Is Increased Diversity in Genetic Programming Beneficial? An Analysis of Lineage Selection

Edmund K. Burke, Steven Gustafson†, Graham Kendall and Natalio Krasnogor
University of Nottingham
{ ekb, smg, gxk, nxk }@cs.nott.ac.uk († corresponding author)

Abstract- This paper presents an analysis of increased diversity in genetic programming. A selection strategy based on genetic lineages is used to increase genetic diversity. A genetic lineage is defined as the path from an individual to individuals which were created from its genetic material. The method is applied to three problem domains: Artificial Ant, Even-5-Parity and symbolic regression of the Binomial-3 function. We examine how increased diversity affects problems differently and draw conclusions about the types of diversity which are more important for each problem. Results indicate that diversity in the Ant problem helps to overcome deception, while elitism in combination with diversity is likely to benefit the Parity and regression problems.

1 Introduction

Is increased diversity in genetic programming beneficial to performance? The motivation behind evolutionary algorithms (Darwin's theory of natural selection) would certainly suggest so. But nature is not necessarily about optimisation, while evolutionary computation generally focuses on optimisation problems. A compromise in these objectives can be seen in the typical two-phase behaviour of evolutionary algorithms. The first phase of the process explores for good solutions, while the second phase exploits the better solutions. In which phase is diversity more important? A loss of diversity during exploration would represent a poor global search, while high amounts of diversity could later prevent a thorough exploitation phase. Both phases appear to require different diversity, and referring to the general loss of diversity would seem problematic. Additionally, in a complex representation such as genetic programming, in what sense should diversity be defined: the space of possible parse trees representing programs, the actual amount of different elements which define programs, or the fitness values attached to programs?

Our previous research suggests that the measures which are used to control and describe diversity are often conflicting and do not necessarily correlate well with fitness improvement [2, 3]. To better understand the effects diversity has on performance, we have used a simple method based on genetic lineages to increase the genetic diversity of populations. We add no elitism, size, shape or content bias to an otherwise standard framework. Three problem domains are investigated and compared with previous research to understand why increasing diversity is beneficial on some but not others.

1.1 Previous Work

One of the main challenges in genetic programming is preventing the system from getting stuck in local optima. An often cited cause is the loss of diversity and convergence within the population [15, 20, 2]. Diversity has been measured as genetic variety [12], edit distances between trees [22, 6, 5, 3], unique subtrees [11], initial genetic material [20], and the entropy of a population [27]. These measures identify different aspects of a population's structure and behavioural properties that might cause convergence and eventual fitness stagnation. However, diversity measures can imply different objectives and do not necessarily correlate well with fitness [2, 3].

Convergence (the population-wide loss of diversity) was suggested to mark the beginning of a local search phase in genetic programming [8, 26]. In typical genetic programming systems, the size of genetic material exchanged during recombination becomes smaller and more concentrated to similar locations. The exchanges of genetic material in these areas represent a local search as they are less likely to be detrimental to fitness and modify only a small part of the overall structure [10, 17, 26] (assuming a correlated landscape).

The quick loss of diversity followed by small improvements to a genetically similar population has been likened to blind random search [8, 20] where genetic programming will "behave like a set of parallel stochastic hill-climbers"[26]. With added elitism or over-selection of similar individuals, the algorithm begins to look more like a 'hill-climbing' method. However, it is a loose metaphor used here to help explain why increasing diversity can cause different behaviour on different problems. Early convergence, however, is the likely cause of large performance variance across runs as it is makes the population more susceptible to local optima that can vary widely [20].

Genetic programming was compared with hill-climbing methods using similar representations and operators [24, 23, 15, 13]. Some problems, Artificial Ant for instance, were often solved better using genetic programming. On other problems, such as Multiplexer, hill-climbing methods performed considerably better than genetic programming.

While we use a metaphor of hill-climbing to describe a type of genetic programming search, it is clearly not the case that standard genetic programming *is only* hill-climbing.

Many problem and representation specific methods have been used to improve diversity. Hamming distances between individuals were used to select diverse crossover partners in a genetic algorithm [7]. The distance between trees was used to define a homologous crossover for genetic programming [14]. Crossover partners were selected in the Pygmie algorithm from lists based on fitness and size separately [29]. Edit distances were used in a multi-objective method [5], with fitness sharing [6], and in a linear representation to first select for fitness and then diversity [1]. Fitness sharing and negative correlation learning were studied as ways to improve diversity [19], and a selection method that is uniform over the fitness values [9] was suggested as an alternative way to preserve diversity. Island models, or demes, are commonly suggested as ways to improve diversity [20, 12], but are typically used for easy parallelisation of the algorithm.

