Genetic Programming: Parametric Analysis of Structure Altering Mutation Techniques

Alan Piszcz and Terence Soule
Computer Science Department
University of Idaho
Moscow, ID 83844, USA
pisz3769@uidaho.edu,
tsoule@cs.uidaho.edu

ABSTRACT

We hypothesize that the relationship between parameter settings, specifically parameters controlling mutation, and performance is non-linear in genetic programs. Genetic programming environments have few means for a priori determination of appropriate parameters values. The hypothesized nonlinear behavior of genetic programming creates difficulty in selecting parameter values for many problems. In this paper we study three structure altering mutation techniques using parametric analysis on a problem with scalable complexity. We find through parameter analysis that two of the three mutation types tested exhibit nonlinear behavior. Higher mutation rates cause a larger degree of nonlinear behavior as measured by fitness and computational effort. Characterization of the mutation techniques using parametric analysis confirms the nonlinear behavior. In addition, we propose an extension to the existing parameter setting taxonomy to include commonly used structure altering mutation attributes. Finally we show that the proportion of mutations applied to internal nodes, instead of leaf nodes, has a significant effect on performance.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Experimentation, Performance.

Keywords

Genetic programming, parametric analysis, mutation.

1. INTRODUCTION

Parameter setting includes specification of values for population size, selection rate, crossover probability, mutation probability, reproduction rate, the number of generations and other environment attributes. In many problems, the parameters of genetic programming (GP) algorithms interact, making optimal value selection difficult [3]. This interaction leads to unpredictable behavior with respect to computational effort and may limit the ability to evolve a solution. If linear changes at the input parameter space translate into linear changes in performance, a transfer function or model would enable appropriate parameter selection. Luke and Spector studied crossover and mutation noting interesting nonlinearities in the results, they suggest that further analysis is needed [12]. Identification of specific nonlinearity in GP parameter settings will improve the researchers probability of choosing appropriate values. The GP community typically treats mutation as a secondary operator, when implemented it often has less emphasis than crossover and reproduction. Optimizing population size and crossover rates appear frequently as methods to improve GP solutions, whereas mutation rates are rarely considered. Feldt and Nordin identified the importance of population size and mutation in [4]. In the following experiments, we examine the effects of a single parameter related to mutation across three structure altering mutation techniques, and several mutation selection methods. We observe that nonlinearity is the behavior for the majority of the structure altering mutation technique (SAMT) cases examined.

GECCO'05, June 25–29, 2005, Washington, DC, USA.

2. BACKGROUND

Mutation is an operator for genetic programming that introduces diversity in the building blocks created during evolution. Due to a lack of research on mutation in GP, choosing mutation parameters is difficult, which is likely to lead to suboptimal parameter choices. Additionally, improved characterization of mutation parameters may further our understanding on the effects of mutation on the evolutionary process. It is typical to use

$$\frac{1}{size\ of\ individual}$$

for the mutation probability when it is possible to estimate or control the problem size.

Koza's view of mutation and crossover considers the value of mutation versus crossover. Koza introduces a framework for comparing mutation with crossover in GP [8] [9]. The framework describes the difference of the source of the new subtree introduced by mutation or crossover. Koza advises

that mutation is unlikely to be beneficial when compared to a large population and crossover. Few of Koza's early problems include mutation, and he states two reasons for the omission of mutation in problems investigated in his first book [6]. First, it is unlikely to lose diversity when using a sufficient population size, and therefore mutation is simply not needed in GP. Second, when the crossover operation occurs using endpoints in both trees the effect is "very similar to point mutation" [6].

Feldt and Nordin applied the design of experiments techniques from mathematical statistics to evaluate the effects of genetic programming parameters on three problems [4]. The findings show the population size was the most important parameter (with an effect score of 11.51), followed by mutation (5.21) and the number of generations (5.14). Mutation and large crossover probabilities had a positive effect on one of the three screening experiments. We attempt to confirm mutation has a similar positive influence on our problem with the SAMT.

In addition to effecting performance there is research suggesting that mutation can also effect rates of code growth. Luke considers modification point depth as a new model for genome growth in GP [11]. The findings suggest correlation of code growth with node depth bias. The survivability of the child improves with deep mutation and crossover points. Luke's meaning of deep node depth bias refers to locations near the terminal nodes. In this paper, we attempt to confirm that the relationship between the depth and size of mutation replacement trees and performance is nonlinear.

