Lexicase Selection Beyond Genetic Programming

Blossom Metevier*, Anil Kumar Saini*, Lee Spector*†

October 15, 2018

*University of Massachusetts Amherst, MA
†Hampshire College, MA

Summary

Motivation

Why Lexicase?

- Proven to be helpful for several GP problems
- Not specific to GP
- Should be useful wherever there are many objectives (test cases), all of which we want to handle correctly

Why GAs? Why Boolean CSP?

- Lexicase selection is not necessarily unique to genetic programming
- We want to study lexicase selection in a less complex setting; GA provides this

- Boolean Constraint Satisfaction Problem (CSP) can easily be mapped to GAs
- Boolean CSP is more constrainted than most GP problems
- Lexicase does well with uncompromising problems
- Boolean CSP can serve as a proxy for problems with many interconnected constraints

Background: Lexicase Selection

Lexicase Selection

- Parent selection algorithm
- Employs repeated filtering steps of randomly chosen test cases

```
Result: Parent chosen for recombination
candidates := the entire population
cases := list of all test cases in a random order
while True do
    candidates := candidates who perform best on case[0]
    if only one candidate exists in candidates then
        return candidate
    end
    if cases is empty then
        return a randomly selected candidate from candidates
    end
    delete case[0]
end
```

Algorithm 1: Lexicase Selection

Lexicase Selection: An Example

- We want to evolve programs that do well over 4 objectives
- Our population size is 10
- Now we come to the point in our program that uses lexicase selection
- First we set our cases to be the number of objectives, and shuffle this list $[0, 1, 2, 3] \rightarrow [2, 0, 1, 3]$
- Then we set our candidates equal to the initial population.

Lexicase Selection: An Example

shuffled cases: 2, 0, 1, 3

case	0	1	2	3
e0	36	80	84	40
e1	47	2	84	30
e2	34	72	38	72
e3	32	96	84	72
e4	47	12	84	36
e5	17	37	84	80
e6	47	18	84	37
e7	47	23	84	84
e8	40	20	38	17
e9	87	25	6	84

Lexicase Selection: An Example

shuffled cases: 2, 0, 1, 3

case	0	1	2	3
e0	36	80	84	40
e1	47	2	84	30
e2	34	72	38	72
e3	32	96	84	72
e4	47	12	84	36
e5	17	37	84	80
e6	47	18	84	37
e7	47	23	84	84
e8	40	20	38	17
e9	87	25	6	84

Lexicase Selection: An Example

shuffled cases: 2, 0, 1, 3

case	0	1	2	3
e0	36	80	84	40
e1	47	2	84	30
e3	32	96	84	72
e4	47	12	84	36
e5	17	37	84	80
e6	47	18	84	37
e7	47	23	84	84

Lexicase Selection: An Example

shuffled cases: 2, 0, 1, 3

case	0	1	2	3
e0	36	80	84	40
e1	47	2	84	30
e3	32	96	84	72
e4	47	12	84	36
e5	17	37	84	80
e6	47	18	84	37
e7	47	23	84	84

Lexicase Selection: An Example

shuffled cases: ~~2~~, 0, 1, 3

case	0	1	3
e0	36	80	40
e1	47	2	30
e3	32	96	72
e4	47	12	36
e5	17	37	80
e6	47	18	37
e7	47	23	84

Lexicase Selection: An Example

shuffled cases: ~~2~~, 0, 1, 3

case	0	1	3
e0	36	80	40
e1	47	2	30
e3	32	96	72
e4	47	12	36
e5	17	37	80
e6	47	18	37
e7	47	23	84

Lexicase Selection: An Example

shuffled cases: ~~2~~, 0, 1, 3

case	0	1	3
e1	$\boxed{47}$	2	30
e4	$\boxed{47}$	12	36
e6	$\boxed{47}$	18	37
e7	$\boxed{47}$	23	84

shuffled cases: ~~2~~, ~~0~~, 1, 3

case	1	3
e1	2	30
e4	12	36
e6	18	37
e7	23	84

Lexicase Selection: An Example

shuffled cases: ~~2~~, ~~0~~, 1, 3

case	1	3
e7	23	84

Lexicase Selection: A Short Analysis

- Let's assume we have a population of individuals, and that there exist only two objectives, or fitness cases
- Where do individuals selected by lexicase fall on the pareto front?
- What does this mean?
- Why is this important? In aggregation, these case **specialists are not often the most fit**.

