Noname manuscript No.
(will be inserted by the editor)

On the Importance of Specialists for Lexicase Selection

Thomas Helmuth · Edward Pantridge · Lee Spector

Received: date / Accepted: date

Abstract Lexicase parent selection filters the population by considering one random training case at a time, eliminating any individual with an error for the current case that is worse than the best error of any individual in the selection pool, until a single individual remains. This process often stops before considering all training cases, meaning that it will ignore the error values on any cases that were not yet considered. Lexicase selection can therefore select specialist individuals that have high errors on some training cases, if they have low errors on others and those errors come near the start of the random list of cases used for the parent selection event in question. We hypothesize here that selecting such specialists, which may have high total error, plays an important role in lexicase selection's observed performance advantages over error-aggregating parent selection methods such as tournament selection, which select specialists less frequently. We conduct experiments examining this hypothesis, and find that lexicase selection's performance and diversity maintenance degrade when we deprive it of the ability to select specialists. We also conduct experiments

Thomas Helmuth
Hamilton College
198 College Hill Rd.
Clinton, NY 13323
Tel.: +1-315-859-4507
E-mail: thelmuth@hamilton.edu

Edward Pantridge
Swoop
125 Cambridgepark Drive
Cambridge, MA 02140
Tel.: +1-877-848-9903
E-mail: ed@swoop.com

Lee Spector
Amherst College, Hampshire College, and UMass Amherst
P.O. Box 5000
Amherst, MA 01002-5000
E-mail: lspector@amherst.edu

with a form of tournament selection that has been modified to allow for the selection of specialists, and find that it performs better than ordinary tournament selection, but not as well as lexicase selection. These findings, and other data that we present here, help explain the improved performance of lexicase selection compared to tournament selection, and suggest that specialists help drive evolution with lexicase selection toward global solutions.

Keywords lexicase selection · specialists · parent selection · program synthesis · genetic programming

1 Introduction

Most parent selection methods used in genetic programming, and in evolutionary computation more generally, select individuals on the basis of scalar fitness values. For problems that involve multiple training cases, these fitness values are aggregated over all of the training cases, often by summing them. By contrast, lexicase selection selects parents on the basis of performance on un-aggregated training-case errors [34, 16, 23]. It does this by considering training cases one at a time, in a different random order for each parent selection event. For each selection it creates a pool that initially contains the entire population, and then, for each training case, it filters the pool to retain only the individuals with the best[1] error for each training case. If the pool is reduced to a single individual, then that individual is the selected parent. If multiple individuals survive filtering by all of the training cases, then a randomly chosen survivor is designated as the selected parent.

Prior work has shown that lexicase selection often works well in practice, but the reasons that it does so, and the contexts in which it does and doesn't work well, are still topics of active investigation. In the present article[2] we address one hypothesis regarding the efficacy of lexicase selection: that selecting *specialists* is important for solving problems. A "specialist" is an individual that achieves low errors on a subset of training cases while having high errors on other training cases. The total, or aggregated, error of a specialist individual is often relatively high compared to the rest of the population, since a high error on a few training cases can dominate the sum of the errors. In contrast to specialists, a generalist is an individual that performs approximately the same on all training cases.

Our motivation for the present study stems from anecdotal evidence observed in an earlier study, which suggested that specialists might contribute in important ways to the evolution of solutions [26]. This prior work also suggested that the selection of specialists might explain, to a significant degree,

[1] We assume lower errors are better, meaning the best error on a training case is the lowest one.

[2] This article is an expanded version of a paper presented at the 2019 Genetic and Evolutionary Computation Conference, which was awarded Best Paper in the Genetic Programming track [14].

the better problem-solving performance of lexicase selection relative to other parent selection methods.

More specifically, in this prior work we examined the lineage leading to a solution to the "Replace Space with Newline" software synthesis problem, evolved with a PushGP genetic programming system. In the run that we examined, the generation in which a solution first appeared actually contained 45 distinct solutions. All of these solutions were children of the same parent in the previous generation, and both this parent and *its* parent (that is, the grandparent of all of the solutions) had total error values that were in the worst quartile of their respective generations by total error. The grandparent of every solution had nearly the worst total error of its generation. Nonetheless, both the grandparent and the parent produced large numbers of offspring, including large numbers of solutions in the final generation.

A later study using a larger set of benchmark problems observed lexicase selection selecting individuals with high total error significantly more frequently than tournament selection [32]. This study also observed that lexicase selection rarely utilizes a majority of the training cases when selecting parents.

These observations motivated the present study, but anecdotal evidence is not sufficient to ground scientific understanding or to guide engineering practice. Systematic studies are required to determine the extent to which the selection of specialists is truly important, and the contexts in which this is the case. In this paper we document such a study, providing clear evidence supporting the hypothesis that the selection of specialists is responsible, in large measure, for the superiority of lexicase selection to tournament selection.

In the following sections we present background on lexicase selection and then the design, results, and analysis of our new experiments.

2 Background on Lexicase Selection

The basic and most commonly used version of the lexicase selection algorithm proceeds as follows each time a parent is required:

1. A collection of **candidates** is set initially to contain the entire population.
2. A collection of **cases** is set initially to contain all of the training cases, shuffled in random order.
3. (**Optional pre-selection**) Partition **candidates** into groups of individuals with the same error vector. Retain only one individual from each group.
4. Until a parent has been designated, loop:
   (a) Discard all individuals in **candidates** except those with exactly the lowest error for the first case in **cases**.
   (b) If just a single individual remains in **candidates**, then designate it as the parent.
   (c) If only a single item remains in **cases**, then designate a randomly chosen individual from **candidates** as the parent.
   (d) Otherwise, remove the first item from **cases**.

With regard to the lexicase selection algorithm, it is important to note that the condition in step 4b often triggers before examining all training cases and, as we will show later, often considers less than half of the cases. This scenario provides the mechanism by which lexicase selection can select specialists: A specialist that performs well on a subset of the cases may have those cases appear first in some selection event, allowing it to be selected before considering the cases on which it performs poorly.

The **optional pre-selection** step (step 3) of the lexicase selection algorithm has no functional effect on the probability of lexicase selecting any given individual, but in our experience gives significant computational time savings. Consider a group of individuals with identical error vectors. Without the pre-selection step, once this group of individuals are the only ones left in candidates, the algorithm must continue through the remaining cases, and then choose one of the individuals in the group randomly. With the pre-selection step, there is only one individual from the group in candidates, so at the point when only individuals in the group would have remained, that will be the only individual left in candidates and will be selected. Because of this, the pre-selection step can significantly reduce the number of comparisons of errors on training cases, without changing the probability of lexicase selection selecting a given individual. Note that this pre-selection step cannot be used with epsilon lexicase selection or some other relaxations of the algorithm (see below), as pre-selection changes the probabilities of selection with these algorithms.

Lexicase selection has been studied in several settings, and several variants of the basic algorithm have been proposed (for example, relaxations of the algorithm [36]). Among the most significant of these variations is epsilon lexicase selection, in which "exactly the lowest error" in the description of the algorithm is replaced with "within epsilon of the lowest error" for a suitably defined epsilon; this has proven to be particularly effective on problems with floating-point errors such as symbolic regression [24,23]. Additionally, lexicase selection has been effectively used to solve benchmark problems in areas such as general program synthesis [15,5], boolean logic and finite algebras [16, 25], learning classifier systems [1], evolutionary robotics [29,30], and boolean constraint satisfaction using genetic algorithms [28].

Lexicase selection often produces and maintains particularly diverse populations, which has been hypothesized to be responsible, in part, for its problem-solving power [10,11]. If lexicase selection does in fact select specialists more often than other parent selection techniques, this may contribute to its effects on diversity, regardless of effects on problem-solving performance. We investigate this question in Section 6.4.

An additional aspect of lexicase selection that bears consideration is the fact that selected individuals will always be non-dominated in their populations and elite with respect to at least one training case, a property that has been characterized as inhabiting the "corners" of the Pareto front [23].

