On Naïve Crossover Biases with Reproduction for Simple Solutions to Classification Problems

M. David Terrio, Malcolm I. Heywood

Dalhousie University, Faculty of Computer Science
6040 University Avenue, Halifax, NS. B3H 1W5 Canada
{mterrio, mheywood}@cs.dal.ca

Abstract. A series of simple biases to the selection of crossover points in tree-structured genetic programming are investigated with respect to the provision of parsimonious solutions. Such a set of biases has a minimal computational overhead as they are based on information already used to estimate the fitness of individuals. Reductions to code bloat are demonstrated for the real world classification problems investigated. Moreover, bloated solutions provided by a uniform crossover operator often appear to defeat the application of MAPLE™ simplification heuristics.

1 Introduction

The variable length representation employed by tree-structured Genetic Programming (GP) [1], whilst satisfying the goal of providing a machine-learning paradigm with a minimum number of *a priori* constraints, is also known to lead to approximately square law increases in code length [2], or what has typically become known as code bloat [3-7]. The ideal would therefore be to evolve fit – as measured by the application performance objective – yet parsimonious solutions. By doing so, the computational requirements necessary to evolve solutions would be significantly reduced – fitness evaluation is the most computationally expensive process, where this is proportional to the size of the data set and chromosome length of a candidate solution – whilst increasing the acceptance rate of GP solutions in the application domain. Moreover, the causes or biases behind code bloat are known to vary depending on the GP structure [8]. Here, we concentrate on tree structured as opposed to linearly structured GP.

In this work our motivation is to investigate the applicability of a series of naïve biases introduced to the crossover operator using fitness information freely available during evaluation of individuals. The basic objective is to direct the identification of crossover points such that parsimonious solutions are identified with minimal impact on solution performance. In addition, crossover is only allowed to produce one child. The second is reproduced, the motivation being to let mutation provide further investigation of the current shape [2]. In relation to previous works, Iba and de Garis considered the case of collecting information during fitness evaluation to *deterministically* select sub-trees for application of mutation and crossover [9]. However, the principle motivation appears to have been improvements to the fitness of an individual and not to encourage parsimony in the solutions found.

2 Methodology and Approach

The initial motivation for this work was to provide transparent solutions using tree structured individuals in medical applications. Such a requirement is particularly important in this area due to the need to not just solve the problem, but also to win the confidence of the patient and medical personnel [10]. In practice, however, this might be seen as a general requirement for all solutions produced by machine learning systems. As will become apparent from Section 3, it is not in general possible to apply off-line simplification of GP solutions and expect a succinct solution.

In order to mitigate this effect we introduce two properties. Firstly, a degree of determinism is introduced into the crossover operator such that both stochastic selection and 'directed' selection of crossover points is provided. Figure 1 summarizes this process in terms of a generic GP algorithm with steady state tournament selection. Secondly, only one of the children is a result of the crossover operation, the other is reproduced, Figure 2.

The 'uniform' sub-tree crossover operator used here selects a branch of the tree representing an individual using a uniform probability density function. In providing suitable definitions for the selection of crossover operators, we limit ourselves to investigating the applicability of the following naïve crossover definitions,

Fitness Directed Crossover: This crossover operator selects the node in the individual with highest fitness. Node fitness is the error between the target value and node value (including any wrapper). The node value naturally includes the contribution of any attached sub-tree;

Fitness Difference Directed Crossover: Directed crossover now selects the node with greatest change in fitness as evaluation progresses from terminal to root node. This may or may not help protect the individual from repeatedly selecting the branches near the root node for crossover;

Roulette-Fitness Directed Crossover: In this case each node of an individual is given space in a roulette wheel in proportion to node fitness. In this way it is possible to provide an additional path for the stochastic selection of nodes.

