Impact of Crossover Bias in Genetic Programming

Nicholas Freitag McPhee
Div. of Science and Mathematics
University of Minnesota, Morris
Morris, MN USA-56267
mcphee@morris.umn.edu

M. Kirbie Dramdahl
Div. of Science and Mathematics
University of Minnesota, Morris
Morris, MN USA-56267
dramd002@morris.umn.edu

David Donatucci
Div. of Science and Mathematics
University of Minnesota, Morris
Morris, MN USA-56267
donat056@morris.umn.edu

ABSTRACT

In tree-based genetic programming (GP) with sub-tree crossover, the parent contributing the root portion of the tree (the *root parent*) often contributes more to the semantics of the resulting child than the *non-root parent*. Previous research demonstrated that when the root parent had greater fitness than the non-root parent, the fitness of the child tended to be better than if the reverse were true. Here we explore the significance of that asymmetry by introducing the notion of *crossover bias*, where we bias the system in favor of having the more fit parent be the root parent.

Here we apply crossover bias to several problems. In most cases we found that crossover bias either improved performance or had no impact. We also found that the effectiveness of crossover bias is dependent on the problem, and significantly dependent on other parameter choices.

While this work focuses specifically on sub-tree crossover in tree-based GP, many biological and artificial evolutionary systems have substantial asymmetries, many of which remain unstudied. This work suggests that there is value in further exploration of the impacts of those asymmetries.

Categories and Subject Descriptors

I.2.2 [**Artificial Intelligence**]: Automatic Programming-Program transformation

Keywords

genetic programming, crossover bias, root parent, crossover asymmetry, sub-tree crossover

1. INTRODUCTION

As Figure 1 illustrates, the widely used sub-tree crossover operator in tree-based genetic programming (GP) [15] is an inherently asymmetric operation, with one parent contributing the root node and, in most cases, a substantially larger number of nodes than the other parent. This asymmetry has been noted before, e.g., [10] where they refer to the parent that contributes the root node as the *root parent* and the other parent as the *non-root parent*. It has also been previously observed [4, 9, 11] that the root parent often contributes more to the semantics of the resulting individual. This is in part because the root parent typically contributes more overall nodes, and because the nodes closest to the root of the tree frequently have a stronger impact on the result of evaluating the tree in question. While the specific impact of this asymmetry depends on the details of the function set and the particular trees chosen as parents, asymmetry has a substantial effect on the semantics of the offspring in many common tree-based GP settings.

An important impact of this asymmetry was recently noted in [9], where they observed that the fitness of the offspring tended to improve when the root parent had better fitness than the non-root parent. To explore this further, we implemented what we refer to as *crossover bias*, which allowed us to probabilistically force the GP system to assign the individual with the greater fitness to be the root parent. In this paper, we apply crossover bias (Section 2) to a variety of test problems (Section 3). The results (Section 4) make it clear that crossover bias can have a substantial impact on the performance of GP runs, but that the results are

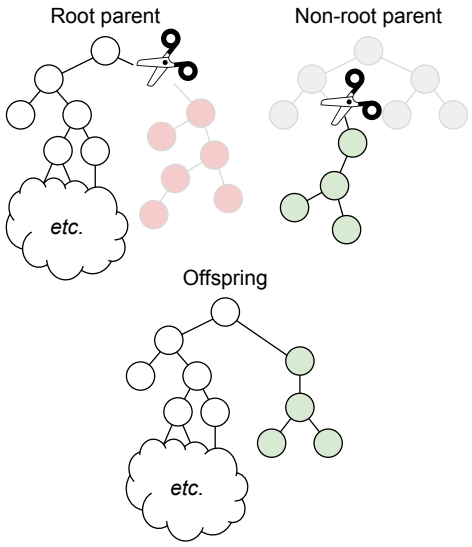

Figure 1: Sub-tree crossover, illustrating the asymmetric role of the *root* and *non-root* parents.

GECCO '15, July 11 - 15, 2015, Madrid, Spain

ISBN 978-1-4503-3472-3/15/07…$15.00

DOI: http://dx.doi.org/10.1145/2739480.2754778

Algorithm 1 Crossover bias

Require: Two individuals, RP (root parent) and NRP (non-root parent), are chosen as parents.
  if RANDOM(0, 1) $\leq$ XO_bias **then** ▷ Do we force bias?
    if FITNESS(RP) worse than FITNESS(NRP) **then**
      RP, NRP := NRP, RP ▷ Swap so RP is more fit
    end if
  end if

heavily parameter dependent and, to a lesser degree problem dependent. In general, however, we find (Section 5) that incorporating some crossover bias is, at least in these problems, usually either advantageous or neutral.

This work is related to some previous work which aims to deliberately create an asymmetry between parents (e.g., [18, 5]) as a mechanism to increase diversity and (in the case of [18]) combat bloat. There has also been a blossoming of work in the last few years based on the semantics of GP trees (e.g., [11, 12]). This includes a number of proposed mechanisms that encourage or force semantic differences (asymmetries) between parents and children [20, 1]. Some [21] also *limit* the differences between parents and offspring, in some sense reducing the impact of asymmetries between the parents, and smoothing out the fitness landscape as a result. These proposals typically focus on the semantic relationship between parents and children, however, and don't directly take into account any asymmetries between the parents.