3 Specialists in Genetic Programming

As we have discussed, specialists achieve low errors on some training cases and high errors on others, resulting in a relatively high total error. On the other hand, generalists perform similarly on all training cases. Consider the following training cases for the function $y = (x_1)^2 - x_2$.

x_1	x_2	y
2	1	3
3	5	4
1	3	-2

The following two tables describe the actual output ($\hat{y}$) and expected output (y) of a generalist and a specialist on each training case.

Generalist

$\hat{y}$	y	Error
10	3	7
-8	4	12
6.5	-2	8.5
Total:		27.5

Specialist

$\hat{y}$	y	Error
-100	3	103
`err`	4	1,000,000
-1.99	-2	0.01
Total:		1,000,103.01

The generalist has similar error values across all training cases while the specialist has a near zero error on one training case but high errors on the other training cases. On an actual problem with many training cases, a specialist will likely perform well on a subset of the training cases, not just one of them. Notice that the specialist in this example has received a penalty error of one million on the second training case because it could not be evaluated on the given set of inputs.

The total error of the specialist is drastically higher than the generalist. However, the generalist was not able to achieve a near zero error on any of the training cases. In an evolutionary population that is ranked by total error, the generalists will tend to have lower rank than the specialists. On the other hand, the specialist may have discovered something truly useful about solving the problem as indicated by its one (or more with real problems) nearly perfect output, and might be worth selecting to pass on its genetics to the next generation.

4 Experimental design

In Section 1 we described a single run that featured an individual in the bottom quartile of the population (when sorted by total error) that was the parent of 45 solution programs. Later, in Section 6 we will show that specialists make up large portions of the individuals selected by lexicase selection compared to tournament selection. Still, this does not answer the question of

whether selecting specialists is an important component to lexicase selection's improved performance compared to tournament selection and other selection methods, or whether it is a side effect that has little bearing on the trajectory of evolution.

Does lexicase selection perform well because it selects specialists, or can it maintain good performance without selecting individuals with poor total error? We hypothesize that lexicase selection's ability to select specialist individuals allows it to more effectively explore the search space than if it were limited to selecting individuals with good performance when measured by total error. We do not expect tournament selection to exhibit similar decreases in performance when limited to selecting individuals with good total error, since it does not often select individuals with poor total error. Additionally, we expect that limiting lexicase selection to individuals with better total error will decrease population diversity.

To test our hypotheses, we have designed and conducted an experiment in which we do not allow parent selection to select individuals with poor total error relative to the population. We implemented this restriction by adding a new survival selection step that is run before parent selection called *elitist survival selection*. During elitist survival selection, we sort the population by total error and only allow the best $X\%$ of the population to "survive" to be available to make children. We call the percent of the population that survives this step the *elitist survival rate*. We then conduct parent selection using this reduced population as normal. With 100% elitist survival we would keep the entire population (i.e. no individuals are removed); 30% elitist survival would keep only the best 300 individuals sorted by total error (out of a population of 1000) to be available for parent selection. If our hypothesis holds, we would expect lower levels of elitist survival rate to produce decreased performance with lexicase selection but not with tournament selection.

4.1 Benchmark Problems

The problems used in the experiments described here were taken from a benchmark suite of software synthesis problems, which were derived from exercises in introductory computer science textbooks [15]. These problems require general-purpose programming tools to solve, such as multiple data types (strings, integers, floats, Booleans, vectors, etc.) and various control flow structures. These problems have been addressed in several studies, using multiple genetic programming systems including PushGP [15,26,10,12,11,9], grammar guided GP [8,5,6,7], grammatical evolution [18,22], and tag-based linear GP [19,4], as well as by at least one non-evolutionary program synthesis technique [33].

We selected 8 out of the 29 benchmark problems to use in this study to reflect a wide range of requirements and difficulties. The specific problems addressed in this study are Last Index of Zero, Mirror Image, Negative to Zero, Replace Space with Newline, String Lengths Backwards, Syllables, Vector Average, and X-Word Lines. Some of these problems have been solved with

genetic programming using lexicase selection over 75% of the time out of 100 runs, while others have success rates around 25%.

In this study, we follow the guidance published with the benchmark suite about how to determine whether a run is successful or not [15]. Each GP run uses a different randomly-generated set of training cases, as well as a larger set of unseen test cases used to assess generalization. Once a program has evolved that passes all of the training cases, we stop evolution and test it on the unseen test set—if it passes those as well, it counts as a solution. If the program that passes the training cases does not pass the test set, it counts as a failed run, just like runs that reach the maximum allowed generations. In this paper we additionally automatically simplify the programs that pass the training data before testing them for generalization, a process that shrinks program size without changing the behavior of the program on the training set. Previous work has shown that automatic simplification effectively increases generalization on these benchmark problems [9].

4.2 Push and PushGP

The experiments conducted in this study were run using a PushGP genetic programming system, which evolves stack-based programs expressed in the Push programming language [37,35]. The key feature of Push for the experiments presented here is its multi-stack architecture, which includes a stack for each data type and instructions that always take their arguments from the correct stacks and push their results to the correct stacks. This facilitates the evolution of programs that use multiple, nontrivial data types and control structures, making it suitable for solving the benchmark problems described above. In addition, a wealth of prior data on the performance of PushGP on these problems can provide context for the results obtained in different experimental conditions [26,10,12,11,9]. We use the Clojure implementation of PushGP[3], which was also used in the aforementioned studies.

The parameters and configurations of the PushGP system that we used for the experiments here are the same as those described in the original benchmark description [15]. Table 1 presents the key parameters, and the code used for our experiments is available on GitHub.[4]

5 Specialists Under Tournament Selection

Tournament selection tends to select generalists because it uses an aggregate error metric, such as root mean square error, classification accuracy, or total error. To compute these kinds of error metrics, an individual's errors on all training cases must be considered. If an individual performs particularly poorly on any subset of training cases, its aggregated error will be high relative to the

[3] `https://github.com/lspector/Clojush`

[4] `https://git.io/Je8BR`

Table 1 PushGP system parameters and the usage rates of genetic operators.

Parameter	**Value**
population size	1000
max number of generations	300
tournament size for tournament selection	7
Genetic Operator Rates	**Prob**
alternation	0.2
uniform mutation	0.2
uniform close mutation	0.1
alternation followed by uniform mutation	0.5
Genetic Operator Parameters	**Prob**
alternation rate	0.01
alternation alignment deviation	10
uniform mutation rate	0.01

population and its probability of getting selected will be low. Methods such as implicit fitness sharing have been proposed to help alleviate this problem, though such methods still use a single aggregate fitness value that still punishes individuals that perform particularly poorly on a subset of training cases. While implicit fitness sharing gives some improvement over tournament selection, it has performed significantly worse than lexicase selection on a variety of problems [16, 15].

Table 2 Theoretical probability of tournament selection with tournament size 7 selecting an individual that would be removed by X% elitist survival. For example, the probability of selecting an individual removed by 50% elitist survival is 0.00781, meaning that individuals with total error worse than the median make up less than 0.8% of the parents when using tournament selection.

