# Parent Selection and Diversification in Genetic Programming

Author Omitted
.
.
.
.

Author Omitted
.
.
.
.

Author Omitted
.
.
.
.

## ABSTRACT

More things!

## Keywords

lexicase selection, hyperselection, PushGP, other stuff

## 1. INTRODUCTION

I bet we start here!

Lexicase selection [8] is nifty, eh?

Hyperselection provided initial motivation [3], but later we became interested more generally in what lexicase and tournament do differently starting with the same population.

[TMH: I'm not sure if we want to talk about error diversity in the Intro or somewhere else. But, the following paragraph could be moved to be the first paragraph of Experimental Design if we don't need to talk about diversity earlier]

In this paper we concentrate on diversity measures related to the outputs of the programs. One such diversity measure, *behavioral diversity*, has been shown to have correlation with problem-solving performance [6]. In behavioral diversity, the output of each program is recorded on each training case input and stored as a *behavior vector*. Behavioral diversity is then the percentage of distinct behavior vectors in the population. *Error diversity*, a slight variation of behavioral diversity, considers the percentage of distinct error vectors in the population where each error vector is computed by applying the error function to each output in the behavior vector. We believe error diversity does a good job of measuring how well evolution is exploring meaningful differences between programs that might be lost with a diversity measure that only takes into account syntactic (genotypic) diversity of the population, where two wildly different programs may actually compute the same function.

## 2. LEXICASE SELECTION



*GECCO'16, July 20-24, 2016, Denver, Colorado, USA.*



DOI: 10.1145/1235

## 3. EXPERIMENTAL DESIGN

Previous work has shown that using lexicase selection results in higher population error diversity than tournament selection across a variety of problems [2, 5]. These papers examined the diversity of entire GP runs, each starting with a different initial population and random number seed.

Here we examine the effcts of these parent selection methods on population error diveristy starting from specific population conditions besides a random initial population. In particular, we want to see how each method changes diversity in populations that occur naturally during an evolutionary run.

In order to produce the populations on which to experiment, we started GP runs and let them continue until they met certain stopping conditions; we then stored those populations and later conducted multiple trials with different random number seeds starting with those stored populations. We used three different stopping conditions in order to generate naturally occuring populations with interesting properties:

1. In a run using lexicase selection, we stopped if the population error diversity was greater than 0.9. This results in very diverse populations, allowing us to observe whether evolution is able to maintain such high diversity in the following generations.

2. In a run using tournament selection, we stopped if the population error diversity was less than 0.15. These populations allow us to see if methods promote diversification starting from such undiverse populations. They also allow us to see if methods perform differently on a population produced by tournament selection versus one produced by lexicase selection.

3. As described above, we were initially motivated here by observations of runs using lexicase selection that underwent major drops in diversity following hyperselection events, where one or a few individuals in the population received the majority of the parent selections in a generation. We had anecdotally noticed rapid diversity recovery following these events, but not examined them systematically [3].

   In this condition, we stopped a run using lexicase selection when the error diversity reached a level at least 0.25 less than it had been at some point in the previous 10 generations. This allowed us to detect populations that had recently undergone large drops in diversity. We do not definitively know whether those drops are

Table 1: PushGP parameters

| Parameter | Value |
|---|---|
| runs per problem/parent selection combination | 100 |
| population size | 1000 |
| maximum genome size | 1600 |
| maximum initial genome size | 400 |
| **Genetic Operator** | **Prob** |
| alternation | 0.2 |
| uniform mutation | 0.2 |
| uniform close mutation | 0.1 |
| alternation followed by uniform mutation | 0.5 |

related to hyperselection events, but we expect that they are.

In all three conditions, we only considered populations occuring after the first 10 generations in order to give evolution a chance to settle down after the extreme shifts that can happen at the beginning of a run.

In each trial, we continued running GP on a stored population for 20 generations and recorded the population error diversity. For each parent selection setting (lexicase and tournament selections), we conducted 100 trials with different random number seeds from each stored population.

We conducted these tests on two problems taken from a recent program synthesis benchmark suite [4]. The first problem, Replace Space With Newline (RSWN), searches for a program that takes as input a string and both prints the string after replacing all of the spaces in the input with newline characters and functionally returns the number of non-whitespace characters in the string. Previous examinations of error vector diversity on the RSWN problem indicate that lexicase selection maintains significantly higher diversity than tournament selection, which across 100 runs never achieved a median diversity higher than 0.25 [2].