2. CROSSOVER BIAS

To better understand these asymmetries and impact of crossover bias, we implemented crossover bias with a parameter specifying the probability of forcing the more fit parent to be the root parent when performing crossover, as detailed in Algorithm 1. Every time a crossover event was to be performed, we would use this probability to decide whether to force the more fit parent to be the root parent. Note that when crossover bias isn't applied, there is still a 50% chance that the root parent is the more fit parent.

In this paper we focus on five levels of crossover bias:

- 0.00 bias - no crossover bias is applied, i.e., just standard sub-tree crossover; a 50% probability of the more fit parent being the root parent;
- 0.25 bias - system forced to use the more fit parent as the root parent in 25% of cases; a 62.5% total probability of the more fit parent being the root parent;
- 0.50 bias - system forced to use the more fit parent as the root parent in 50% of cases; a 75% total probability of the more fit parent being the root parent;
- 0.75 bias - system forced to use the more fit parent as the root parent in 75% of cases; a 87.5% total probability of the more fit parent being the root parent; and
- 1.00 bias - system forced to always use the more fit parent as the root parent, sometimes referred to as "with crossover bias".

We also experimented with *reverse bias*, where the weaker parent was always forced to be the root parent. In general, the results for reverse bias showed no statistically significant difference from standard crossover (no bias), so in the interest of simplification we've omitted reverse bias from the remainder of the paper.

3. EXPERIMENTAL SETUP

To better understand the impact of crossover bias on GP performance, we experimented with five problems: K Landscapes with $K = 6$ [22], ORDERTREE [7], U.S. Change, Sine regression [15], and Pagie-1 regression [14]. Three of these (K Landscapes, ORDERTREE, and Pagie-1 regression) are taken from recent benchmark suggestions [23]. U.S. Change is a program synthesis problem that has proven interesting for evolutionary program synthesis [25], and sine regression is the example problem used in earlier work on the impact of root and non-root parents [9]. All experiments were run using a copy of ECJ 21[1] that we modified to support crossover bias.

The one problem not precisely described elsewhere in the literature is the U.S. Change problem, where the task is to take as input an amount, and to return as output the minimum number of coins required to make that amount using (a subset of) standard U.S. coins with amounts 25, 10, 5, and 1. So, for example, given 118 as input, the result should be 9 since

$$\underline{4} \times 25 + \underline{1} \times 10 + \underline{1} \times 5 + \underline{3} \times 1 = 118,$$

and this set of $4+1+1+3 = 9$ coins is the minimal set making a total of 118. Our function set for this problem was +, -, ×, %,[2] *mod*,[3] *min*, and *max*. The terminal set consisted of the input x, integer ephemeral random constants taken uniformly from the range $[-50, 50]$, and the set of constants $\{0, 1, 4, 5, 9, 10, 24, 25\}$. The test cases were all inputs in the range $[0, 150)$.

All our experiments used the previously published settings, function sets, fitness functions, etc., with the exception of the parameters listed in Table 1,[4] which are shared throughout these experiments. Our evolutionary system is generational, with all new individuals being generated via subtree crossover (i.e., no mutation or reproduction) unless there is a non-zero elitism percentage. Our primary goal was to better understand the impact of crossover bias, so we performed runs with a variety of crossover bias probabilities. In addition, we wanted to see how crossover bias might interact with some other commonly manipulated parameters as listed in Table 1. Unless otherwise noted, all plots, etc., in what follows will include results from all possible combinations of values in Table 1. Each box plot in Figure 3, for example, includes the results of 100 runs for each of the combinations of all four tournament sizes, both elitism proportions, and both population sizes.

[1] http://cs.gmu.edu/~eclab/projects/ecj/

[2] Protected integer division (quotient), returning 1 if the denominator is 0.

[3] Also returns 1 if the denominator is 0.

[4] The function set for Pagie-1 specified in [14] is simply +, -, ×, and % (protected division). The function set given in [8] and implemented in ECJ is "koza2", which also includes various trigonometric and logarithmic functions. The performance of GP varies substantially between the two function sets; here we follow the original Pagie-1 paper and use the simpler function set.

Parameter	Values
Crossover bias	0, 0.25, 0.5, 0.75, and 1
Tournament size	2, 3, 5, and 7
Elitism %	0 and 1%
Population size	1,024 and 10,240
# of generations	100
# of runs per treatment	100

Table 1: Shared parameters

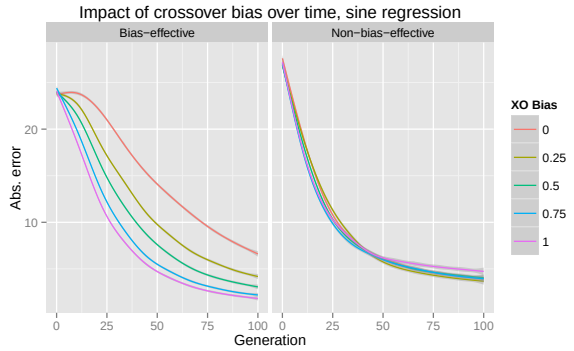

Figure 2: Impact of crossover bias over time on the sine symbolic regression problem with *bias-effective* and *non-bias-effective* parameter sets. In the left-hand panel (bias-effective settings), the curves stack in order (top to bottom) from crossover bias 0 to crossover bias 1.

