# Parent Selection and Diversification in Genetic Programming

Thomas Helmuth
Computer Science
Washington and Lee Univ
Lexington, Virginia
helmutht@wlu.edu

Nicholas Freitag McPhee
Div of Sci and Math
Univ of MN, Morris
Morris, MN 56267
mcphee@morris.umn.edu

Lee Spector
Cognitive Science
Hampshire College
Amherst, MA
lspector@hampshire.edu

## ABSTRACT

More things!

## Keywords

lexicase selection, hyperselection, PushGP, other stuff

## 1. INTRODUCTION

I bet we start here!

Lexicase selection [7] 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.

## 2. 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 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 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 dou-



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



DOI: 10.1145/1235

Table 1: PushGP parameters

| Parameter | Value |
|---|---|
| runs per problem/parent selection combination | 100 |
| population size | 1000 |
| maximum generations | 300 |
| 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 |

bling 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 [10, 9], 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 [8]. 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.

## 3. RESULTS

SOOOO MANY GRAPHS!

### 3.1 Starting with high diversity

### 3.2 Starting with low diversity

### 3.3 Starting after a diversity crash

## 4. CONCLUSIONS

I'm hoping we have conclusions.

[1]Lexicase selection has also been shown to be effective in tree-based genetic programming [5, 6].


## Acknowledgments

Lots of cool people helped us.


## 5. 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] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

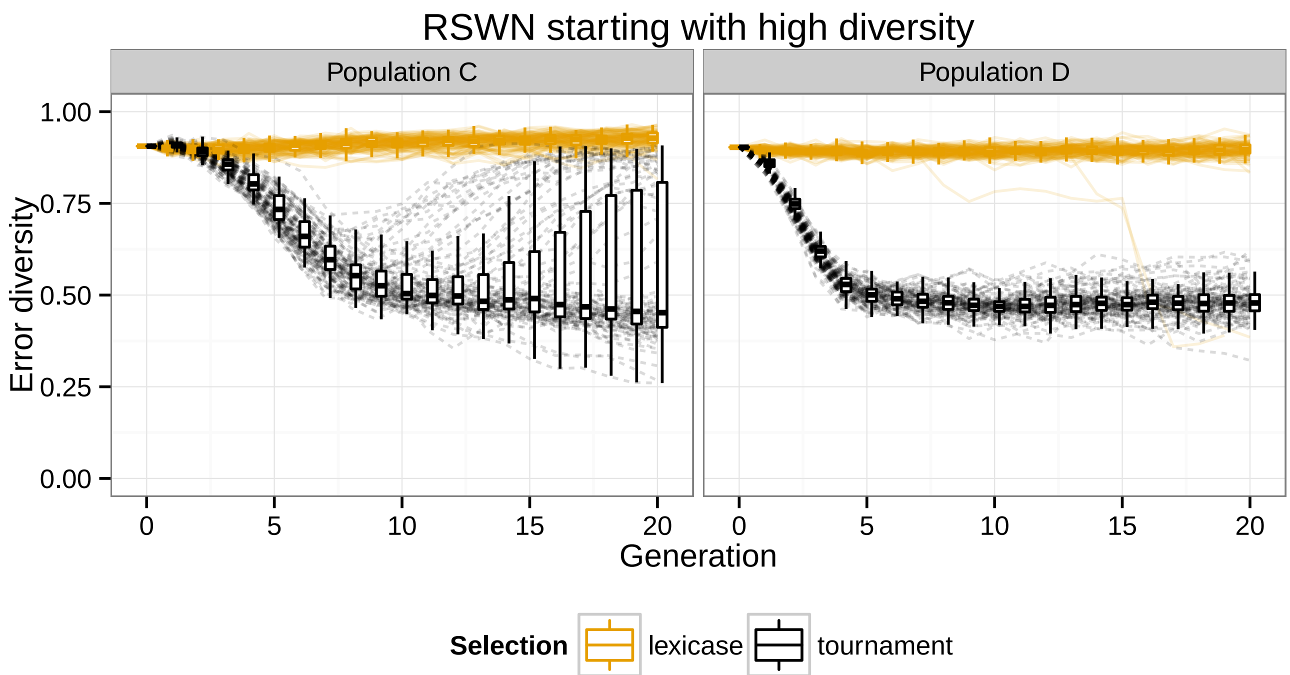


**Figure 1: Changes in diversity over 100 "re-runs" of the replace-space-with-newline problem with both lexicase and tournament selections, starting from a population with high diversity coming from a lexicase selection run.**