The second problem, Double Letters, asks for a program that takes a string as input and prints the string after doubling every alphabetic character and tripling every exclamation point. All other characters should be printed once. As with the RSWN problem, lexicase selection consistently achieves high diversity on this problem. Differently than RSWN, runs using tournament selection show slow but steady increases in diversity, though not approaching that of lexicase selection runs [2].

For our experiments we used PushGP [11, 10], a stack-based genetic programming system.[1] PushGP supports a variety of control structures and multiple data types, making it a good choice for program synthesis tasks such as the problems we explore here. Except for parent selection, we used the exact same PushGP parameters in both the initial runs used to store interesting populations as well as the continuations of the stored populations. We give the most relevant parameters in Table 1. The parameters not listed here exactly follow those used in the experiments in [1].

These runs use the most recent version of PushGP, in which individuals are stored as linear genomes that we translate into hierarchical Push programs prior to execution [1]. These linear genomes admit a range of uniform genetic operators; we use four, listed in Table 1 with their probabilities. Alternation is a linear crossover operator modeled after the sexual portion of ULTRA [9]. Uniform mutation may replace each instruction with 1% probability. Uniform close mutation may add or remove parentheses from the program. Finally, the last operator runs alternation on two parents and then uniform mutation on that child to produce a new child.

[1]Lexicase selection has also been shown to be effective in tree-based genetic programming [5, 7].

# 4. RESULTS

Using the techniques presented in the previous section we obtained populations on which to perform continuation experiments. For each combination of the 2 problems and 3 stopping conditions we stored populations from 2 runs, for a total of 12 populations. In the following subsections we group the results based on the stopping conditions, since they produce the most similar populations. Note that we label each population with a letter; these letters have no relation to the populations themselves, and are simply used for reference.

Starting with each stored population we conducted 100 "continuation" GP runs with lexicase selection and 100 with tournament selection. We let each continuation go for 20 generations, and plot the population error diversity across the runs. In particular, each figure has a standard box-and-whisker plot for each generation, with the box showing the median and quartiles. The whiskers stretch to the maximum and minimum values, ignoring outliers. In each figure we also plot the error diversity of each individual run at each generation, giving another way of visualizing the spread of diversities across runs and making it easier to trace runs with outliers.

Note that in a few settings, one or two runs out of 100 found solutions to the problem before the end of 20 generations. In these cases, we terminate the runs, and they do not contribute data past their termination generation. We do not believe these solutions have a large effect on the plots since no plot had more than two of these early-terminating runs.

## 4.1 Starting with high diversity

In this subsection we continue runs using stored populations evolved using lexicase selection that achieved error diversity greater than 0.9. As such, the initial populations of the runs have very high diversity, with most individuals producing distinct error vectors.

Figure 1 plots the continued runs started from two populations (C and D) stored from GP evolving on the RSWN problem. Lexicase selection consistently maintains high levels of diversity starting from both populations, with little variance. On the other hand, both plots show runs using tournament selection quickly losing significant diversity within the first 5 to 10 generations of the continuation, dropping from over 90% distinct error vectors down to around 50% distinct error vectors. Interestingly, the tournament selection runs on Population C show large differences in diversity the last 10 generations, with some becoming even less diverse while others recovering much of the lost diversity. On the other hand, the tournament selection runs on Population D maintain much more consistent diversity, with most runs having between 40% and 60% diversity in the remaining generations. It is unclear to us why these two populations result in such different diversity plots for tournament