While some methods of diversity show improvement of fitness, they typically add elitism, suffer from additional computation and address a problem which is not clearly defined or understood. How does one know what type of diversity is needed and how much of it is necessary for different problems? As stated by Ryan [29], "...what is needed is a method which does not attempt to explicitly measure genetic differences, for this leads to much difficulty when defining exactly what constitutes difference". We would add that it is also difficult to understand why a problem would benefit from different types and levels of diversity.

1.2 About this Paper

We address the loss of diversity in genetic programming with a simple technique to redirect selection pressure from the *fit* to the *fit and diverse*. The technique does not require us to actually measure diversity, but it significantly changes populations during the evolutionary process. We are not interested in a general method to improve fitness or diversity, but rather to demonstrate that increasing diversity can lead to dramatic changes in the search ability of genetic programming. Population convergence and increased selection pressure of similar individuals creates a 'hill-climbing' atmosphere which, when disturbed, can improve or worsen fitness, depending on the problem. We do not propose that one method (genetic programming or hill-climbing) is better than the other. Instead, we intend to use the similarities between methods to explain the effects of changing diversity.

In this paper we show that increasing the genetic diversity of populations can be effective in avoiding local optima in problems that are deceptive. By deceptive, we mean the abundance of dissimilar partial solutions which may draw the attention of the algorithm but not allow improvement. However, we show that in problems where low diversity encourages code growth or the culling of contextual shifting nodes, increasing diversity can be counter to improving fitness. The method we present, **lineage selection**, is interesting in its own right. It is efficient and serves to focus the search more evenly across the diversity present in the initial population, while still allowing exploitation to emerge and a high level of fitness to be achieved.

2 Methods

We present a method of lineage selection which is based on the definition and tracking of genetic lineages.

2.1 Genetic Lineages

In this study we will focus on the standard form of genetic programming with a variable length tree representation of S-expressions, using standard subtree crossover for recombination. In crossover, one of the two chosen parents serves as the root parent, which provides the root portion of the tree and receives a subtree from the other parent. A **genetic lineage** is defined as the connection from the root parent to those individuals which were created, via crossover, from that individual. McPhee and Hopper [20] show how entire populations quickly loose genetic lineages and soon descend from one individual. This is critical because genetic lineages, defined in context with crossover, tend to share common root shapes and contents [28, 15, 26]. Thus, they allow us to approximate the loss of genetic diversity and

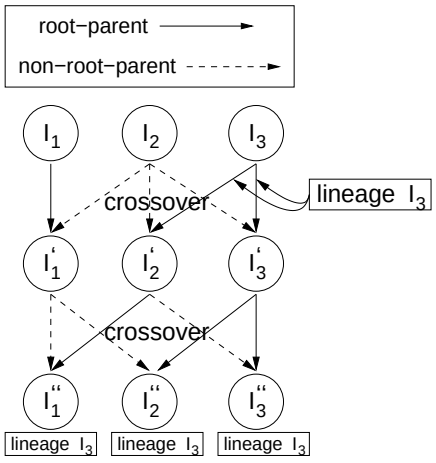

Figure 1: An example of three individuals undergoing recombination, where after the second application of crossover all three are descendants from the same individual and represent the same genetic lineage.

convergence without expensive measures. We then improve diversity by slowing the loss of genetic lineages. Fig. 1 is an example of the loss lineages. Three individuals (I_1, I_2, I_3) produce offspring (I_1', I_2', I_3') via crossover. The root parent is denoted with a solid arrow. After another generation, the three new individuals all belong to the same genetic lineage, I_3.

2.2 Lineage Selection

Lineage selection is implemented as an additional step to bias selection toward different lineages from the initial population. To perform selection, we place individuals into groups based on common genetic lineages. Tournament selection picks individuals by first picking a random genetic lineage and then a random individual from within that lineage. A tournament is held between these random individuals. Each genetic lineage has an equal change of contributing an individual to each tournament. We introduce no elitism, or no direct measure of size, shape, content, or fitness. The aim of lineage selection is to maintain diversity, where the population contains *good* individuals that are not *just* diverse. This additional step to tournament is an inexpensive operation that can be done at each generation by maintaining a similar tag between root-parents and children. It is even more efficient to simply keep individuals in groups based on their genetic lineage from the initial generation. The following measures of diversity characterise populations according to genetic differences and behaviour.

