Going Faster and Hence Further with Lexicase Selection

Li Ding, Ryan Boldi, Thomas Helmuth, Lee Spector

UMassAmherst

Manning College of Information & Computer Sciences

Abstract

Lexicase selection is a semantic-aware parent selection method, which has demonstrated success in multiple research areas including genetic programming, symbolic regression, and recently deep learning. One potential drawback of lexicase selection and its variants is that the selection procedure requires evaluating training cases in a single data stream, making it difficult to handle tasks where the evaluation is computationally heavy or the dataset is large-scale. In this work, we investigate how the weighted shuffle methods can be employed to improve the efficiency of lexicase selection. We propose fast lexicase selection, a method that incorporates lexicase selection and weighted shuffle with partial evaluation. Experiments on both classic genetic programming and deep learning tasks demonstrate that the proposed method has superior efficiency with a significantly reduced number of evaluation steps during selection, as well as sustains equivalent performance.

Vanilla Lexicase Selection

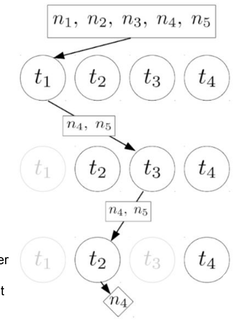
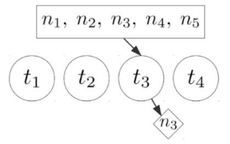


Figure demonstrating that changing the order of cases might result in fewer evaluations. If t3 is a case that has a higher selection pressure, increasing the probability of seeing this case earlier might result in fewer overall evaluations.

Fast Lexicase Selection



Algorithm 1: Fast Lexicase Selection for Parent Selection

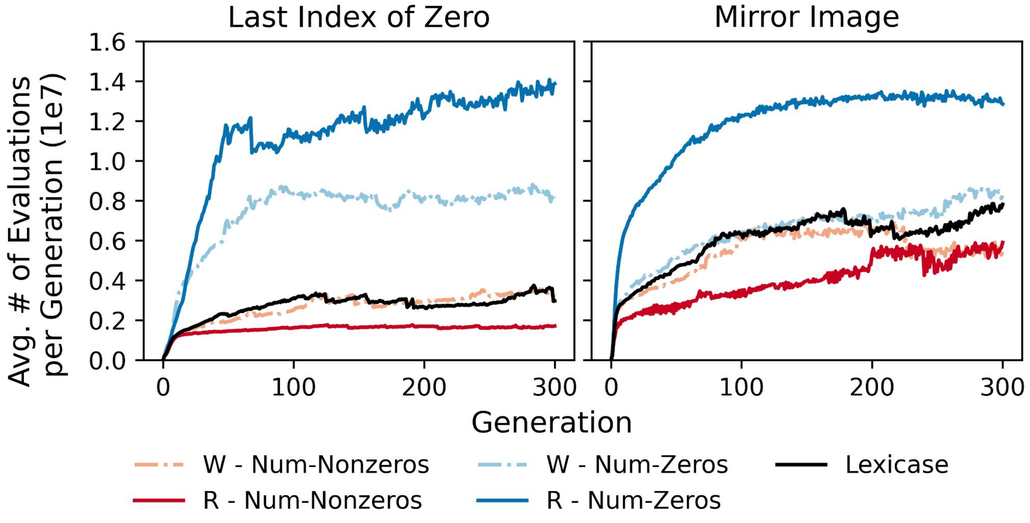
Data:
- `cases` - a sequence of all the data samples to be used in selection with default ordering
- `candidates` - the entire population of programs
- W - a weight vector for all the `cases`

Result:
- an individual program to be used as a parent
- an updated weight vector W

```
shuffled_cases ← Weighted_Shuffle(cases, W)
for case in shuffled_cases do
    i ← the index of case in cases
    results ← evaluate candidates on case
    W[i] ← Bias_Metric(results)
    candidates ← the subset of the current candidates
      that have exactly best performance on case
    if candidates contains only one single candidate then
        return candidate, W
    end
end
candidate ← a randomly selected individual in
  candidates
return candidate, W
```