In our experimentation we observed that there were quite substantial interactions between crossover bias and some of the other parameter settings. In particular, we found that there are parameter values where crossover bias appears to have a substantial and significant impact, and other parameter settings where crossover bias has almost no effect on the results. Based on these observed patterns, we found it useful to identify two specific subsets of the parameter settings listed in Table 1: "*bias-effective settings*" and "*non-bias-effective settings*". The bias-effective settings are binary tournaments, no elitism, and population size of 10,240; the non-bias-effective settings, conversely, are tournament size 7, non-zero elitism percentage, and population size 1,024. These terms will be used in the remainder of the paper, which will also include some discussion of why crossover bias might interact with these parameters in this way.

To illustrate the impact of these parameter settings, Figure 2 shows the fitness (absolute error) over time for the sine regression problem (see Section 4.3.2 for more on the sine regression results). The bias-effective settings (left-hand panel) show a clear difference in the performance as a function of crossover bias, where the non-bias-effective settings (right hand panel) show almost no difference as a function of the crossover bias probability.

4. RESULTS

All tests of differences in fitness or hits in this section are performed using pairwise Wilcoxon tests with Holm corrections. Tests of differences in the number of successes are performed using pairwise tests of proportions (chi-squared) with Holm corrections. All statistics calculations were performed

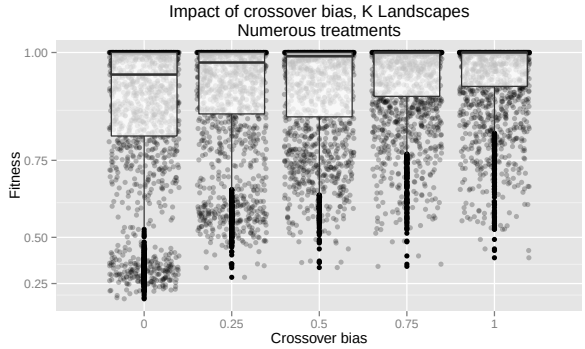

Figure 3: Impact of crossover bias on fitness for K Landscapes problem, over all the combinations of parameters in Table 1.

with R [17] and all plots were generated using the ggplot2 package [24]. Statistical significance will require $p \leq 0.05$.

4.1 Structural Problems

Most GP problems evaluate the evolved trees on a set of inputs, looking for some desired behavior or semantics. Structural problems (also referred to as constructed problems), however, have fitness functions that are entirely based on the overall structure of the evolved trees (e.g., size, shape, distribution of particular leaves) without any sort of "evaluation" being involved.

4.1.1 K Landscapes

Fitnesses for the K Landscape problem are between 0 and 1, with a fitness of 1 being a perfect solution. Figure 3 shows the impact of crossover bias on this problem across all the combinations of parameter values in Table 1. Increasing the amount of crossover bias consistently improves the fitness of the results. All the differences are statistically significant ($p < 0.012$) except for the difference between bias probability 0.75 and 1.00.

Compare this to the results shown in Figure 4, which plots the same data separated out by tournament size. It is clear that for binary tournaments, increasing the crossover bias probability continues to improve the fitness – all the differences are strongly statistically significant ($p < 6 \times 10^{-08}$). This continues to be true for tournament size 3; all the differences are statistically significant except for those between bias 0.50 and 0.75, which was very close ($p = 0.056$). None of the differences for tournament size 5 are significant. For tournament size 7, however, it looks like the reverse is true, where increasing crossover actually hurts fitness. Almost none of the differences for tournament size 7 are statistically significant with the only exception being the difference between 0.25 and 1.00 ($p = 0.021$).

Figure 5 filters out only the data gathered using the bias-effective treatment, discussed in Section 3. It is clear that the impact of crossover bias is much stronger in this case than in the more general case shown in Figure 3. Here all the differences are strongly statistically significant ($p < 10^{-11}$). In addition to improvements in fitness, increasing the crossover bias also increases the number of "perfect" solutions discovered when using the bias effective settings. Out of 100 runs with a crossover bias setting of 1.00, 15

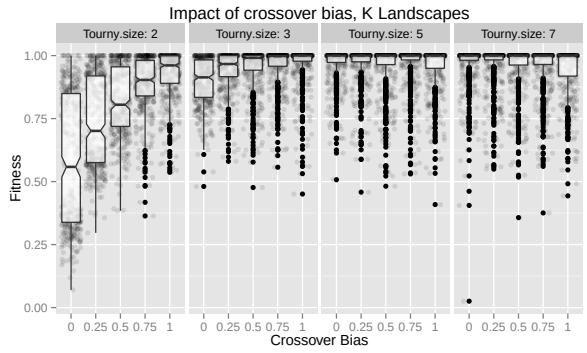

Figure 4: Impact of crossover bias on fitness for K Landscapes problem, for various tournament sizes.