2.3 Entropy and Edit Distance

The phenotypic entropy is defined as the distribution of the proportion of the population with the same fitness value [27], (also investigated in [2, 3]). Specifically, entropy is defined as $-\sum_k p_k \log p_k$, where p_k is the proportion of the population with the same fitness value. Higher entropy values represent more chaos of the system, where chaos refers to more unique elements with fewer copies in the population. An increase of entropy means the population is either acquiring new fitness values or spreading out the members more uniformly over existing values. An important side-effect of entropy is that it also describes an emergent change of the selection pressure. Higher entropy populations will allow selection to distinguish between members better, while lower entropy populations make selection more random by grouping many different individuals into the same fitness classes (and is a focus of our current research).

We will use two measures of distance within the population. Both measures compare every member in the population to the current most fit member, before dividing by the population size. The first edit distance is found by bringing two trees to the same structure - filling either tree with *null* nodes. The distance between two nodes is 1 if they are not equal and 0 if they are equal. The distance between two trees is the summation of the distance between their nodes, normalised by dividing by the smaller tree [5]. The second edit distance divides the total distance (defined above) within the same depth by an increasing weight. The distance between two trees is then the total distance within each depth, placing more weight on distances near the root of the trees. The same distances were studied previously [3], the second was adapted from a similar distance measure [6] and both are similar to previous ones in the literature [22].

3 Experimental Results

To explore the effectiveness of this strategy, we examine three different problem domains with two experiments: a control experiment with tournament selection and an experiment which employs lineage selection. The three problems chosen for the experimental study are Artificial Ant on the Sante Fe Trail, Even-5-Parity and symbolic regression of the Binomial-3 function. These problems and algorithmic parameters are commonly used in numerous theoretical studies of genetic programming and diversity [25, 15, 2, 3, 20, 13]. The Ant problem attempts to pick up 89 food pellets on a grid with the functions {if-food-ahead, prog2n} and the terminals {left, right, move}. The Parity problem attempts to classify all 2^5 combinations of 5-bit length strings of {1,0} with the functions {and, or, nand} and five boolean terminals. The Binomial-3 regression problem [4] attempts to approximate the function $f(x) = (1 + x)^3$ using the terminals x, ephemeral random constants in the range of $[-10, 10]$, and the functions $\{+, -, \times, p/\}$, where division is protected and returns 1.0 if the denominator is extremely small.

Other parameters are as follows: a generational algorithm for 101 generations, a population size of 500, maximum depth of 10, initial maximum depth of 4 for Ramped-Half-n-Half tree generation, internal node selection for crossover of 90%, and tournament size of 4 for selection. The objective of each problem is the minimisation of missed pellets (Ant), wrongly classified bit-strings (Parity) and the mean squared error from the Binomial-3 function. One hundred runs were performed for each experiment in the ECJ [16] framework.

Fig. 2 shows the behaviour of the system for the control and lineage experiments. Mean run values are plotted here, but final generation statistics[1] are also reported in Table 1. Note that only the Ant problem had an improvement in fitness with lineage selection, and all problems had a significant decrease in size and increase in edit distances using

[1] Significance testing was done using the Student's T-test at the 95% confidence level.

Table 1: Statistics for the population in the final generation of runs. Significant difference between the control experiment and the experiment using lineage selection is denoted with a '*' next to the lineage selection mean.

Problem	Final Pop	Fitness		Nodes		Phen. Entropy		Edit Dist.		Edit Dist. (Weighted)	
		control	lineage	control	lineage	control	lineage	control	lineag e	control	lineage
ant	min	0.000	0.000	43.408	41.968	0.292	0.542	0.120	0.187	0.615	1.047
	max	37.000	29.000	116.180	88.408	1.169	1.509	0.353	0.365	3.572	4.394
	mean	15.060	10.930	79.068	62.370*	0.709	1.127*	0.245	0.275*	1.643	2.884*
	stdev	12.362	10.010	14.878	8.672	0.170	0.235	0.048	0.036	0.628	0.711
parity	min	0.000	5.000	68.064	63.136	0.437	0.643	0.102	0.259	0.356	2.335
	max	13.000	11.000	220.268	109.580	0.969	0.940	0.409	0.471	3.494	5.490
	mean	6.740	8.970*	124.125	82.896*	0.749	0.787*	0.221	0.363*	1.042	4.507*
	stdev	2.207	1.195	26.762	9.443	0.092	0.059	0.066	0.042	0.516	0.580
bin3	min	0.000	0.007	3.000	2.992	0.287	0.264	0.200	0.227	0.664	0.677
	max	5.480	6.930	141.308	84.372	2.614	2.662	0.533	0.711	2.078	5.134
	mean	0.651	1.428*	57.351	34.401*	1.920	1.888	0.361	0.403*	1.123	2.442*
	stdev	0.972	1.875	24.950	21.659	0.554	0.819	0.060	0.104	0.308	1.042