In addition to the rate of mutation, the ratio of internal and external mutations may have a significant effect on performance. Few researchers report this ratio. Exceptions do exist, Luke, Ross, and Burke et al. reference the internal/external mutation parameter in descriptions for the genetic programming environment problems in their work [12] [13] [1]. However none of the research describes or reports the effect of this parameter value. In this work we report preliminary results regarding the importance of this ratio.

3. PARAMETER SETTING TAXONOMY

To understand the effects of mutation we first need to define the different parameters that can be used to control mutation. Eiben et al. studied parameter settings for evolutionary algorithms (EA) and introduces a parameter setting taxonomy that defines attributes for EA [3]. Eiben et al. examines parameter settings for mutation, crossover, evaluation functions, replacement operator or survivor selection, and population size in evolutionary algorithms. Eiben et al. divided the research into parameter tuning and parameter control. Figure 1(a) shows the parameter setting taxonomy. Eiben et al. prefaces the discussion of mutation operators and their probabilities with "There have been several efforts to tune the probability of mutation in genetic algorithms. Unfortunately, the results (and hence the recommended values) vary, leaving practitioners in the dark." [3].

We propose an extension to Eiben et al. global taxonomy for parameter setting, starting at the "Parameter tuning" node shown in Figure 1(b). The proposed extensions only apply to genetic programming, and the underlined attributes indicate the three SAMTs studied with parameter analysis. *Parameter tuning*: The taxonomy extends this node by further decomposition of mutation methods. *Population proportion* determines the number of individuals selected for mutation. *Individual proportion* determines the number of nodes to mutate in an individual. *Selection bias* determines what members of the population are chosen for mutation, examples include the *best*, *worst*, or *random* individuals. *Individual proportion* is further subdivided into *intron* and *exon* ratios that specifically mutate introns or exons. *Depth ratios* that specifically mutate nodes based on sub-tree depth.

Variable shallow node mutation modifies 1 or more nodes at a controlled depth. An example of this is to randomly modify 5 nodes throughout an individual. Shallow indicates a depth of 0 or a single node mutated, instead of the typical variable size replacement depth.

Replacement tree depth controls the size of the replacement subtree randomly created and inserted by the mutation operation.

Internal, external ratio controls the selection of internal and external nodes of a tree for mutation.

4. STRUCTURE ALTERING MUTATION TECHNIQUES (SAMT)

The possible permutations, and interaction with other parameters make rationalization of the mutation rate parameter difficult to optimize for many problems. To address the evaluation of structure altering mutation we introduce three SAMT techniques, each modifies the genetic program tree in the mutation phase. Figure 2 depicts visual examples for the descriptions of SAMT 1, 2, and 3.

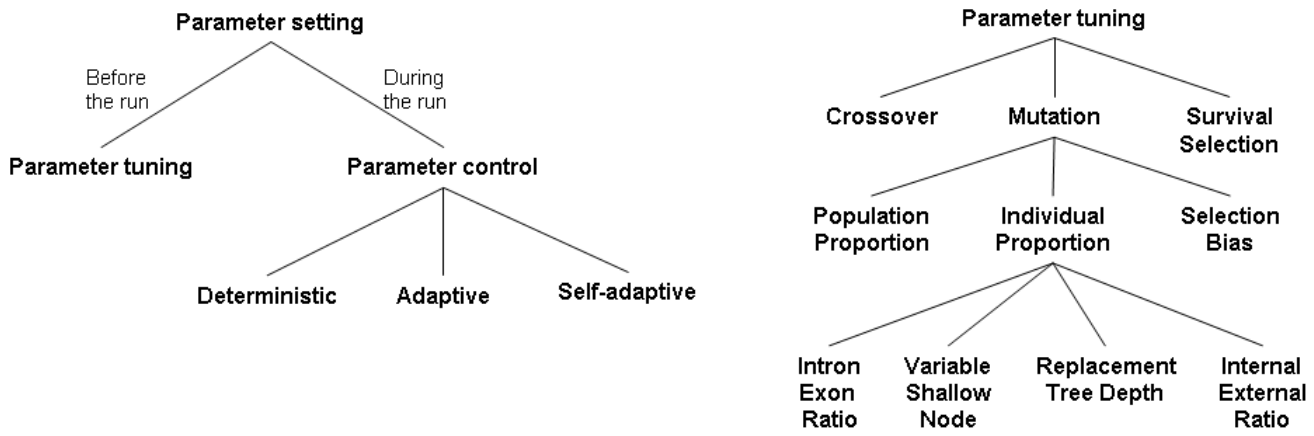
SAMT-1 implements shallow variable node count mutation (SVNCM) and operates by selecting 1 to n random locations in a tree, mutating a single node at each of the locations.

