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

This is probably too long. People often recommend keeping the abstract to somewhere between 150 and 250 words, and this is closer to 500. For the initial submission that's OK, but we may want to move some of this to the introduction and trim down the abstract somewhat.

In tree-based genetic programming with sub-tree crossover, the parent contributing the root portion of the tree (which we refer to as the *root parent*) often contributes more to the semantics of the resulting child than the other parent (the *non-root parent*). In previous research, we found 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. To better understand the impact of this bias, we implemented several levels of crossover bias, including 0% bias (root individual chosen randomly, as in traditional sub-tree crossover), 100% bias (the stronger parent is always chosen to be the root parent), 50% bias (bias implemented in half the cases, and the other half chosen randomly), and reverse bias (the weaker parent is always chosen as 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. 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.

Our results also demonstrate that the effectiveness of crossover bias is somewhat dependent on the problem, and significantly dependent on other parameter choices. In particular it appears that crossover bias has the largest impact when selection pressure is weaker, and the differences in the fitness of the parents is thus likely to be larger. We also found that the use of elitism reduced the influence of crossover bias. It's possible that crossover bias acts to some degree as an "elitism" operator, making it more likely that the semantics of more fit individuals are copied into the next generation; thus if traditional elitism is being employed this effect is less visible. Another possible explanation for this is that if the most fit individuals are automatically being carried over, there is perhaps less need to produce new, fitter individuals via crossover, reducing or even eliminating the usefulness of crossover bias. Other factors which we found to have potential impact on the effectiveness of crossover bias were tournament size, population size, and possibly the difference in parental fitness.

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

Categories and Subject Descriptors

[]

General Terms

Keywords

genetic programming, crossover bias, root parent

1. INTRODUCTION

Nic should try to capture some nice things Lee Spector said about the importance of asymmetrical recombination operators (which are very common in biology, e.g., sex-linked traits) and how we've identified an asymmetry in subtree crossover and a way to potentially exploit that.

In tree-based genetic programming, sub-tree crossover remains a very common reproduction mechanism [3]. As Figure 1 illustrates, sub-tree crossover is an inherently asymmetric operation, with one parent contributing the root node and, in most cases, a substantially larger total number of nodes than the other parent. In this work we will refer to the individual which contributes the root node as the *root parent* and the other as the *non-root parent.*

Crucial to this work, it has been previously noted [1, 2] that in many cases the root parent also tends to contribute more to the semantics of the resulting individual than the non-root parent. This is in part because of the tendency of the root parent to contribute more overall nodes, but is also because for most function sets the root node (and the nodes near it) have an especially strong impact on the result of evaluating the tree in question. While the details and impact of this asymmetry will depend a great deal on the details of

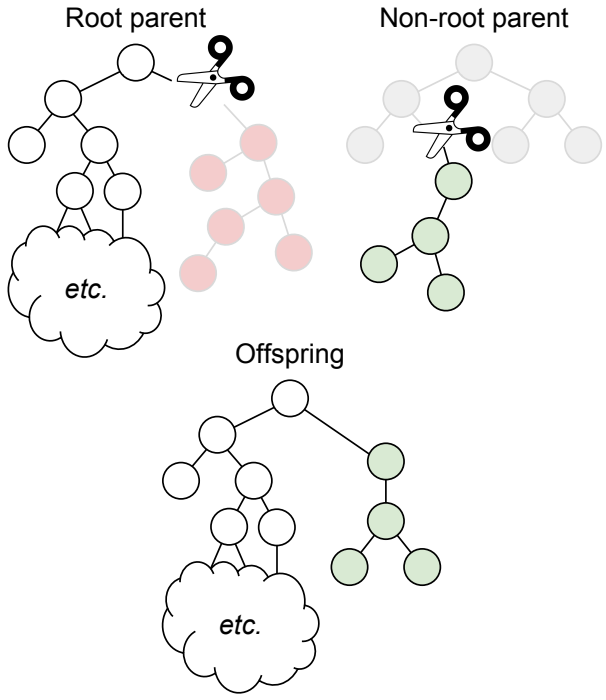

Figure 1: Sub-tree crossover, illustrating the asymmetric role of the *root* and *non-root* parents.

the function set and the particular trees chosen as parents, in many common tree-based GP settings the asymmetry has a quite substantial impact on the semantics of the offspring.