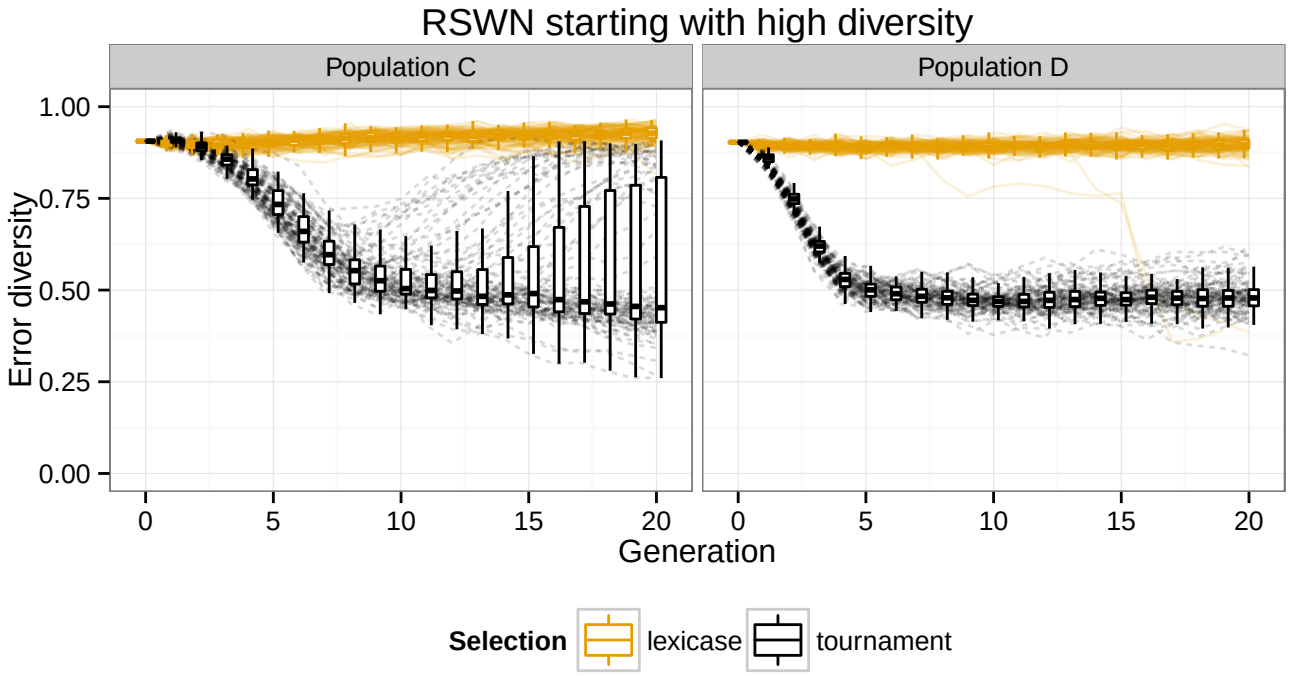


**Figure 1: Error diversity over 100 continuations of the RSWN problem with both lexicase and tournament selections, starting from populations with high diversity naturally occuring in a run using lexicase selection.**

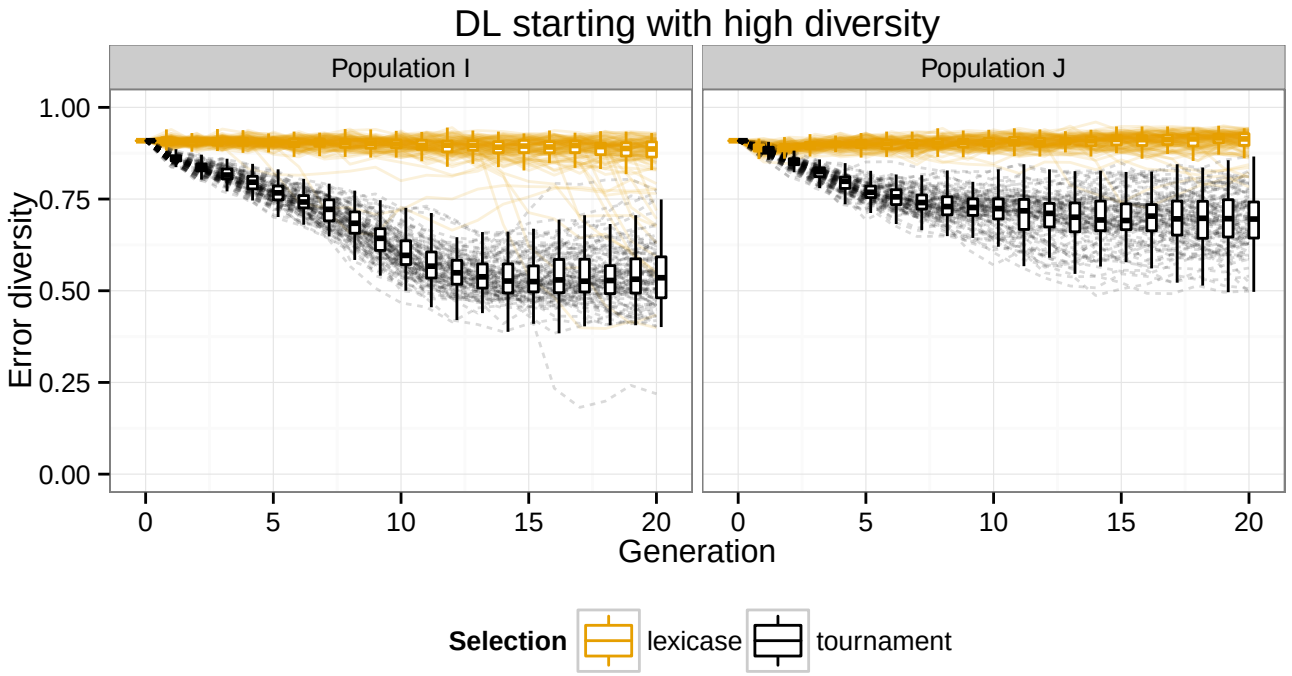


**Figure 2: Error diversity over 100 continuations of the double-letters problem with both lexicase and tournament selections, starting from populations with high diversity naturally occuring in a run using lexicase selection.**

selection, but we assume it has to do with the composition of the stored population.