SAMT-2 controls the depth of replacement tree for mutation. Genetic programming algorithms often consider the replacement tree depth for mutation a tunable parameter. SAMT-2 mutates the tree structure using increasing mutation replacement tree depth.

SAMT-3 controls selection frequency for internal versus external nodes or non terminal nodes to mutate. Researchers often omit defining the internal/external (INEX) mutation rate parameter, instead relying on the default values.

5. THE MAX BINARY TREE PROBLEM

The goal of MAX binary tree problem (BTP) is to create a full tree of a given depth that produces the maximum value. This problem has several advantages: it is easy to understand, fitness evaluation is fast and simple, and the problem complexity is easy to influence by changing the tree depth. The MAX BTP problem is similar to the MAX problem introduced by Gathercole and Ross for researching the interaction of crossover and tree depth [5]. Gathercole and Ross relate the MAX problem to a broader class of GP issues where premature convergence leads to suboptimal solutions. Langdon and Poli extended the analysis of the MAX problem by Gathercole and Ross and developed a quantitative model that indicates the rate of improvement and shows that solution time grows exponentially with depth [10]. Langdon and Poli cite two reasons why GP finds the MAX problem difficult. (1) the tendency for GP populations to converge in the first few generations to suboptimal solutions from which they can never escape. (2) convergence to suboptima from which escape can only be made by slow



(a) Original

(b) Extended

Figure 1: Parameter Setting Taxonomy. The original taxonomy on the left is extended on the right starting at the 'Parameter tuning' node.

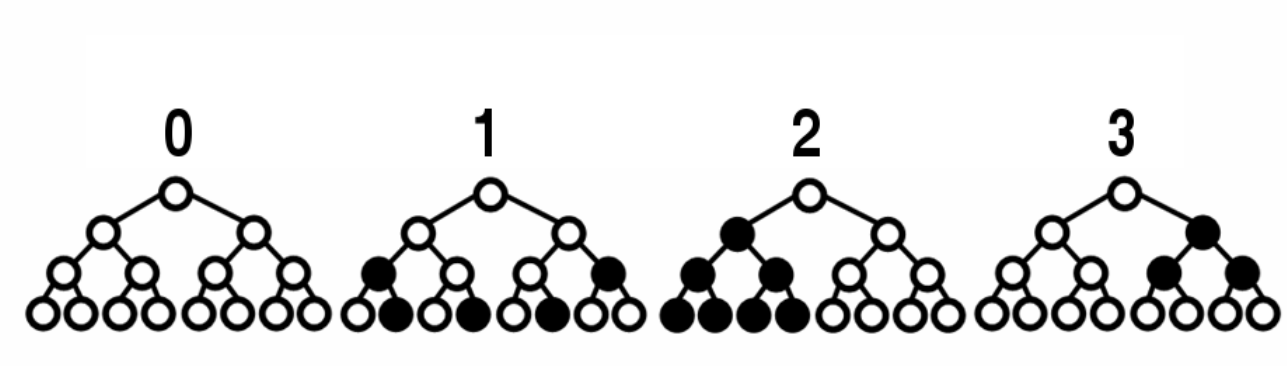


Figure 2: SAMT Examples. The SAMT technique number is above each tree. A white node indicates no mutation, a black node indicates a mutation. 0 is the original tree. 1 depicts 5 single node locations mutated. 2 depicts a mutation replacement tree depth of 2. 3 depicts mutation of the internal nodes, the INEX ratio is 1.0.

search similar to randomized hill climbing [10].

In the MAX BTP the root node is at level (or depth) 0. All internal nodes have a degree of three, supporting one parent and two child nodes, and the terminal nodes have a degree of one (the parent). The problem operators are $+$, $-$ and the terminals are ephemeral random constants 0 and 1. The maximum value for the tree is obtained when all internal nodes are $+$ and all terminals are 1 in which case the $maxvalue = 2^{depth}$. Tree evaluation only occurs once for each individual. Thus, fitness evaluation is fast compared to many other GP problems, such as symbolic regression that may require tens to hundreds of tree evaluations to measure the fitness of a single individual.

