Impact of Crossover Bias in Genetic Programming

Nicholas Freitag McPhee, M. Kirbie Dramdahl, David Donatucci
Division of Science and Mathematics
University of Minnesota, Morris
Morris, MN USA-56267
{mcphee, dramd002, donat056}@morris.umn.edu

ABSTRACT

In tree-based genetic programming 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 other parent (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*, which allows us to bias the system in favor of having the more fit parent be the root parent.

We applied crossover bias to a variety of problems. In most cases we found that using crossover bias either improved performance or had no impact - although the effectiveness is somewhat dependent on the problem, and significantly dependent on other parameter choices. Our results do, however, indicate the possibility that crossover bias may increase selection pressure and premature convergence - undesirable behavior, as it encourages a genetic programming run to arrive at a solution too quickly, in the process potentially excluding more accurate solutions for a more generalized one.

Categories and Subject Descriptors

[]

General Terms

Keywords

genetic programming, crossover bias, root parent

1. INTRODUCTION

As Figure 1 illustrates, the widely used sub-tree crossover operator in tree-based genetic programming (GP) [10] 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., [6] 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 [2, 5, 7] that the root parent often contributes more to the semantics of the resulting individual. This is in part due to the tendency of the root parent contributing more overall nodes, but also 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.

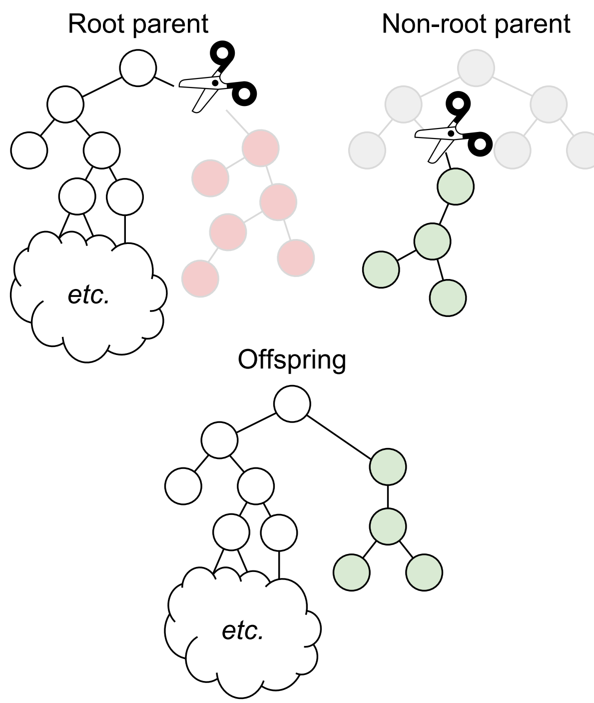

Figure 1: Sub-tree crossover, illustrating the asymmetric role of the *root* and *non-root* parents.

An important impact of this asymmetry was recently noted in [5], where they observed that the fitness of the offspring tended to be better when the root parent had a better fitness than the non-root parent. To explore this fur-

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

Algorithm 1 Crossover bias

Require: RP and NRP are two individuals chosen as parents, with RP initially (and randomly) chosen to be the root parent.

```
if RANDOM(0, 1) ≤ XO_bias then          ▷ Do we force bias?
    if FITNESS(RP) worse than FITNESS(NRP) then
        RP, NRP := NRP, RP       ▷ Swap parents so RP is more fit
    end if
end if
```

ther, 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. Here we apply crossover bias (defined in Section 2) to a variety of test problems (described in 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 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.

Should this next paragraph be moved the Conclusions section instead of having it here?

While this paper focuses on the 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 [11], arguably one of the most crucial of biological asymmetries. Many evolutionary computation systems other than tree-based GP also have significant asymmetries. Many linear GP systems [1] and stack-based GP systems [13], 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 [8] strings will have a disproportionate impact by determining the important early choices in the grammar productions. The results presented here suggests that it may be important to more thoroughly explore the impact of these 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.

