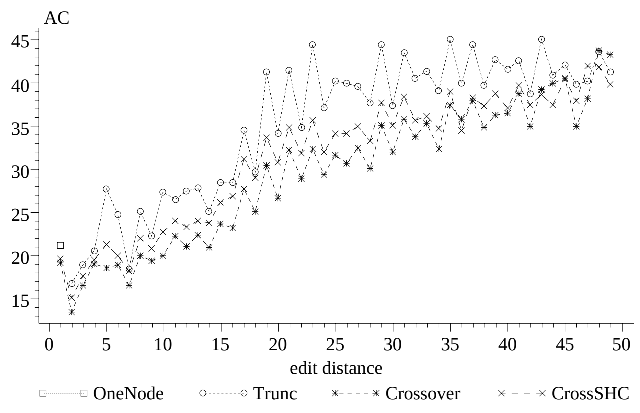

Figure 3: The average fitness change as a function of the edit distance between parent and offspring on the artificial ant problem. As the *OneNode* operator produces a maximal structural change of one, it is represented by a single square with an edit distance of one. Because the arity of each function node was at least two, the smallest edit distance *Trunc* generated was two.

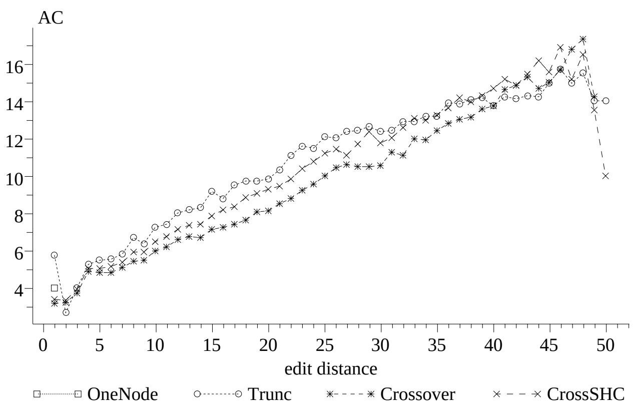

Figure 4: The average fitness change as a function of the edit distance between parent and offspring on the 6-multiplexer problem. The *Trunc* operation can only generate an edit distance of one when it is applied to a `not` operation with a terminal as argument. This special case leads to the outlier in the curve of the *Trunc* operator at depth zero.

(non-terminal) nodes close to the root node. In view of the fact that changes (using any variation operator) closer to the root generated larger AC values than changes at deeper nodes (see below), changes of edit distance one using *OneNode* produced larger AC values.

The *OneNode* results show that the *fitness landscapes* determined by the Levenshtein distance of the three test problems can be considered to be very rugged. As the depth-dependent results described in the next section show, an edit distance of one can result in small and large fitness changes. However, on average the AC increases with increasing edit distance for the discrete problems (ant and mux). Hence for two of the three problems Hypothesis 1 seems to hold. For the continuous problem, it seems to hold only locally.

4.2 DEPTH-DEPENDENCE

The Figs. 6 to 11 show the results for the analysis of the depth dependence. For each operator, only depths that were sampled more than 100 times were considered in these plots.

The choice of the depth of application of a variation operator has a large influence on the degree of fitness change produced and the probability of generating silent variations. On average, variations closer to the root generate larger fitness changes and fewer silent mutations. For the two discrete problems, the depth of application appears to be as important as the type of operator used. In other words, it is just as important to know the depth at which an operator is applied as the type of variation generated by the operator. For the sunspot problem, these results do not hold so clearly; the corresponding curves are rugged and there are several outliers (Figs. 10 and 11). However, Hypothesis 2 can be regarded as valid for the continuous domain problem, at least when the depths under consideration are significantly apart.

The results show that it is possible to tune the search behavior of an operator from local to global search by controlling its application depth.