If we consider a full tree with binary values as follow: $+$, $-$ for internal node's, and 0, 1 for leaf nodes, the number of permutations for a given depth is:

$$Permutations = 2^{(Maximum\ Tree\ Size)}$$

For example, a depth 3 problem with 15 vertices has 256 possible permutations. Structural complexity is defined by Koza as a count of the number of points in a solution, this is the sum of functions and terminals. Structural complexity is one characteristic of a tree representation that can be measured [7]. The permutations nearly double with each level of depth and correlates with the structural complexity. The ratio of permutations to structural complexity increases exponentially. Through control of the tree depth, we can control the number of permutations, which determines the size of the search space, which determines the problem complexity. Thus, increasing the depth significantly increases problem complexity.

6. EXPERIMENT APPROACH

We present three mutation selection techniques to test for nonlinear performance by presenting an increasing linear input that mutates the tree structure. Each experiment performs a sweep of the parameter setting under study. Each problem run has a unique random seed assigned. We test three mutation selection modes: *none* (n), *random* (r) and *best* (b). *none* indicates no mutation mode or mutation phase. *random*, P percent of the population is randomly selected to undergo mutation. *best*, the best P percent of the population is selected for mutation. Table 1 lists the experiment parameters. Details of these initialization techniques and parameters can be found in [2]. The few differences in Table 1 parameter do not effect analysis which takes place within an SAMT experiment.

6.1 SAMT-1, Shallow Variable Node Count Mutation

The goal of this experiment is to determine if the relationship between the number of nodes mutated and the mean performance is linear or non-linear. Knowing the optimal mutation function will be useful for researchers using this type of mutation, and inform the community on optimal parameter settings for other types of mutation. This experiment consists of three trials, a control run with no mutation or *none*, then 5 repetitions for *random*, and *best* selection mutation mode (abbreviated n, r, and b in the figures). Each of the 5 repetitions consists of mutating 1, 2, 3, 4, and 5 nodes while iterating 100 separate runs differentiated by the random seed.

6.2 SAMT-2, Mutation Replacement Tree Depth

The goal of this experiment is to determine if changes in the mutation replacement tree depth cause nonlinear changes in the mean fitness of the population. In this experiment, we test for nonlinearity by measuring the mean fitness of the population while the depth of the mutation replacement tree is increased linearly. We test three mutation selection modes including *none*. The replacement tree depth is tested with a range of values from 0 to the maximum tree depth (total replacement). The subtree created is the result of the same method used for the initial population at generation 0. We have chosen to continuously generate new replacement trees until one meets the maximum depth criteria. The new tree is tested for compliance with the maximum tree depth limit. If a new individual exceeds the depth limit another subtree is generated. For example in lilgp, a common GP engine used for these experiments the default behavior omits the mutation if the total tree depth exceeds the maximum. If the keep_trying flag is enabled lilgp will continuously search for a replacement until one meets the overall depth requirement. By default, lilgp also generates the new random subtree with a default depth range of 0 to 4 [2]. This depth range represents 1 to 15 new nodes for the subtree in the binary tree problem. For example, depth 0 results in a single node replacement, depth 1 results in 3 nodes undergoing replacement, etc. For each of the replacement tree sizes we measure the resulting mean population fitness.

6.3 SAMT-3, Internal/External Mutation Selection Ratios

The goal of this experiment is to understand if a linear input change in the INEX mutation ratio has a linear effect on the mean fitness of population. This experiment employed parametric control of the INEX mutation ratio while measuring the resulting computational effort to evolve a solution. We test three mutation selection modes including *none*. Each experiment increased the problem depth and corresponding problem complexity. Each data point presented in the following graphs is the average of 100 runs. The INEX ratio varies from 0.0 to 1.0 in steps of 0.01 Thus, each plot line is composed of 10100 runs. For each problem depth, the CEAIS value computes the effort expended to evolve a solution. The following terms and calculations determine the minimum computational effort average for each acceptable solution. Mean, mean generation for an acceptable solution across 100 runs. Average Computational Effort (ACE) based on mean generation for all 100 runs. Number of Acceptable Solutions (NS). ACE counts generation 0 by adding 1 to the final generation value.

$$ACE = (MeanGeneration + 1) * PopulationSize$$

Computational Effort Average Individual Solution (CEAIS) is:

$$CEAIS = \frac{ACE}{NS}$$

Low values of CEAIS indicate the minimum average computational effort per solution that corresponds to an optimal configuration of parameter settings. A prior experiment determined optimal crossover and mutation rates for SAMT-3.

