Genetic Programming: Analysis of Optimal Mutation Rates in a Problem with Varying Difficulty

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

Abstract

In this paper we test whether a correlation exists between the optimal mutation rate and problem difficulty. We find that the range of optimal mutation rates is inversely proportional to problem difficulty. We use numerical sweeps of the mutation rate parameter to probe a problem with tunable difficulty. The tests include 3 different types of individual selection methods. We show that when problem difficulty increases, the range of mutation rates improving performance over crossover alone narrowed; e.g. as the problem difficulty increases the genetic program becomes more sensitive to the optimal mutation rate. In general, we found that the optimal mutation rate across a range of mutation types and level of difficulty is close to 1/C, where C is the maximum size of the individual.

Introduction

Mutation in evolutionary computation and specifically genetic programming (GP) is frequently treated as a secondary or background operator. If mutation is not excluded, it is often given less emphasis than crossover. However, it has been shown that mutation can significantly improve performance when combined with crossover (Banzhaf, Francone, & Nordin 1996) (Poli & Langdon 1997). Investigating the effect of the mutation rate as a positive, neutral or negative factor for variable length representation GP is limited compared with genetic algorithm (GA) research.

In GP, different mutation implementations make it difficult to compare the published results. This inhibits the ability to select the optimal mutation rate (OMR). Factors that make it difficult or impossible to determine the OMR by reviewing published results include:

- Varying solution depth and size.
- Mutation implementation and control parameters.
- Interaction with crossover and other phases of genetic programming.
- Limited reporting of mutation in published research.
- Limited or lacking statistics and metrics describing population response to mutation.
- Mutation parameter description and justification.

This paper extends our understanding of mutation through analysis of experimental results for GP culminating in the following findings:

- The OMR is approximately $\frac{1}{C}$ across a range of problem difficulties.
- The OMR is independent of mutation selection method.
- The sensitivity to the mutation rate increases when problem difficulty increases.

Background

First, we review previous research addressing mutation in GP, followed by a review of difficulties in understanding mutation research. Next, we review related GA research efforts that have similar goals to this paper.

Banzhaf, Francone and Nordin applied population mutation rates of 5%, 20%, 50%, 80% and significantly improved the generalization capabilities of their genetic program (Banzhaf, Francone, & Nordin 1996). The findings reveal that as the problem difficulty increased, the improvement due to higher mutation rates increased for two of three data sets studied. Generalization improvements increased the probability of creating outstanding runs. The OMR for all three test problems (Gaussian 3D data set, phoneme recognition data set and the IRIS machine learning image training set) was 50%. These results show that mutation can be beneficial, although in this case the limited number of mutation rates tested makes it difficult to identify the precise OMR. There was no description of the mutation method implementation.

Poli and Langdon tested new crossover techniques using six different single-point mutation probabilities per node: 0, 1/256, 1/128, 1/64, 1/32 and 1/16 (Poli & Langdon 1997). Point mutation was applied to every individuals in the population. The results suggest that crossover without mutation causes premature convergence. They show that it was five times easier to generate a solution for the 4-parity problem with a 1/256 mutation probability applied to each node. They found the OMR is approximately 2 divided by the size of the tree. For a problem depth of six, the OMR provided 10 to 15 times the performance of a standard genetic program. Therefore, these results show that mutation is beneficial. However, it is unclear if the rates they selected were optimal.

Luke and Spector varied crossover, mutation and population size simultaneously. They studied 6 common problems from the literature and found peak performance with a mutation ratio between 20% and 30% of the population (Luke & Spector 1997) (Luke & Spector 1998). The implementation was point mutation where a random subtree is replaced with a new random tree.

Koza introduced a framework for comparing mutation with crossover in GP (Koza *et al.* 1999) (Koza *et al.* 2003). This suggests mutation is unlikely to be beneficial when compared to a large population with crossover. Few of Koza's early experiments include mutation. Koza states two reasons for the omission of mutation in the majority of problems investigated in his first book (Koza 1992a). First, it is rare to lose diversity when using a sufficient population size; 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."