- However, they may contain features good at solving niche portions of our problem.
- Contributes to diversity.

Background: Boolean CSP

Boolean Expressions

- Evaluate to either TRUE or FALSE (1 or 0)
- $(\neg x_1 \vee x_3 \vee x_0) \wedge (x_2 \vee x_0 \vee x_4 \vee \neg x_1)$
- This formula has 5 variables, all of which are either 1 or 0
- This formula is in CNF (conjunctive normal form)
- In 3CNF, all clauses must have 3 variables
- $(\neg x_1 \vee x_3 \vee x_0) \wedge (x_2 \vee x_0 \vee x_4) \wedge (x_0 \vee x_4 \vee \neg x_1)$

Boolean CSP

- Let's look at the following boolean constraints: $(x_0 \vee x_3 \vee \neg x_1)$ and $(x_2 \vee x_1 \vee \neg x_0)$
- Let's assign to each variable a value. $\alpha = [1, 1, 1, 1]$. In this case, both the constraints evaluate to TRUE. Hence, $\alpha = [1, 1, 1, 1]$ is a solution to the CSP.
- Correct assignment does not have to be unique. Another assignment for this problem is $\beta = [1, 1, 0, 0]$.

Experiments and Results

Mapping Boolean CSP to GA

- We experiment on GA with different selection algorithms: tournament selection (with replacement), roulette selection, and lexicase selection
- $(x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \neg x_4 \vee x_5) \wedge (x_3 \vee x_5 \vee \neg x_3) \wedge (x_2 \vee x_4 \vee x_1)$
- How would we encode this?
- Candidate solutions are binary vectors of fixed length

Fitness Function

- Let's come back to our example expression
- $(x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \neg x_4 \vee x_5) \wedge (x_3 \vee x_5 \vee \neg x_3) \wedge (x_2 \vee x_4 \vee x_1)$
- Split the formula into pieces
- piece 1: $(x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \neg x_4 \vee x_5)$
- piece 2: $(x_3 \vee x_5 \vee \neg x_3) \wedge (x_2 \vee x_4 \vee x_1)$
- Essentially, each constraint is a subformula of our original expression
- We define our fitness function by the number of constraints our solution satisfied. We can interpret this as error. An assignment that solves a given problem then has a fitness value of 0.

Why Boolean Constraints?

- Remember that Boolean CSPs are a proxy for real world problems.
- Many real world problems have different components of error, and this is what our constraints represent.

Experimental Setup

- Tournament selection (various sizes)
- Lexicase selection
- Roulette (fitness proportionate) selection
- 15 different initializations
- 50 different runs for each initialization
- Hence, 750 runs for each parameter combination

Tournament Selection

- For integer-valued size t, we first form a tournament set of t individuals, each chosen with uniform probability (with replacement) from the entire population. We then return, as the selected parent, the individual in the set with the lowest total error.
- For a non-integer-valued size t between 1 and 2 we use tournament size 2 with probability t-1, and select a parent entirely randomly otherwise.

Roulette Selection

The probability of selection for an individual i that satisfies s_i constraints is s_i divided by sum of s_j for all individuals j across the population. In the degenerate case of no individuals satisfying any constraints, which would produce a denominator of zero, an individual is selected at random.