% Elitist Survival	**Probability of Selecting A Removed Individual**
10	0.47829
20	0.20971
30	0.08235
40	0.02799
50	0.00781
60	0.00163
70	0.00021
80	0.00001
90	0.0000001
100	0

With tournament selection, the number of times an individual can be selected is limited by the number of tournaments in which it participates. If the best member of the population participates in 1% of the tournaments for a given generation, it will be selected 1% of the time that generation, but no more. Since the expected number of tournaments in which each individual

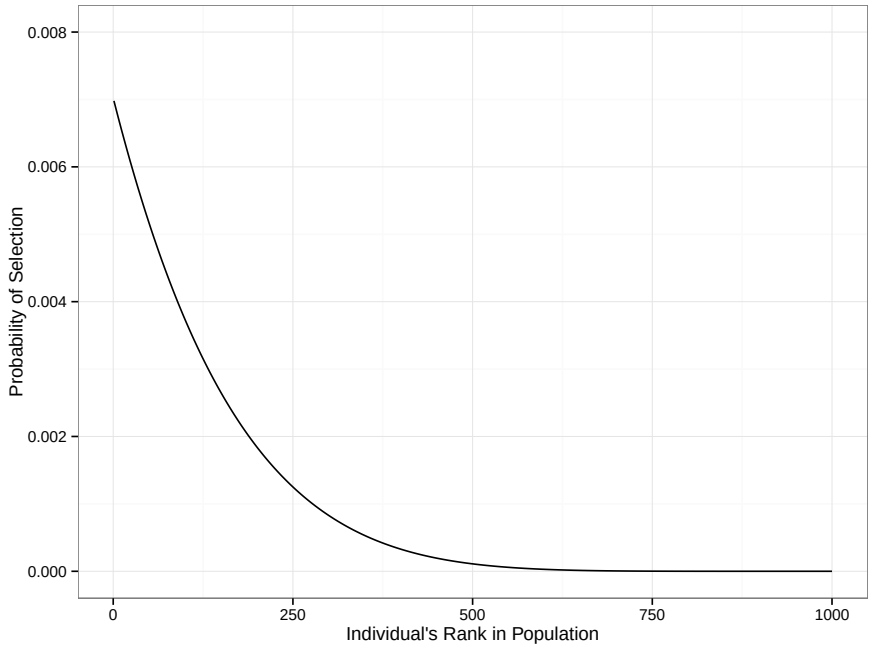

Fig. 1 Probability mass function of selecting individual with rank i out of a population of 1000 individuals using tournament selection with tournament size 7, assuming no two individuals have the same rank. This plots Equation 1.

participates is constant for a particular population size P and tournament size t, the probability of an individual being selected by tournament selection is entirely determined by its rank in the population. In particular, Bäck [2,3] shows that the probability of selecting an individual with rank $i \in [1, P]$, with $i = 1$ being the best rank, is

$$p(i) = \frac{(P - i + 1)^t - (P - i)^t}{P^t} \tag{1}$$

assuming no two individuals have the same fitness. With ties in the rankings, this equation does not hold exactly, but is approximately correct unless there are many tied individuals. We plot this probability mass function in Figure 1.

Table 2 shows that tournament selection rarely selects poor-ranking individuals. For example, it only selects individuals in the worst 50% of the population (by total error) 0.78% of the time. In our experiments without elitist survival (the same ones discussed in Section 6), tournament selection selected individuals in the worst 50% of the population at a rate of 3.3%. This is greater than the 0.78% predicted by the theoretical probability of selection due to the selected benchmark problems producing a high numbers of ties in total error. Still, with only 3.3% of selections going to such individuals, those with error worse than the median will have very little influence on evolution.

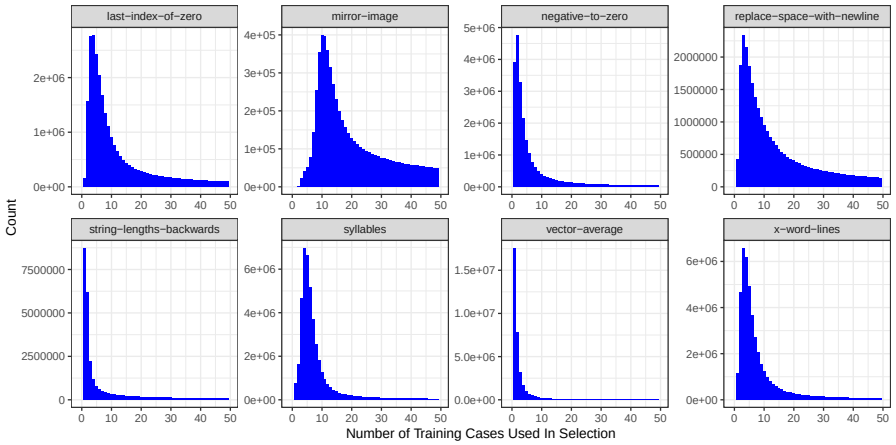

Fig. 2 Each histogram shows the distribution of the number of training cases used across all lexicase selection events for each problem. With the exception of the Mirror Image problem, lexicase selection almost never considers more than 50 training cases.

Both theoretical and empirical evidence suggest that tournament selection will almost never select specialists. Thus, the elitist survival filtering should not have a strong impact on tournament selection's ability to find solution programs, which we test empirically in Section 6.4.

6 Specialists Under Lexicase Selection

By considering training cases one at a time, lexicase selection often selects an individual without considering all of the training cases; this idea explicitly influenced the design of lexicase selection. When halting before seeing all of the training cases, the lexicase algorithm will ignore the error values on all other training cases, regardless of whether they are relatively good or relatively poor compared to the rest of the population. Lexicase selection therefore has the ability to select specialist individuals that perform extremely well on some cases while having very poor error on other cases.

However, just because it is possible for lexicase selection to select specialist individuals does not mean that it happens often or has an effect on lexicase selection's performance. In this section, we will provide evidence showing that lexicase selection can select specialists, that it selects them relatively often, and that selecting specialists is important for its performance.

6.1 Lexicase Case Usage

Figure 2 plots histograms of the number of training cases used in each selection event across each problem. It should be noted that this statistic clearly varies between problems and that these results assume the optional pre-selection step

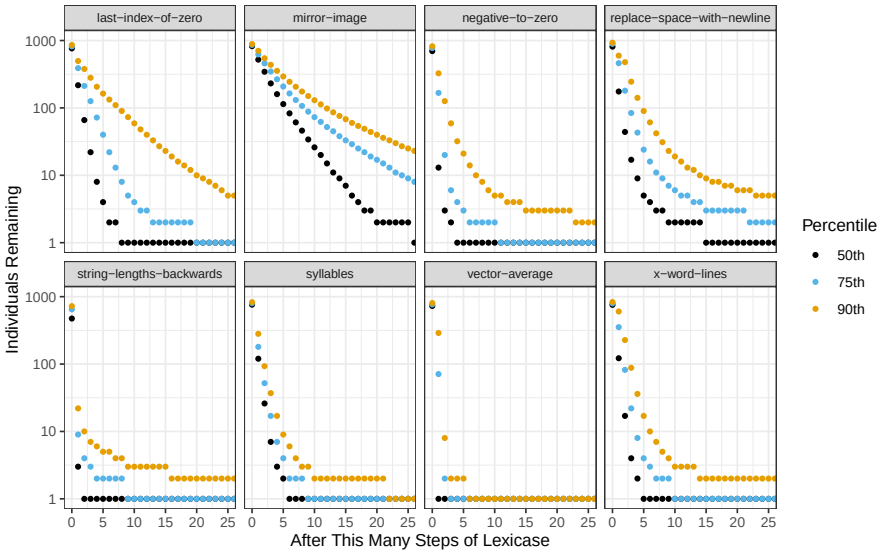

Fig. 3 The number of individuals remaining after each step of lexicase selection (i.e. each training case considered) across all selections in 100 runs using lexicase selection. Percentiles are taken from every selection event in all runs at that number of steps. Hence 50th represents the median number of individuals remaining at that step, etc. Note the log-scale y-axis.

described in Section 2; otherwise, any time multiple individuals have the same error vector, 100% of the training cases would be used, since none of them would differentiate the individuals.

Figure 2 shows that, in practice, lexicase selection rarely considers more than 25 training cases, and often less than 10; in comparison, each of these problems has 100 or more training cases. These measurements agree with the empirical results on a different set of benchmarks obtained in a previous study [32]. These results provide evidence that many of lexicase's selections ignore more than 75% of the training cases; it is certainly possible that some of these selected individuals achieved poor errors on some of the ignored training cases.

6.2 Individuals Remaining

Each step of the lexicase selection algorithm considers a new training case and filters the pool of candidates to include only those individuals with the best error on that case. After each step, fewer individuals remain in contention to be selected. Figure 3 gives the number of individuals remaining after every step of lexicase selection, across all parent selections in every run for each problem. We plot the median, 75th percentile, and 90th percentile of numbers of individuals remaining. Note that after lexicase selection has reduced the candidate pool

down to 1 individual (say, at step N), that selection is considered to have 1 individual remaining for every step $> N$ for these calculations.