It is interesting to note that in Figs. 6, 8 and 10 the average fitness change for *OneNode* (and also for *Crossover* and *Trunc* in the case of the sunspot problem) increased for the first few depth values before decreasing. These results can be explained as follows (Igel and Chellapilla, 1999): In early generations, the average fitness change was smaller than in later generations. As better solutions were found, they became progressively more brittle with most changes generating large deviations in behavior and resulting in large AC values. Since the average tree size grew during evolution, nodes with small depth values were sampled more frequently in the early generations and less frequently in later generations, see Fig. 2. This caused the AC curve to increase for the first depth values. In order to verify this theory, the samples from early generations were omitted from the calculation of the fitness distribution features. Upon omitting these samples, the effect could be removed or at least reduced

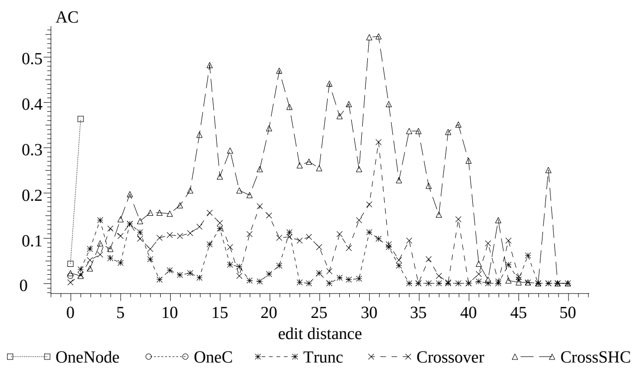

Figure 5: The average fitness change as a function of the edit distance between parent and offspring on the sunspot problem. The values of AC samples were limited to 100, i.e., AC $= \mathcal{E}(\min\{100, |F_o - F_{\{p\}}|\})$. In this problem a structural change of zero was possible, namely the replacement of a numerical constant with a different constant.

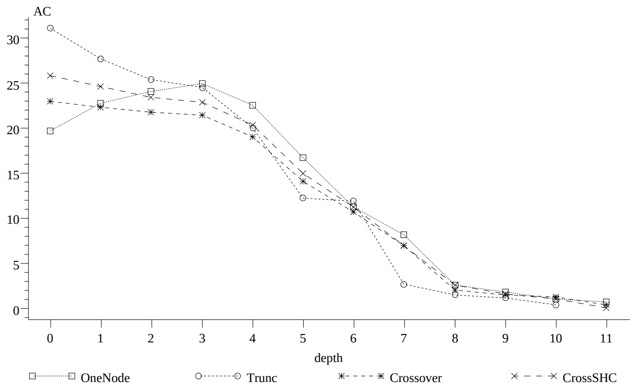

Figure 6: Artificial ant, depth-dependent average fitness change.

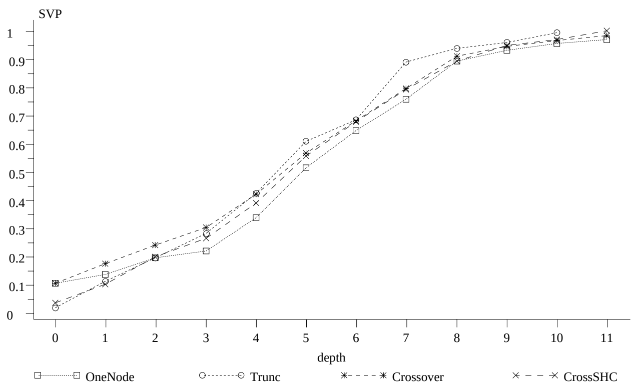

Figure 7: Artificial ant, depth-dependent probability of silent variations.