7. RESULTS

Table 1: Experiment Parameters

Parameter	SAMT-1	SAMT-2	SAMT-3
Objective	2^{Depth}	2^{Depth}	2^{Depth}
Terminal Set	$0, 1$	$0, 1$	$0, 1$
Functional Set	$+, -$	$+, -$	$+, -$
Fitness Cases	1	1	1
Fitness	Value of tree	Value of tree	Value of tree
Population Size	10000	1000	4000
Max Generations	1000	1000	1000
Initialization Method	Half & Half	Full	Full
Initialization Depth Ramp	2-6	7	$maxdepth - 1$
Maximum Tree Depth	8	8	3,4,5,6,7,8
Crossover selection mode	fitness	fitness	fitness
Crossover rate	$0.9 - MR$	0.89	optimal
Reproduction selection	fitness	fitness	fitness
Reproduction rate	0.1	0.1	0.1
Mutation selection mode	(n),(r),(b)	(n),(r),(b)	(n),(r),(b)
Mutation rate	(r,b)=0.01	(r,b)=0.01	optimal
(MR)	0.11,0.21,0.31	0.11, 0.21, 0.31	for each problem size
Internal mutation ratio	0.9	0.9	variable
External mutation ratio	0.1	0.1	variable

The following subsections give a brief description of experiment results. The legend in Figures 3 and 4 represent the four population selection mutation rates.

SAMT-1, we examine four mutation rates applied to a population size of 10000. The mutation rates of 0.01, 0.11, 0.21 and 0.31, affect only the *best* individuals for mutation. The four rates select the *best* 100, 1100, 2100 and 3100 individuals of the population. Results for the *none* selection mode, and for the *random* mutation selection mode produced no acceptable solutions and these data are subsequently omitted from SAMT-1 results. The numbers of nodes mutated at each mutation rate are 1, 2, 3, 4 and 5 are selected independently. The maximum value that can be obtained with a depth of 8 is 256, this represents all 255 internal nodes containing a + and all 256 external nodes contain a 1. Figure 3(a) and 3(b) show the results of the number nodes mutated for 0 to 5, the 0 indicates no mutation. Figure 3(a) plots the mean population fitness against the number of nodes mutated. The deleterious impact to fitness beyond 2 nodes is significant and severely limits mean fitness the population achieves.

Figure 3(b) plots the change in mean population fitness between each interval of the number of nodes mutated. $y - axis$ values above 0 indicate improvement over the previous number of nodes mutated, values below 0 indicate a reduction mean fitness.

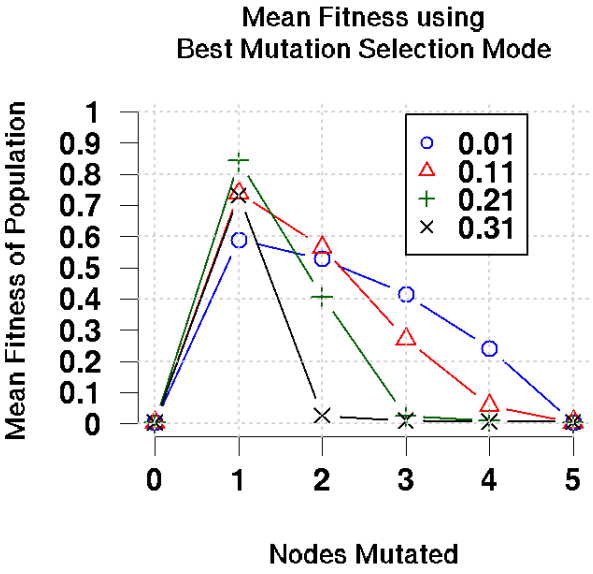
For SAMT-2, we examine four mutation rates applied to a population size of 1000. The four rates randomly select 10, 110, 210 and 310 individuals for mutation. The mutation rates of 0.01, 0.11, 0.21 and 0.31, effect only the *best* individuals for mutation. Results for the *random* mutation selection mode produced no acceptable solutions, and have been omitted from SAMT-2 results. Figure 4(a) and (b) indicate no mutation mean population fitness by the $x - axis$ (n) label. Figure 4(a) and (b) show the impact of controlled replacement tree depth mutation. In Figure 4(a) the mean fitness improves from a depth 0 or size 1 replacement tree through a depth 3 replacement tree. Fitness decreases from depth 2 through depth 8. Figure 4(b) shows the relative change in fitness for each depth replacement tree. All positive contributions occur at depths of 0 and 1. Further increases in depth cause a negative fitness influence on the population relative to the previous value. Mutations beyond a depth of 4 evolved no acceptable solutions.