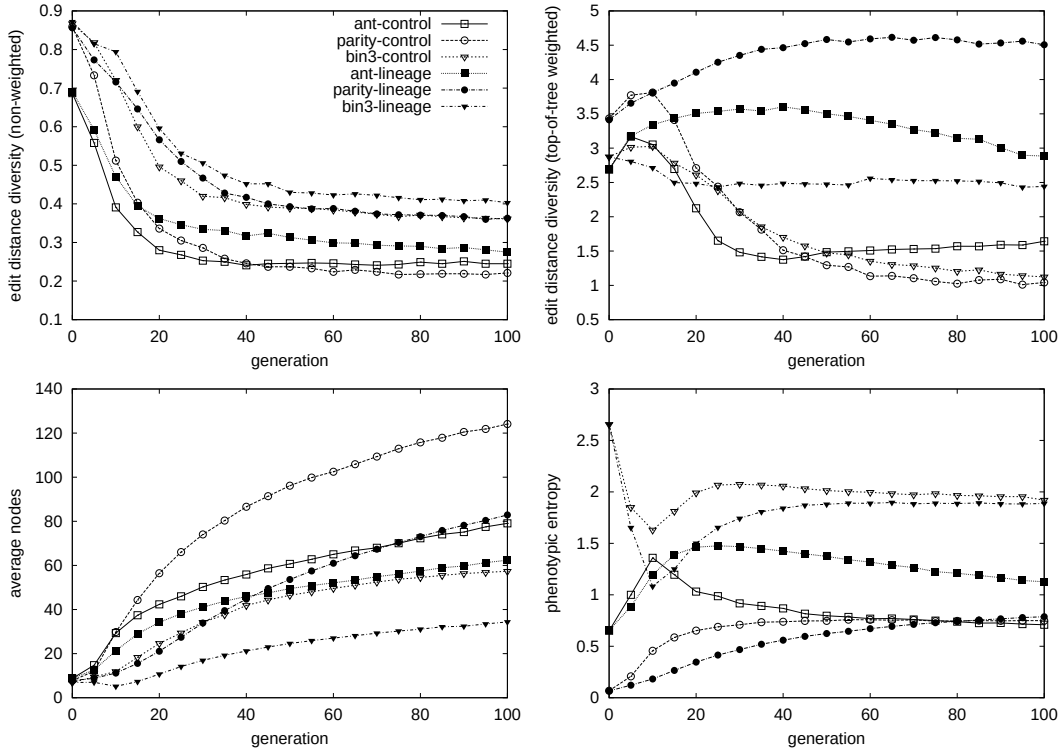

Figure 2: Measures versus generation are shown for each problem and experiment type (control and lineage). Note that the same key from the top-left graph is used for all graphs.

lineage selection. In the control experiment, both measures of edit distance diversity decreased early in the runs and remained low. Initial increases in entropy for the control experiments were followed by either decreases or stagnation. This signifies the inability to improve either the spread of fitness values or the uniformity of their distribution. On the other hand, lineage selection had significantly higher levels of both edit distance diversity. Also, after an initial period of greater decrease of entropy, lineage selection increased entropy longer and to higher values. Fig. 2 also shows that lineage selection produced significantly smaller individuals.

Fig. 3 shows that under lineage selection, the distance between successive best fit individuals in the population is also higher. Note that the weighted edit distance measure