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, for example. **Do we want a bib entry here? Something as simple as a Wikipedia reference, or something "fancier" like a review article on sex-linked traits?** Many evolutionary computation systems other than tree-based GP also have significant asymmetries. Many linear GP systems (**cite something - Wolfgang's book?**) and stack-based GP systems [5], for example, have asymmetries where the last instructions executed can have a disproportionate impact on the results, and changes near the front of grammatical evolution (**cite some GE reference**) strings will have a disproportionate impact by determining the important early choices in the grammar productions. In general these 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.

In previous work [1], it was noted that when performing crossover, if the root parent had a better fitness than the non-root parent, then the fitness of the offspring produced tended to be greater than in the reverse scenario where the less fit parent was the root parent. To explore this further, we implemented what we refer to crossover bias, which allowed us to probabilistically force the genetic programming system to assign the individual with the greater fitness to be the root parent.

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.

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) $\leq$ 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

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, however, the results with reverse bias were effectively the same as using 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 $K = 6$
- Order Tree
- U.S. Change

- Sine regression
- Pagie-1 regression

Three of these (K-Landscapes, Order Tree, and Pagie-1 regression) are taken from recent benchmark suggestions [6]. U.S. Change is a program synthesis problem taken from [**?**], and sine regression is the example problem used in earlier work on the impact of root and non-root parents [1].

Add a footnote about the two function sets for Pagie-1 and their impact.

In the following sections, the terms *"bias-effective settings"* and *"non-bias-effective settings"* are used to describe specific subsets of the parameters settings described above. As is seen in Section 4, there are parameter choices where crossover bias appears to have a fairly strong impact, and other choices where crossover bias has almost no effect on the results. The "bias-effective settings" are:

- binary tournaments,
- no elitism, and
- population size of 10,240;

Non-bias-effective settings, conversely, are:

- tournament sizes of seven individuals,
- elitism of either 0.1% or 1%, and
- population size of 1,024;

All experiments were run using a copy of ECJ 21[1] that we modified to support crossover bias.[2]

4. RESULTS

Unless noted otherwise, all mentions of statistical significance in this section are based on pairwise Wilcoxon tests with Holm corrections. All statistics calculations were performed with R [4] and plots were generated using the ggplot2 package [7].

Talk about the fact that we used pairwise Wilcoxon test with Holm corrections for the bulk of our statistical tests, along with a reference/citation to R as generated inside R. I'm not sure if we want to mention pairwise test of proportions (chi-squared) or not, because I'm not sure how much we'll use it in the paper.

4.1 Structural Problems

4.1.1 K-Landscapes Problems

We did a full sweep of parameters for the K-Landscapes problem with both $K = 2$ and $K = 6$, using various combinations of the following:

- crossover biases of -1.00, 0.00, 0.25, 0.50, 0.75, and 1.00;
- population sizes of 1,024 and 10,240;
- elitism percentages of 0% or 1%; and
- tournament sizes of 2, 3, 5, and 7.

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

[2] 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.

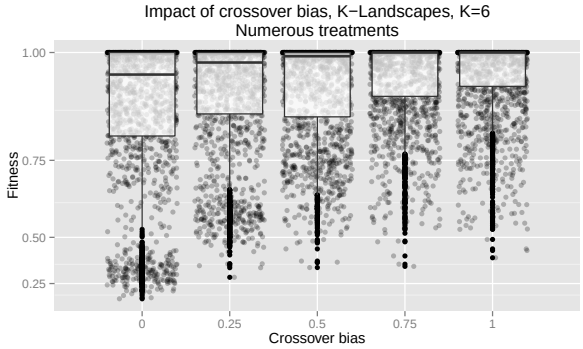

Figure 2: Impact of crossover bias on fitness for K-Landscapes problem, $K = 6$ for a variety of treatments.