In the SAMT-3 experiment, five increasingly complex versions of the MAX binary tree problem had replacement tree depths ranging from 3 to 8. Due to space limitations of this paper, only the depth 6 results appear. Figure 5(a) plots the CEAIS values for the MAX BTP of depth 6 against the internal/external selection ratio. Figure 5(b) shows the *best* CEAIS decreasing from 2402 to 1084 over the internal mutation rate range. *None* has a CEAIS value of 1830 and outperforms *random* across the entire internal mutation rate range. The *random* selection mode has CEAIS higher then *none* and has increasing CEAIS as internal mutation rate increases. Note the observation for this even depth problem is that the slope of the *best* CEAIS decreases with increasing internal mutation rate. Table 2 summarizes results from each experiment.

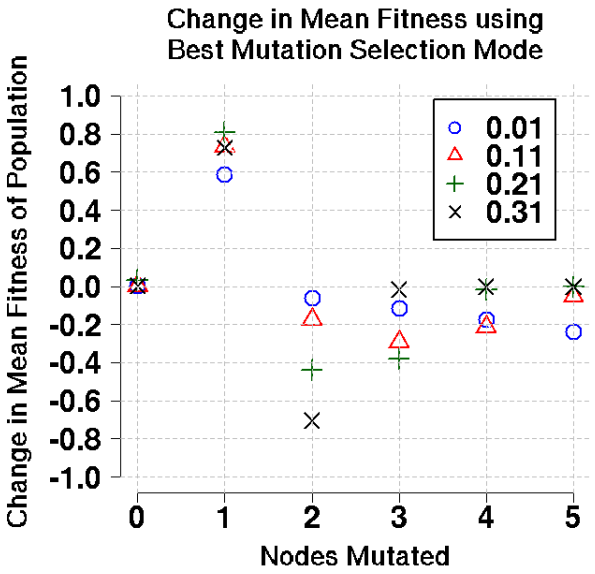
Table 2: SAMT-3 Experiment Summary. Mean of *best* (MB) CEAIS values, and Standard deviation (SD) of *best* CEAIS values.

Depth	MB	SD
3	7	0.8
4	40	7.3
5	161	24.6
6	1827	415.4
7	12024	2023.2
8	8638296	10639.0