With respect to single child crossover with reproduction of one parent, our principle motivation is to let the reproduced individual be modified by mutation, so emphasizing the further investigation of the current shape [2]. Note that mutation is applied in addition to crossover in the algorithm of Figure 1, and at a relatively high probability of 50%, Table 1. The test for producing a single child is summarized by Figure 2, and affects both the directed and undirected process for sub-tree crossover.

```
1. Initialize Population of size M.
2. Uniform random selection of N individuals (N<<M).
3. Evaluate fitness of the N individuals.
4. Rank the N individuals in descending order of fitness.
5. IF one of the N individuals satisfies the stop criteria, END.
6. Overwrite worst N/2 individuals from tournament with best N/2.
7. Call these the children.
8. For each N/4 pair of child individuals,
9. IF (apply crossover == TRUE)
10. THEN IF (apply directed crossover == TRUE)
          THEN (apply directed crossover)
          ELSE (apply uniform crossover)
11. For each of the N/2 child individuals
12. IF (apply mutation == TRUE)
13.       THEN (apply mutation operator)
14. Replace worst N/4 members selected from population with children of N/4 best.
15. Return to point (2).
```

Fig. 1. Basic algorithm for Genetic Program with Steady-State Tournament selection and directed crossover.

```
identify crossover points.
IF (number of nodes[child#1] > K)
THEN (use child#2 and parent#1)
ELSE (use child #1 and parent#2)
```

Fig. 2. Single child crossover with reproduction.

In each case, the directed form of crossover is applied in conjunction with the uniform selection of crossover points, Figure 1, point 10. That is to say, by adjusting the ratio governing the application of uniform verses directed crossover, it is possible to empirically assess the sensitivity of the approach to different problems. This issue in particular will be investigated in Section 3. Should the directed crossover definitions prove appropriate in practice, the principle benefit of such operators is that they do not require any additional information than that calculated during the course of fitness evaluation.

Mutation may take one of two forms: single-point or multi-point (applied with equal likelihood should the test for mutation prove true, Figure 1, point 12). A single-point mutation selects a node with uniform probability and replaces it with an alternative operator with the same arity. The multi-point mutation operator selects a node with uniform probability and recursively applies the single-point operator over the sub-tree. Neither mutation operator therefore changes the size or shape of the parent.

3 Evaluation

Two benchmark problems are used for the purpose of evaluating the forms of naïve directed crossover: Breast Cancer and Liver Disease Classification [11], where the latter is known to be particularly difficult. Table 1 summarizes the problem and GP parameters used. In all the experiments the parameter K from Figure 2 remains at

200, where no particular significance appears to be attached to different values for '*K*' i.e. the parents are chosen uniformly. The data for the classification problems are separated into training and test data, where test data are only used to evaluate performance once the termination criterion is met.

As indicated in Section 2 one of the purposes of the following study is to identify the significance of applying different degrees of directed crossover. To this end we conduct 50 trials for each of the following crossover conditions,

- Uniform selection of crossover points – represents the performance baseline against which crossover points are selected with uniform probability. Hereafter this is referred to as 'Uniform';
- Fitness Directed (FD) crossover – applied at a ratio to the uniform selection of crossover points: 25%, 50%, 75% and 100%; Figure 1, point 10;
- Fitness Difference Directed (FDD) crossover – applied at a ratio to the uniform selection of crossover points: 25%, 50%, 75% and 100%; Figure 1, point 10;
- Roulette-Fitness Directed (R-FD) crossover – applied at a ratio to the uniform selection of crossover points: 25%, 50%, 75% and 100%; Figure 1, point 10.

Thus, Fitness Directed crossover applied at a ratio of 25% results in the application of uniform crossover 75% of the time, when the test for crossover returns true. Naturally, the case of a 100% ratio implies that only the directed crossover operator is applied.

Performance is expressed in terms of training and test classification accuracy of the best individual in the last tournament, resulting in each of the 50 trials contributing to the classification accuracy. By way of comparison the c5.0 Induction System [12] produced best-case classification errors on test data of 4.6% in the Beast Cancer Classification problem and 34.9% in the Liver Disease Classification problem. From Tables 2 and 3 it is apparent that the medians of all GP solutions are better than those from c5.0 for the two problems. Moreover, c5.0 produced solutions with 8 rules for Breast Cancer and 27 rules for Liver Disease. The median of all biased GP solutions for Liver Disease using FD and FDD were significantly simpler whilst retaining lower median error rates. Under Breast c5.0 solution was simpler, albeit at the expense of classification accuracy.

Table 1. Tableau of GP parameters.