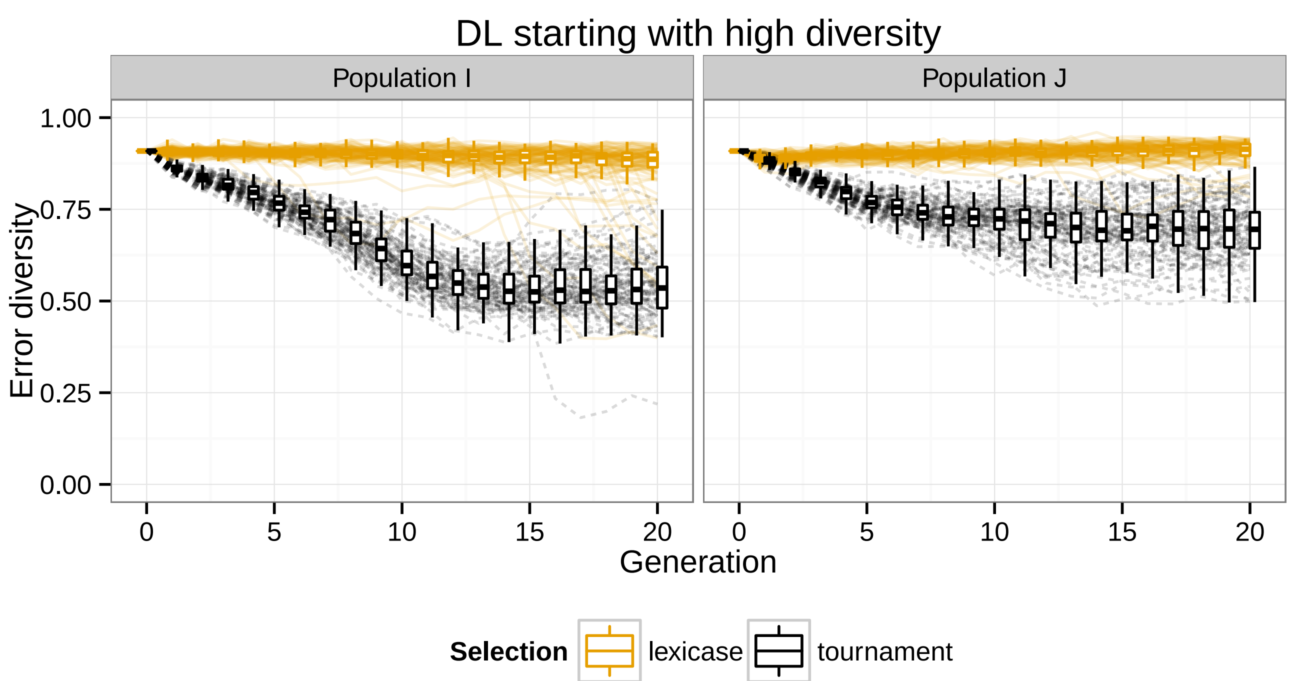


**Figure 2: Changes in diversity over 100 "re-runs" of the double-letters problem with both lexicase and tournament selections, starting from a population with high diversity coming from a lexicase selection run.**