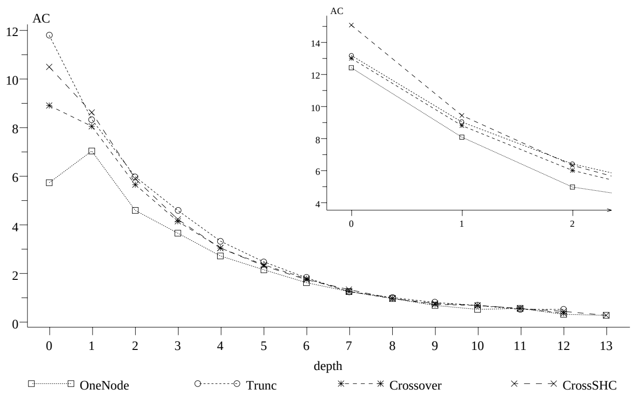

Figure 8: 6-multiplexer, depth-dependent average fitness change. The inline figure shows the AC for the lowest depths when changes generated in the first ten generations were omitted from the calculation.

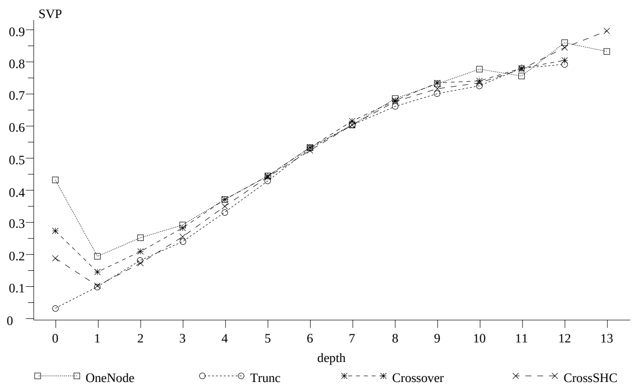

Figure 9: 6-multiplexer, depth-dependent probability of silent variations.

(the inline graphic in Fig. 8 shows an example). The early decrease in the silent variation curves in the case of the multiplexer problem can be explained in the same way.

The AC values were determined in part by the SVP: a high silent variation rate lead, on average, to small changes in the fitness space, whereas low SVP values typically resulted in larger AC values. Most of the silent variations were generated through changes applied to deeper nodes, wherein the likelihood of producing changes to intron subtrees (Banzhaf et al., 1998) increased. Since it is of interest to know whether there was a cause and effect relationship between the AC and SVP values, new AC curves were generated with the samples corresponding to the silent variations omitted. Upon removing the silent variation samples, the multiplexer problem AC curves qualitatively remained the same, whereas the artificial ant problem's AC curve became nearly constant (see Fig. 12) indi-

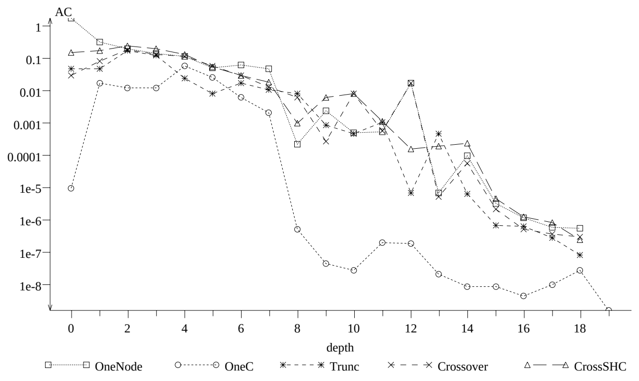

Figure 10: Sunspot prediction, log plot of depth-dependent average fitness change. The values of AC samples were limited to 100.

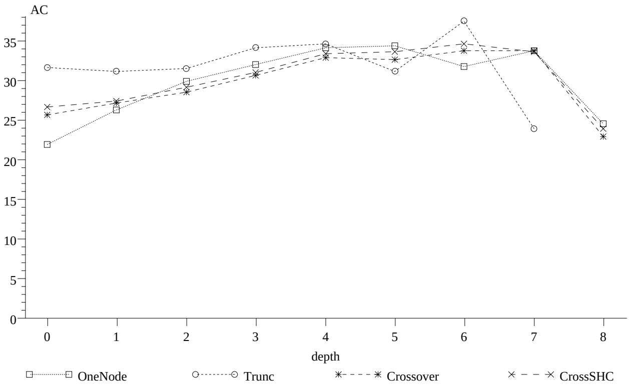

Figure 12: Artificial ant, depth-dependent average fitness change without silent mutations.

