ϵ-Lexicase Selection for Regression

William La Cava*
Department of Mechanical and Industrial Engineering
University of Massachusetts
Amherst, MA 01003
wlacava@umass.edu

Lee Spector
School of Cognitive Science
Hampshire College
Amherst, MA 01002
lspector@hampshire.edu

Kourosh Danai
Department of Mechanical and Industrial Engineering
University of Massachusetts
Amherst, MA 01003
danai@engin.umass.edu

ABSTRACT

Lexicase selection is a parent selection method that considers test cases separately, rather than in aggregate, when performing parent selection. It performs well in discrete error spaces but not on the continuous-valued problems that compose most system identification tasks. In this paper, we develop a new form of lexicase selection for symbolic regression, named ϵ-lexicase selection, that redefines the pass condition for individuals on each test case in a more effective way. We run a series of experiments on real-world and synthetic problems with several treatments of ϵ and quantify how ϵ affects parent selection and model performance. ϵ-lexicase selection is shown to be effective for regression, producing better fit models compared to other techniques such as tournament selection and age-fitness Pareto optimization. We demonstrate that ϵ can be adapted automatically for individual test cases based on the population performance distribution. Our experiments show that ϵ-lexicase selection with automatic ϵ produces the most accurate models across tested problems with negligible computational overhead. We show that behavioral diversity is exceptionally high in lexicase selection treatments, and that ϵ-lexicase selection makes use of more fitness cases when selecting parents than lexicase selection, which helps explain the performance improvement.

Keywords

genetic programming, system identification, regression, parent selection

1. INTRODUCTION

Genetic programming (GP) traditionally tests programs on many test cases and then reduces the performance into a single value that is used to select parents for the next generation. Typically the fitness f of an individual is quantified as its aggregate performance over the training set $\mathcal{T} = \{(y_t, \mathbf{x}_t)\}_{t=1}^{N}$, using e.g. the mean absolute error (MAE), which is quantified for individual program $i \in P$ as:

$$f(i, \mathcal{T}) = \frac{1}{N} \sum_{t \in \mathcal{T}} |y_t - \hat{y}_t(i, \mathbf{x}_t)| \quad (1)$$

where $\mathbf{x} \in \mathbb{R}^D$ represents the variables or features, the target output is y and $\hat{y}(i, \mathbf{x})$ is the program's output. As a result of the aggregation of the absolute error vector $e(i) = |y - \hat{y}(i, \mathbf{x})|$ in Eq. (1), the relationship of $\hat{y}$ to y is represented crudely when choosing models to propagate. As others have pointed out [12], aggregate fitnesses strongly reduce the information conveyed to GP about i relative to the description of $i's$ behavior available in $e(i)$, thereby underutilizing information that could help guide the search. In addition, many forms of aggregation assume all tests are equally informative (although there are exceptions, including implicit fitness sharing which is discussed below). Therefore individuals that are elite (i.e. have the lowest error in the population) for portions of e are not selected if they perform poorly in other regions and therefore have a higher f. By providing equivalent selection pressure with respect to test cases, GP misses the opportunity to identify programs that perform especially well in certain regions of the problem, most importantly those portions of the problem that are more difficult for the process to solve. We expect GP to solve problems through the induction, propagation and recombination of building blocks (i.e. subprograms) that provide partial solutions to our desired task. Hence we wish to select those programs that imply a partial solution by performing uniquely well on subsets of the problem.

Several methods have been proposed to reward individuals with uniquely good test performance, such as implicit fitness sharing (IFS) [19], historically assessed hardness [10], and co-solvability [11], all of which assign greater weight to fitness cases that are judged to be more difficult in view of the population performance. Perhaps the most effective parent selection method recently proposed is lexicase selection [7, 27]. In particular, "global pool, uniform random sequence, elitist lexicase selection" [27], which we refer to simply as lexicase selection, has outperformed other similarly-motivated methods in recent studies [6, 17]. Despite these gains, it fails to produce such benefits when applied to continuous symbolic regression problems, due to its method of selecting individuals based on test case elitism. We demonstrate in this paper that by re-defining the test case pass condition in lexicase selection using an ϵ threshold, the benefits of lexicase selection can be achieved in continuous domains.

We begin by describing the ϵ-lexicase selection algorithm

*corresponding author

GECCO '16, July 20 - 24, 2016, Denver, CO, USA

ISBN 978-1-4503-4206-3/16/07...$15.00

DOI: http://dx.doi.org/10.1145/2908812.2908898

in §2 and discuss how it differs with respect to standard lexicase selection. Several definitions of ϵ are proposed. We briefly review related work in §3 and describe the relation between lexicase selection and multiobjective methods. The experimental analysis is presented in §4, beginning with a parameter variation study of ϵ and ending with a comparison of several GP methods on a set of real-world and symbolic regression problems. Given the results, we propose future research directions in §5 and summarize our findings in §6.