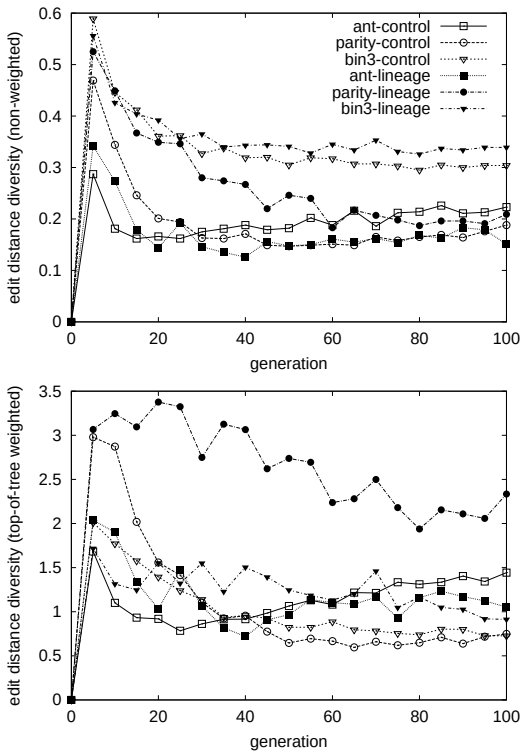

Figure 3: The edit distance between best fit individuals of successive generations.

is not normalised by individual size. Because lineage selection produced smaller individuals, we divided this measure by their average size to produce a graph similar to the non-weighted measure, but where all the lineage selection experiments remained significantly higher.

The Ant problem was the only one to benefit in terms of fitness improvement from lineage selection. While the fitness for Parity and Binomial-3 was statistically worse, a high level of fitness was achieved by lineage selection in very diverse populations. This behaviour is reflected in the phenotypic entropy. When an early increase of entropy is followed by stagnation or monotonic decrease, the experiment tended to have better fitness. The early difference in entropy values between the control and lineage selection experiments appears to be somewhat correlated to fitness improvement [3].

Fig. 4 shows the last generation where fitness improved plotted against the best fitness of the run for the Ant problem. Under lineage selection, the Ant problem finds better fitness on average 20 generations later than the control experiment. This is a good indicator that premature convergence is being avoided. The Parity problem had a similar change using lineage selection, where the best fitness was found between 10 and 15 generations later but with

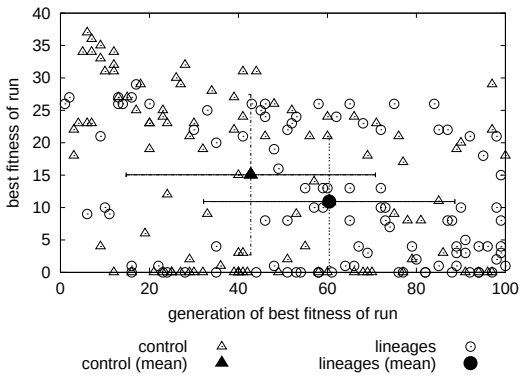

Figure 4: The generation in the Ant problem where the best fitness of the run was found, plotted against the best fitness of the run. Single standard deviation bars are plotted for both means in all directions.

a slightly worse fitness. The Binomial-3 results were not significant with respect to fitness or the last generation of improvements.

4 Discussion

Lineage selection changes the evolutionary dynamics of diversity according to edit distance and phenotypic entropy. The measures of diversity, when increased, are expected to decrease the chance that genetic programming will become stuck in local optima. However, only on the Ant problem did fitness improve. Lineage selection increases diversity by reducing the effects of selection and recombination which lead to the quick loss of diversity. Why does the Ant problem benefit from reduced selection pressure and added diversity, and why does improving diversity and reducing selection negatively affect fitness on the Binomial-3 and Parity problems? We now look at the metaphor of genetic programming performing a type of hill-climbing search to help understand the results.

4.1 Genetic Programming as a Hill-Climber

In standard genetic programming algorithms, the convergence of the population to similar programs leads recombination to perform like blind random search. It is this behaviour that leads us to characterise the search as a type of hill-climber. Thus, we may think of the beginning of a genetic programming run as a short, parallel search period until convergence occurs. At that point, recombination coupled with selection pressure (or elitism) and a converged population behaves like a hill-climber on a single program. If we consider this as a metaphor for standard genetic programming search, then what changes to the algorithm might weaken or strengthen performance?

4.2 Artificial Ant

Langdon and Poli [15] described the Ant problem to be highly deceptive for genetic programming. This is because of numerous solutions with a lot of symmetry, and also because there is no 'guiding' force to encourage the ant to travel any particular path. They also showed that the Ant problem is solved better using genetic programming than similar simulated annealing and hill-climbing techniques (page 158 [15]). Only population based, mutation-only search performed considerably better, as did a variant of strict hill-climbing which allowed smaller and larger trees than likely chosen by crossover. As population search only uses mutation, it should maintain a high amount of diversity and be similar to performing several hill-climbing searches in parallel. This search method should deal with deception better and not get stuck in local optima as frequently, which explains better performance. Adding components to the fitness function that encouraged similar types of solutions made the problem easier for genetic programming.