Mutation was often overlooked in early GP research. Today it receives more attention. Evidence strongly suggests that efficiency in GP requires determining the OMR. In general, mutation can have three effects: *efficient mutation* (mutation at an optimal or near optimal rate) improves the probability of generating a solution, *inefficient mutation* adds another processing phase to the GP environment and causes deleterious changes to the population and *benign mutation* may not improve the results but still consume processing resources. The previous results suggest that finding an OMR depends on the type of mutation and the difficulty of the problem. In this paper, we focus on OMR and is relationship to problem difficulty.

In contrast to GP, there has been extensive research in the GA community regarding mutation. Therefore, we examine that literature for further insights. The GA research community has investigated OMR for problems while considering: population size, adaptive mutation rates, improved statistical random distributions and mixing of mutation modes. Of the many papers studying mutation in GA, the following are most relevant to this paper.

The GA research community typically specifies the mutation rate μ as 1/n, where n is the length of the genome (Koza 1992b). In most implementations, mutation is applied with μ probability to each bit in the genome. Sasaki and Nowak describe *localization* as "the ability for a quasispecies to adapt to the peaks in the fitness landscapes (Sasaki & Nowak 2003)." Sasaki and Nowak describe the rate of less than 1/n as a requirement to satisfy the equilibrium quasispecies distribution for a particular region of sequence space. A value greater than 1/n causes delocalization, this results in any finite population wandering in sequence space (Sasaki & Nowak 2003).

Heinz Mühlenbein and Dirk Schlierkamp-Voosen performed theoretical analysis of selection, mutation and recombination. They found that: a) mutation is more efficient with crossover, b) the efficiency of the mutation operator is dependent upon the mutation rate and c) mutation in large populations is inefficient (Mühlenbein & Schlierkamp-Voosen 1995).

Stanhope and Daida use parameter sweeps to evaluate a mutation-rate strategy. The mutate-rate strategy is variable between individuals within a given generation based on the individual's relative performance for the purpose of function optimization (Stanhope & Daida 1997).

Eiben et al. studied parameter settings for: mutation, crossover, evaluation functions, replacement operator or survivor selection and population size in evolutionary algorithms (Eiben, Hinterding, & Michalewicz 1999). Eiben et al. split the research into parameter tuning and control. Eiben et al. introduced mutation operators influence in GA with: "*There have been efforts to tune the probability of mutation in GA's. Unfortunately, the results (and hence the recommended values) vary, leaving practitioners in [the] dark.*" Extensions to improve both the structure of experiments and reporting of mutation operations in GP are made to Eiben's et al. global taxonomy for parameter setting in (Piszcz & Soule 2005).

Our goal is to test for the existence of correlation between the level of difficulty and the OMR. We attempt to increase the knowledge of OMR and its relationship as problem difficulty varies. The GA community has studied many aspects of mutation, however little direct evidence exists that these results apply to GP.

The MAX Binary Tree Problem

The goal of the MAX Binary Tree Problem (BTP) is to create a full tree of a given depth that produces a tree with the maximum value. The advantages of this problem are: 1) it is easy to understand, 2) fitness evaluation is fast and simple and 3) the problem difficulty is easy to change by changing the tree depth. Gathercole and Ross describe the MAX problem that is similar to MAX BTP with variations for researching the interaction of crossover and tree depth (Gathercole & Ross 1996). Langdon and Poli extend the analysis of the MAX problem by Gathercole and Ross and develop a quantitative model that indicates the rate of improvement. This shows that solution time grows exponentially with depth (Langdon & Poli 1997). Langdon and Poli's work confirms that the BTP is a tunably difficult problem suitable for our research purposes.

The root node is at level (or depth) 0. All internal nodes have a degree of 3, supporting 1 parent and 2 child nodes and the terminal nodes have a degree of 1 (the parent). The problem functions are $+$, $-$ and the terminals used 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, where the maxvalue is:

$$maxvalue = 2^{depth}$$

Tree evaluation occurs once for each individual. Therefore, this allows a very fast fitness evaluation when compared to other GP problems, such as symbolic regression that may require tens to hundreds of tree evaluations to measure the fitness of a single individual.