Figure 2, which plots the diversities of continuations of two populations (I and J) on the Double Letters problem, shows similar trends in both lexicase selection and tournament selection. Note that tournament selection maintains higher diversity on this problem than on the RSWN problem, though not as high as lexicase selection. This trend mirrors what has been previously seen on full GP runs [2].

*[should the next two paragraphs be in Discussion?]* The continuations starting from high-diversity populations clearly show that lexicase selection can maintain a high population diversity while tournament selection cannot reliably do so.

One interesting observation in these plots is the occasional steep drop in diversity in a small number of runs using lexicase selection, which can be seen especially clearly in populations D and I. Based on similar runs we have encountered previously, we would guess that these runs underwent hyperselection events in which one or a small number of individuals were selected to make most of the children in a single generation. Hyperselection events can, understandably, lead to diversity crashes since most of the individuals in the population are closely related. Interestingly, previous work has shown that such events are neither a driving force or a hinderance in runs using lexicase selection [3].

## 4.2 Starting with low diversity

Next, we present continuations of runs that start from populations exhibiting very low population diversity, at most 0.15. In other words, most of the individuals in these populations produced the same error vectors. These populations were stored from runs using tournament selection, since we were not able to achieve population diversity this low in a run using lexicase selection. Additionally, this will allow us to examine whether the parent selection technique used to produce the initial populations has effect on the continued diversity.

Figure 3 plots diversity of runs starting from populations A and B on the RSWN problem. Starting from both populations, tournament selection does not increase diversity across the 20 generations except for a handful of outlier runs. The continuations using lexicase selection increase the median diversity across runs rapidly, with over 50% unique error vectors by generation 8 using population A and generation 4 using population B. For population A, lexicase selection runs continue to steadily rise in diversity over the 20 generations. On the other hand, many runs starting with population B undergo steep drops in diversity, such that by generation 9 the lower quartile of diversity falls precipitously from around 60% to around 35%. The indivdiually plotted run diversities show that many runs continue to see single-generation drops in diversity throughout the 20 generations. We believe this population was likely contained one or more individuals that, when varied in the right way, produce a child that dominates the rest of the population, leading to many hyperselection events and therefore drops in diversity. Even with these drops in diversity, lexicase selection maintains higher diversity than tournament selection in the majority of continuations.

We present continuations of low-diversity populations (G and H) evolved on the Double Letters problem in Figure 4. Lexicase selection again increases error diversity more quickly than tournament selection, though here tournament selection does show some incrases in diversity. This is more pronounced when starting from population H, where median diversity is over 0.25 by generation 20. This still pales in comparison to lexicase selection's diversity growing to more than 0.6 on population H and 0.75 on population G. Both plots show lexicase selection runs gaining and maintaining diversity across the 20 generations, and not encountering any diversity falls that we observed in Figure 3.

*[should the next two paragraphs be in Discussion?]* Figures 3 and 4 show how lexicase selection can diversify an underiverse population over small numbers of generations. This ability to create error diversity exhibits how lexicase selection can rapidly explore the space of meaningfully different programs.

## 4.3 Starting after a diversity crash

Our work here was originally motivated by observations of runs using lexicase selection that suddenly lost significant amounts of diversity. Anecdotally, these runs often seemed to quickly recover diversity in the generations following the diversity crash. We presume that these diversity crashes were related to hyperselection events that we observed in the populations. As such, we are interested to see whether lexicase selection can reliably recover diversity following a diversity crash. The stored populations in these continuations occured after error diversity dropped at least 25% over the preceding 10 generations when using lexicase selection.

We plot error diversity from populations E and F that were stored after diversity crashes on the RSWN problem in Figure 5. Neither of these plots shows rapid rediversification of lexicase selection runs; instead, we see consistent gains in diversity for most of each run. The median diversity on population E increases about 20% over 20 generations, gaining back most of the diversity it lost during the diversity crash. Population F gains back about 15% population in that timespan. Interestingly, population E sees immediate small gains in diversity in the first few generations, where population F shows consistent small losses in diversity before rediversifying. This presumably means that population E had reached its minimum in its diversity crash, where population F was recorded near the end of the crash but before its minimum.