Objective	Find a program that classifies the Breast Cancer (Liver Disease) Classification problem.
Terminal Set	$x_0, \ldots, x_8$ ($x_0, \ldots, x_5$)
Functional Set	+, –, *, %, sin, cos, sqrt
Fitness Cases	524 training, 175 test (259 training, 86 test)
Fitness (and Hits)	Number of correct classifications.
Selection	Steady State Tournament of size 4.
Wrapper	IF GP output > 0.0 and data label is '1'
	THEN correct classification
	ELSE IF output ≤ 0.0 and data label is '0'
	THEN correct classification
Population size	500
Initial Population	Created using "ramped half and half" with depths between 2 and 6.
Parameters:	90% crossover, 50% mutation.
Termination:	15,000 tournaments.
Experiments:	50 independent trials.

3.1 Breast Cancer Classification Problem

Table 2 summarizes performance of uniform sub-tree crossover and the three forms of naïve directed crossover on the Breast Cancer Classification problem. It is apparent that the application of fitness directed crossover, irrespective of the form, results in decreases to code bloat while classification accuracy remains within one percent of the (median) baseline established by uniform sub-tree crossover. It is also apparent that Fitness Directed (FD) and Fitness Difference Directed (FDD) crossover provide the greatest levels of bloat reduction. Roulette-Fitness Directed (R-FD) crossover appeared to be the least sensitive to specific ratios of directed and uniform sub-tree crossover, but also never improved on the nodes per solution provided by uniform crossover.

Table 2. Breast Cancer Classification Performance.

Crossover Type		Median Test Error	Median depth per solution	Median # nodes per solution	Average # nodes per individual
Uniform (baseline)		1.41	*7*	*22*	18.23
FD	0.25	***1.15***	7	20	16.35
	0.5	***1.15***	6.5	21	19.59
	0.75	1.73	6	16	15.76
	1.0	2.3	***4***	***10***	9.654
R-FD	0.25	***1.15***	8	21.5	18.68
	0.5	1.73	7	20.5	18.3
	0.75	1.73	7	24	17.76
	1.0	***1.15***	7	20.5	16.29
FDD	0.25	***1.15***	7	20	18.53
	0.5	***1.15***	7	21	18.19
	0.75	***1.15***	7	24	20.97
	1.0	1.44	5	12	11.97

3.2 Liver Disease Classification

Table 3 summarizes performance of uniform sub-tree crossover and the three forms of naïve directed crossover on the Liver Disease Classification problem.

The pattern of performance remains consistent with that experienced for the Breast Cancer problem. Fitness Directed and Fitness Difference Directed crossover still provide the highest levels of bloat reduction, with increasing cost to classification accuracy as the ratio of directed crossover increases. Roulette-Fitness Directed crossover again provides the most competitive classification performance (with respect to the sub-tree crossover baseline), but does not appear to provide a particular trend in terms of code bloat reduction with increasing ratios of directed crossover (clarified further in next sub-section).

Table 3. Liver Disease Classification Performance.

Crossover Type		Median Test Error	Median depth per solution	Median # nodes per solution	Average # nodes per individual
Uniform (baseline)		33.33	8	34.5	26.77
FD	0.25	33.33	6.5	19.5	18.89
	0.5	33.33	6	21	18.84
	0.75	34.52	5.5	18	15.98
	1.0	33.33	***3***	***10***	***10.17***
R-FD	0.25	***30.95***	8	22.5	18.71
	0.5	32.14	8	28	21.79
	0.75	33.33	8	24	21.03
	1.0	32.14	9	29.5	24.62
FDD	0.25	31.55	9	24	20.48
	0.5	32.14	7	21	19.414
	0.75	32.14	6	17.5	15.68
	1.0	34.52	4	11.5	11.2

3.3 Analysis of Dynamic Properties

Given the encouraging results for the classification benchmark problems an analysis is conducted into the evolution of fitness and node count on a tournament-by-tournament basis. To do so, the classification accuracy and node count of each tournament winner is averaged over a 300-tournament window for each of the 50 trials. This information is then plotted as an *average* with respect to tournament for the 15,000 tournaments comprising a trial. For conciseness we will consider the case of Liver Disease alone, with similar results being produced for Breast Cancer.