The tree depth is variable, this allows us control the number of permutations. The tree depth control also determines the size of the search space, which determines the problem difficulty. Thus, increasing the depth significantly increases problem difficulty (Langdon & Poli 1997).

Experiment Approach

Two mutation implementations frequently appear in GP environments. The first is node based mutation, in which the mutation rate specifies the probability per node of the individual. The second method is tree based mutation, in which the mutation rate specifies the frequency that individuals are selected from the population for mutation. When applying mutation in GP, researchers frequently set the mutation rate to be $\frac{1}{C}$, where C denotes the size of the individual. The mutation rate $\frac{1}{C}$ is analogous to the rate preferred in fixed length GA but there is little direct evidence that it is appropriate for variable length GP.

Four experiments of increasing difficulty test the convergence of the genetic program on the MAX BTP. We perform a parameter sweep of the mutation rate and observe the computational effort. The parameter sweep of mutation rates include the ranges: 0.0001 to 0.001 with a step size of 0.0001, 0.001 to 0.01 with a step size of 0.001 and 0.01 to 1.0 with a step size 0.01. These parameter sweep ranges provide a balance between coverage of all possible rates and the time to complete the experiment. The population mutation rate or the percentage p_m of the population that undergoes mutation is tested for the mean computational effort over 100 runs. The mutation operation is subtree replacement where a randomly selected node is replaced by a new randomly generated subtree. The control case is no mutation.

Three mutation selection modes are considered:

- best (b) - the best p_m of the population is subjected to subtree mutation, each individual has one subtree mutated
- none (n) - indicates no mutation phase
- random (r) - p_m of the population is selected at random with uniform probability and subjected to mutation.

During each mutation phase the number of individuals selected for mutation is:

$$I_m = Mp_m$$

where I_m denotes the number of individuals mutated, M the population size. Each selected individual has one subtree mutated. In the case of *random*, the same individual may be selected from the population more than once, leading to several subtree mutations in the same individual.

The difficulty of the MAX BTP is varied by using tree depths of 4, 5, 6 and 7. We identify the OMR by measuring the average computational effort to evolve a solution. The optimal population size was empirically determined using a parameter sweep of population size for each depth in a prior experiment. Luke performed a similar parameter sweep of population and mutation parameters in (Luke & Spector 1997) (Luke & Spector 1998).

The following terms and calculations define the minimum computational effort average for each optimal solution. **ACE** Average Computational Effort based on mean generation for all 100 runs. The mean generation describes the number of generations for a given test to produced an acceptable solution. To achieve an acceptable solution the GP must evolve an individual that meets the criteria defined by the fitness function at or prior to the maximum generation specified. **NS** Number of Acceptable Solutions found in 100 runs. The '+ 1' is included in the ACE equation to count generation 0:

$$ACE = (Mean\ Generation + 1) * Population\ Size$$

CEAIS Computational Effort Average per Individual Solution.

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

We use the following acronyms in the experiment results section.

- **IOMRN** Improvement Optimal Mutation Rate Normalized, the ratio of the CEAIS value for *best* and *random* mutation selection modes compared to *none*: $\frac{CEAIS_{none}}{CEAIS}$
- **MAXOW** Maximum Optimal Mutation Window, the mutation rate at the upper CEAIS value of the optimal range window.
- **MINOW** Minimum Optimal Mutation Window, the mutation rate at the lower CEAIS value of the optimal range window.
- **OW** OMR Window, the OMR window that minimizes computational effort defined as: $MAXOW - MINOW$.

Table 1 shows the parameters for the four experiments, E(n) references the experiment denoted by the value n.