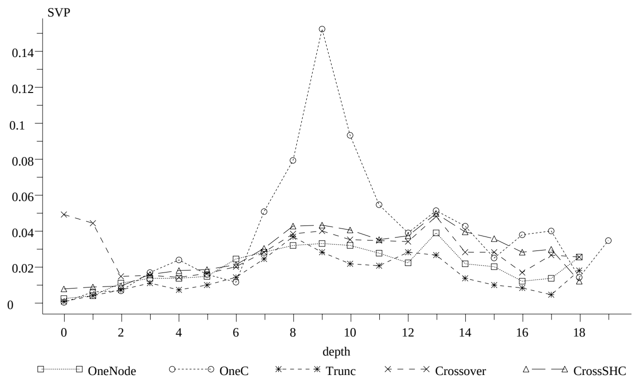

Figure 11: Sunspot prediction, depth-dependent probability of silent variations.

cating that the AC decrease in Fig. 6 was mostly determined by the silent variations shown in Fig. 7. The corresponding plot for the sunspot problem looks very much like Fig. 10 because of the low rate of silent variations.

On the sunspot time series forecasting task, the fitness changes induced by *Trunc* mutations at the root node were surprisingly small. Subsequent investigation showed that when the root node was replaced by a non-numerical terminal, the predicted values for the number of sunspots, x_t, was the same as the number of sunspots observed in previous years, i.e., x_{t-1}, x_{t-2}, x_{t-4}, or x_{t-8}. These one node predictors turned out to be moderately good predictors with the smaller time lag terminals performing better.

Not surprisingly, the *OneC* variation, which is the only non structure changing operator, shows a slightly different behavior from the other operators in Figs. 10 and 11. An application depth of zero is a special case: The parent tree consisted only of a single numerical constant that predicted the time series. Such constant solutions lead to rather large errors and therefore are not frequent at later generations. Not surprisingly, adding a zero mean Gaussian random number to such a constant predictor leads on average to a small fitness change.

Ito et al. (1998) concluded that depth-dependent crossover is better than traditional crossover because it works as "protection against destructive XO" and "accumulates building blocks". Our results show that making any operator depth-dependent changes the search behavior of the operator in question drastically, e.g. when nodes near the root are sampled more often the operator becomes more explorative. If a more explorative search is better suited for a problem this may be an alternative explanation why depth-dependent crossover works better.

A size constraint for the trees in the population was a principal parameter of the program evolution trials conducted in this study. In most algorithms, the maximum height is limited (Koza, 1992) and the height limit is chosen ad hoc. The analysis of the depth-dependency of AC for a problem class may provide better insight for picking a sensible depth limit.

This investigation contributes to the question, to which extent the *principle of strong causality*, which states that small changes in the genotype space should lead to small changes in the fitness space (Rechenberg, 1994), holds for controlled steps in the GP search space. The importance of this principle for evolutionary search is discussed e.g. in the work of Rechenberg (1994) and Sendhoff et al. (1997).

5 CONCLUSION

Hypothesis 2 which states "the longer the path to the root node the less the influence on the fitness" was found to be valid for all the three test problems. The differences between the depth of the application of a variation operator appears to be just as important as the choice of the operator in the case of the three test problems studied.

The rule of thumb "the larger the difference between parent and offspring the larger the fitness change" clearly holds for the ant and multiplexer problems, but for the sunspot problem only locally.

Acknowledgments

C. Igel would like to acknowledge support from the BMBF under grant SONN II 01IB701A0 and thank M. Kreutz and B. Sendhoff for fruitful discussions.

References

Aho, A. V., J. E. Hopcroft, and J. D. Ullman (1983). *Data Structures and Algorithms*. Addison-Wesley.