We see that for many of these problems (each of which features 100+ training cases), most selections have reduced the pool of candidates to 10 or fewer individuals after considering a small number of cases. This indicates that even if a selection requires many more cases to decide, for the majority of the algorithm, only a small number of individuals are in consideration. Note that the Mirror Image problem, which has by far the most individuals remaining at every step, is a boolean output problem, meaning that every program is either correct or incorrect, unlike other problems which have varying degrees of incorrectness.

These results indicate that any individual that is selected by lexicase selection must not only be good at however many cases it takes for lexicase selection to prefer it to any other individual, but also that it is one of a few individuals that are best in the population on the first few cases that lexicase considers. Thus the individual must specialize in those cases compared to the population, in the sense that only a few other individuals in the population perform as well on those cases. This meaning of "specialization" is subtly different from (but related to) our definition in the rest of the paper of "performing very well on a subset of cases while performing poorly on another subset." That said, it adds to the evidence that lexicase selection selects specialists.

6.3 Ranks of Selected Individuals

Across all problems, lexicase selection selects individuals in the worst 50% of the population (by total error) at a rate of 7.9%. Although this may seem low, it is more than twice the rate of tournament selection, as discussed in Section 5.

Figure 4 shows how the median rank (when sorted by total error) of individuals selected by lexicase selection changes throughout evolution. Lexicase selection begins evolution by selecting individuals with higher (i.e. worse) rank than tournament selection on most problems. As evolution searches the space, the median rank of individuals selected by lexicase selection dips lower than the median empirically recorded rank of individuals selected by tournament selection on some problems. This is due to the low levels of semantic and behavioral diversity produced by tournament selection leading to many individuals with equal total error [10]. Individuals with equal total error are ranked arbitrarily, yet tournament selection will select between them with equal probability, which inflates the observed median rank. The red horizontal line in Figure 4 shows the theoretical expected rank of individuals selected by tournament selection assuming each individual's total error is unique. Lexicase selection rarely selects individuals having a lower rank than the expected rank of tournament selection.

For many of the problems in Figure 4, the median rank of individuals selected by lexicase slowly decreases throughout evolution. This indicates that

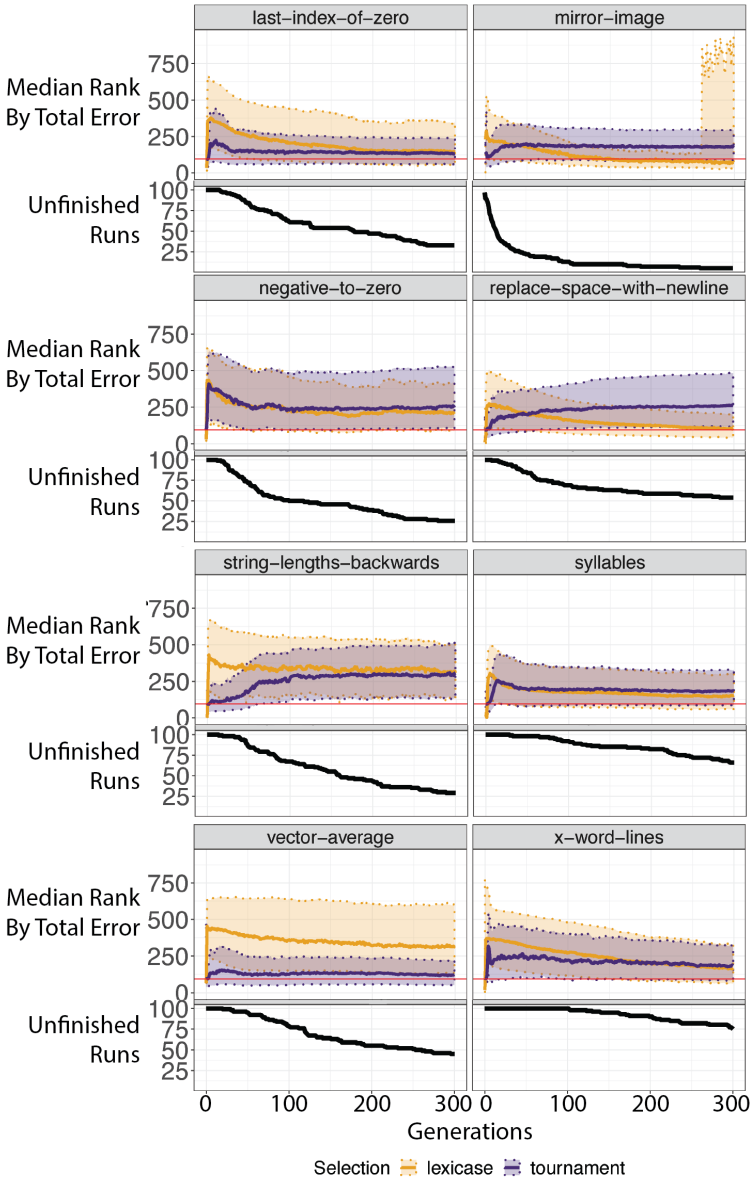

Fig. 4 The median rank of individuals selected at each generation by lexicase selection and tournament selection. The colored regions surrounding the median lines show the range between the first and third quartiles. The red horizontal lines show the expected median rank of individuals selected by tournament selection assuming all individuals have a unique total error. The small line plots below each problem show the number of unfinished runs at each generation with lexicase selection. These plots show that as runs finish due to candidate solutions being found, the aggregate measurements over the later generations become less representative.

lexicase selection turns more toward low-rank (better individuals with lower total error) individuals later in runs. While we are not sure exactly what causes this phenomenon, we hypothesize that, in many cases, lexicase selection has concentrated on one part of the search space, attempting to refine one or more promising programs into solutions (and likely often doing so). Additionally, many runs of lexicase terminate when finding solution programs before the maximum number of generations, meaning that the remaining runs that have not finished are the only ones factoring into the upper plots. Once many runs have finished, the remaining runs may not be representative of what a larger sample of runs would show, which is clearly the case for the Mirror Image problem.

Since lexicase selection only retains individuals with elite errors on the first test case it considers, every selected individual is elite on a subset of the training cases. In combination with the observed tendency to select individuals with high total error ranks compared to tournament selection, it can be concluded that lexicase selection is definitively selecting specialists at a much higher rate than tournament selection.

6.4 Importance of Selecting Specialists with Lexicase Selection

In Section 4, we presented the hypothesis that lexicase selection's ability to select specialist individuals with poor total error improves its performance compared to the setting in which it is limited to selecting individuals with good total error. To test this hypothesis, we conduct runs of PushGP using elitist survival with elitist survival rates of 10% to 100% in increments of 10. By using elitist survival, we can force selection (lexicase or tournament) to not select individuals with total error worse than some percent of the population. Thus, for example, if it is important for lexicase to select specialist individuals with rank (when sorted by total error) worse than 60% of the population, then we would expect lexicase selection with 60% elitist survival to perform worse than with 100% survival of the population.

Based on Equation 1, we can calculate how often we would expect tournament selection to select the individuals excluded by elitist survival.[5] The probabilities of tournament selection choosing an individual removed by elitist survival at different rates are given in Table 2. Tournament selection would select a decent proportion of the individuals removed by 30% elitist survival, at around 0.08. We can see that most of those individuals have ranks between 30% and 50%, since tournament selection selects individuals worse than the median with probability of only about 0.0078. Thus, we would not expect 50% survival elitism to affect the performance of tournament selection, and certainly not 70% survival elitism. Even 20% survival elitism may have negligible effects.

[5] Figure 1, which plots the probability distribution defined by Equation 1, is useful to visualize these cumulative probabilities.