Lineage selection also adds a similar component of parallel search to the Ant problem. High edit distance diversity is maintained and selection pressure is reduced, creating a parallel hill-climbing effect that escapes local-optima better. When an individual becomes stuck in local optima because of deception, a diverse population is likely to contain another individual which is significantly different and allows the run to continue. Lineage selection increases or maintains higher entropy longer with more fitness values or more uniform distributions. In the control experiments, entropy quickly rises and then declines, suggesting a short period of exploration and a higher likelihood of being stuck in local optima. Also, note that in Fig. 3 the weighted edit distance between best of generation individuals is considerably higher with lineage selection between generations 10-30, the same generations where the entropy values between experiments diverge. The difference between the best individuals in this phase is found closer to the root with lineage selection. A more explorative search phase appears to be taking place using lineage selection.

Reducing deception by restricting solutions to be similar ([15]) would have the opposite effect when the hill-climbing behaviour is also reduced, as seen next in the Parity problem.

4.3 Parity

O'Reilly and Oppacher studied the 6 and 11 Multiplexer problems with genetic programming and similar hill-climbing type methods [24, 23]. The Parity and Multiplexer problems have similar functions, terminals and objectives, though they are not identical. Hill-climbing techniques appeared superior in this type of problem. Could genetic programming improve performance by becoming more of a hill-climber? We first look at a modification to genetic programming that claims superior performance on the Even-5-parity problem. De Jong et al [5] used a multi-objective method that keeps only non-dominated individuals according to an individual's fitness, size and diversity. Diversity is based on an edit distance between trees. Small populations are used which keep all non-dominated individuals. The authors note that diversity is required to prevent convergence resulting in run failure with their "uncommon degree of greediness or elitism". Hill-climbing appears to perform well on this problem as does a multi-objective method which simulates hill-climbing.

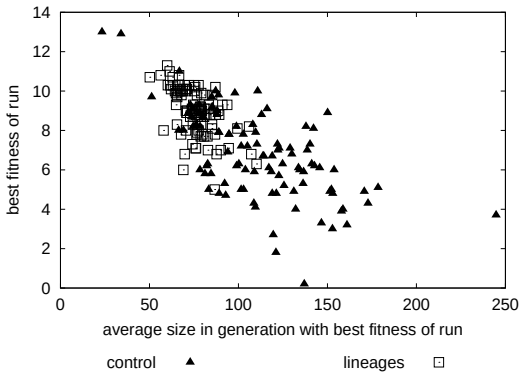

Figure 5: The best fitness found in each Parity problem run, plotted against the average size of an individual in that generation.

Lineage selection on this problem decreases selection pressure and prevents the loss of diversity and convergence. Lineage selection appears to remove the attributes of genetic programming which allow it to behave like a "hill-climber". Additionally, the size of individuals is significantly reduced under lineage selection, as seen in Fig. 2 and Fig. 5. The latter graph shows the best fitness of each run plotted against the average size of an individual in that generation. Only in the Parity problem was there such a distinctive increase in size associated with an improvement in fitness. We hypothesise that an additional factor is responsible for poorer fitness under lineage selection. By using code-growth reducing methods, Luke and Panait showed that reducing the size of solutions in Parity and Multiplexer problems (in both size restricted and unrestricted spaces) had the effect of also worsening fitness when compared with standard runs [18]. In both problems, the solving of all the fitness cases requires the use of all the terminal values. For example, in the Parity domain, the absence of one of the boolean variable from a program would result in only half of the test cases from being potentially solved. There is a benefit to programs which contain several copies of each terminal to increase the chance that they are used properly.

Adding additional elitism or more computation time with lineage selection should improve the fitness results. Also, for problems where it is known that solutions will need particular terminals or functions, it would make sense to encourage their inclusion.

4.4 Binomial-3

Daida et al [4] provide a thorough investigation of why increasing the ephemeral constant range makes the Binomial-3 problem 'harder'. They draw attention to the inter-play between content and context of functions and terminals in the representation. Many different solutions exist to the Binomial-3 problem and combining parts of different solutions does not always make sense. A level of deception exists similar to the Ant problem, due to the many different solutions. But, in the Ant problem the functions and terminals preserve semantic meaning in different contexts. Moving constants and arithmetic functions between programs in regression problems does not ensure their meaning in new contexts. A DAG representation of genetic programming was used on a regression problem [21] where the author introduced a diversity method that was also highly elitist. Performance showed that best fitness was achieved much faster with smaller population sizes using the elitist diversity measure.