Parameters

Table 1: Problem parameters

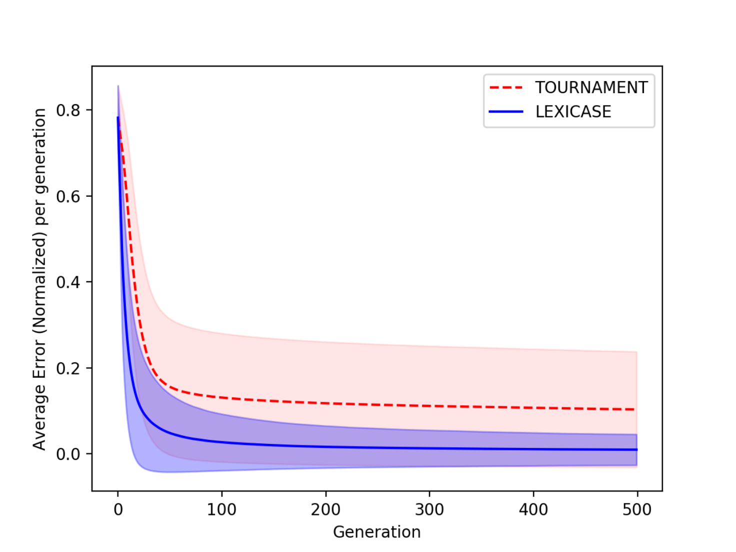
Parameter	Value
Number of variables (v)	20,30,40
Number of constraints (c)	8,12,16,32
Number of clauses per constraint (n)	20,25,30,35,40
Number of problems per combination of v, c, and n	15
Number of runs per method per problem	50
Total runs per method per combination of v, c, and n	750

Parameters

Table 2: Genetic algorithm parameters

Parameter	Value
Population size	200
Number of generations	500
Mutation operator	bit-flip
Probability of Mutation	0.1
Crossover operator	one-point
Probability of Crossover	0.9

Error Profile



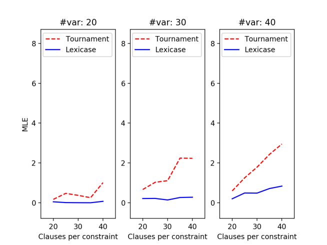
Quality Measures

- **Mean Least Error**:

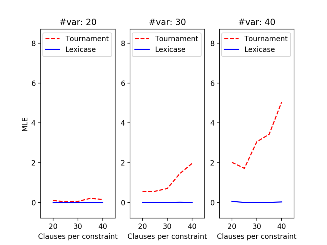
$$MLE = (1/N)\sum_i error(best\_prog_i)$$

- **Success Generation**: Number of generations the algorithm took to find a solution
- **Success Rate**: Fraction of the total runs that succeeded

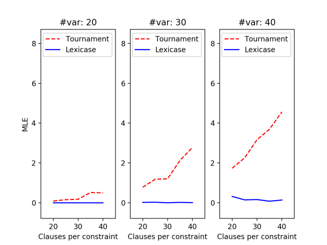
Mean Least Error



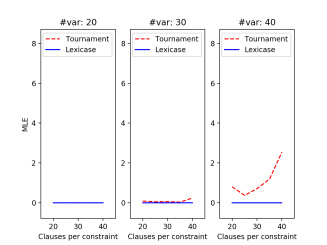
(a) $C = 8$



(a) $C = 16$

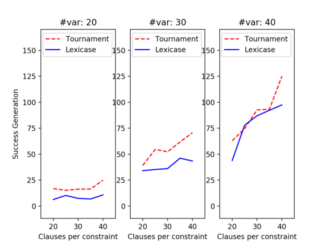


(b) $C = 12$

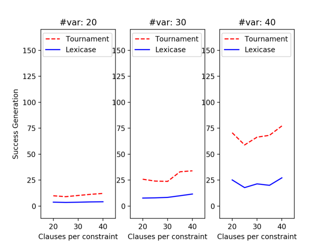


(b) $C = 32$

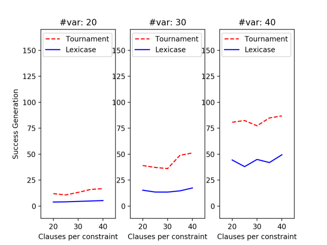
Success Generation



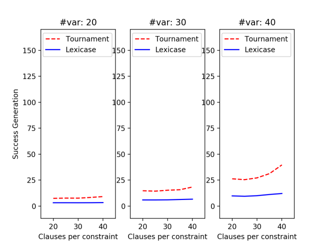
(a) $C = 8$



(a) $C = 16$



(b) $C = 12$



(b) $C = 32$

Success Rates: different selection algorithms

Table 3: Success rates. Underlines indicate statistically significant improvements, determined using a pairwise chi-square test with Holm correction and $p < 0.05$.