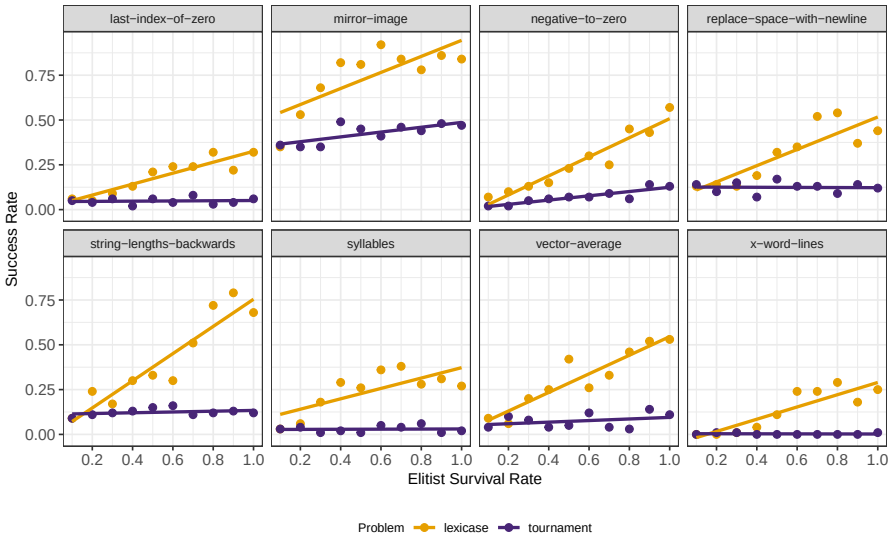

Fig. 5 The impact of elitist survival filtering on the ability of lexicase selection and tournament selection to find generalizing solution programs. As the elitist survival rate increases, lexicase selection tends to find more solutions, while tournament selection does not.

Figure 5 gives the number of successful runs on 8 benchmark problems using elitist survival rates of 10% to 90% in increments of 10 with lexicase and tournament selection. We also include 100% elitist survival, which is equivalent to not using elitist survival, since the entire population is kept. We plot a linear regression line for each problem, and use an F-test at the 0.05 level to determine if there is a relationship between the elitist survival rate and the number of solutions for each method.

On all 8 of the problems, there is a significant relationship between the elitist survival rate and the number of solutions found by lexicase selection, indicating that lexicase performs significantly worse when limited to smaller elite proportions of the population. On the other hand, when using tournament selection, only two of the problems (Mirror Image and Negative to Zero) showed a significant relationship between elitist survival rate and number of solutions. As predicted by the small effects of lower-rank individuals on tournament selection (as discussed in Section 5), removing those lower-rank individuals has little effect on the performance of tournament selection. In fact, limiting tournament selection to only the top 10% of the population by total error gave numbers of successes insignificantly different from using the entire population on every problem except for Negative to Zero (using a chi-squared test).

The *behavioral diversity* of a population is the proportion of distinct behavior vectors that are present in a population, where a *behavior vector* is simply the outputs of a program when run on the training cases [20]. The behavior vector is sometimes called the *semantics* of the program [25], as in

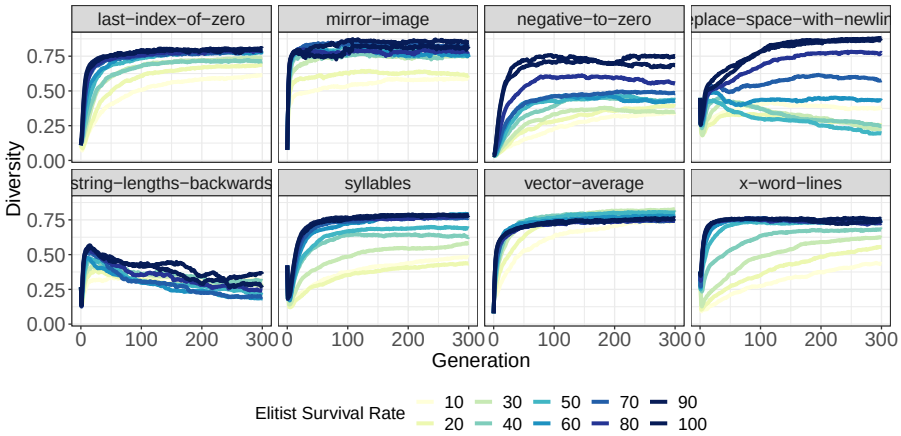

Fig. 6 Mean behavioral diversity for lexicase selection using elitist survival at different rates.

geometric-semantic genetic programming [31]. We plot the population behavioral diversity of lexicase selection in Figure 6. We do not present the diversities of populations under tournament selection, which were very low for every problem and every elitist survival rate, mirroring to previous studies comparing the behavioral diversity of lexicase and tournament selection [10, 11]. This result is consistent with the unchanged performance of tournament selection with elitist survival, both of which can be explained by the small portion of selections affected by elitist survival.

For most problems in Figure 6, the behavioral diversity of runs using lexicase selection decreases as the number of individuals removed by elitist survival increases. We see this decrease across all problems besides Vector Average, though the impact varies per problem. On the Replace Space With Newline problem, lexicase selection with elitist survival rates below 80% grow in diversity early, but then lose diversity and finish the remainder of the run with low levels of diversity. On many of the other problems, the lower rates of elitist survival have similar curves to the higher rates, just at lower levels.

One interesting finding here is that for the Vector Average problem, removing the worst individuals *increases* behavioral diversity, down to 30% elitist survival. This strange pattern says that if lexicase selection is given fewer individuals to select, the resulting populations will be more diverse. Despite the higher levels of diversity, Vector Average followed a similar pattern of poorer performance with low levels of elitist survival, as shown in Figure 5.

These results provide evidence supporting our hypothesis that lexicase selection makes use of specialist individuals with poor total error relative to the rest of the population—individuals that presumably have poor errors on some training cases but good errors on others. Lexicase selection shows clear correlation between elitist survival and success rate on every problem, performing better when able to select from the worst individuals when sorted by

total error. As discussed earlier, individuals selected by lexicase must be elite on a subset of the training cases; those that also have poor total error are therefore specialists. Lexicase selection performs better when allowed to select these specialists, which clearly help drive the direction of evolution toward more solutions.

Our plots show that behavioral diversity in lexicase selection runs decreases, sometimes significantly, as we decrease the number of individuals that survive elitist survival. These plots support our hypothesis that the high diversity seen in runs using lexicase selection is influenced by lexicase selection's ability to select individuals with relatively poor total error. Selecting specialists thus allows lexicase selection to better explore the search space, likely contributing to its better performance. The decreased diversity in runs using lower rates of elitist survival likely contributes to the corresponding decreases in performance observed in Figure 5.

As expected, tournament selection was not significantly affected by elitist survival selection at any level, even when removing 90% of the population, on 6 out of 8 problems. As we saw in Table 2, tournament selection simply does not often select specialist parents from the bottom ranks of the population. Instead, it concentrates on the individuals with the best total error, and mostly selects from the top 20% of the population, with more than half of selections coming from individuals in the top 10% of the population. Tournament selection's tendency to select individuals in the top ranks of the population, ignoring specialist individuals with worse total error, explains at least part of the difference in performance between it and lexicase selection seen in previous studies [15,16].

6.5 Lexicase Selection and UMAD

Our original experiments, described above and in [14], use the combination of genetic operators given in Table 1. Alternation is a crossover operator that copies stretches of each parent at a time; uniform mutation replaces individual instructions with some probability; and uniform close mutation changes the position of parentheses in the Push programs [17]. Since running those experiments, we have started using the new genetic operator "uniform mutation with addition and deletion" (UMAD) [13]. UMAD performs quite well when using PushGP on the program synthesis problems explored here, and in fact has produced the best results of any genetic operators to date, despite only consisting of single-parent mutation. As such, we decided to replicate the elitist survival rate experiment, this time with UMAD as the only genetic operator and lexicase for parent selection.

We use size-neutral UMAD as the only genetic operator, with an addition rate of 0.09, as recommended in [13]. This means that each gene has a probability of 0.09 of having a new random gene inserted before or after it, and then every gene has a corresponding probability of being deleted, such that the child genome is the same size as the parent on average.