Figures 3, 4 and 5 summarize the evolution of training classification accuracy for Fitness Directed, Fitness Difference Directed, and Roulette-Fitness Directed crossover respectively. In each case the uniform crossover classification accuracy is included to establish the base line (□). The clear distinction between increasing ratios of directed crossover and classification accuracy is readily apparent for Fitness Directed and Fitness Difference Directed crossover, Figures 3 and 4; whereas Roulette-Fitness Directed crossover closely mimics the performance of uniform sub-tree crossover throughout, Figure 5.

Figures 6, 7 and 8 repeat the analysis in terms of average node counts as evolution progresses. Here again, the contrast between different forms of directed crossover is readily apparent. Each increase in the ratio of Fitness Directed and Fitness Difference Directed crossover results in step reductions to the rate of bloat, such that by the 100% application of directed crossover, (average) node count is constant with increasing evolutionary cycles, Figures 6 and 7. Moreover, even in the case of Roulette-Fitness Difference crossover, the contribution of each increase to the directed crossover ratio provides a clear decrease to the rate of average node count, Figure 8.

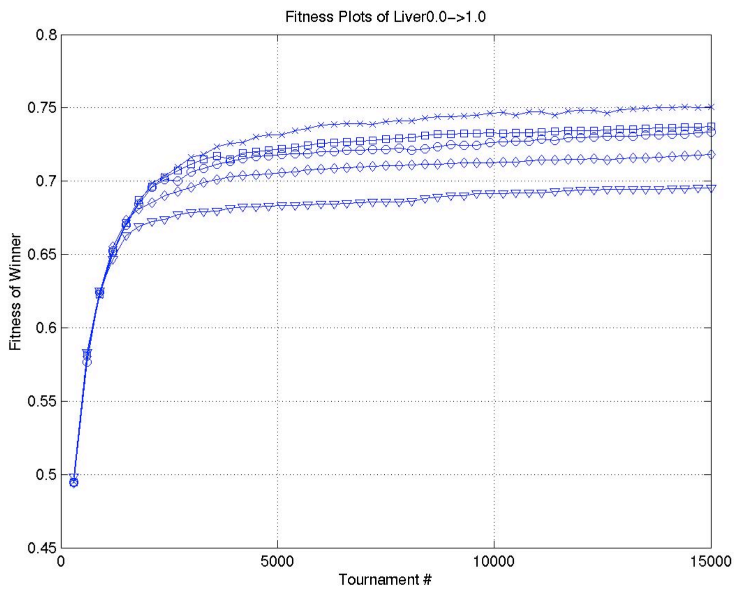

Fig. 3. Average Training Classification Accuracy – Fitness Directed Crossover. Crossover types: × - 100% uniform; ○ - 25% directed; □ - 50% directed; ◊ - 75% directed; ▽ - 100% directed

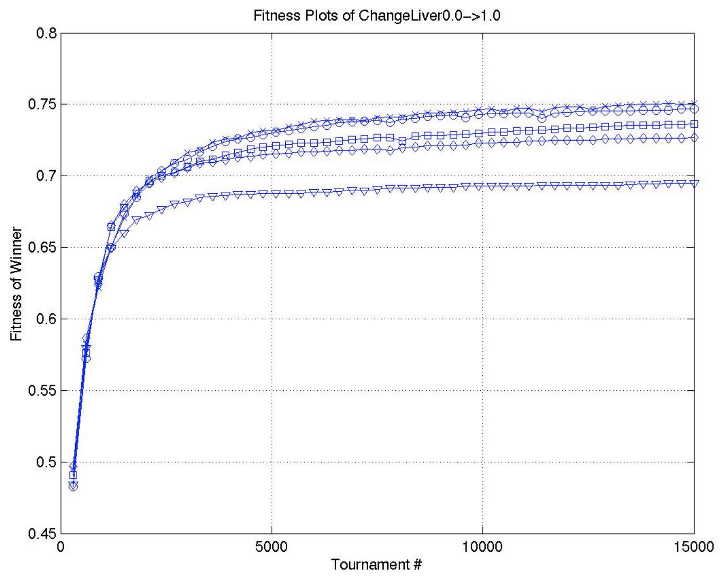

Fig. 4. Average Training Classification Accuracy – Fitness-Difference Directed Crossover. Crossover types: × - 100% uniform; ○ - 25% directed; □ - 50% directed; ◊ - 75% directed; ▽ - 100% directed