Table 1: Experiment Problem Parameters

Objective	Maximum tree value
Terminal Set	0, 1
Function Set	$+, -$
Fitness Cases	1
Fitness	Evaluation of tree value
Population Size, E(1, 2, 3, 4)	40, 90, 350, 1000
Maximum Generations	1000
Initialization Method	Half and half
Initialization Depth Ramp	20 to 60 %
Maximum Tree Depth: E(1, 2, 3, 4)	4, 5, 6, 7
Maximum Tree Size	Controlled by depth.
Crossover	Selection mode = fitness
Crossover Rate	0.9
Reproduction	Selection mode = fitness
Reproduction Rate	0.1
Mutation Selection Mode	none, random, best
Mutation Rate Range E(1, 2, 3, 4)	none, 0.0001-0.001, 0.001-0.01, 0.01-1.0

Experiment Results

Figure 1 depicts the results for the experiment runs. The y value of the plot indicates the average computational effort per individual solution. The OMR is the point where the minimum CEAIS value occurs.

The results show an increasing sensitivity to mutation rate range as the problem difficulty increases. Increasing sensitivity is evident in Figure 1 by the increasing slope and by

Table 2: Experiment Result Summary

MODE	Depth	4	5	6	7
(n)	OMR	NA	NA	NA	NA
(n)	CEAIS	108	259	1464	NA
(n)	IOMRN	1.0	1.0	1.0	NA
(r)	OMR	0.030	0.002	0.0002	NA
(r)	CEAIS	41	212	1438	NA
(r)	IOMRN	2.7	1.2	1.0	NA
(r)	OW	0.032	0.003	0.0002	NA
(b)	OMR	0.060	0.011	0.020	0.003
(b)	CEAIS	30	150	363	7197
(b)	IOMRN	3.7	1.7	4.0	∞
(b)	MINOW	0.0001	0.0002	0.003	0.002
(b)	MAXOW	0.09	0.07	0.062	0.004
(b)	OW	0.0899	0.0698	0.059	0.002

the narrowing window of effective OMR rates shown by the thick gray arrows. The OW for the CEAIS curve is formed by the plotted points adjacent to the minimum CEAIS value. The results identify the OMR for at four levels of difficulty.

The improvement in time-to-solution and computational effort as measured by CEAIS shows that the *best* mutation selection mode provides a significant improvement compared with none and random. As problem difficulty increases the mutation selection methods of none and random produce fewer solutions, neither produces solutions for trees of depth 7 (see Figure 1. The performance improvement is shown in Table 2 and is relative to *none*.

In Figure 1 each subplot is marked with a thick gray arrow indicating the extent of the OW. Selection of the OW uses the following heuristics: 1) the minimum and maximum window points are at or below the average of the CEAIS values that occur at mutation rates of 0.1 through 1.0 and 2) represent the first or last inflection point that define a window with heuristic 1.

Experiment Summary

Table 2 shows as problem difficulty increases the OW narrows. This is shown by OMR and CEAIS values in the *best* rows of Table 2. To achieve optimal computational efficiency with complex problems, overestimation of the OMR may be more useful than underestimation. In the case of the depth 7 problem, selecting below an optimal mutation rate causes an increase in computational effort. However, above the OW mutation differences from 0.02 to 1.0 only decrease CEAIS values by approximately 1/2 an order of magnitude compared to the OMR. The OW is typically 6-9% which is a fairly small range.

For the *best* mutation selection method, the OMR decreased by more than an order of magnitude from 0.06 to 0.003 as the binary tree problem depth increased from 4 to 7. In the depth 7 binary tree problem, *none* and *random* produced no viable solutions. *Best* mutation selection mode achieved a peak of 94 acceptable solutions near the OMR. Figure 2 shows a plot of the resulting the OMR from the experiment and the typical mutation rate: $\frac{1}{C}$.

The plot shown in Figure 2 for the *typical* OMR is approximately fit by the following equation:

$$0.5336 * e^{(-0.7023 * Depth)}$$

The plot shown in Figure 2 for the *experimental* OMR is approximately fit by the following equation:

$$1.4234 * e^{(-0.8389 * Depth)}$$

The sensitivity to the proper mutation rate increases as difficulty increases. Figure 2 shows the mutation range limits at each of the four levels of difficulty. The triangle symbol designates the *typical* mutation rate of $\frac{1}{C}$ and the circle symbol indicates the measured *experimental* mutation rate.