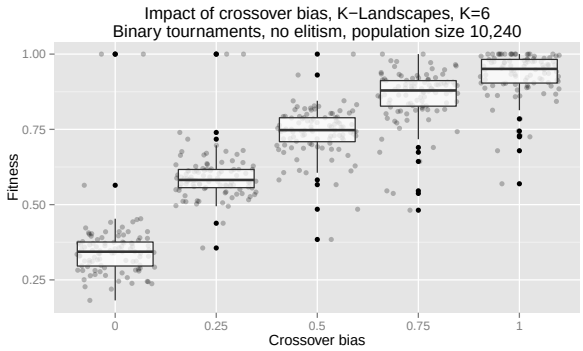

Figure 3: Impact of crossover bias on fitness for K-Landscapes problem, $K = 6$, restricted to binary tournament selection, no elitism, and population size 10,240.

Figure 2 shows the impact of crossover bias on this problem across all the combinations of parameter values (excepting reverse bias). 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.

Figure 3 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 ($p < 10^{-11}$), with the exception of the difference between reverse bias (-1.00; not shown) and no bias (0.00). 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 crossover probabilities. This difference is statistically significant with $p \leq 0.03$ using a pairwise test of proportions.

4.1.2 OrderTree Problem

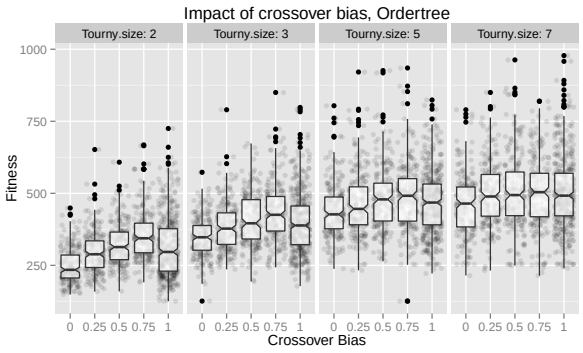

Figure 4: Impact of crossover bias on fitness for OrderTree problem for multiple tournament sizes.

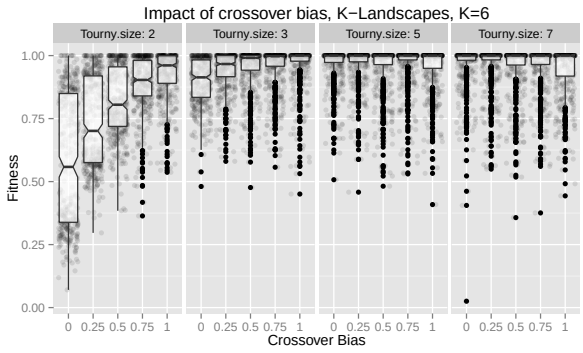

Figure 5: Impact of crossover bias on fitness for K-Landcapes problem, K=6, for various tournament sizes. Move this into the K-Landscapes section next to Figure 2, along with the associated discussion.

For the OrderTree problem, a full sweep of the following parameters was used:

- crossover biases of -1.00, 0.00, 0.25, 0.50, 0.75, and 1.00;
- population size of 1,024;
- elitism percentages of 0% or 1%; and
- tournament sizes of 2, 3, 5, and 7.

While similar to the parameters used for the K-Landscapes problem, one difference is apparent: 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 (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 4 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.

Compare this to the results shown in Figure 5, which plots the corresponding data from the K-Landscapes runs. 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.**

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 ($p = 0.08019$).