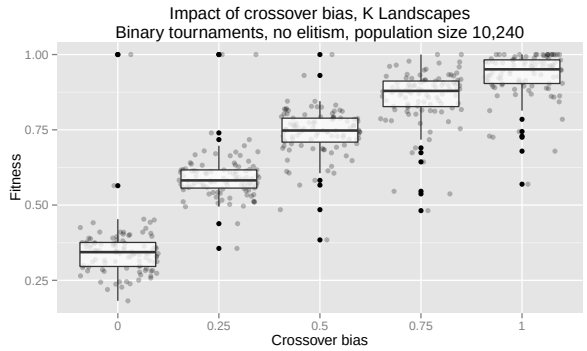

Figure 5: Impact of crossover bias on fitness for K Landscapes problem, restricted to bias-effective parameter settings.

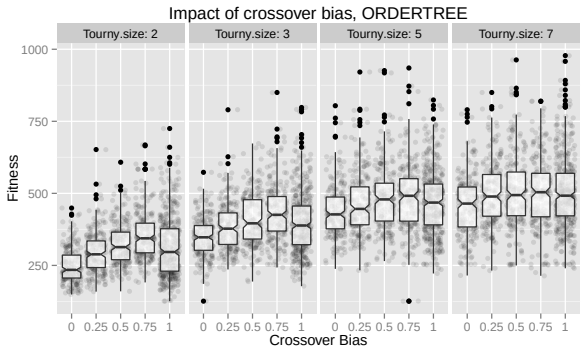

Figure 6: Impact of crossover bias on fitness for ORDERTREE problem for multiple tournament sizes.

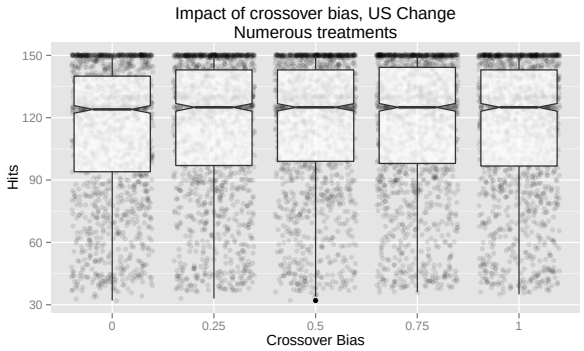

Figure 7: Impact of crossover bias on the number of hits for the U.S. Change problem for all the combinations of parameters in Table 1.

of these runs resulted in the discovery of a "perfect" solution. By comparison, only 1 or 2 runs out of 100 resulted in "perfect" solutions for each of the other crossover bias probabilities. This difference is statistically significant ($p \leq 0.03$).

4.1.2 ORDERTREE

It should be noted that, for the ORDERTREE problem we used the full set of parameters outlined in Section 3, with one exception: the population size for the ORDERTREE runs was limited to 1,024. This was a consequence of the large trees generated for this problem, resulting in out of memory errors for the larger population size of 10,240.

ORDERTREE has a difficulty tuning parameter n; we used $n = 10$, which meant that the fitnesses ranged from 0 to 1,022 with larger values being better, so the best possible fitness is 1,022. The details for the ORDERTREE fitness computation are described in [7].

Figure 6 shows the impact of crossover bias across all parameter combinations. For binary tournaments, adding crossover bias leads to improved performance when compared to no crossover bias, but we observed a drop in fitness from bias 0.75 to bias 1.00, where the fitness for 1.00 is actually slightly below that for 0.50 as well. All the binary tournament differences are statistically significant ($p < 0.03$), with the exception of the difference between bias 0.25 and bias 1.00, so bias 0.75 is the clear winner in this scenario. This drop in fitness for bias 1.00 is consistent across the other three tournament sizes as well. However, it is also apparent from the figure that in general the impact of crossover bias lessens with the larger tournament sizes, both in the increase in fitness up to bias 0.75, and the drop from there to bias 1.00.

4.2 U.S. Change Problem

In contrast to the structural problems discussed in the previous section, which used fitness as the measure of success of a run, for the U.S. Change problem we measured success in "hits" (the number of test cases that are correctly solved). There are 150 test cases in our implementation, so an optimal program will have a score of 150 hits.

Figure 7 shows the impact of crossover bias on the number of hits for the U.S. Change problem across the full collection of parameter settings. This suggests that in general there is little consistent impact of crossover bias; while most of the differences in this plot are not statistically significant, two are: the differences between crossover bias 0.00 and crossover bias 0.50 and 0.75 are both statistically significant ($p \leq 0.015$), even if numerically small.