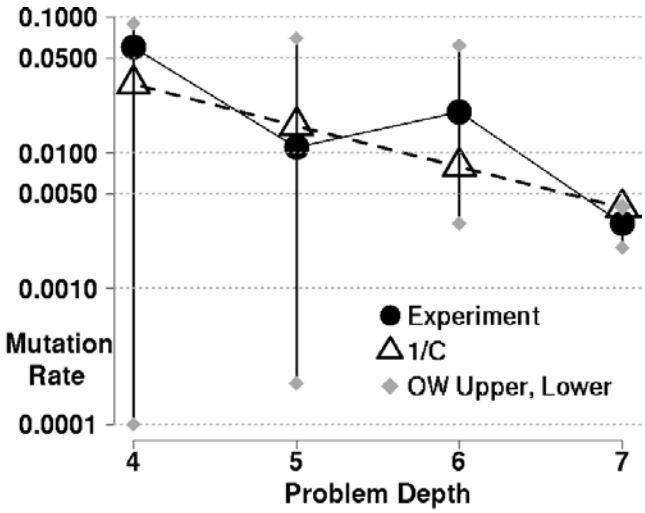

Figure 2: Mutation range summary linear/log plot. The experimental mutation rate (circle) is similar to 1/C. OW decreases with increasing difficulty.

Conclusion

The MAX BTP provided controlled problem difficulty to determine the OMR as a function of problem difficulty and the individual selection method for mutation. We have shown that the OMR for several mutation selection modes remain consistent for the four levels of problem difficulty tested. Experimental optimal mutation rates for the binary tree problem correlate with the typical mutation rate used in GA. In general, the range of mutation rates that produce acceptable solutions decrease as problem difficulty increases. This observation confirms the challenge researchers face when choosing effective mutation rates. Performing parametric mutation experiments provides detailed problem characterization compared to a limited number of fixed mutation rates for evaluation. The resulting CEAIS plots over a regular interval show that interpolation of too few mutation rates may cause omission of critical behavior changes. This research shows:

- The OMR diminishes and is inversely proportional to problem difficulty.