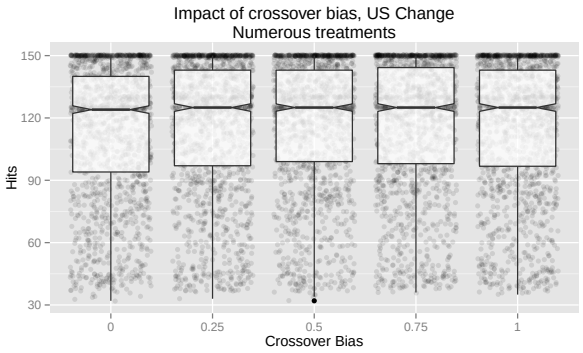

Figure 6: Impact of crossover bias on the number of hits for the U.S. Change problem across a variety of treatments.

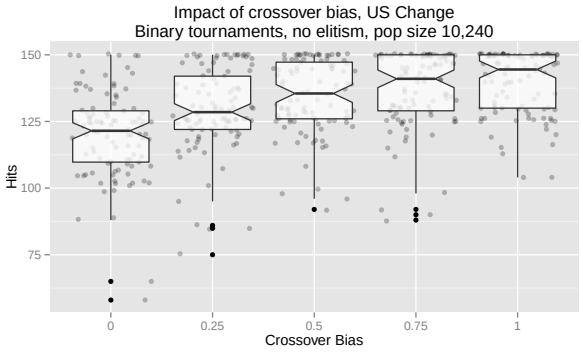

Figure 7: Impact of crossover bias on the number of hits for the U.S. Change problem, limited to binary tournaments, no elitism, and population size 10,240.

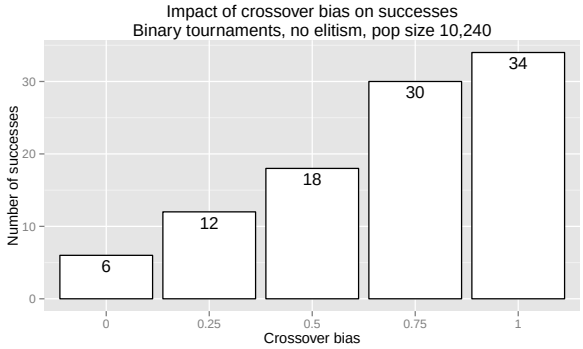

Figure 8: Impact of crossover bias on the number of successes runs for the U.S. Change problem when using binary tournaments, no elitism, and population size 10,240.

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 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

We decided that we're going to just focus on hits for both regression problems, which means we should remove the fitness plots/discussion from the Pagie-1 section.

4.3.1 Pagie-1 Problem

We agreed to just drop everything about the koza2 function set, which includes redrawing Figure 9 to only have basic4.

For the Pagie-1 problem we did runs across the following parameters:

- crossover biases of -1.00, 0.00, 0.25, 0.50, 0.75, and 1.00;
- population sizes of 1,024 and 10,240;
- elitism percentages of 0% or 1%;
- tournament sizes of 2, 3, 5, and 7;
- basic4 ($+, -, \times, \div$) and koza2 (complete) function sets; and
- with and without Tarpeian bloat control.

However, for purposes of space conservation, we will not discuss the koza2 function set results here.

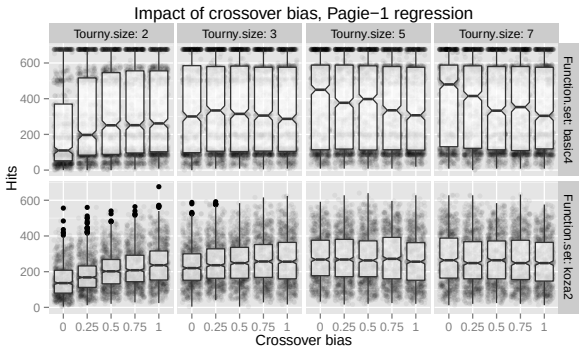

Figure 9: Impact of crossover bias on the number of hits for the Pagie-1 symbolic regression problem, broken out for the four different tournament sizes (2, 3, 5, and 7) and the two different function sets (basic4 and koza2). The maximum number of possible hits is 676.