2. CROSSOVER BIAS

For this work, 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 if the random value in Algorithm 1 didn't cause crossover bias to be applied, then the parents were left in their original random order, meaning that there is a 50% chance that the root parent is the more fit parent even without any crossover bias.

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

- 0.00 bias - no crossover bias is applied;
- 0.25 bias - system forced to use the more fit parent as the root parent in 25% of cases;
- 0.50 bias - system forced to use the more fit parent as the root parent in 50% of cases;
- 0.75 bias - system forced to use the more fit parent as the root parent in 75% of cases; and
- 1.00 bias - system forced to always use the more fit parent as the root parent, sometimes referred to as "with crossover bias".

The 0.00 case, where no crossover bias is applied, is just standard sub-tree crossover, where there will be a 50% chance of the root parent being the more fit parent.

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 revealed that there was no significant difference from no bias (i.e., differences between reverse bias and no bias weren't statistically significant), 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 genetic programming performance, we experimented with five problems: K Landscapes with $K = 6$ [14], ORDERTREE [3], U.S. Change, Sine regression [10], and Pagie-1 regression [9]. Three of these (K Landscapes, ORDERTREE, and Pagie-1 regression) are taken from recent benchmark suggestions [15]. U.S. Change is a program synthesis problem that has proven interesting for evolutionary program synthesis[1], and sine regression is the example problem used in earlier work on the impact of root and non-root parents [5].

(**well? there is a paper that sort of describes it. search "us change problem" in gecco last year**)

Good catch. That's actually one of Lee's undergrad students, and he doesn't actually properly define the problem in the paper, unfortunately, so I wasn't sure whether to even bother linking to it. We certainly can, though.

The one problem not described elsewhere in the literature is the U.S. Change problem, a program synthesis 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*, *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 $[0, 150)$.

[1]personal communication

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

Parameter	Values
Crossover bias	0, 0.25, 0.5, 0.75, and 1
Tournament size	2, 3, 5, and 7
Elitism %	0 and 1% (& sometimes 0.1%)
Population size	1,024 and 10,240
# of generations	100
# of runs per treatment	100

Table 1: Shared parameters

All our experiments used the previously published settings, function sets, etc., with the exception of the parameters listed in Table 1,[3] 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 the table. 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 2, for example, includes the results of runs with all four tournament sizes, both elitism proportions, and both population sizes; given that there are 100 runs for each combinations of parameters, each box represents the results of 1,600 runs, i.e., 1,600 data points.

As Section 4 shows, there were some 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.

All experiments were run using a copy of ECJ 21[4] that we modified to support crossover bias.[5]

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 with R [12] and plots were generated using the ggplot2 package [16].

[3]The function set for Pagie-1 specified in [9] is simply +, -, *, and % (protected division). The function set given in [4] 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.

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

[5]Our intent is to submit our modifications back to the ECJ group to make it easy for other ECJ users to experiment with crossover bias.

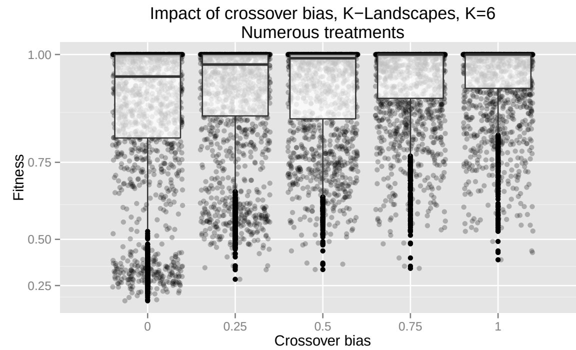

Figure 2: Impact of crossover bias on fitness for K Landscapes problem, $K = 6$ for a variety of treatments.