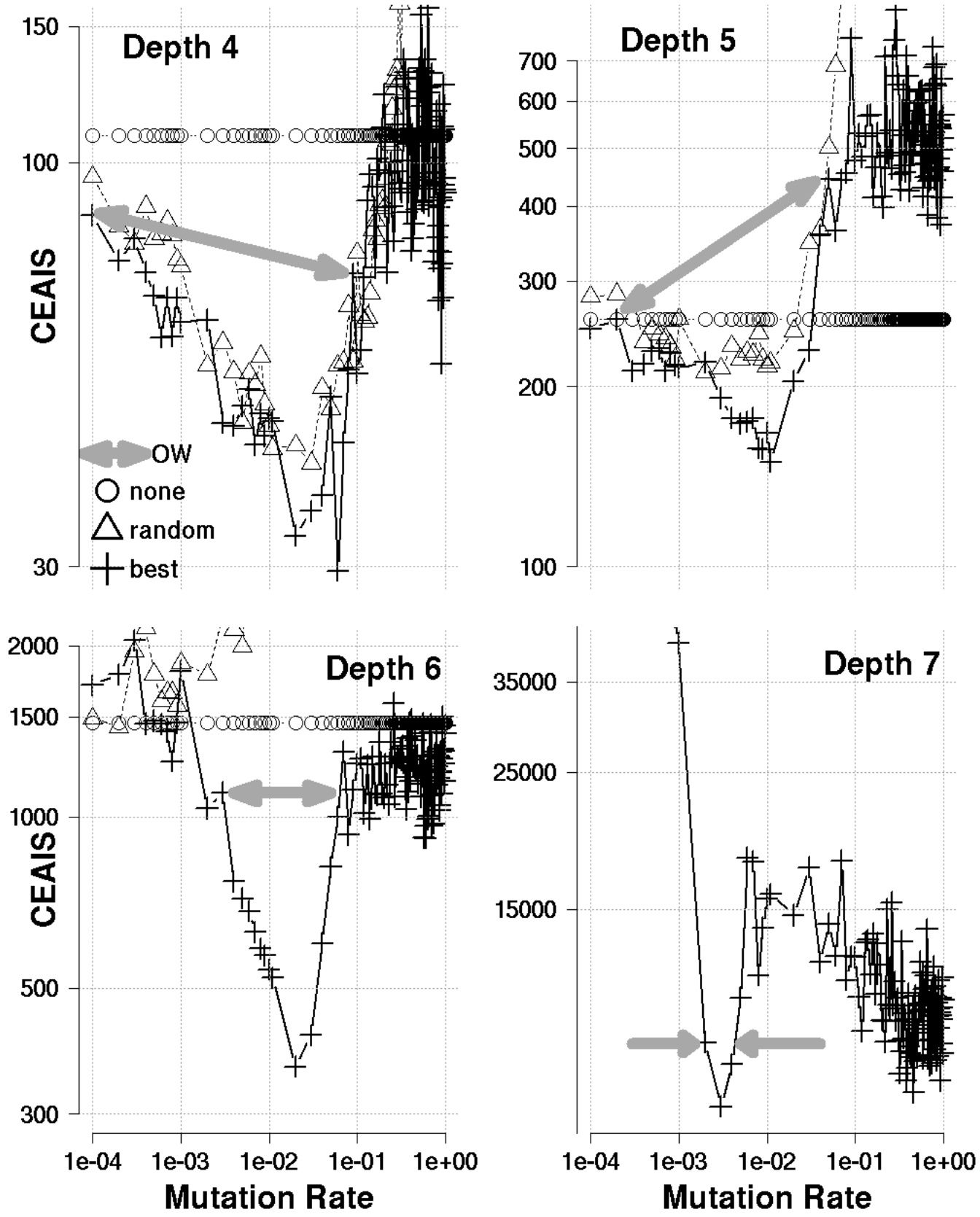

Figure 1: Experiment 1, 2, 3, 4, Depth 4, 5, 6, 7, CEAIS versus Mutation Rate, Log-Log. The plots show an increasing problem difficulty (left to right, top to bottom) with a decreasing OMR window.

- The results show significant performance improvements with mutation versus crossover alone, for problem depths 6 and 7.
- The range of efficient mutation rates narrow as difficulty increases.
- OMR decreases with increasing problem difficulty for the unimodal problem.

An interesting picture emerges. Mutation can improve performance, however the range of mutation rates that improve performance is fairly small and gets smaller as problems get more difficult. This confirms researchers' observations that it is very hard to determine a good mutation rate. The optimal rate is usually near 1/C, so we have a reasonable initial value to chose for the mutation rate.

In the binary tree problem of problem depth 7 we show extremely low mutation rates are suboptimal. High mutation rates may not always be computationally efficient; however, in the problem depth 7 case more acceptable solutions were evolved with high mutation rates than mutation rates below 1e-03.

Open research questions: Is there a correlation between increasing difficulty and the diminishing returns of mutation with larger populations for GP as Heinz Mühlenbein and Dirk Schlierkamp-Voosen (Mühlenbein & Schlierkamp-Voosen 1995) describe for GA? Is it possible to model and predict where a large enough population size provides sufficient diversity to reduce or eliminate the need for mutation?

References

Banzhaf, W.; Francone, F. D.; and Nordin, N. 1996. The effect of extensive use of the mutation operator on generalization in genetic programming using sparse data sets. In Voigt, H.-M.; Ebeling, W.; Rechenberg, I.; and Schwefel, H.-P., eds., *Parallel Problem Solving from Nature IV, Proceedings of the International Conference on Evolutionary Computation*, volume 1141 of *LNCS*, 300–309. Berlin, Germany: Springer Verlag.