Regression problems appear to pose a two-fold problem, finding a good approximation to fit the data points and attempting to reduce semantic changes of nodes during crossover. In this case, increasing genetic diversity could increase the chance that crossover will have problems with nodes changing context. A converged population might contain fewer nodes but with similar contexts and ease the search. However, too little or too much selection or diversity would cause problems as well, making this a complex problem domain.

4.5 Remarks

Is increasing diversity beneficial to genetic programming? We have seen that increasing the genetic differences in populations allows for more global search and local optima avoidance. We have also seen that it may negatively reduce individual size and prevent the removal of nodes that otherwise might reduce deception. Increasing genetic diversity adds longer periods of entropy increase, which is desirable if the objective is a uniform spread of solutions instead of a single solution. However, the slower increase of entropy, the higher genetic diversity and the survival of more dissimilar solutions appears to decrease the hill-climbing behaviour that previous research has shown to be effective in solving these problems. Future methods used to increase diversity to improve fitness should clearly state the motivation for such an increase and why that type of diversity would be beneficial. Diversity methods may not be justified in their own right, but as a partner in a more elitist strategy or as a supplier of programs to a local search method.

5 Conclusions

Lineage selection is used to increase diversity by reducing the selection pressure from the most *fit* to the *fit and diverse*. This has caused performance variance across three problem domains. We analysed these results in the light of previous research to conclude that, if genetic programming is viewed as performing a type of hill-climbing search, then adding diversity can worsen fitness on some problems that clearly benefit from elitism in a hill-climbing environment. However, when deception is embedded into the problem, improving diversity may help avoid local optima (as in the Ant problem), or it may compound the deception by maintaining its presence (as in the Binomial-3 problem).

Our future work is investigating methods which allow the appropriate type and amount of diversity to be an emergent property of the system. Detecting deception, contextual shifts and the changing distribution of population content and behaviours are promising areas of future research.

Acknowledgments

The authors would like to thank Nic McPhee, Una-May O'Reilly and Bill Hart for discussions about diversity in GP.

Bibliography

[1] M. Brameier and W. Banzhaf. Explicit control of diversity and effective variation distance in linear genetic programming. In *Genetic Programming, Proceedings of the 5th European Conference*, volume 2278 of *LNCS*, pages 162–171, Kinsale, Ireland, April 2002. Springer-Verlag.

[2] E. Burke, S. Gustafson, and G. Kendall. A survey and analysis of diversity measures in genetic programming. In W. B. Langdon et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 716–723, New York, 9-13 July 2002. Morgan Kaufmann Publishers.

[3] E. Burke, S. Gustafson, G. Kendall, and N. Krasnogor. Advanced population diversity measures in genetic programming. In J.J Merelo Guervós et al., editors, *Seventh International Conference on Parallel Problem Solving from Nature*, volume 2439 of *LNCS*, pages 341–350, Granada, Spain, September 2002. Springer.

[4] J.M. Daida, R.R. Bertram, S.A. Stanhope, J.C. Khoo, S.A. Chaudhary, O.A. Chaudhri, and J.A. Polito II. What makes a problem GP-hard? analysis of a tunably difficult problem in genetic programming. *Genetic Programming and Evolvable Machines*, 2(2):165–191, June 2001.

[5] E.D. de Jong, R.A. Watson, and J.B. Pollack. Reducing bloat and promoting diversity using multi-objective methods. In

L. Spector et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 11–18, San Francisco, CA, 7-11 July 2001. Morgan Kaufmann.

[6] A. Ekárt and S. Németh. Maintaining the diversity of genetic programs. In J. Foster et al., editors, *Proceedings of the 5th European Genetic Programming Conference*, volume 2278 of *LNCS*, pages 162–171, Kinsale, Ireland, 3-5 April 2002. Springer-Verlag.

[7] L.J. Eshelman and J.D. Schaffer. Crossover's niche. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 9–14, San Mateo, CA, 1993. Morgan Kaufman.

[8] C. Gathercole and P. Ross. An adverse interaction between crossover and restricted tree depth in genetic programming. In J.R. Koza et al., editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 291–296, Stanford University, CA, USA, 28–31 July 1996. MIT Press.

[9] M. Hutter. Fitness uniform selection to preserve genetic diversity. In D.B. Fogel et al., editors, *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 783–788. IEEE Press, 2002.