4.1 Structural Problems

4.1.1 K Landscapes

Figure 2 shows the impact of crossover bias on this problem across all the combinations of parameter values. 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 3, 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 ($p < 6e{-}08$). This continues to be true for tournament size 3 - all the differences are statistically significant except for that between bias 0.50 and 0.75 and that was very close ($p = 0.05620$). None of the differences for tournament size 5 are significant, and so we will past over it here. For tournament 7, however, it does look like the reverse is true, where increasing crossover actually hurts fitness. Almost none of the differences for tournament size 7 are statistically significant, however, with the only exception being the difference between 0.25 and 1.00 ($p = 0.21$ using a pairwise Wilcoxon test with Holm correction). **This paragraph might move to the Discussion or Conclusions section if we keep it, or something like it.**

Figure 4 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 2. 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. Out of 100 runs with a crossover bias setting of 1.00, 15 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

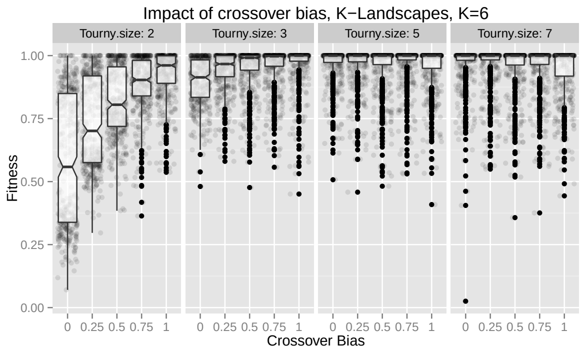

Figure 3: Impact of crossover bias on fitness for K-Landcapes problem, K=6, for various tournament sizes.

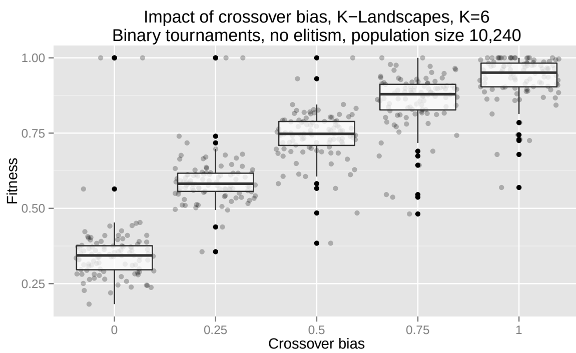

Figure 4: Impact of crossover bias on fitness for K Landscapes problem, $K = 6$, restricted to binary tournament selection, no elitism, and population size 10,240.

crossover probabilities. This difference is statistically significant with $p \leq 0.03$ using a pairwise test of proportions.

4.1.2 ORDERTREE

It should be noted that, for the ORDERTREE problem, a full sweep of the following parameters outlined in Section 3, with on 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.

Initial runs of this problem with (1.00) and without (0.00) crossover bias across all four tournament sizes demonstrated that in each case, adding crossover bias improved fitness. Additionally, these improvements were statistically significant ($p < 0.002$ for each pairing using a pairwise Wilcoxon with Holm correction).

However, in the next set of runs, using the full range of crossover bias values but limited to binary tournaments, we observed a drop in fitness from bias 0.75 to bias 1.00, actually dropping slightly below that for 0.50 as well. All these 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. **I**

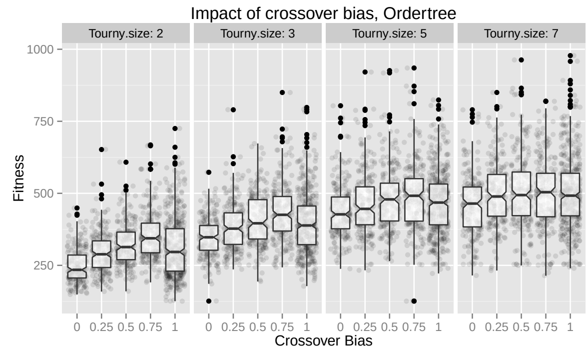

Figure 5: Impact of crossover bias on fitness for ORDERTREE problem for multiple tournament sizes.

(Nic) suspect this is actually quite interesting and might open interesting doors to things like premature convergence and need for being able to have a least a little variation in the system for a problem like this. Maybe we should talk about that in the "Future work" section?