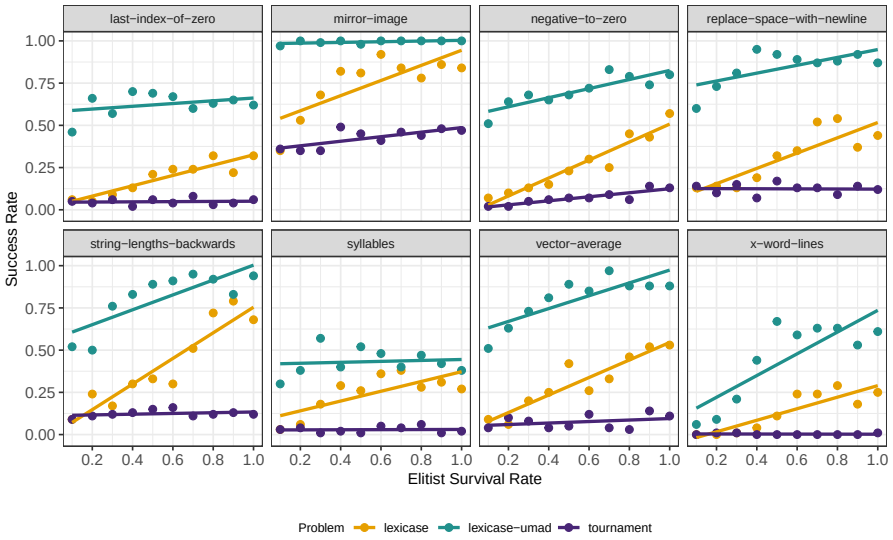

Fig. 7 The impact of elitist survival filtering on the ability of lexicase selection with UMAD as the only genetic operator (lexicase-umad) to find generalizing solution programs. We include lexicase and tournament selection with the original genetic operators (as in Figure 5) for comparison.

In Figure 7, "lexicase-umad" gives results using UMAD along with lexicase selection. As expected, PushGP with UMAD found many more solution programs, no matter what the elitist survival rate. We see that for three problems (Last Index of Zero, Mirror Image, and Syllables), decreasing the elitist survival rate has little to no effect on the success rate, leading to regression lines that are essentially flat. For all five other problems there is a significant relationship between elitist survival rate and success rate, indicating the importance of selecting specialists for those problems when using UMAD. Thus for some, but not all, problems, the importance of lexicase selecting specialists also is evident when using UMAD as the genetic operator, but this relationship isn't as strong as when using our previous set of genetic operators.

7 Specialists Under Subsampled Training Case Selection

To give further evidence for the importance of selecting specialist individuals, we devised a new parent selection scheme that makes modifications to standard tournament selection in a way that makes it more likely to select specialists. Our goal is to investigate the importance of selecting specialists beyond the details of the lexicase algorithm. To be clear, this new selection method is not devised because we think it will compare favorably with methods like lexicase selection, but instead because it will show how a selection method

similar to tournament selection but with more emphasis on specialists can create improvements compared to tournament selection.

The new parent selection method, which we call *subsampled training case selection*, has two major differences from tournament selection. First, for each selection event, we use a subsample of the training cases and select the individual with the best summed total error on the subsample. To allow for a range of sizes of subsamples, we choose subsample size by sampling a normal distribution, rounding to the nearest integer and forcing a minimum of 1 and a maximum of the number of cases. After deciding on the subsample size, we sample that number of cases at random.

Second, we make the "tournament set" contain every individual in the population, instead of only consisting of a small subset of the population. Thus, we select the best individual in the population on the subsample of training cases, with ties broken at random.

We created subsampled training case selection because it allows specialist individuals, which perform well on some training cases and poorly on others, the chance to be selected when the cases that they specialize in are in the random subsample of cases. While it has this in common with lexicase selection, it does not randomize the relative importance of cases as lexicase selection does, and thus in some sense shares similarities with both lexicase and tournament selection. The use of the entire population instead of a tournament set ensures that a specialist will be considered when its best cases are used in a selection event.

We conducted a set of GP runs with subsampled training case selection. In these runs, we tested two different settings of μ (mean percent of cases used) and σ (standard deviation) for sampling the rounded normal distribution for subsample sizes. The values for these settings were inspired by the numbers of cases typically used in lexicase selection (see Section 6.1). One setting used $\mu = 0.1$ and $\sigma = 0.05$, and the other used $\mu = 0.2$ and $\sigma = 0.1$. As the success rate of GP when using $\mu = 0.1$ and $\sigma = 0.05$ was better on 10 of the 11 problems we tested, those will be the settings used in our results here.

Table 3 gives the number of successful runs out of 100 comparing tournament, subsampled training case, and lexicase selection. These runs used UMAD as the only genetic operator, as in Section 6.5. Subsampled training case selection achieved significantly better results than tournament selection on 8 of the 11 problems, while lexicase selection was significantly better than tournament on 10 of the problems. Additionally, lexicase selection outperformed subsampled training case selection on 4 out of 11 problems.

The differences in success rates show how subsampled training case selection's ability to select individuals that specialize in a subset of the training cases dramatically improves performance compared to tournament selection without incorporating lexicase selection's randomly-ordered importance of cases. That said, lexicase's random shuffling of cases does seem to play some role in its success, as it proved significantly better than subsampled training case selection on four problems.

Table 3 Number of successful runs out of 100 for tournament, subsampled training case (STC), and lexicase selection. Bold results are significantly better than tournament, and underline means significantly better than STC. No set of STC runs were significantly better than lexicase. Significance was determined using a pairwise chi-squared test with $\alpha = 0.05$ and Holm correction.

Problem	Tournament	STC	Lexicase
Compare String Lengths	3	**45**	**32**
Double Letters	0	**10**	**19**
Last Index of Zero	30	**63**	**62**
Mirror Image	100	100	100
Negative to Zero	30	**50**	**80**
Replace Space with Newline	41	**88**	**87**
Scrabble Score	0	5	**13**
String Lengths Backwards	27	**62**	**94**
Syllables	2	**46**	**38**
Vector Average	32	**72**	**88**
X-Word Lines	0	1	**61**

More generally, these results provide evidence that tournament selection's method of only considering a small portion of the population in the tournament set is detrimental to solving problems. Both lexicase selection and subsampled training case selection consider the entire population for each selection event, meaning that they select whichever individual in the population best fulfills their requirements (case ordering for lexicase and total error on the subsample for subsampled training case). These methods allow specialist individuals to be selected when the cases at which they excel arise, instead of randomly being left out of contention as in tournament selection.

In fact, we conducted a set of subsampled training case selection runs using a tournament size of 7 instead of the entire population. We found the results indistinguishable from tournament selection—not significantly different from tournament selection on any problem and significantly worse than subsampled training case selection without a tournament set on the same 8 problems. Thus allowing subsampled training case selection to select from the entire population is critical to its improvement over tournament selection.

8 Discussion: What Makes Lexicase Selection Work?

We have made the argument that specialists are important for lexicase selection's success, and for successful parent selection more generally. Others have noted this as well: the importance of specialists in lexicase selection was one of the stated motivations for the development of batch tournament selection, which has also shown good performance by selecting specialists [27]. However, lexicase selection has other traits that may also contribute to its good performance. Here we will consider other work that has aimed to better understand lexicase selection in order to develop a more comprehensive picture about how it works. We will also suggest future work that could contribute to understanding it better.

One of lexicase selection's key algorithmic ideas is that it considers cases in a new random order for each selection event. The importance of each case for any particular selection event is determined by this randomly-ordered sequence of cases, which is processed lexicographically (hence **lexi**-case). The cases encountered by individuals in a single lexicase selection event can be thought of as *random challenges* faced by each individual, until an individual proves itself better than the others on those random challenges [36]. While subsampled training case selection can select specialists and outperformed tournament selection (see Section 7), it does not place any emphasis on performing best on any given training case in its subsample. Thus, as discussed further in [36], it seems that the random lexicographic ordering of cases contributes to lexicase selection's success.

Previous studies have shown lexicase selection to be capable of producing and maintaining diverse populations compared to tournament selection and other selection methods [10,11,30]. Such diversity tends to correlate with problem-solving performance, as having high population diversity can indicate good exploration of the search space. However, increased diversity has not been shown to *cause* improved performance. Indeed, some parent selection methods (in particular novelty search) have demonstrated higher levels of diversity than lexicase selection while achieving significantly worse problem-solving performance [21]. Additionally, some lexicase selection variants produced higher diversity along with worse performance compared to standard lexicase selection in an evolutionary robotics setting [30]. Thus it seems that while lexicase selection causes increases in both problem-solving performance and diversity, the diversity itself is not directly responsible for the improved performance, but instead is indicative of the exploration that lexicase produces while effectively searching for a solution program.