However, if we limit our attention to the bias-effective parameter settings (Section 3), then we find that crossover bias has a substantial and statistically significant impact, as is seen in Figure 8. Here the bulk of these pairwise differences

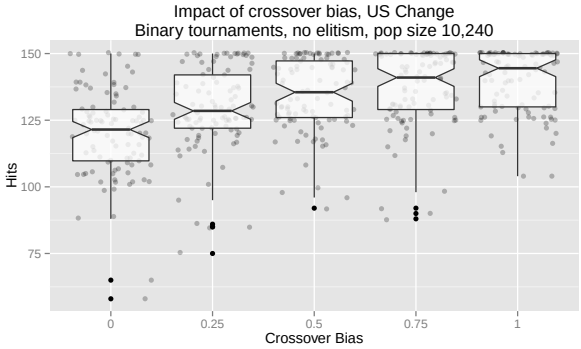

Figure 8: Impact of crossover bias on the number of hits for the U.S. Change problem, limited to bias-effective parameters.

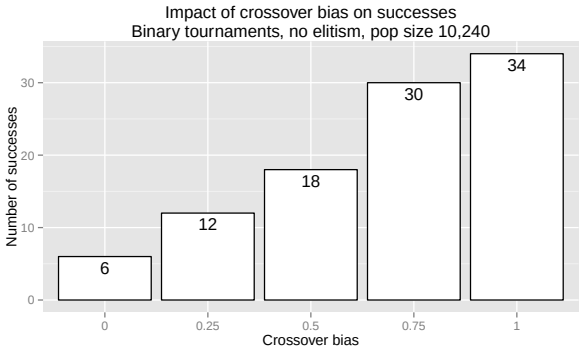

Figure 9: Impact of crossover bias on the number of successes runs for the U.S. Change problem when using bias-effective parameters.

are statistically significant ($p < 0.0002$). The major exception is the difference between crossover bias 0.75 and 1.00 ($p = 0.43$). Two other adjacent pairs have p-values slightly above 0.05: crossover bias 0.25 *vs.* 0.50 ($p = 0.0504$), and 0.50 *vs.* 0.75 ($p = 0.0802$).

If complete success was considered vital (which might be the case if we were evolving software for use in a production system), then it would make sense to see if crossover bias has a significant impact on the success rate. Figure 9 shows the number of successes for the various crossover bias values when using the bias-effective parameter settings. None of the adjacent differences (for example, crossover bias 0.50 *vs.* 0.75) are statistically significant, while most of the non-adjacent differences (for example, crossover bias 0.25 *vs.* 0.75) are statistically significant, with the exceptions being crossover bias 0.00 *vs.* 0.50, and bias 0.50 *vs.* 1.00 ($p = 0.094$ in both cases).

4.3 Regression Problems

Regression problems attempt to evolve a function that is able to match a given set of input-output pairs. Success is here measured in hits, where a "hit" is awarded when the output of the evolved function is within 0.01 of the target value.

4.3.1 Pagie-1 Problem

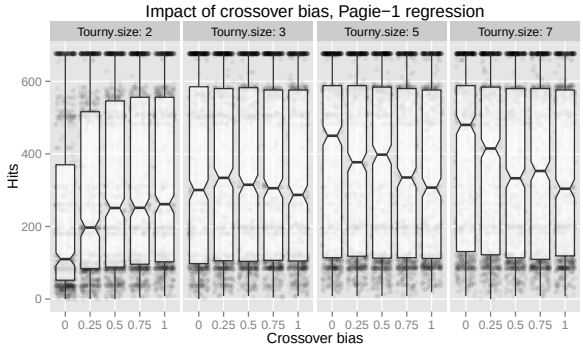

Figure 10: Impact of crossover bias on the number of hits for the Pagie-1 symbolic regression problem, broken out across the four different tournament sizes. The maximum number of possible hits is 676.

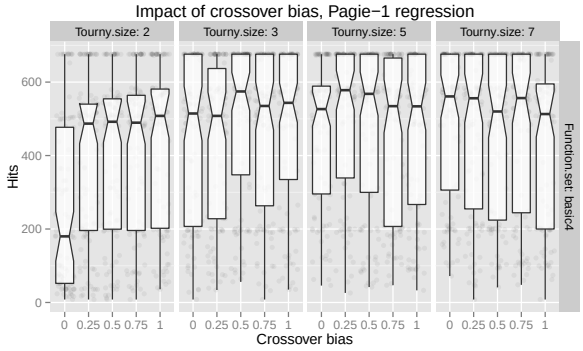

Figure 11: Impact of crossover bias on the number of hits for the Pagie-1 symbolic regression problem, broken out by tournament size, limited to bias-effective elitism (0) and population size (10,240).

Figure 10 shows the impact of crossover bias on the number of hits for the Pagie-1 regression problem, separated out by tournament size. Figure 11 is limited to population size 10,240 and no elitism.

Focusing on the binary tournament data in Figure 11, the differences between crossover bias 0.00 (standard subtree crossover) and all the other positive crossover bias values are statistically significant ($p < 0.0002$). None of the differences among 0.25, 0.50, 0.75, and 1.00 are statistically significant, however, and none of the differences for tournament sizes 3, 5, or 7 are statistically significant. This indicates that for binary tournaments, including crossover bias substantially and significantly improves the hit rate, although all bias rates above 0.25 are very similar. For larger tournament sizes, however, it appears that adding crossover bias doesn't have a significant effect and in some cases (e.g., tournament sizes 5 and 7 in Figure 10) may hurt performance.