Figure 5 generalizes this to cover all four tournament sizes. The figure demonstrates that the drop in fitness for bias 1.00 observed for binary tournaments 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 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 hits score of 150.

Figure 6 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, but a pairwise Wilcoxon test with Holm correction indicates that 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 binary tournaments, no elitism, and the larger population size of 10,240; then we find that crossover bias has a substantial and statistically significant impact, as is seen in Figure 7. Here the bulk of these pairwise differences are statistically significant ($p < 0.0002$ using a pairwise Wilcoxon test with Holm correction). The major exception is the difference between crossover bias 0.75 and 1.00 ($p = 0.43078$). Two other adjacent pairs have p-values slightly above 0.05: crossover bias 0.25 *vs.* 0.50 ($p = 0.05036$), and 0.50 *vs.* 0.75 ($p = 0.08019$).

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 8 shows

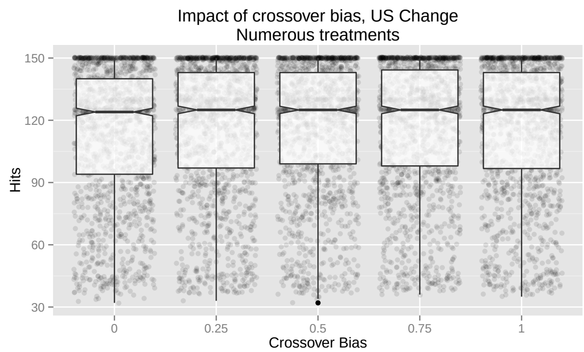

Figure 6: Impact of crossover bias on the number of hits for the U.S. Change problem across a variety of treatments.

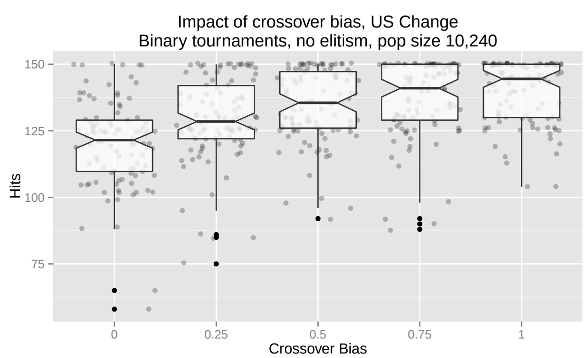

Figure 7: Impact of crossover bias on the number of hits for the U.S. Change problem, limited to binary tournaments, no elitism, and population size 10,240.

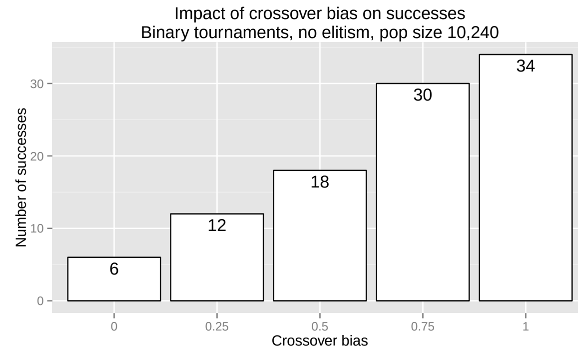

Figure 8: Impact of crossover bias on the number of successes runs for the U.S. Change problem when using binary tournaments, no elitism, and population size 10,240.

the number of successes for the various crossover bias values when using binary tournaments, no elitism, and population size 10,240. A pairwise test of proportions (chi-squared) with Holm correction indicates that while non of the adjacent differences (for example, crossover bias 0.50 *vs.* 0.75) are statistically significant, 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 and 1.00 ($p = 0.09361$ in both cases).

4.3 Symbolic Regression Problems

4.3.1 Pagie-1 Problem

In addition to the parameters outlined in Section 3, the following parameters were also implemented:

- basic4 $(+, -, \times, \div)$ and koza2 (complete - not discussed) function sets, and
- with and without Tarpeian bloat control.