Results

Program Synthesis (GP)

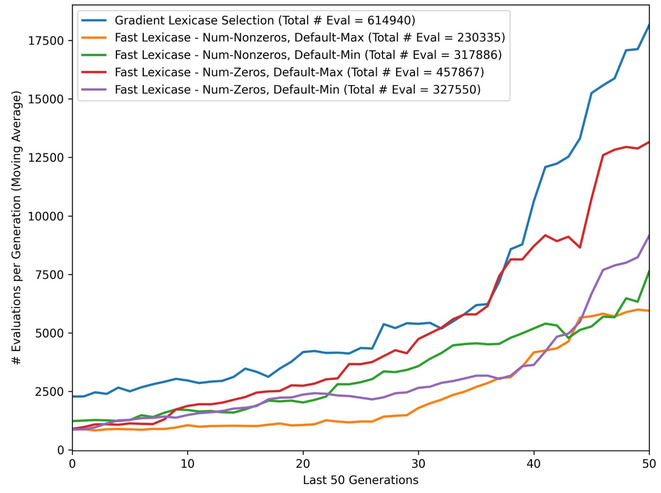


Average number of evaluations performed in a given active generation over evolutionary time while solving the Last Index of Zeros problem (left) and the Mirror Image problem (right). Ranked shuffling using the Number-of-Nonzeros bias metric consistently performs less comparisons per generation than all other shuffling methods, including lexicase selection (random shuffling).

Selection Method	Success Rate (/50 Runs)	
	Mirror Image	*Last Index of Zer*
Lexicase	43	20
Weighted - Num-Nonzeros	45	11
Ranked - Num-Nonzeros	46	21
Weighted - Num-Zeros	48	**2**
Ranked - Num-Zeros	35	**5**

Success Rates of GP runs on the Mirror Image and Last Index of Zero problems, utilizing a variety of selection techniques. Underlined are the results that were significantly worse than lexicase selection. No results were significantly better than lexicase selection across either problem.

Image Classification (DL)



Comparing number of evaluations of regular lexicase selection and fast lexicase selection. We show the last 50 generations of each method, aligned by the end. The y-axis is smoothed by applying a moving average over 20 generations. All fast lexicase selection methods outperform regular lexicase selection, with the Default-Max and Num-Nonzeros being the most efficient choice.

Method	acc.	p-val
Gradient Lexicase Selection	93.34	-
Fast Lexicase - Nonzeros, Default-Max	93.19	0.276
Fast Lexicase - Nonzeros, Default-Min	92.99	0.085
Fast Lexicase - Zeros, Default-Max	93.29	0.421
Fast Lexicase - Zeros, Default-Min	93.17	0.25

Image classification results. Besides the percentage accuracy (acc.), we also report the p-value of one proportion z-test against the regular lexicase selection. A p-value greater than the significance level (0.05) means there is no significant difference between the two methods in performance.

Conclusions

- Weighted Shuffling is a viable method to **decrease computational overhead** of the selection procedure.
- Using a hard first metric results in **significantly fewer evaluations** while maintaining similar success rate in GP.
- Ranked shuffling seems to reduce the number of evaluations more than weighted shuffle does.
- In DL, both metrics **significantly reduce** the number of evaluations per generation while maintaining similar accuracy.
- Setting the default difficulty to maximally hard when using a hard first, or maximally easy when using an easy first results in fewer evaluations.

References

Sarah Anne Troise and Thomas Helmuth. 2018. **Lexicase selection with weighted shuffle**. In Genetic Programming Theory and Practice XV. Springer, 89–104.

Li Ding and Lee Spector. 2022. **Optimizing Neural Networks with Gradient Lexicase Selection**. In International Conference on Learning Representations.

Thomas Helmuth and Lee Spector. 2015. **General program synthesis benchmark suite**. In Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation. 1039–1046.

La Cava, W., Helmuth, T., Spector, L., & Moore, J. H. (2018). **A probabilistic and multi-objective analysis of lexicase selection and epsilon-lexicase selection**. Evolutionary Computation, 1–28.