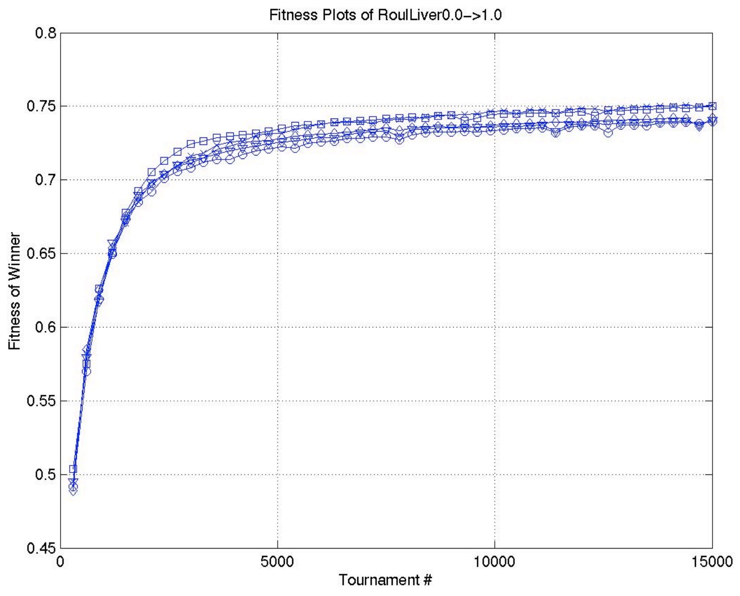

Fig. 5. Average Training Classification Accuracy – Roulette Fitness Directed Crossover. Crossover types: ▯ - 100% uniform; ○ - 25% directed; □ - 50% directed; ◊ - 75% directed; ▽ - 100% directed

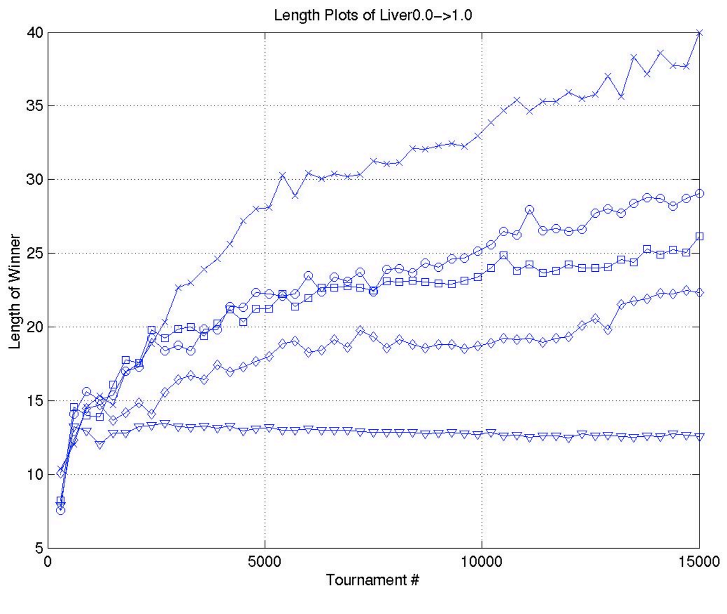

Fig. 6. Average Node Count – Fitness Directed Crossover. Crossover types: ▯ - 100% uniform; ○ - 25% directed; □ - 50% directed; ◊ - 75% directed; ▽ - 100% directed