When we remove koza2 from Fig 9 then I think we'll be able to remove any mention of koza2, which means we can drop the bullet list and just make this a sentence.

Figure 9 shows the impact of crossover bias on the number of hits for the Pagie-1 regression problem, separated out by both tournament size and function set used.

The results, as can be seen in Figure 9, vary significantly. However, if limited to the following parameters:

- crossover biases of 0.00, 0.25, 0.50, 0.75, and 1.00;
- population size 10,240;
- no elitism;
- tournament sizes of 2, 3, 5, and 7;
- basic4 $(+, -, \times, \div)$ function set; and
- Tarpeian bloat control;

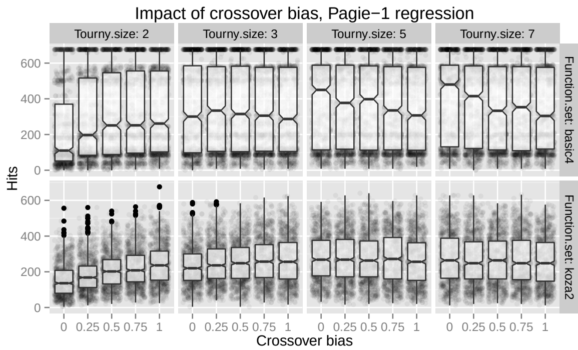

Figure 9: Impact of crossover bias on the number of hits for the Pagie-1 symbolic regression problem, broken out for the four different tournament sizes (2, 3, 5, and 7) and the two different function sets (basic4 and koza2). The maximum number of possible hits is 676.

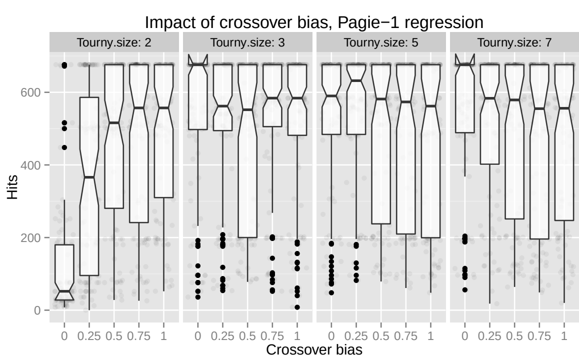

Figure 10: Impact of crossover bias on the number of hits for the Pagie-1 symbolic regression problem, broken out by tournament size (2, 3, 5, and 7). These runs use the "basic4" function set ($+, -, \times, \div$), no elitism, population size 10,240, and Tarpeian bloat control. The maximum number of possible hits is 676.

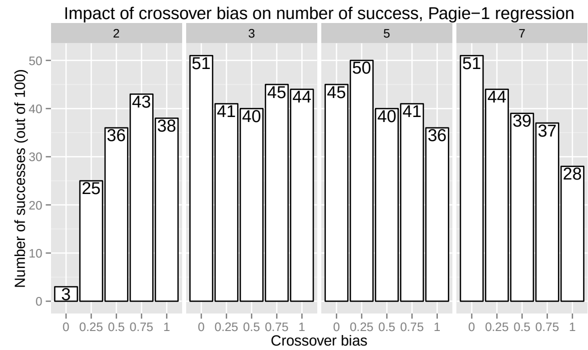

Figure 11: Impact of crossover bias on the number of successes (runs that exactly solve the problem) for the Pagie-1 symbolic regression problem broken out by tournament size (2, 3, 5, and 7). These runs use the "basic4" function set (just $+, -, \times, \div$), no elitism, population size 10,240, and Tarpeian bloat control.

then the complexities of Figure 9 simplify to Figure 10. For now we will focus on this specific subset of the data.

Focusing on the data in Figure 10, the differences between crossover bias 0.00 (standard subtree crossover) and all the other positive crossover bias values are statistically significant ($p < 10^{-12}$), and all of the differences between crossover bias 0.25 and higher biases are significant ($p < 0.007$); however, none of the differences among 0.50, 0.75, and 1.00 are statistically significant. None of the differences for tournament sizes 3, 5, or 7 are statistically significant, although the difference between crossover biases 0.00 and 1.00 are very close ($p = 0.053$) for tournament size 7. 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 tournament size 7, however, it appears that adding crossover bias tends to reduce the hit rate, although in a less substantial and significant manner.