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

Gathercole, C., and Ross, P. 1996. An adverse interaction between crossover and restricted tree depth in genetic programming. In Koza, J. R.; Goldberg, D. E.; Fogel, D. B.; and Riolo, R. L., eds., *Genetic Programming 1996: Proceedings of the First Annual Conference*, 291–296. Stanford University, CA, USA: MIT Press.

Koza, J. R.; III, F. H. B.; Andre, D.; and Keane, M. A. 1999. *Genetic Programming III: Darwinian Invention and Problem Solving*. Morgan Kaufmann Press.

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

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

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

Langdon, W. B., and Poli, R. 1997. An analysis of the MAX problem in genetic programming. In Koza, J. R.; Deb, K.; Dorigo, M.; Fogel, D. B.; Garzon, M.; Iba, H.; and Riolo, R. L., eds., *Genetic Programming 1997: Proceedings of the Second Annual Conference*, 222–230. Stanford University, CA, USA: Morgan Kaufmann.

Luke, S., and Spector, L. 1997. A comparison of crossover and mutation in genetic programming. In Koza, J. R.; Deb, K.; Dorigo, M.; Fogel, D. B.; Garzon, M.; Iba, H.; and Riolo, R. L., eds., *Genetic Programming 1997: Proc. of the Second Annual Conf.*, 240–248. San Francisco, CA: Morgan Kaufmann.

Luke, S., and Spector, L. 1998. A revised comparison of crossover and mutation in genetic programming. In Koza, J. R.; Banzhaf, W.; Chellapilla, K.; Deb, K.; Dorigo, M.; Fogel, D. B.; Garzon, M. H.; Goldberg, D. E.; Iba, H.; and Riolo, R., eds., *Genetic Programming 1998: Proceedings of the Third Annual Conference*, 208–213. University of Wisconsin, Madison, Wisconsin, USA: Morgan Kaufmann.

Mühlenbein, H., and Schlierkamp-Voosen, D. 1995. Analysis of selection, mutation and recombination in genetic algorithms. In Banzhaf, W., and Eeckman, F. H., eds., *Evolution as a Computational Process, Lecture Notes in Computer Science*, 188–214. Springer, Berlin.

Piszcz, A., and Soule, T. 2005. Genetic programming: Parametric analysis of structure altering mutation techniques. In Rothlauf, F.; Blowers, M.; Branke, J.; Cagnoni, S.; Garibay, I. I.; Garibay, O.; Grahl, J.; Hornby, G.; de Jong, E. D.; Kovacs, T.; Kumar, S.; Lima, C. F.; Llorà, X.; Lobo, F.; Merkle, L. D.; Miller, J.; Moore, J. H.; O'Neill, M.; Pelikan, M.; Riopka, T. P.; Ritchie, M. D.; Sastry, K.; Smith, S. L.; Stringer, H.; Takadama, K.; Toussaint, M.; Upton, S. C.; and Wright, A. H., eds., *Genetic and Evolutionary Computation Conference (GECCO2005) workshop program*, 220–227. Washington, D.C., USA: ACM Press.

Poli, R., and Langdon, W. B. 1997. Genetic programming with one-point crossover. In Chawdhry, P. K.; Roy, R.; and Pant, R. K., eds., *Soft Computing in Engineering Design and Manufacturing*, 180–189. Springer-Verlag London.

Sasaki, A., and Nowak, M. A. 2003. Mutation landscapes. *Journal of Theoretical Biology* 62:7.

Stanhope, S., and Daida, J. 1997. An individually variable mutation-rate strategy for genetic algorithms. In Angeline, P. J.; Reynolds, R. G.; McDonnell, J. R.; and Eberhart, R., eds., *Evolutionary Programming VI: Proceedings of the Sixth Annual Conference on Evolutionary Programming*, volume 1213 of *Lecture Notes in Computer Science*. Indianapolis, Indiana, USA: Springer-Verlag.