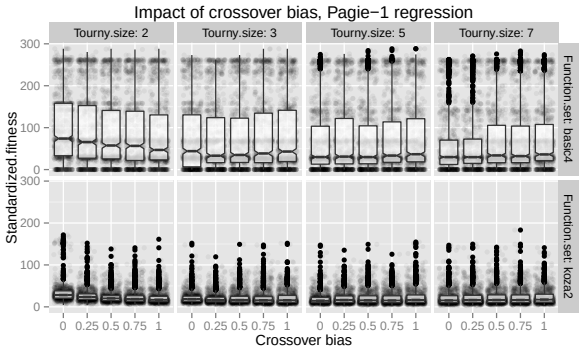

Figure 10: Impact of crossover bias on the fitness (total error) for the Pagie-1 symbolic regression problem, broken out for the four different tournament sizes (2, 3, 5, and 7) and the two different function sets (basic4 and koza2). The best possible is an error of 0.

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; Figure 10 is similar, but plots fitness (total error) instead of hits.

The results, as can be seen in Figures 9 and 10 vary significantly. However, if limited to the following parameters:

- crossover biases of -1.00, 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;

then the complexities of Figures 9 and 10 simplify to Figures 11 and 12. For now we will focus on this specific subset of the data.

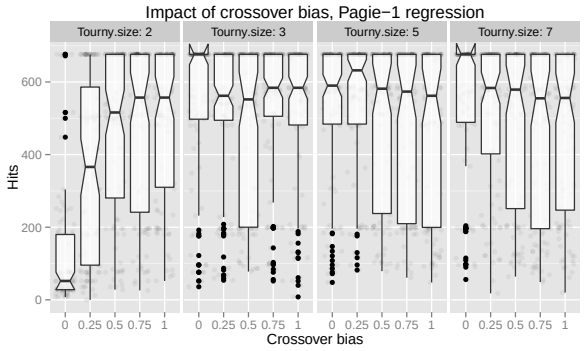

Figure 11: Impact of crossover bias on the number of hits for the Pagie-1 symbolic regression problem, broken out by tournament size (2, 3, 5, and 7). These runs use the "basic4" function set $(+, -, \times, \div)$, no elitism, population size 10,240, and Tarpeian bloat control. The maximum number of possible hits is 676.

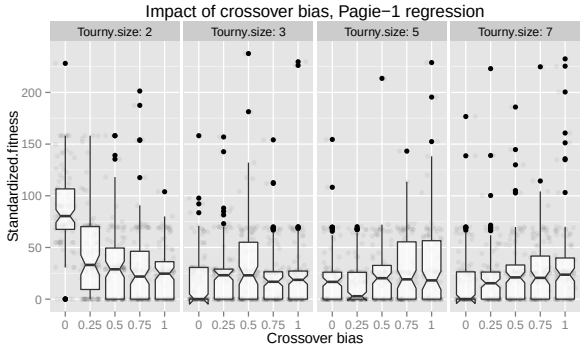

Figure 12: Impact of crossover bias on the number of fitness (total error) for the Pagie-1 symbolic regression problem, broken out by tournament size (2, 3, 5, and 7). These runs use the "basic4" function set $(+, -, \times, \div)$, no elitism, population size 10,240, and Tarpeian bloat control. The best possible is an error of 0.

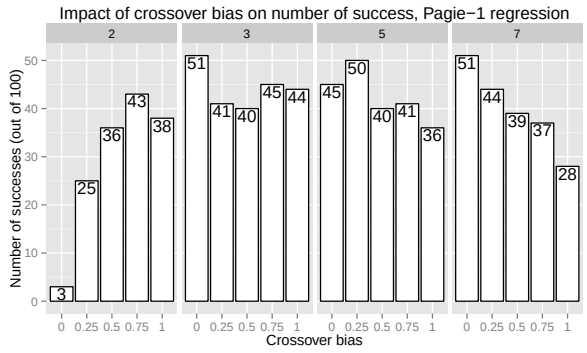

Figure 13: Impact of crossover bias on the number of successes (runs that exactly solve the problem) for the Pagie-1 symbolic regression problem broken out by tournament size (2, 3, 5, and 7). These runs use the "basic4" function set (just $+, -, \times, \div$), no elitism, population size 10,240, and Tarpeian bloat control.