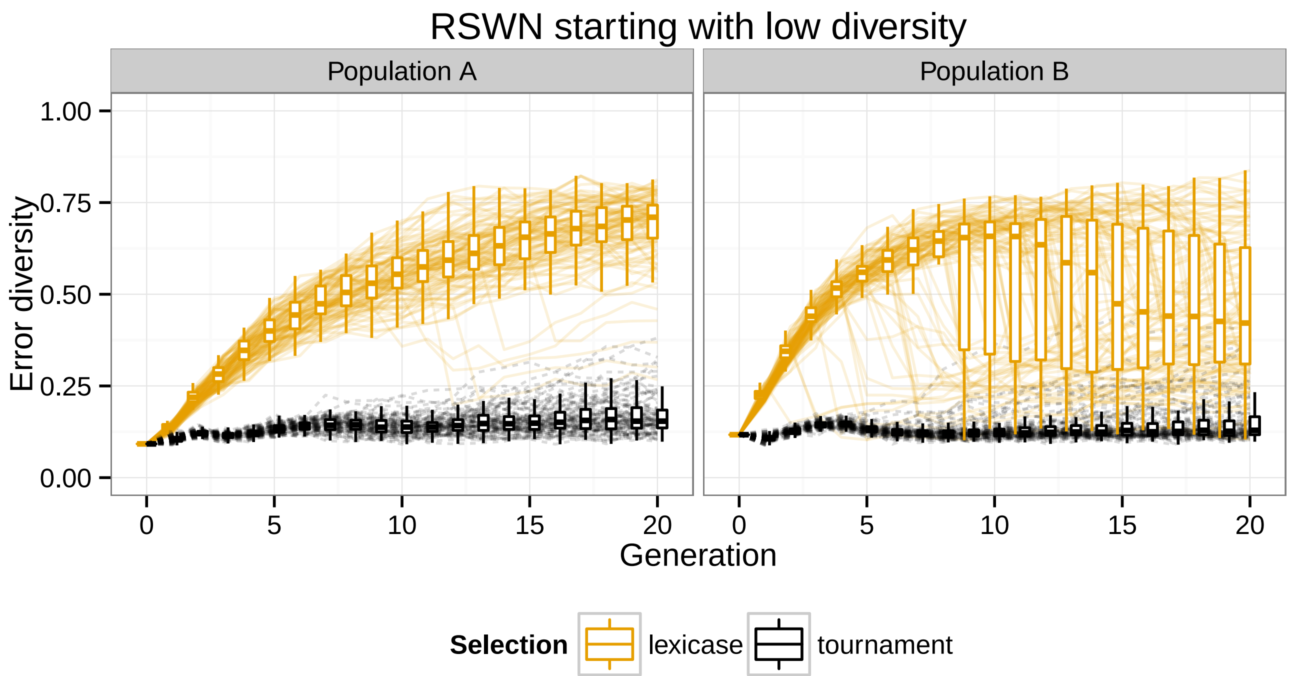


**Figure 3: Changes in diversity over 100 "re-runs" of the replace-space-with-newline problem with both lexicase and tournament selections, starting from a population with low diversity coming from a tournament selection run.**