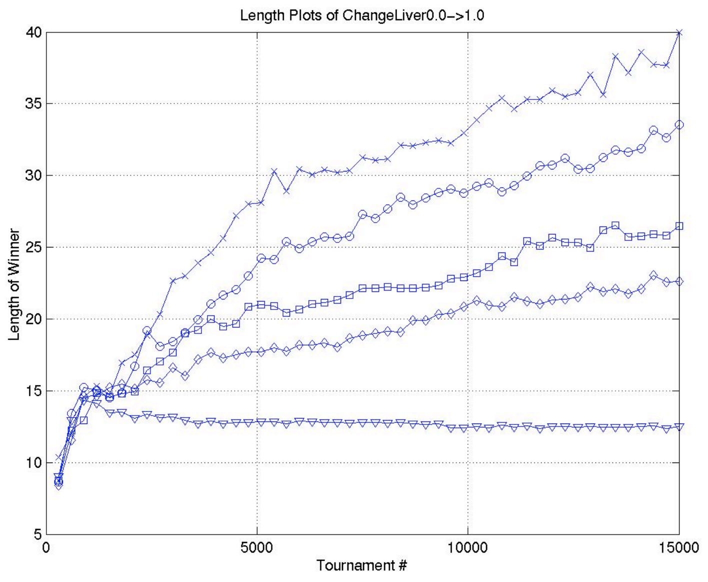

Fig. 7. Average Node Count – Fitness-Difference Directed Crossover. Crossover types: ▯ - 100% uniform; ○ - 25% directed; □ - 50% directed; ◊ - 75% directed; ▽ - 100% directed

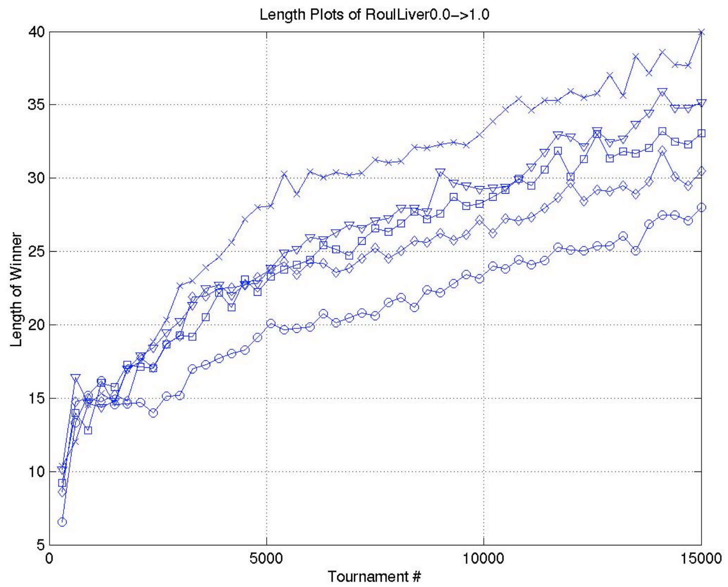

Fig. 8. Average Node Count – Roulette-Fitness Directed Crossover. Crossover types: ▯ - 100% uniform; ○ - 25% directed; □ - 50% directed; ◊ - 75% directed; ▽ - 100% directed

3.4 Solution Transparency

As indicated in the introduction, one of the desired properties assumed to coincide with parsimonious solutions is an increased transparency in programs provided by GP. In order to assess this hypothesis we utilize the simplification engine provided as part of the MAPLE□ system for symbolic math [13]. The motivation is to use the MAPLE□ symbolic simplification tool to represent the case of an average user who is used to judge the solutions provided by GP.

A scatter plot is used to summarize the effect of applying the simplification engine of MAPLE□ to each of the 50 solutions to the Liver Disease data set for both uniform sub-tree crossover and the Fitness Directed crossover (100%), Figure 6. That is, we compare solutions from 100% uniform selection of crossover points with 100% deterministic selection of crossover points. Any points along a line "$y = x$" indicate that no simplification took place; points above this line indicate that an expression became more complicated after application of MAPLE□ ; and points below indicate that an expression was simplified after application of MAPLE□ . Note also that a log scale is employed on both axes.

In the case of uniform sub-tree crossover it is apparent that on this problem, solutions actually became more complicated following application of MAPLE□ . Specifically, an average increase to term count of 168.7, over the 33 cases that increased after simplification, verses an average reduction in term count of 6.2 over 5 cases, with respect to the GP solution provided before simplification. In the case of 100% Fitness Directed crossover there is an average increase of 6.8 terms over 30 cases whereas no cases produced a decrease in code length with respect to the GP solution before simplification. It is also apparent that cases resulting in an increase following 'simplification' also predominate instances of longer programs with uniform crossover. Thus, the uniform application of crossover appears to result in individuals that are both 'bloated' and do not simplify on application of our 'base user' (the MAPLE□ symbolic simplification tool).