[10] C. Igel and K. Chellapilla. Investigating the influence of depth and degree of genotypic change on fitness in genetic programming. In W. Banzhaf et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1061–1068, Orlando, Florida, USA, 13-17 July 1999. Morgan Kaufmann.

[11] M. Keijzer. Efficiently representing populations in genetic programming. In P.J. Angeline and K.E. Kinnear, Jr., editors, *Advances in Genetic Programming 2*, chapter 13, pages 259–278. MIT Press, Cambridge, MA, USA, 1996.

[12] W.B. Langdon. *Data Structures and Genetic Programming: Genetic Programming + Data Structures = Automatic Programming!*, volume 1 of *Genetic Programming*. Kluwer, Boston, 24 April 1998.

[13] W.B. Langdon. The evolution of size in variable length representations. In *1998 IEEE International Conference on Evolutionary Computation*, pages 633–638, Anchorage, Alaska, USA, 5-9 May 1998. IEEE Press.

[14] W.B. Langdon. Size fair and homologous tree genetic programming crossovers. *Genetic Programming and Evolvable Machines*, 1(1/2):95–119, April 2000.

[15] W.B. Langdon and R. Poli. *Foundations of Genetic Programming*. Springer-Verlag, 2002.

[16] S. Luke. ECJ: A java-based evolutionary computation and genetic programming system, 2002. http://www.cs.umd.edu/projects/plus/ecj/.

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

[18] S. Luke and L. Panait. Fighting bloat with nonparametric parsimony pressure. In J.-J. Merelo Guervós et al., editors, *Parallel Problem Solving from Nature - PPSN VII*, number 2439 in Lecture Notes in Computer Science, LNCS, page 411 ff., Granada, Spain, 7-11 September 2002. Springer-Verlag.

[19] R.I. McKay. Fitness sharing in genetic programming. In D. Whitley et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 435–442, Las Vegas, Nevada, USA, 10-12 July 2000. Morgan Kaufmann.

[20] N.F. McPhee and N.J. Hopper. Analysis of genetic diversity through population history. In W. Banzhaf et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1112–1120, Florida, USA, 1999. Morgan Kaufmann.

[21] P. Monsieurs and E. Flerackers. Reducing population size while maintaining diversity. In C. Ryan et al., editors, *Genetic Programming, Proceedings of EuroGP'2003*, volume 2610 of *LNCS*, pages 145–156, Essex, 14-16 April 2003. Springer-Verlag.

[22] U.-M. O'Reilly. Using a distance metric on genetic programs to understand genetic operators. In *IEEE International Conference on Systems, Man, and Cybernetics, Computational Cybernetics and Simulation*, volume 5, pages 4092–4097, Florida, USA, 1997.

[23] U.-M. O'Reilly and F. Oppacher. Program search with a hierarchical variable length representation: Genetic programming, simulated annealing and hill climbing. In Y. Davidor et al., editors, *Parallel Problem Solving from Nature – PPSN III*, number 866 in Lecture Notes in Computer Science, pages 397–406, Jerusalem, 9-14 October 1994. Springer-Verlag.

[24] U.-M. O'Reilly and F. Oppacher. Hybridized crossover-based search techniques for program discovery. In *Proceedings of the World Conference on Evolutionary Computation*, volume 2, pages 573–578, Perth, Australia, 29 November - 1 December 1995. IEEE Press.

[25] R. Poli. A simple but theoretically-motivated method to control bloat in genetic programming. In C. Ryan et al., editors, *European Conference on Genetic Programming*, volume 2610 of *LNCS*, pages 200–210, Essex, 14-16 April 2003. Springer-Verlag.

[26] R. Poli and W.B. Langdon. On the search properties of different crossover operators in genetic programming. In J.R. Koza et al., editors, *Proceedings of the Third Annual Genetic Programming Conference*, pages 293–301, Wisconsin, USA, 22-25 July 1998. Morgan Kaufmann.

[27] J.P. Rosca. Entropy-driven adaptive representation. In J.P. Rosca, editor, *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, pages 23–32, Tahoe City, California, USA, 9 July 1995.

[28] J.P. Rosca and D.H. Ballard. Rooted-tree schemata in genetic programming. In L. Spector et al., editors, *Advances in Genetic Programming 3*, chapter 11, pages 243–271. MIT Press, Cambridge, MA, USA, June 1999.

[29] C. Ryan. Pygmies and civil servants. In K.E. Kinnear, Jr., editor, *Advances in Genetic Programming*, chapter 11, pages 243–263. MIT Press, 1994.