4.3.2 Sine Problem

For the sine regression problem, we limited our runs to bias-effective and non-bias-effective settings as described in Section 3.

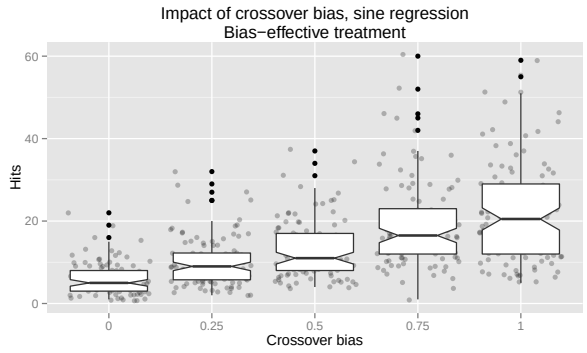

Figure 12: Impact of crossover bias on the sine symbolic regression problem with bias-effective settings.

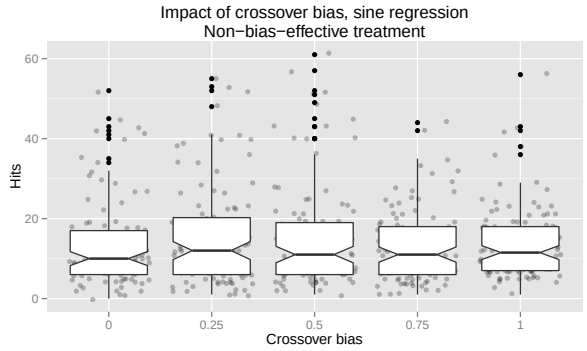

Figure 13: Impact of crossover bias on sine symbolic regression problem with non-bias-effective settings.

Figure 12 shows the hits results for the sine regression problem using bias-effective settings. All these differences are statistically significant ($p < 10^{-5}$) except for the difference between the bias of 0.75 and 1.00. In Figure 2, the left panel illustrates the impact of crossover bias over time when using the bias-effective settings, demonstrating that the effects appear early in the run. Here, the fitnesses are consistently different throughout the time of the runs, with higher crossover bias settings arriving at fitter individuals earlier in the runs, although the differences are shrinking towards the end as many runs find fairly fit individuals.

Figure 13 shows the results for the sine regression problem using the non-bias-effective parameter settings; none of these differences are statistically significant. In Figure 2, the right panel illustrates the impact of crossover bias over time when using the non-bias-effective settings, demonstrating that with these settings the runs tend to be highly similar throughout the duration of the runs, regardless of crossover bias setting.

5. DISCUSSION

In general our results indicate that adding crossover bias improves performance, either generally (Figure 3) or when combined with certain other parameter choices (e.g., Figure 10). In some cases the improvements induced by crossover bias are quite substantial (e.g., Figures 5, 9, and 12), while in other situations the improvements were

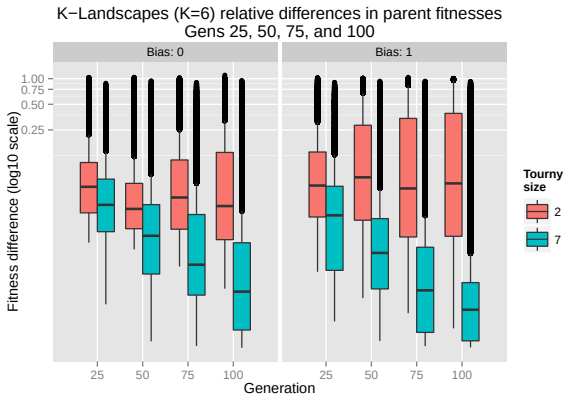

Figure 14: Plot of the normalized differences in parent fitnesses, K Landscape problem. The left-hand (and higher) boxplot in each pair of boxplots is for the binary tournament runs.

quite small in magnitude even when statistically significant (Figure 3). Not surprisingly, however, crossover bias is not universally helpful, and there are settings where some crossover bias improves performance more than when always applying crossover bias (Figure 6), and other settings where all tested crossover bias amounts were detrimental (tournament sizes 5 and 7 in Figure 10).

One general trend seems to be that crossover bias is more effective when using the *bias-effective* parameter settings, namely smaller tournaments (size 2 instead of 7), larger populations (10,240 instead of 1,024), and no elitism. One possible explanation for this pattern is that when using the *non-bias-effective* settings, the difference in fitness between the two parents is likely to be closer than when using bias-effective settings; this reduction in the difference between the fitness of the parents is thus likely to minimize the impact of crossover bias.

To test this hypothesis, we looked at the impact of tournament size on the relative difference in parent fitness for the K Landscapes problem and the sine regression problem. For each crossover event, we compute the relative difference in parent fitness as

$$|f_A - f_B|/(f_A + f_B)$$

where f_A and f_B are the fitnesses of the two chosen parents A and B. This has a minimum value of 0 when the two fitnesses are the same, and a maximum value approaching 1 for the case where one of the values is nearly 0.