The following observations concern the *best* mutation se-

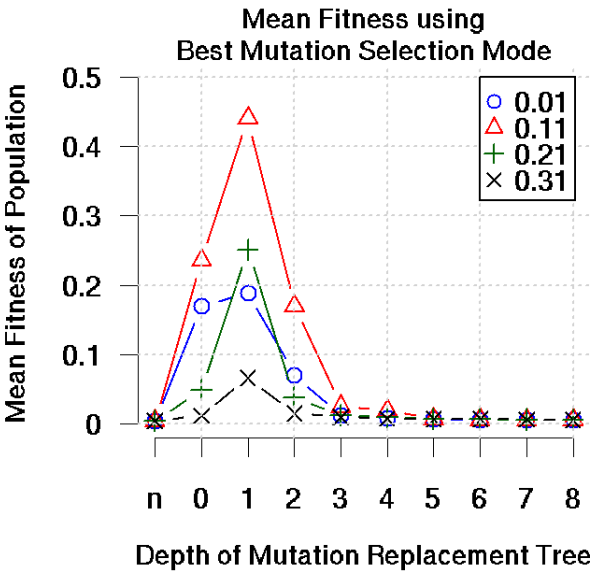


(a) Absolute Scale of Mean Fitness versus Number of Nodes Mutated

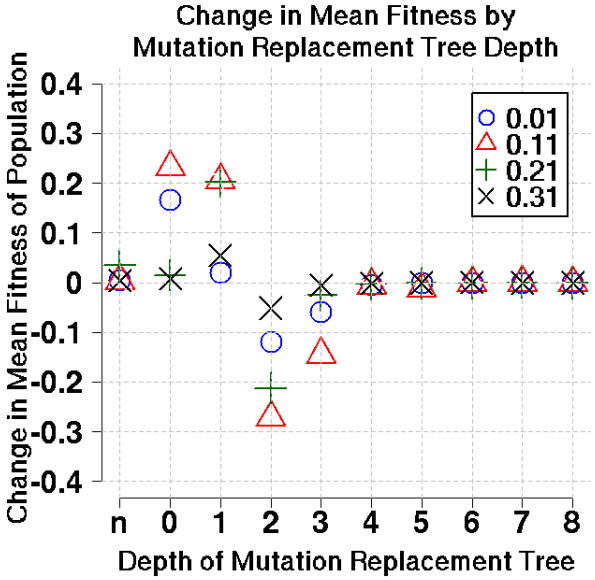


(b) Relative Change in Mean Fitness versus Number of Nodes Mutated

Figure 3: Shallow variable node count mutation results

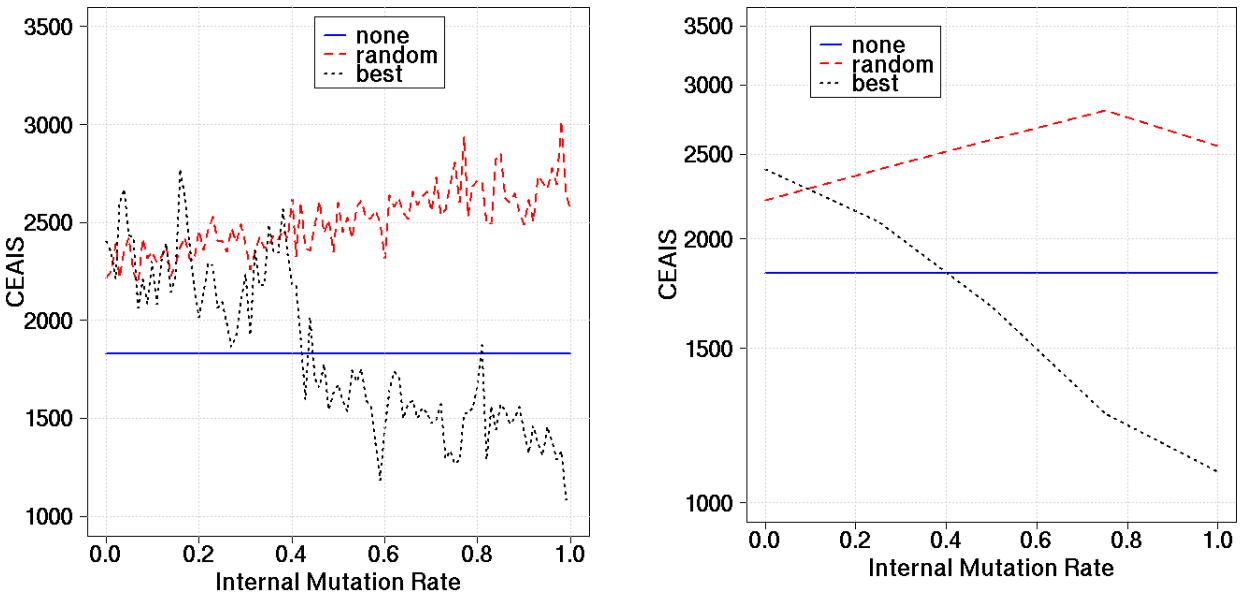


(a) Absolute Scale of Mean Fitness versus Depth of Mutation Replacement Tree



(b) Relative Change in Mean Fitness versus Depth of Mutation Replacement Tree

Figure 4: Variable depth mutation replacement tree results



(a) Depth 6, CEAIS versus ratio of mutation at non-terminal nodes, Linear-Log Plot

(b) Depth 6, CEAIS versus ratio of mutations at non-terminal nodes, Linear-Log Plot

Figure 5: Depth 6 internal/external mutation selection rate versus CEAIS

lection mode in the cases examined above. For the odd depth MAX BTP tested, we note the internal mutation ratio range from 0.0 to 1.0 produces CEAIS values that decrease by 100%. The payoff for identification of the optimal INEX value needs careful balance with the effort of discovery. For the even depth problems we note a trend that identifies higher internal mutation rates providing significantly lower CEAIS values than low internal mutation ratio rates. Even depth problems show a consistent reduction of CEAIS values as the internal mutation ratio increases from 0.0 to 1.0. The even depth problems also have a higher CEAIS variance across the entire mutation rate range as a fraction of the mean than the odd depth problems. In this situation, it indicates the improvement with decreasing CEAIS values as the mutation rate changes from 0.0 to 1.0.