Populations evolving by lexicase selection have often exhibited *hyperselection*, in which single individuals in one generation are selected as parents for many, and sometimes nearly all, of the children in the next generation. Earlier work showed that while lexicase selection hyperselects individuals at high rates, this hyperselection does not significantly impact the performance of GP using lexicase selection [12]. Thus hyperselection seems to be a benign side effect of lexicase selection, neither benefiting or hindering its performance.

That said, this earlier work only considered the hyperselection of single individuals, but not the hyperselection of *groups of individuals with identical error vectors*. We know that individuals with identical error vectors frequently receive selections, and it is possible that such semantically-equivalent individuals undergo *semantic hyperselection*. With semantic hyperselection, it is possible that one error vector has an outsized influence on the next generation without giving many selections to a single individual. Studying semantic hyperselection could tell us more about this phenomenon and its effects on lexicase selection.

9 Conclusions

Numerous demonstrations of lexicase selection's search performance have concluded it is superior to tournament selection on a variety of tasks. Previous attempts to explain this behavior have observed increases in population diversity, guarantees of non-dominated selections, and the possibility of selecting individuals with high total error. This paper formalizes the hypothesis that lexicase selection's performance is in part due to its tendency to select specialists over generalists, especially compared to tournament selection. These specialists, with excellent errors on some training cases yet poor total error, receive little attention from most other parent selection methods, which aggregate performance into a single fitness metric.

This paper presents theory explaining the exceedingly low probability of tournament selection selecting specialists, along with empirical results that support this theory. In contrast, we observe the comparatively high rate at which lexicase selection selects specialists. We additionally provide evidence of test case usage during lexicase selection, indicating that few test cases are typically used in any one parent selection event, showing how lexicase can select specialists by ignoring training cases on which the specialist performed poorly.

To support the hypothesis that the selection of specialists is a key component of lexicase selection's search performance, an elitist survival filter was applied with various degrees of strictness before conducting parent selection. This filtering removed all potential specialists and forced lexicase selection to select among more generalist individuals, which have better total error. The filter significantly reduced the number of solutions evolution was able to find, implying that the presence of specialists was crucial to lexicase selection's performance. Furthermore, tournament selection was not significantly impacted by elitist survival filtering. Additionally, we discussed the effects of specialists on a population's diversity under lexicase selection, finding that specialists typically contribute to lexicase selection's ability to maintain high rates of population diversity. We also designed subsampled training case selection, which selects the individual from the population that performs best on a random subset of the training cases, allowing it to select specialists. It too outperformed tournament selection (though not lexicase selection), giving more evidence for the importance of specialists.

These findings suggest that future work to improve parent selection techniques should consider their ability to select specialist individuals, which provided significant benefits to lexicase selection in this study. Additionally, we solely focused on the automatic program synthesis domain here; programs in this domain can use control flow structures to act in different modalities for different inputs [34], potentially leading to the development and importance of specialists. It could prove informative to replicate this study using genetic programming in other domains, especially ones with relatively small instruction sets such as symbolic regression.

How does selecting specialists lead to solving problems? How are the skills of specialists adapted or combined into better individuals? These questions go beyond the selection of parents and depend on how the GP system generates children from specialist parents. A solution program must by definition be a generalist, since it perfectly passes all of the test cases. Future work should consider how generalists are constructed from specialists, and if there are better ways of doing so.

The specialists selected by lexicase selection here were subjected to the same genetic operators used in other studies with PushGP. However, we could imagine designing genetic operators with specialists in mind to better make use of their novel abilities. For example, when combining two specialists, should we use a different recombination operator than when combining two generalists, to have a better chance at reaping the benefits of both parents? We could imagine such operators increasing the efficiency of evolution as it combines the specializations of individuals until it finds a general solution.

Acknowledgements We thank Nicholas McPhee and members of the Hampshire College Institute for Computational Intelligence for discussions that advanced this work. 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.

References

1. Aenugu, S., Spector, L.: Lexicase selection in learning classifier systems. In: GECCO '19: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 356–364. ACM, Prague, Czech Republic (2019). DOI doi:10.1145/3321707.3321828
2. Bäck, T.: Selective pressure in evolutionary algorithms: a characterization of selection mechanisms. In: Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on, pp. 57–62 vol.1 (1994). DOI 10.1109/ICEC.1994.350042
3. Blickle, T., Thiele, L.: A mathematical analysis of tournament selection. In: Proceedings of the 6th International Conference on Genetic Algorithms, pp. 9–16. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1995). URL `http://dl.acm.org/citation.cfm?id=645514.658088`
4. Ferguson, A.J., Hernandez, J.G., Junghans, D., Lalejini, A., Dolson, E., Ofria, C.: Characterizing the effects of random subsampling and dilution on lexicase selection. In: W. Banzhaf, E. Goodman, L. Sheneman, L. Trujillo, B. Worzel (eds.) Genetic Programming Theory and Practice XVII. East Lansing, MI, USA (2019)
5. Forstenlechner, S., Fagan, D., Nicolau, M., O'Neill, M.: A grammar design pattern for arbitrary program synthesis problems in genetic programming. In: M. Castelli, J. McDermott, L. Sekanina (eds.) EuroGP 2017: Proceedings of the 20th European Conference on Genetic Programming, *LNCS*, vol. 10196, pp. 262–277. Springer Verlag, Amsterdam (2017). DOI 10.1007/978-3-319-55696-3_17
6. Forstenlechner, S., Fagan, D., Nicolau, M., O'Neill, M.: Extending program synthesis grammars for grammar-guided genetic programming. In: A. Auger, C.M. Fonseca, N. Lourenco, P. Machado, L. Paquete, D. Whitley (eds.) 15th International Conference on Parallel Problem Solving from Nature, *LNCS*, vol. 11101, pp. 197–208. Springer, Coimbra, Portugal (2018). DOI 10.1007/978-3-319-99253-2_16. URL `https://www.springer.com/gp/book/9783319992587`