Focusing on the 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 ($p < 10^{-12}$), 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.

The statistical significance in Figure 12 is very similar to that in the previous figure (Figure 11). Here again adding bias for binary tournaments improves fitness, but none of the differences between biases 0.50, 0.75, and 1.00 are statistically significant. None of the differences for tournament sizes 3 and 5 are significant, and most of those for tournament size 7 are not significant. The one exception for tournament size 7 is that the difference between crossover biases 0.00 and 1.00 is statistically significant ($p = 0.021$).

Figure 13 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 ($p = 0.015$).

Waiting for treatment information on Figures 14 and 15. Figure 15 will eventually be removed, but the information is interesting, and I (Kirbie) would like as much information on it as possible before I remove it.

4.3.2 Sine Problem

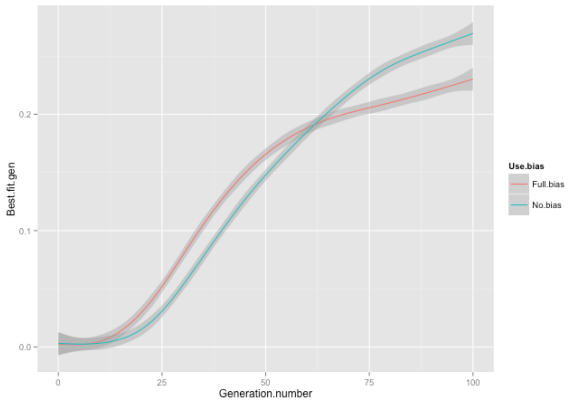

Figure 14: Impact of crossover bias on the fitness over time for the Pagie-1 symbolic regression problem. What's the data set this comes from? Drop this for now.

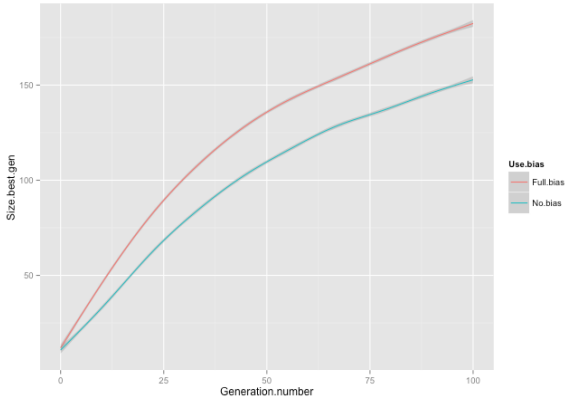

Figure 15: Impact of crossover bias on the tree size over time for the Pagie-1 symbolic regression problem. Drop this for now.

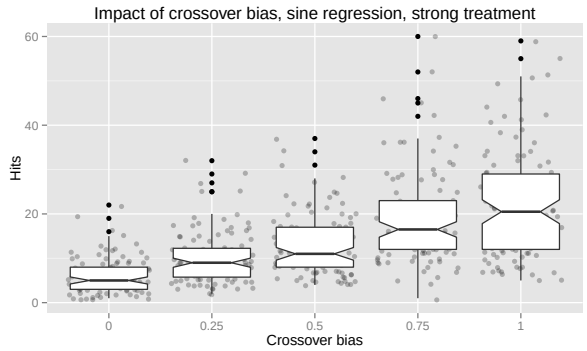

Figure 16: Impact of crossover bias on the sine symbolic regression problem with bias-effective settings: binary tournaments, no elitism, and population size 10,240.