Number of Variables (v)	Number of Constraints (c)	Fitness Proportionate	Tournament (size 2)	Lexicase
20	8	0.835	0.867	<u>0.992</u>
20	12	0.940	0.954	<u>1.000</u>
20	16	0.980	0.987	<u>1.000</u>
20	32	0.999	1.000	1.000
30	8	0.415	0.475	<u>0.889</u>
30	12	0.614	0.697	<u>0.995</u>
30	16	0.815	0.869	<u>1.000</u>
30	32	0.983	0.995	1.000
40	8	0.205	0.257	<u>0.689</u>
40	12	0.224	0.310	<u>0.927</u>
40	16	0.433	0.576	<u>0.993</u>
40	32	0.861	0.944	<u>1.000</u>

Success Rates: different tournament sizes

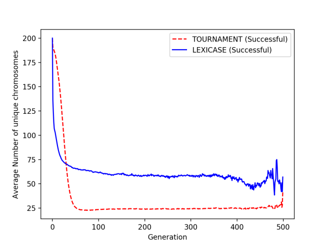
Table 4: Success rate for different tournament sizes. Boldfaced numbers indicate the highest success rate in a particular row.

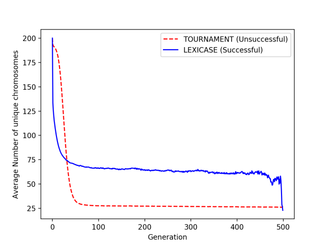
Number of Variables (v)	Number of Constraints (c)	Tournament Size 1.25	Tournament Size 1.5	Tournament Size 2	Tournament Size 4	Tournament size 8
20	8	0.850	**0.860**	0.856	0.818	0.777
20	12	0.948	0.955	**0.959**	0.952	0.934
20	16	0.982	0.987	0.988	**0.989**	0.979
20	32	**1.000**	**1.000**	0.999	**1.000**	0.999
30	8	0.443	**0.485**	0.471	0.428	0.367
30	12	0.644	0.702	**0.773**	0.712	0.618
30	16	0.850	**0.888**	0.879	0.846	0.766
30	32	0.993	**0.996**	**0.996**	0.990	0.974
40	8	0.226	**0.271**	0.137	0.120	0.105
40	12	0.254	**0.322**	0.293	0.245	0.213
40	16	0.510	**0.614**	0.503	0.423	0.335
40	32	0.938	**0.958**	0.901	0.794	0.680

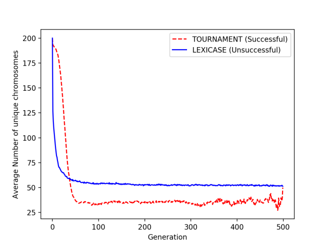
Analysis and Discussion

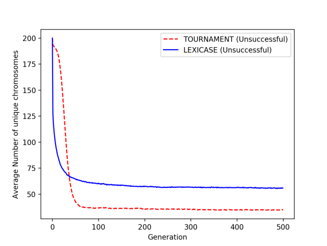
Diversity Analysis

Average number of unique chromosomes (individuals) in the population, over evolutionary time, under different conditions.









Conclusions

Future Work

- Apply lexicase to problems that can be mapped to GA
- Lexicase is being used in GP and GA as a parent selection algorithm. However, it really is just a selection algorithm for optimization over many objectives
- Diversity analysis of error vectors. We only looked at the structure of bit strings. Considering error distributions might be interesting
- Study diversity of populations produced by other parent selection algorithms

Conclusions

- Lexicase is not necessarily unique to GP
- Lexicase outperforms tournament selection
- Lexicase maintains high genome diversity
- Studying where lexicase works and where it has difficulty in the Boolean CSP domain may help us improve it

Acknowledgments

We thank Thomas Helmuth and Nic Mcphee for providing useful suggestions.

This material is based upon work supported by the National Science Foundation under Grant No. 1617087. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.