7. Forstenlechner, S., Fagan, D., Nicolau, M., O'Neill, M.: Towards effective semantic operators for program synthesis in genetic programming. In: GECCO '18: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1119–1126. ACM, Kyoto, Japan (2018). DOI 10.1145/3205455.3205592
8. Forstenlechner, S., Fagan, D., Nicolau, M., O'Neill, M.: Towards understanding and refining the general program synthesis benchmark suite with genetic programming. In: M. Vellasco (ed.) 2018 IEEE Congress on Evolutionary Computation (CEC). IEEE, Rio de Janeiro, Brazil (2018)
9. Helmuth, T., McPhee, N.F., Pantridge, E., Spector, L.: Improving generalization of evolved programs through automatic simplification. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '17, pp. 937–944. ACM, Berlin, Germany (2017). DOI 10.1145/3071178.3071330. URL http://doi.acm.org/10.1145/3071178.3071330
10. Helmuth, T., McPhee, N.F., Spector, L.: Lexicase selection for program synthesis: A diversity analysis. In: R. Riolo, W.P. Worzel, M. Kotanchek, A. Kordon (eds.) Genetic Programming Theory and Practice XIII, Genetic and Evolutionary Computation. Springer, Ann Arbor, USA (2015). DOI 10.1007/978-3-319-34223-8. URL http://www.springer.com/us/book/9783319342214
11. Helmuth, T., McPhee, N.F., Spector, L.: Effects of lexicase and tournament selection on diversity recovery and maintenance. In: GECCO '16 Companion: Proceedings of the Companion Publication of the 2016 Annual Conference on Genetic and Evolutionary Computation, pp. 983–990. ACM, Denver, Colorado, USA (2016). DOI 10.1145/2908961.2931657
12. Helmuth, T., McPhee, N.F., Spector, L.: The impact of hyperselection on lexicase selection. In: T. Friedrich (ed.) GECCO '16: Proceedings of the 2016 Annual Conference on Genetic and Evolutionary Computation, pp. 717–724. ACM, Denver, USA (2016). DOI 10.1145/2908812.2908851
13. Helmuth, T., McPhee, N.F., Spector, L.: Program synthesis using uniform mutation by addition and deletion. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '18, pp. 1127–1134. ACM, Kyoto, Japan (2018). DOI doi:10.1145/3205455.3205603. URL http://doi.acm.org/10.1145/3205455.3205603
14. Helmuth, T., Pantridge, E., Spector, L.: Lexicase selection of specialists. In: GECCO '19: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1030–1038. ACM, Prague, Czech Republic (2019). DOI doi:10.1145/3321707.3321875. URL https://dl.acm.org/citation.cfm?doid=3321707.3321875
15. Helmuth, T., Spector, L.: General program synthesis benchmark suite. In: GECCO '15: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, pp. 1039–1046. ACM, Madrid, Spain (2015). DOI 10.1145/2739480.2754769. URL http://doi.acm.org/10.1145/2739480.2754769
16. Helmuth, T., Spector, L., Matheson, J.: Solving uncompromising problems with lexicase selection. IEEE Transactions on Evolutionary Computation **19**(5), 630–643 (2015). DOI 10.1109/TEVC.2014.2362729. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6920034
17. Helmuth, T., Spector, L., McPhee, N.F., Shanabrook, S.: Linear genomes for structured programs. In: Genetic Programming Theory and Practice XIV, Genetic and Evolutionary Computation. Springer, Ann Arbor, USA (2016)
18. Hemberg, E., Kelly, J., O'Reilly, U.M.: On domain knowledge and novelty to improve program synthesis performance with grammatical evolution. In: GECCO '19: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1039–1046. ACM, Prague, Czech Republic (2019). DOI doi:10.1145/3321707.3321865
19. Hernandez, J.G., Lalejini, A., Dolson, E., Ofria, C.: Random subsampling improves performance in lexicase selection. In: GECCO '19: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 2028–2031. ACM, Prague, Czech Republic (2019). DOI doi:10.1145/3319619.3326900
20. Jackson, D.: Promoting phenotypic diversity in genetic programming. In: R. Schaefer, C. Cotta, J. Kolodziej, G. Rudolph (eds.) PPSN 2010 11th International Conference on Parallel Problem Solving From Nature, *Lecture Notes in Computer Science*, vol. 6239, pp. 472–481. Springer, Krakow, Poland (2010)

21. Jundt, L., Helmuth, T.: Comparing and combining lexicase selection and novelty search. In: GECCO '19: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1047–1055. ACM, Prague, Czech Republic (2019). DOI doi:10.1145/3321707.3321787. URL https://dl.acm.org/citation.cfm?doid=3321707.3321787
22. Kelly, J., Hemberg, E., O'Reilly, U.M.: Improving genetic programming with novel exploration - exploitation control. In: L. Sekanina, T. Hu, N. Lourenço, H. Richter, P. García-Sánchez (eds.) EuroGP 2019: Proceedings of the 22nd European Conference on Genetic Programming, pp. 64–80. Springer International Publishing (2019)
23. La Cava, W., Helmuth, T., Spector, L., Moore, J.H.: A probabilistic and multi-objective analysis of lexicase selection and epsilon-lexicase selection. Evolutionary Computation (2018). Forthcoming
24. La Cava, W., Spector, L., Danai, K.: Epsilon-lexicase selection for regression. In: T. Friedrich (ed.) GECCO '16: Proceedings of the 2016 Annual Conference on Genetic and Evolutionary Computation, pp. 741–748. ACM, Denver, USA (2016). DOI 10.1145/2908812.2908898
25. Liskowski, P., Krawiec, K., Helmuth, T., Spector, L.: Comparison of semantic-aware selection methods in genetic programming. In: GECCO 2015 Semantic Methods in Genetic Programming (SMGP'15) Workshop, pp. 1301–1307. ACM, Madrid, Spain (2015). DOI 10.1145/2739482.2768505. URL http://doi.acm.org/10.1145/2739482.2768505
26. McPhee, N.F., Donatucci, D., Helmuth, T.: Using graph databases to explore genetic programming run dynamics. In: Genetic Programming Theory and Practice XIII, Genetic and Evolutionary Computation. Springer, Ann Arbor, USA (2015). URL http://www.springer.com/us/book/9783319342214
27. de Melo, V.V., Vargas, D.V., Banzhaf, W.: Batch tournament selection for genetic programming: The quality of lexicase, the speed of tournament. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '19, pp. 994–1002. ACM, New York, NY, USA (2019). DOI 10.1145/3321707.3321793. URL http://doi.acm.org/10.1145/3321707.3321793
28. Metevier, B., Saini, A.K., Spector, L.: Lexicase selection beyond genetic programming. In: Genetic Programming Theory and Practice XVI, pp. 123–136. Springer International Publishing, Cham (2019). DOI 10.1007/978-3-030-04735-1_7. URL https://doi.org/10.1007/978-3-030-04735-1_7
29. Moore, J.M., Stanton, A.: Lexicase selection outperforms previous strategies for incremental evolution of virtual creature controllers. Proceedings of the European Conference on Artificial Life pp. 290–297 (2017). DOI 10.1162/ecal_a_0050_14. URL https://www.mitpressjournals.org/doi/abs/10.1162/ecal_a_0050_14
30. Moore, J.M., Stanton, A.: Tiebreaks and diversity: Isolating effects in lexicase selection. The 2018 Conference on Artificial Life pp. 590–597 (2018). DOI 10.1162/isal_a_00109. URL https://www.mitpressjournals.org/doi/abs/10.1162/isal_a_00109
31. Moraglio, A., Krawiec, K., Johnson, C.G.: Geometric semantic genetic programming. In: C.A. Coello Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, M. Pavone (eds.) Parallel Problem Solving from Nature, PPSN XII (part 1), *Lecture Notes in Computer Science*, vol. 7491, pp. 21–31. Springer, Taormina, Italy (2012). DOI doi:10.1007/978-3-642-32937-1_3
32. Pantridge, E., Helmuth, T., McPhee, N.F., Spector, L.: Specialization and elitism in lexicase and tournament selection. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO '18, pp. 1914–1917. ACM, New York, NY, USA (2018). DOI 10.1145/3205651.3208220. URL http://doi.acm.org/10.1145/3205651.3208220
33. Rosin, C.D.: Stepping stones to inductive synthesis of low-level looping programs. In: Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, *AAAI '19*, vol. 33. AAAI Press, Palo Alto, California USA (2019)
34. Spector, L.: Assessment of problem modality by differential performance of lexicase selection in genetic programming: A preliminary report. In: K. McClymont, E. Keedwell (eds.) 1st workshop on Understanding Problems (GECCO-UP), pp. 401–408. ACM, Philadelphia, Pennsylvania, USA (2012). DOI 10.1145/2330784.2330846. URL http://hampshire.edu/lspector/pubs/wk09p4-spector.pdf

35. Spector, L., Klein, J., Keijzer, M.: The push3 execution stack and the evolution of control. In: GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation, vol. 2, pp. 1689–1696. ACM Press, Washington DC, USA (2005). DOI 10.1145/1068009.1068292. URL http://www.cs.bham.ac.uk/~wbl/biblio/gecco2005/docs/p1689.pdf
36. Spector, L., La Cava, W., Shanabrook, S., Helmuth, T., Pantridge, E.: Relaxations of lexicase parent selection. In: W. Banzhaf, R.S. Olson, W. Tozier, R. Riolo (eds.) Genetic Programming Theory and Practice XV, pp. 105–120. Springer International Publishing, Cham (2018)
37. Spector, L., Robinson, A.: Genetic programming and autoconstructive evolution with the push programming language. Genetic Programming and Evolvable Machines **3**(1), 7–40 (2002). DOI 10.1023/A:1014538503543. URL http://hampshire.edu/lspector/pubs/push-gpem-final.pdf