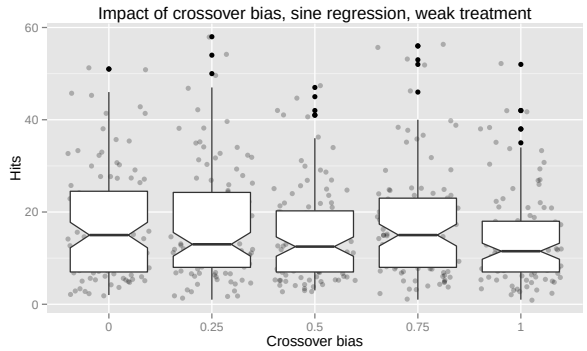

Figure 17: Impact of crossover bias on the sine symbolic regression problem with non-bias-effective settings: tournament size 7, 0.1% elitism, and population size 1,024.

Figure 18: 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.

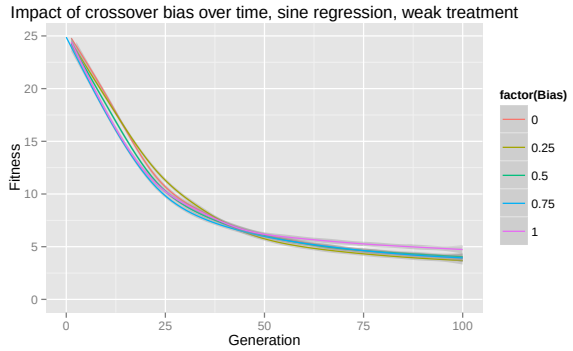

Figure 19: Impact of crossover bias on the sine symbolic regression problem with tournament size 7, 0.1% elitism, and population size 1,024. Move this to where we define non-bias-effective parameters, combined with the previous figure as a pair of panels.

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 16 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 17 shows the results for the sine regression problem using tournament size 7, 0.1% elitism, and smaller populations of size 1,024. None of these differences are statistically significant using a pairwise Wilcoxon rank sum test.

Figures 18 and 19 show the change in fitness over time for the bias-effective and non-bias-effective configurations. In the bias-effective configuration we get a really clean "adding bias is good" demonstration. In the non-bias-effective configuration, the impact of crossover bias is quite minimal; **oddly, though, it does look like a crossover bias of 1.00 is worse by a little bit than everything else at the end, which is weird**. Figure 20 shows fitness over time for the non-bias-effective configuration with population size 10,240 (instead of 1,024 for the "normal" non-bias-effective configuration). Increasing the population size improves the performance significantly. The different bias levels are all essentially the same at the end of the 100 generations, but there is definitely a spread around 15-20 generations that is a really clean "more bias is better" demonstration (**Do we care?**) except for the fact that 0.75 and 1.00 are essentially the same. **Do we want to include both Figures 19 and 20? Do we want to combine them into a single graph (either just one graph, or as two "panels" like, e.g., in Figure 21?**

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. To better understand this, blah, blah, blah **We need to turn this into actual text**.

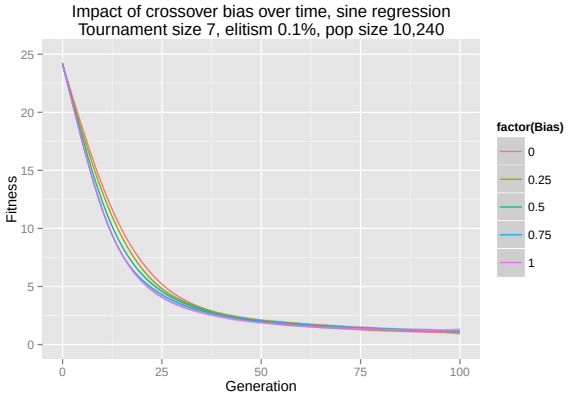

Figure 20: Impact of crossover bias on the sine symbolic regression problem with tournament size 7, 0.1% elitism, and population size 10,240. Drop this.