Turning our attention to tournament selection, we see that it saw remarkably consistent drops in diversity in the very first generation, especially in Population F. These drops are followed by little movement either direction during the remaining generations.

Figure 6 shows the same settings for the Double Letters problem using populations K and L. Population K is interesting in that lexicase selection gains some diversity near the beginning while almost every tournament selection loses at least 25% diversity over the first 4 generations. After that, an increasing number of lexicase selection runs seem to undergo further diversity crashes, pulling down the lower quartile and minimum while the median diversity stays around 60%. On the other hand, tournament selection shows incrases in diversity in the later generations, though its quartiles never overlap with lexicase selection's.

For population L, after a few generations of further diversity decrases, lexicase selection runs tend to increase diversity (with lots of variation) ending with about 20% higher median diversity than at its lowest. Tournament selection,

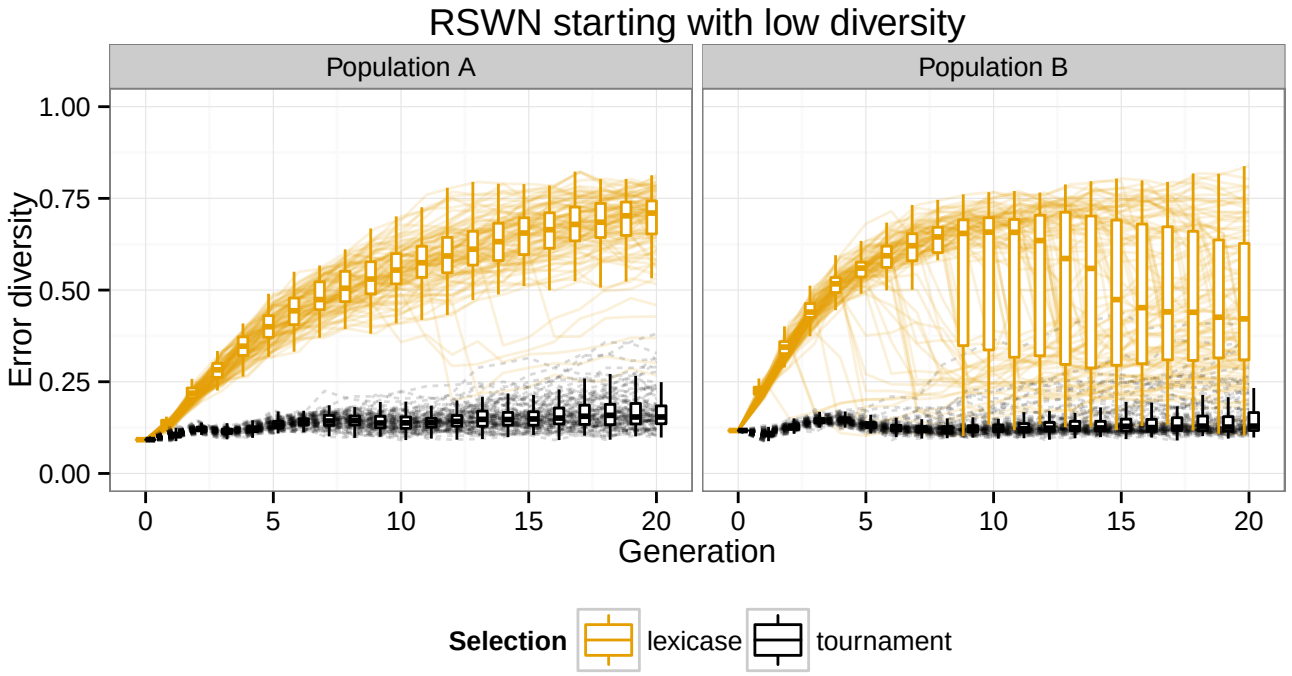


**Figure 3: Error diversity over 100 continuations of the RSWN problem with both lexicase and tournament selections, starting from populations with low diversity naturally occuring in a run using tournament selection.**