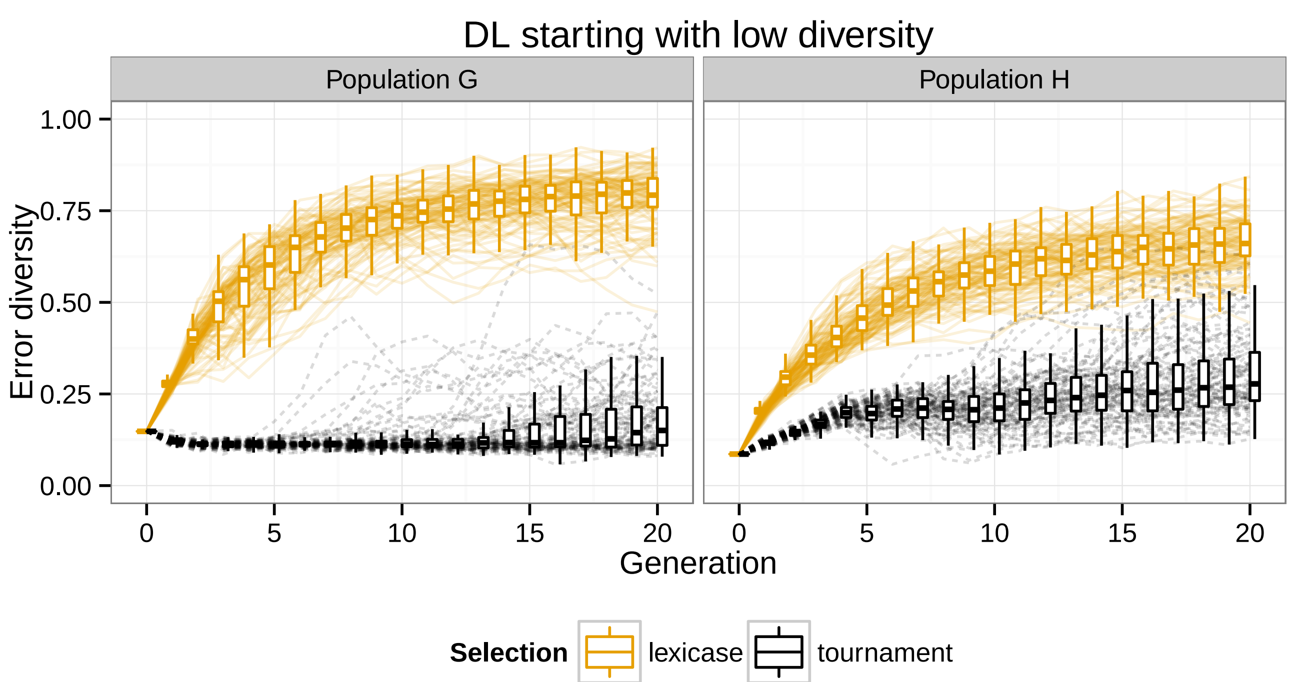


**Figure 4: Changes in diversity over 100 "re-runs" of the double-letters problem with both lexicase and tournament selections, starting from a population with low diversity coming from a tournament selection run.**