Angeline, P. J. (1997, 13-16 July). 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 (Eds.), *Genetic Programming 1997: Proceedings of the Second Annual Conference*, Stanford University, CA, pp. 9–17. Morgan Kaufmann.

Banzhaf, W., P. Nordin, R. E. Keller, and F. D. Francone (1998). *Genetic Programming – An Introduction; On the Automatic Evolution of Computer Programs and its Applications*. Morgan Kaufmann, dpunkt.verlag.

Chellapilla, K. (1997). 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 (Eds.), *Genetic Programming 1997: Proceedings of the Second Annual Conference*, Stanford University, CA, pp. 431–438. Morgan Kaufmann.

Chellapilla, K. (1998). Evolving computer programs without subtree crossover. *IEEE Transactions on Evolutionary Computation 1*(3), 209–216.

Fogel, D. B. and A. Ghozeil (1996). Using fitness distributions to design more efficient evolutionary computations. In *Proceedings of 1996 IEEE Conference on Evolutionary Computation*, Nagoya, pp. 11–19. IEEE Press.

Grefenstette, J. J. (1995). Predictive models using fitness distributions of genetic operators. In L. D. Whitley and M. D. Vose (Eds.), *Foundations of Genetic Algorithms 3*, Estes Park, Colorado, pp. 139–161. Morgan Kaufmann.

Harris, K. and P. Smith (1997). Exploring alternative operators and search strategies in genetic programming. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo (Eds.), *Genetic Programming 1997: Proceedings of the Second Annual Conference*, Stanford University, CA, pp. 147–155. Morgan Kaufmann.

Igel, C. (1998). Causality of hierarchical variable length representations. In *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, Anchorage, Alaska, pp. 324–329. IEEE Press.

Igel, C. and K. Chellapilla (1999). Fitness distributions: Tools for designing efficient evolutionary computations. In L. Spector, W. B. Langdon, U.-M. O'Reilly, and P. J. Angeline (Eds.), *Advances in Genetic Programming 3*, Chapter 9. MIT Press. in press.

Ito, T., H. Iba, and S. Sato (1998). Depth-dependent crossover for genetic programming. In *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, pp. 775–780. IEEE Press.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press.

Nordin, P. and W. Banzhaf (1995, 15-19 July). Complexity compression and evolution. In L. J. Eshelman (Ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms*, Pittsburgh, PA, pp. 310–317. Morgan Kaufmann.

O'Reilly, U.-M. (1997). Using a distance metric on genetic programs to understand genet ic operators. In J. R. Koza (Ed.), *Late Breaking Papers at the Genetic Programming 1997 Conference*, Stanford University, CA, pp. 188–198. Stanford University Bookstore.

O'Reilly, U.-M. and F. Oppacher (1994). Program search with a hierarchical variable length representation: Genetic programming, simulated annealing and hill climbing. In Y. Davidor, H.-P. Schwefel, and R. Männer (Eds.), *Parallel Problem Solving from Nature III*, Jerusalem, pp. 397–406. Springer-Verlag.

Rechenberg, I. (1994). *Evolutionsstrategie '94*. Werkstatt Bionik und Evolutionstechnik. Stuttgart: Frommann-Holzboog.

Rosca, J. P. and D. H. Ballard (1995, 15-19 July). Causality in genetic programming. In L. J. Eshelman (Ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms*, Pittsburgh, PA, pp. 256–263. Morgan Kaufmann.

Sankhoff, D. and J. B. Kruskal (Eds.) (1983). *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison-Wesley Publishing Company, Inc.

Sendhoff, B., M. Kreutz, and W. von Seelen (1997). A condition for the genotype-phenotype mapping: Causality. In T. Bäck (Ed.), *Proceedings of the Seventh International Conference on Genetic Algorithms*, MSU, East Lansing, MI, pp. 73–80. Morgan Kaufmann.

Zhang, K. and D. Shasha (1989). Simple fast algorithms for the edit distance between trees and related problems. *SIAM Journal of Computing 18*(6), 1245–1262.