Figure 14 shows the distribution of the relative differences in parent fitness for the K Landscapes problem both with (right-hand panel) and without (left-hand panel) crossover bias. For tournament size 7, the difference in fitness drops significantly as the run progresses; this, combined with the substantial improvements in fitness, suggests that both of the parents are highly fit. When using binary tournaments, the relative differences in parent fitness remains fairly stable (and higher than for tournament size 7), which would suggest that crossover bias would continue to be effective throughout the run when using binary tournaments. We see similar results in the sine regression problem, as illustrated in Figure 15.

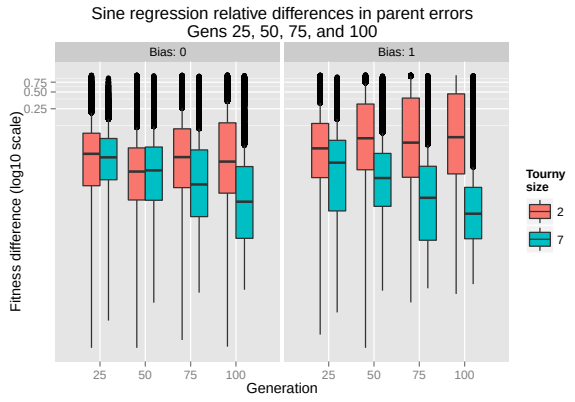

Figure 15: Plot of the normalized differences in parent errors, sine symbolic regression problem. The left-hand (and higher) boxplot in each pair of boxplots is for the binary tournament runs.

Our results also indicate that crossover bias may increase certain types of selection pressure, which could potentially increase the likelihood of premature convergence. There is a heightened focus on the fitter of the two parents which increases the chance of the more fit parent's semantics being carried across into the next generation. This might limit the diversity of the root structures causing premature convergence. Further research is needed to understand if this does in fact lead to premature convergence in some cases, but it's possible, for example, that this explains the fact that using some crossover bias (0.75) is better than always using crossover bias in the ORDERTREE problem (see, e.g., Figure 6).

6. CONCLUSIONS

Based on previous observations of interesting asymmetries in the behavior of sub-tree crossover (e.g., [9]), we introduced the idea of *crossover bias*, where we probabilistically force the more fit parent to be the *root parent* when performing crossover. We then applied this new bias to several different problems. In general, we found that adding crossover bias either helps or is neutral, although there were cases where its effect was detrimental.

Given these results, it seems worthwhile to at least try crossover bias, especially in settings where there are reasons to suspect that there are substantial differences between parental fitnesses. This is likely to be the case wherever selection pressure is weak (e.g., our use of binary tournaments), but could also happen when there are unavoidable steep changes in fitness, or in settings where the fitness function may vary over either time or as a function of some other variable such as location.

A question to consider based on these results is whether or not we should use larger tournaments instead of crossover bias, since in many cases the results with tournament size 7 without crossover bias was as good or better than those obtained with binary tournaments and crossover bias, e.g., Figure 4. According to Gustafson et al. [6], however, matings between similar parents are more likely to produce no change in fitness, whereas a more diverse crossover increases the chances of offspring improving fitness. Larger tournaments are more likely to select two parents that are similar, at least in fitness, as seen in Figures 14 and 15; an excellent illustration of this can be seen in Figure 2 in [2]. Another attempt to find a balance of diversity between parents is called sexual selection proposed by Goh et al. [5]. Their sexual selection system mates individuals chosen at random (with no regard to fitness) with individuals chosen via tournament selection. This creates a strong asymmetry between the parents which appears to increase diversity in certain circumstances in ways that improve performance. Based on these examples, there may be circumstances where combining crossover bias with a weaker selection mechanism might be a reasonable alternative to using a larger tournament size.

Another outstanding question is how crossover bias affects generalization, especially since it's possible that crossover bias increases selection pressure and could consequently encourage overfitting. We didn't explore this question here, but it would be useful to expand the existing work to include validation to see what impact, if any, crossover bias has on how well evolved solutions generalize to unseen data.

While this paper focuses on asymmetry in the context of tree-based GP and sub-tree crossover, it's important to note that asymmetries like this are common in many evolutionary systems, both biological and artificial. Much eukaryote reproduction is sexual, and brings with it numerous sex-linked traits and related asymmetries; this may play an important role in speciation [16], arguably one of the most crucial of biological asymmetries. Many evolutionary computation systems other than tree-based GP also have significant asymmetries. Linear GP systems [3] and stack-based GP systems [19], for example, have asymmetries where the last instructions executed can have a disproportionate impact on the results. Similarly, changes near the front of grammatical evolution [13] strings will have a disproportionate impact by determining the important early choices in the grammar productions. Generally, these asymmetries weren't intentional, but were instead simple artifacts of other system design decisions. Hence, the potential impact of these asymmetries has been largely unstudied. The results presented here suggest that it may be important to more thoroughly explore the impact of such asymmetries, both to better understand the performance and behavior of existing systems, but also as an aid to the design and discovery of new tools like recombination operators that leverage these asymmetries.