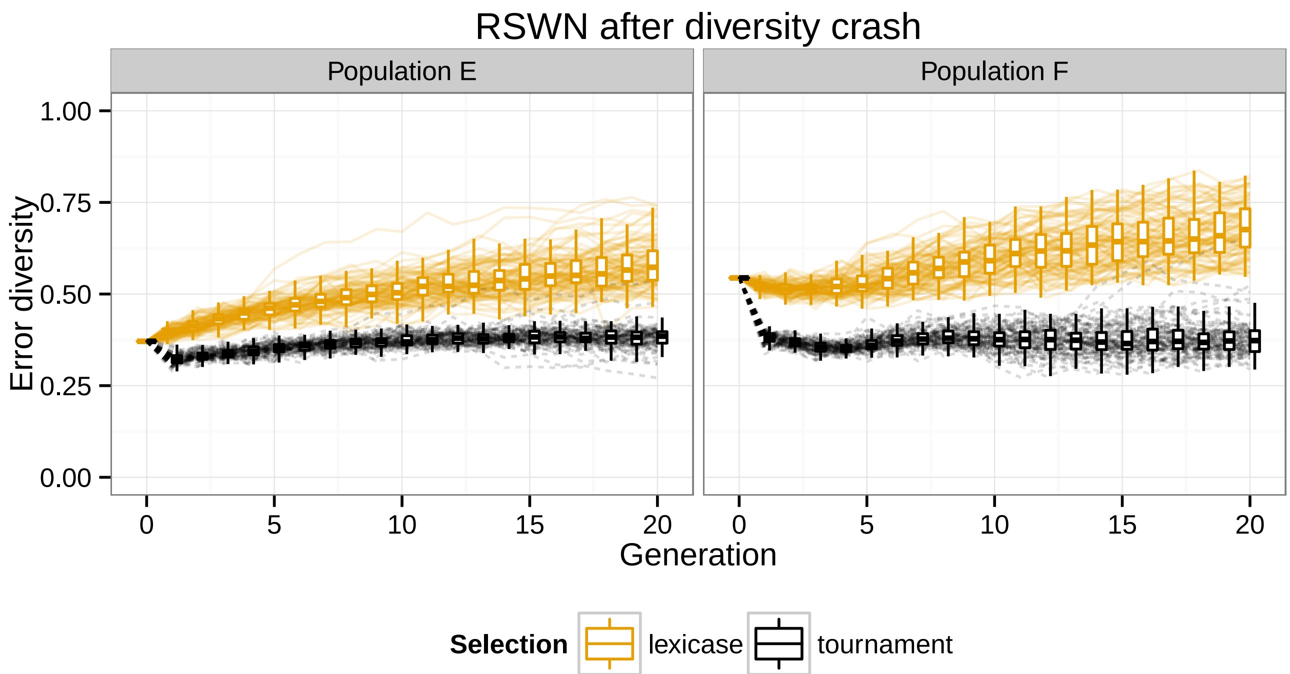


**Figure 5: Changes in diversity over 100 "re-runs" of the replace-space-with-newline problem with both lexicase and tournament selections, starting from a population with that had lost diversity in a diversity crash in a lexicase selection run.**

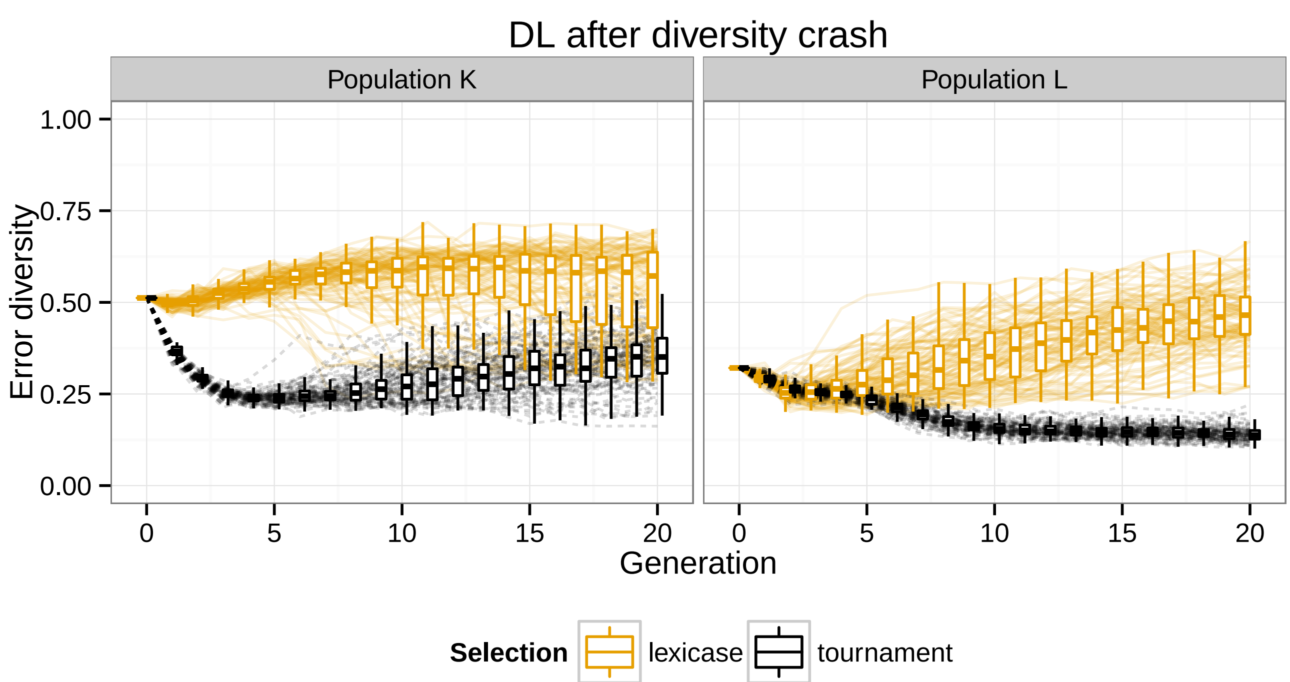


**Figure 6: Changes in diversity over 100 "re-runs" of the double-letters problem with both lexicase and tournament selections, starting from a population with that had lost diversity in a diversity crash in a lexicase selection run.**