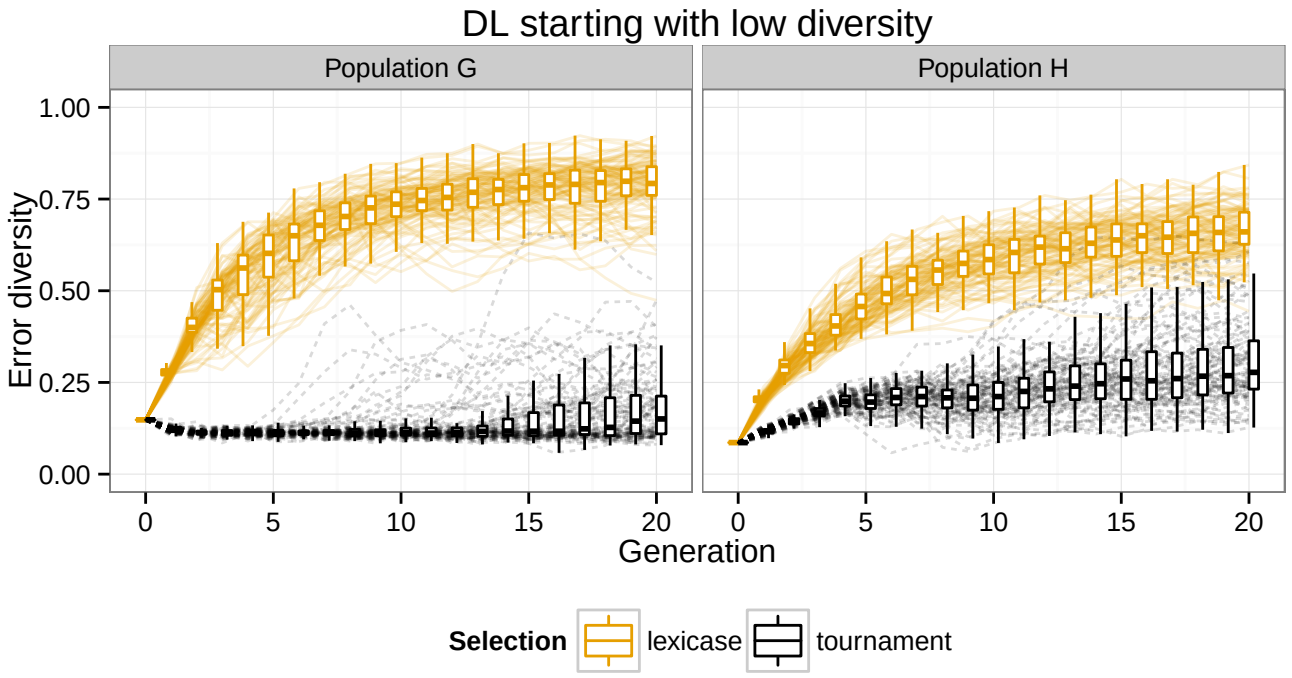


**Figure 4: Error diversity over 100 continuations of the double-letters problem with both lexicase and tournament selections, starting from populations with low diversity naturally occuring in a run using tournament selection.**