Acknowledgements

Many thanks to the members of the Hampshire College Computational Intelligence Lab for suggestions and feedback as this work developed, with particular thanks to Thomas Helmuth, William La Cava, and Lee Spector. Thanks also to Thomas Helmuth for introducing us to the U.S. Change problem. Thanks to W. B. Langdon and our anonymous reviewers for valuable feedback and numerous suggestions.

David Donatucci's work was supported in part by a Morris Academic Partners grant from the Office for Academic Affairs and Dean at the University of Minnesota, Morris.

7. REFERENCES

[1] L. Beadle and C. G. Johnson. Semantically driven crossover in genetic programming. In *IEEE World Congress on Computational Intelligence*, pages 111–116, 2008.

[2] G. D. Boetticher and K. Kaminsky. The assessment and application of lineage information in genetic programs for producing better models. In *Information Reuse and Integration, 2006 IEEE International Conference on*, pages 141–146, Sept 2006.

[3] M. F. Brameier and W. Banzhaf. *Linear genetic programming*. Springer Science & Business Media, 2007.

[4] B. Burlacu, M. Affenzeller, M. Kommenda, S. Winkler, and G. Kronberger. Visualization of genetic lineages and inheritance information in genetic programming. In *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*, pages 1351–1358. ACM, 2013.

[5] K. Goh, A. Lim, and B. Rodrigues. Sexual selection for genetic algorithms. *Artificial Intelligence Review*, 19(2):123–152, 2003.

[6] S. Gustafson, E. Burke, and N. Krasnogor. On improving genetic programming for symbolic regression. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 1, pages 912–919 Vol.1, Sept 2005.

[7] T.-H. Hoang, N. X. Hoai, N. T. Hien, R. I. McKay, and D. Essam. ORDERTREE: a new test problem for genetic programming. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 807–814. ACM, 2006.

[8] J. McDermott, D. R. White, S. Luke, L. Manzoni, M. Castelli, L. Vanneschi, W. Jaskowski, K. Krawiec, R. Harper, K. De Jong, et al. Genetic programming needs better benchmarks. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 791–798. ACM, 2012.

[9] N. F. McPhee, D. Donatucci, and M. K. Dramdahl. Analysis of genetic programming ancestry using a graph database. In *Proceedings of Midwest Instruction and Computing Symposium*. MICS, 2014.

[10] N. F. McPhee and N. J. Hopper. Analysis of genetic diversity through population history. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1112–1120. GECCO, 1999.

[11] N. F. McPhee, B. Ohs, and T. Hutchison. Semantic building blocks in genetic programming. In *Proceedings of the 11th European Conference on Genetic Programming*, EuroGP'08, pages 134–145, Berlin, Heidelberg, 2008. Springer-Verlag.

[12] A. Moraglio, K. Krawiec, and C. G. Johnson. Geometric semantic genetic programming. In *Parallel Problem Solving from Nature-PPSN XII*, pages 21–31. Springer, 2012.

[13] M. O'Neill and C. Ryan. *Grammatical evolution: evolutionary automatic programming in an arbitrary language*, volume 4. Springer Science & Business Media, 2003.

[14] L. Pagie and P. Hogeweg. Evolutionary consequences of coevolving targets. *Evolutionary computation*, 5(4):401–418, 1997.

[15] R. Poli, W. B. Langdon, and N. F. McPhee. *A Field Guide to Genetic Programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008. (With contributions by J. R. Koza).

[16] A. Qvarnström and R. I. Bailey. Speciation through evolution of sex-linked genes. *Heredity*, 102(1):4–15, 2009.

[17] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014.

[18] C. Ryan. Pygmies and civil servants. In *Advances in Genetic Programming*, pages 243–263. MIT Press, 1994.

[19] L. Spector and A. Robinson. Genetic programming and autoconstructive evolution with the push programming language. *Genetic Programming and Evolvable Machines*, 3(1):7–40, Mar. 2002.

[20] N. Q. Uy, N. X. Hoai, M. O'Neill, R. I. McKay, and E. Galván-López. Semantically-based crossover in genetic programming: application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines*, 12(2):91–119, 2011.

[21] N. Q. Uy, N. X. Hoai, M. O'Neill, R. I. McKay, and D. N. Phong. On the roles of semantic locality of crossover in genetic programming. *Information Sciences*, 235:195–213, 2013.

[22] L. Vanneschi, M. Castelli, and L. Manzoni. The K landscapes: a tunably difficult benchmark for genetic programming. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 1467–1474. ACM, 2011.

[23] D. R. White, J. McDermott, M. Castelli, L. Manzoni, B. W. Goldman, G. Kronberger, W. Jaśkowski, U.-M. O'Reilly, and S. Luke. Better GP benchmarks: Community survey results and proposals. *Genetic Programming and Evolvable Machines*, 14:3–29, 2013.

[24] H. Wickham. *ggplot2: Elegant graphics for data analysis*. Springer New York, 2009.

[25] H. Zhan. A quantitative analysis of the simplification genetic operator. In *Proceedings of the 2014 conference companion on Genetic and evolutionary computation companion*, pages 1077–1080. ACM, 2014.