7.1 Observations

In SAMT-1 we found that the *best* mutation mode performed significantly better than *random* and *none* showing that mutation can improve performance. Mutating the best individuals results in the better performance, possibly because those individuals are mostly likely to affect the path of evolution. The critical points for mutation in four separate rate trials show the beneficial effects of mutation occur only for depth 1. Higher mutation rates accelerate the deleterious impact of multi-node mutation, although mutating two nodes is still better than none. *Mutation boost* describes the significant improvement in evolved *best* solutions over the other two selection modes, *none* and *random*.

SAMT-2, 22.2% is the optimal tree replacement depth for the binary tree problem with a depth of 8. Depths 0 to 2 perform better than no mutation.

SAMT-3, Selection of the best individuals and performing mutation on INEX ratio of 50/50 should provide a consistent and neutral effect on the resulting computational effort. Significant variance is noted for the *best* mutation selection mode for even problems between internal mutation rates below and above 50% or 0.5. This suggests deleterious effects of mean population fitness for the MAX BTP using high external mutation rates. Optimal parameter tuning for the INEX rate leads to improved performance with even depth MAX BTP problems. For odd depth problems we observe improvements as the INEX rate approaches ≈0.60. When complexity reaches depth 7, a decrease of CEAIS shows higher INEX rates outperform lower INEX rates.

8. SUMMARY

Parameter settings significantly influence the effectiveness of the structure altering mutation techniques evaluated and generally show a nonlinear response to population fitness and computational effort. Each experiment discovered unique genetic programming behavior using parametric analysis of values in isolation. The key results from the experiments:

- SAMT-1: The behavior of mutating 1 to 5 nodes is near linear when 1% to 11% of the population is mutated. When more of the population is subjected to mutation (21% and 31%) the relationship between the number of nodes mutated and the performance becomes increasingly nonlinear. The rate of decrease of mean population fitness increases with mutation rate as more nodes are mutated.

- SAMT-2: Varying the replacement tree depth produced a nonlinear response as measured by the mean population fitness. Performance with SAMT-2 peaks at approximately 22% of the maximum tree depth across all four mutation rates.

- SAMT-3: Increasing the rate at which non-terminal nodes are mutated results in a linear response of computational effort for the *random* and *best* mutation

selection modes. The computational effort correlation with increasing the INEX rate was negative for *random* and positive for *best* mutation selection modes.

- SAMT-3: As the internal mutation selection ratio increases the computational effort decreases. This indicates that higher internal mutation selection rates are optimal for the *best* mutation selection mode on the MAX BTP. The average response of the population to *random*, *best* and *none* are near linear with respect to the internal mutation parameter value.

9. REFERENCES

[1] E. K. Burke, S. Gustafson, and K. Kendall. Diversity in genetic programming: An analysis of measures and correlation with fitness. *IEEE Transactions on Evolutionary Computation*, 8(1):47–62, 2004.

[2] B. P. Douglas Zongker. lil-gp 1.0 user's manual. Technical report, Michigan State University, 1995.

[3] A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transations on Evolutionary Computation*, 3(2):124–141, 1999.

[4] R. Feldt and P. Nordin. Using factorial experiments to evaluate the effect of genetic programming parameters. In R. Poli, W. Banzhaf, W. B. Langdon, J. F. Miller, P. Nordin, and T. C. Fogarty, editors, *Genetic Programming, Proceedings of EuroGP'2000*, volume 1802 of *LNCS*, pages 271–282, Edinburgh, 15-16 Apr. 2000. Springer-Verlag.

[5] 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, July 1996. MIT Press.

[6] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, pages 106–107. MIT Press, Cambridge, MA, USA, 1992.

[7] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[8] J. R. Koza, F. H. Bennett III, D. Andre, and M. A. Keane. *Genetic Programming III: Darwinian Invention and Problem Solving*. Morgan Kaufmann Press, May 1999.

[9] J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu, and G. Lanza. *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers, July 2003.

[10] 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*, pages 222–230, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann.

[11] S. Luke. Modification point depth and genome growth in genetic programming. *Evolutionary Computation*, 11(1):67–106, 2003.

[12] S. Luke and L. Spector. A comparison of crossover and mutation 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: Proc. of the Second Annual Conf.*, pages 240–248, San Francisco, CA, 1997. Morgan Kaufmann.

[13] B. J. Ross. Searching for search algorithms: Experiments in meta-search. Technical report, Brock University, December 2002. Technical Report CS-02-23.