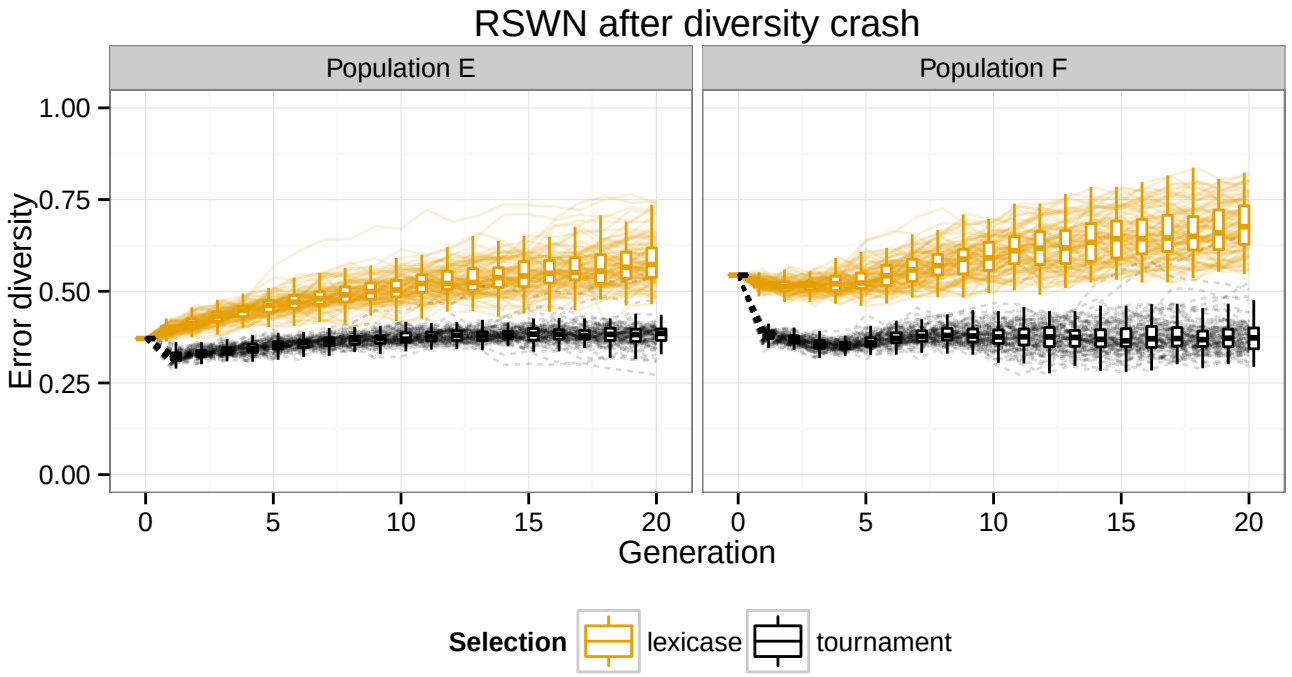


**Figure 5: Error diversity over 100 continuations of the RSWN problem with both lexicase and tournament selections, starting from populations that had lost at least 25% error diversity in a diversity crash in a lexicase selection run.**

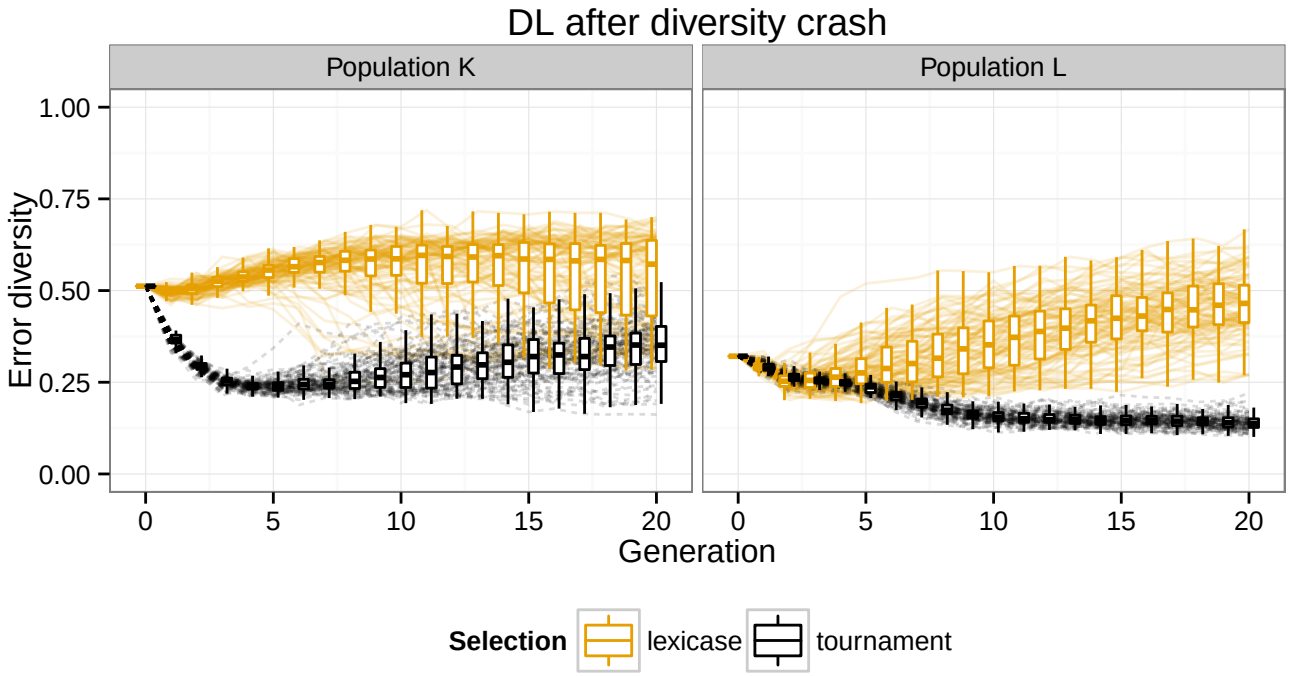


**Figure 6: Error diversity over 100 continuations of the double-letters problem with both lexicase and tournament selections, starting from populations that had lost at least 25% error diversity in a diversity crash in a lexicase selection run.**

however, very consistently loses diversity over the 20 generations with little variation across runs.

Of the plots we present, those in this section have the smallest gaps in diversity between lexicase selection and tournament selection. Still, they show lexicase selection's ability to increase diversity, albeit gradually, following a diversity crash. Tournament selection runs lost diversity over the 20 generations in three of the four plots, showing that it not only was not able to stop the diversity crash, but contributed to its continuation.