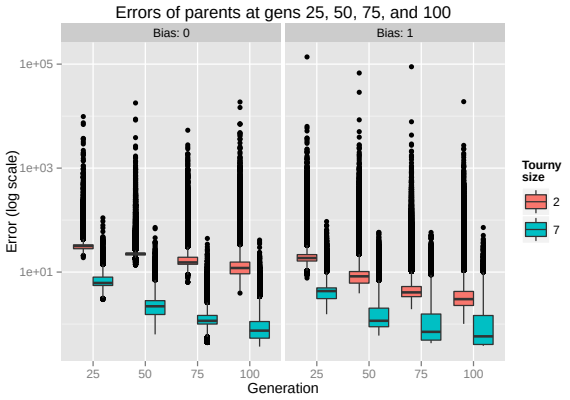

Figure 21: Plot of the errors of all individuals chosen as parents from some sine runs. Explain this. Do we want/need this plot? We agreed to get rid of this.

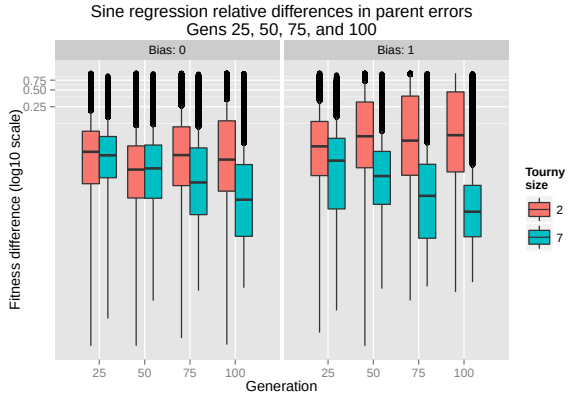

Figure 22: Plot of the normalized differences in parent errors from some sine runs. Explain this.

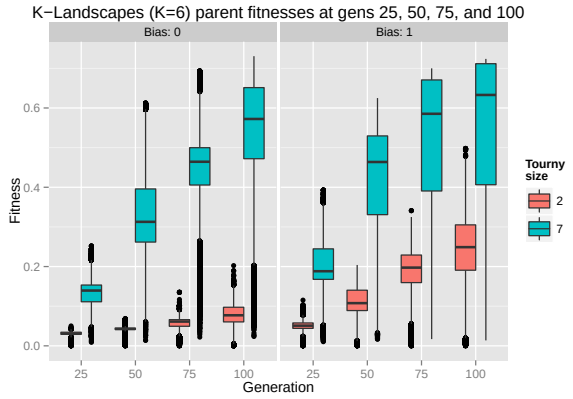

Figure 23: Plot of the fitnesses of all individuals chosen as parents from some K-Landscape runs. Explain this. Do we want/need this plot? We agreed to get rid of this.

Figure 22 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 case where one of the errors is nearly 0.

We see similar results for the K-Landscapes problem, as illustrated in Figures 23 and 24.

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**

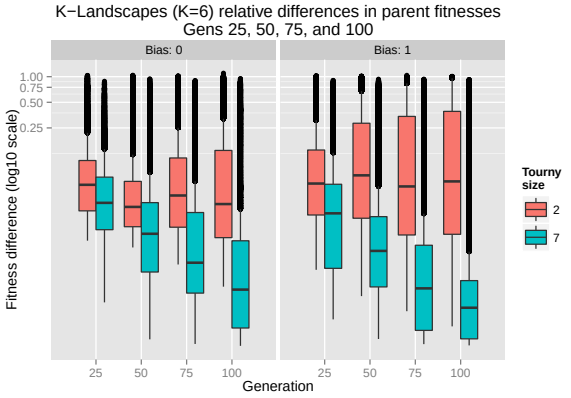

Figure 24: Plot of the normalized differences in parent fitnesses from some K-Landscape runs. Explain this.

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."

Acknowledgements

7. REFERENCES

[1] 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.

[2] 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.

[3] 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).

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

[5] 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.

[6] 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.

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