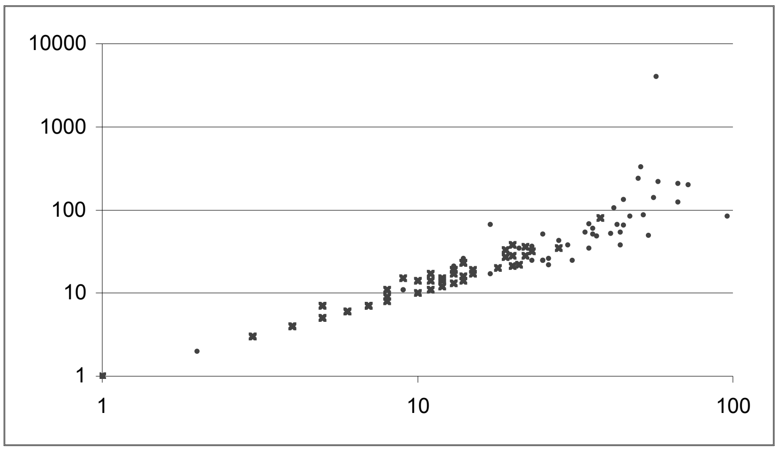

Fig. 9. Scatter plot – Solution Simplicity on Liver Disease Classification Problem. x-axis denotes before simplification; y-axis after simplification; □ - case of 100% fitness directed crossover; o - uniform crossover.

4 Conclusion

Several naïve biases were defined to provide a set of directed crossover operators using only the fitness information readily available during fitness evaluation. Increasing levels of determinism associated with selection of crossover points steadily reduces the complexity of an individual at the same time that absolute fitness may also decrease. Moreover, in the case of the classification problems investigated, it appears that node count was proportional to the ratio of uniform to directed crossover operators. Finally, it is also apparent that 'bloated' GP solutions (case of no crossover bias) did not necessarily result in significant post evolution simplification when using the simplification heuristics of the MAPLE symbolic simplification tool. Future work will evaluate the biases under a wider range of problems.

Acknowledgements

The authors gratefully acknowledge the support of a Discovery Grant from the Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

1. Koza J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, ISBN 0-262-11170-5 (1992)
2. Langdon W.B., Poli R.: Foundations of Genetic Programming. Springer-Verlag. ISBN 3-540-42451-2 (2002)
3. Blickle T., Thiele L.: Genetic Programming and Redundancy, Genetic Algorithms within the Framework of Evolutionary Computation. Workshop at KI-94, Max-Planck-Institut fur Informatik, MPI-1-94-241 (1994) 33-38
4. McPhee N.F., Miller J.D.: Accurate Replication in Genetic Programming, Proceedings of the 6th International Conference on Genetic Algorithms, Morgan Kaufmann, (1995) 303-309
5. Soule T., Foster J.A., Dickinson J.: Code Growth in Genetic Programming, Proceedings of the 1st Annual Conference on Genetic Programming, MIT Press (1996) 215-223
6. Langdon W.B., Poli R.: Fitness Causes Bloat, 2nd On-line World Conference on Soft Computing in Engineering Design and Manufacturing (WSC2) 1997.
7. Soule T., Foster J.A.: Effects of Code Growth and Parsimony Pressure on Populations in Genetic Programming, Evolutionary Computation, 6(4) (1998) 293-309
8. Smith P.W.H., Harries K.: Code Growth, Explicitly Defined Introns, and Alternative Selection Schemes, Evolutionary Computation, 6(4) (1998) 339-360
9. Iba H., de Garis H.: Extending Genetic Programming with Recombinative Guidance. In: Angeline P.J., and Kinnear K.E., eds., Chapter 4, Advances in Genetic Programming II, MIT Press, (1996) 69-88
10. Borjarczuk C.C., Lopes H.S., Freitas A.A.: Genetic Programming for Knowledge Discovery in Chest-Pain Diagnosis, IEEE Engineering in Medicine and Biology Magazine. 19(4), July-August (2000) 38-44
11. Universal Problem Solvers Inc., Machine Learning Data Sets. http://pages.prodigy.com/upso/datasets.htm
12. Quinlan J.R.: C4.5: Programs for Machine Learning. Morgan Kaufman. ISBN 1-55860-238-0. (1993) (for c4.5 v c5.0 comparison see http://www.rulequest.com/)
13. Waterloo Maple, Maple 8. http://www.mapleapps.com/