# 5. DISCUSSION

Our continuations of 12 populations show that lexicase selection can maintain high diversity in a population and can rediversify populations with either low diversity or recently-crashed diversity. On the other hand, tournament selection consistently achieved lower diversity, either by decreasing the number of unique error vectors in the population or by not increasing diversity in undiverse populations. Why is lexicase selection so much better at increasing and maintaining diversity than tournament selection?

First, let us discuss tournament selection's drops in diversity in populations that originally evolved using lexicase selection. Suppose that a number of individuals in a population have identical or very similar error vectors, and have low total error. With tournament selection, these individuals might all be selected a number of times in a given generation, leading to a population containing many of their children. Many of those children likely have similar error vectors to their parents, resulting in a less diverse population than the prior one. With the same population, lexicase selection would require those individuals to compete for the same selections, since any individuals with identical error vectors will have equal chance of selection when their best case errors come near the beginning of the randomly shuffled test cases. So, lexicase selection makes those individuals "compete" for the selections it is eligible for with those individuals that produce identical error vectors.

Another factor likely at play here is the way in which lexicase selection places emphasis on individuals that perform well on single test cases or combinations of small numbers of test cases. Since an individual must be the absolute best in the population on a test case if it comes first in the shuffled test cases in order for the individual to be selected, lexicase selection can select individuals that specialize on doing well at one or more test cases even if they do poorly at others. This phenomenon, shown to contribute to lexicase selection's success in prior work [1], likely allows lexicase to select many different specialists with different error vectors, diversifying the parent pool and therefore the children of the next generation.

Separate from the question of lexicase selection versus tournament selection, the results we present in each figure clearly depend not only on the problem on which the runs are evolving, but the different stored starting populations. In fact, in some figures it is evident that the starting population was the cause of later swings in diversity not manifested in the first few generations; see Figure 1 for tournament selection and Figure 3 for lexicase selection for examples. Thus the composition of a population, or specific members of a population, can drastically change the shape of diversity in the following generations even compared to similarly chosen populations. Even so, lexicase selection clearly contributes more to diversity than tournament selection starting from all twelve of the populations presented here.

# 6. CONCLUSIONS

I'm hoping we have conclusions.

## Acknowledgments

Lots of cool people helped us.

# 7. REFERENCES

[1] T. Helmuth. *General Program Synthesis from Examples Using Genetic Programming with Parent Selection Based on Random Lexicographic Orderings of Test Cases*. Ph.D. dissertation, 2015.

[2] T. Helmuth, N. F. McPhee, and L. Spector. Lexicase selection for program synthesis: a diversity analysis. In *Genetic Programming Theory and Practice XIII*, Genetic and Evolutionary Computation. Springer.

[3] T. Helmuth, N. F. McPhee, and L. Spector. The impact of hyperselection on lexicase selection. In *GECCO '16: Proceedings of the 2016 Conference on Genetic and Evolutionary Computation*, July 2016.

[4] T. Helmuth and L. Spector. General program synthesis benchmark suite. In *GECCO '15: Proceedings of the 2015 Conference on Genetic and Evolutionary Computation*, July 2015.

[5] T. Helmuth, L. Spector, and J. Matheson. Solving uncompromising problems with lexicase selection. *IEEE Transactions on Evolutionary Computation*, 19(5):630–643, Oct. 2015.

[6] D. Jackson. Promoting phenotypic diversity in genetic programming. In *PPSN 2010 11th International Conference on Parallel Problem Solving From Nature*, volume 6239 of *Lecture Notes in Computer Science*, pages 472–481, Krakow, Poland, 11-15 Sept. 2010. Springer.

[7] P. Liskowski, K. Krawiec, T. Helmuth, and L. Spector. Comparison of semantic-aware selection methods in genetic programming. In *GECCO 2015 workshop on Semantic Methods in Genetic Programming*. ACM, 2015.

[8] L. Spector. Assessment of problem modality by differential performance of lexicase selection in genetic programming: A preliminary report. In *1st workshop on Understanding Problems (GECCO-UP)*, pages 401–408, Philadelphia, Pennsylvania, USA, 7-11 July 2012. ACM.

[9] L. Spector and T. Helmuth. Uniform linear transformation with repair and alternation in genetic programming. In R. Riolo, J. H. Moore, and M. Kotanchek, editors, *Genetic Programming Theory and Practice XI*, Genetic and Evolutionary Computation, chapter 8, pages 137–153. Springer, Ann Arbor, USA, 9-11 May 2013.

[10] L. Spector, J. Klein, and M. Keijzer. The Push3 execution stack and the evolution of control. In *GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1689–1696, Washington DC, USA, 2005. ACM Press.

[11] 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.