Figure 11 shows the number of successes (runs that exactly solve the problem). Almost none of these differences are statistically significant, with the major exception being the small number of successes (3) for binary tournaments without bias, which is significantly different from all the other bias values for binary tournaments ($p < 0.0002$). The one other exception is for tournament size 7, the difference between the number of successes with crossover biases 0.00 and 1.00 is significant ($p = 0.015$).

4.3.2 Sine Problem

For the sine regression problem, we limited our runs to two parameter treatments, referred to as bias-effective setting and and non-bias-effective settings respectively, discussed in detail in Section 3.

Figure 12 shows the hits results for the sine regression problem using binary tournaments, no elitism, and population size 10,240. All these differences are statistically significant ($p < 10^{-5}$ using a pairwise Wilcoxon rank sum test) except for the difference between the bias of 0.75 and 1.00.

Figure 13 shows the results for the sine regression problem using tournament size 7, 0.1% elitism, and smaller popula-

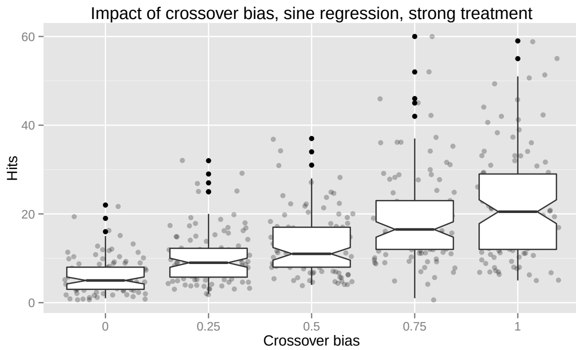

Figure 12: Impact of crossover bias on the sine symbolic regression problem with bias-effective settings: binary tournaments, no elitism, and population size 10,240.

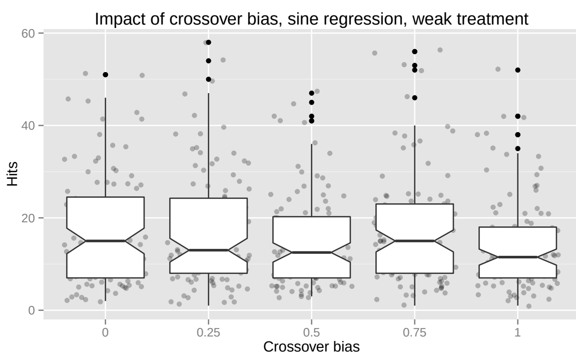

Figure 13: Impact of crossover bias on the sine symbolic regression problem with non-bias-effective settings: tournament size 7, 0.1% elitism, and population size 1,024.

tions of size 1,024. None of these differences are statistically significant using a pairwise Wilcoxon rank sum test.

5. DISCUSSION

Why does all this happen this way? For example, why does crossover bias have a much stronger effect when using binary tournaments than when using larger tournament sizes such as 7. One possible explanation is that with large tournaments, the difference in fitness between the two parents is likely to be closer, because the larger tournaments help ensure that both parents are from the more highly fit part of the population. Since both of the contributing structures will most likely be from highly fit parents, the resulting offspring will likely obtain similiar fitness as one of these highly fit parent . To better understand this, Figure 16 shows the distribution of relative difference in parent errors in the sine regression problem. For each crossover event, the relative difference in parent errors is

$$|e_A - e_B]/(e_A + e_B)$$

where e_A and e_B are the errors of the two chosen parents A and B. This has a minimum value of 0 when the two errors are the same, and a maximum value approaching 1 for the

Figure 14: Impact of crossover bias on the sine symbolic regression problem with binary tournaments, no elitism, and population size 10,240. Move this to where we define bias-effective parameters.

Figure 15: Impact of crossover bias on the sine symbolic regression problem with tournament size 7, 0.1% elitism, and population size 1,024. Move this to where we define non-bias-effective parameters, combined with the previous figure as a pair of panels.

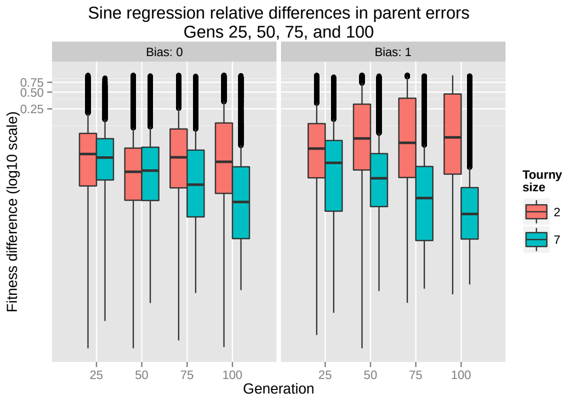

Figure 16: Plot of the normalized differences in parent errors from some sine runs. Explain this.

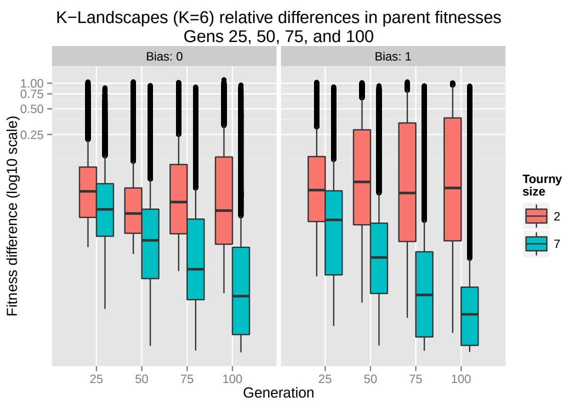

Figure 17: Plot of the normalized differences in parent fitnesses from some K Landscape runs. Explain this.

case where one of the errors is nearly 0. For larger tournament sizes, the difference in fitness drops significantly as the run progresses suggesting that the both of the parents are highly fit. Throughout the run, however, small tournament sizes do not exhibit these changes. (**not entirely sure what to say here**)

We see similar results for the K Landscapes problem, as illustrated in Figure 17.

6. CONCLUSIONS

In most of these experiments we found better results with tournament sizes of 7 than with binary tournaments, and in general using larger tournaments appears to wash out much of the impact of crossover bias, so there's a fair question about whether one should just use larger tournaments and ignore crossover bias. **How do we respond to this? I think the answer is something like "It doesn't hurt (at least in our experiments), and it sometimes helps, even for tournament size 7 and with elitism. For interesting problems, you also don't know in advance what your best parameter choices are, so it's at least worth including in your arsenal."**

One outstanding question is how crossover bias affects generalization. Neither of the published versions of the two regression problems specify a validation set, but it's clearly feasible to create validation sets for both of those problems and the U.S. Change problem to see what impact, if any, crossover bias has on how well evolved solutions generalize to unseen data.

Return to the paragraph in the intro about how asymmetries are very common and that these results suggest that we should look at their impact in other EC system. In general these EC system asymmetries weren't intentional design goals, but were instead simple artifacts of other system design decisions, and the potential impact of these asymmetries has been largely unstudied.

7. REFERENCES

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

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

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

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

[5] McPhee, N. F., Donatucci, D., and Dramdahl, M. K. Analysis of genetic programming ancestry using a graph database. In *Proceedings of Midwest Instruction and Computing Symposium* (2014), MICS '14.

[6] McPhee, N. F., and Hopper, N. J. Analysis of genetic diversity through population history. In *Proceedings of the Genetic and Evolutionary Computation Conference* (1999), vol. 2, Citeseer, pp. 1112–1120.

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

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

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

[10] Poli, R., Langdon, W. B., and McPhee, N. F. *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).

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

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

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

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

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

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