2. ϵ LEXICASE SELECTION

Lexicase selection is a parent selection technique based on lexicographic ordering of test (i.e. fitness) cases. Each parent selection event proceeds as follows:

1. The entire population is added to the selection pool.
2. The fitness cases are shuffled.
3. Individuals in the pool with a fitness worse than the best fitness on this case among the pool are removed.
4. If more than one individual remains in the pool, the first case is removed and 3 is repeated with the next case. If only one individual remains, it is the chosen parent. If no more fitness cases are left, a parent is chosen randomly from the remaining individuals.

As evidenced above, the algorithm is quite simple to implement. In this procedure, test cases act as filters, and a randomized path through these filters is constructed each time a parent is selected. Each parent selection event returns a parent that is elite on at least the first test case used to select it. In turn, the filtering capacity of a test case is directly proportional to its difficulty since it culls the individuals from the pool that do not do the best on it. Therefore selective pressure continually shifts to individuals that are elite on cases that are not widely solved in the population. Because each parent is selected via a randomized ordering of test cases and these cases perform filtering proportional to their difficulty, individuals are pressured to perform well on unique combinations of test cases, which promotes individuals with diverse performance, leading to increased diversity observed during evolutionary runs [7].

Lexicase selection was originally applied to multimodal [27] and "uncompromising" [7] problems. An uncompromising problem is one in which only exact solutions to every test case produce a satisfactory program. For those types of problems, using each case as a way to select only elite individuals is well-motivated, since each test case must be solved exactly. In regression, exact solutions to test cases can only be expected for synthetic problems, whereas real-world problems are subject to noise and measurement error. With respect to the lexicase selection process, continuously-valued errors are problematic, due to the fact that individuals in the population are not likely to share elitism on any particular case unless they are identical equations. On regression problems, the standard lexicase procedure typically uses only one case for each parent selection, resulting in poor performance.

We hypothesize that lexicase selection performs poorly on continuous errors because the case passing criteria is too stringent in continuous error spaces. For individual i to pass case t, lexicase requires that $e_t(i) = e_t^*$, where e_t^* is the best error on that test case in the pool. To remedy this shortcoming, we introduce ϵ-lexicase selection, which modulates the pass condition on test cases via a parameter ϵ, such that only individuals outside of a predefined ϵ are filtered in step 3 of lexicase selection. We experiment with four different definitions of ϵ in this paper. The first two, ϵ_e and ϵ_y, are absolute thresholds that define the pass condition $p_t(i)$ of program i on test case t as follows:

$$\epsilon_e : p_t(i) = \mathbb{I}\left(e_t(i) < e_t^*(1 + \epsilon_e)\right) \quad (2)$$

$$\epsilon_y : p_t(i) = \mathbb{I}\left(e_t(i) < \epsilon_y\right) \quad (3)$$

Here $\mathbb{I}$ is the indicator function that returns 1 if true and 0 if false. As shown in Eq. (2), ϵ_e defines $p_t(i)$ relative to e_t^*, and therefore is always passed by at least one individual in P. Conversely, ϵ_y (Eq. (3)) defines $p_t(i)$ relative to the target value y_t, meaning that $\hat{y}_t(i)$ must be within $\pm\epsilon_y$ of y_t to pass case t. In this way ϵ_y provides no selection pressure if there is not an individual in the population within adequate range of the true value for that case.

ϵ_e and ϵ_y are the simplest definitions of ϵ-lexicase selection, but have two distinct disadvantages: 1) they have to be specified by the user, and 2) their optimal values are problem dependent. An absolute ϵ is unable to provide a desired amount of filtering in each selection event since it is blind to the population's performance. Ideally ϵ should automatically adapt to take into account the values of $e_t(i)$ across P, denoted $\mathbf{e}_t \in \mathbb{R}^{|P|}$, so that it can modulate its selectivity based on the difficulty of t. A common estimate of difficulty in performance on a fitness case is variance [22]; in this regard ϵ could be defined according to the standard deviation of $\mathbf{e}_t$, i.e. $\sigma(\mathbf{e}_t)$. Given the high sensitivity of σ to outliers, however, we opt for a more robust estimation of variability by using the median absolute deviation (MAD) [21] of $\mathbf{e}_t$, defined as

$$MAD(\mathbf{e}_t) = \lambda(\mathbf{e}_t) = \text{median}_j\left(|\mathbf{e}_{t_j} - \text{median}_k(\mathbf{e}_{t_k})|\right) \quad (4)$$

We use Eq. (4) in the definition of two ϵ values, $\epsilon_{e\lambda}$ and $\epsilon_{y\lambda}$, that are defined analogously to ϵ_e and ϵ_y as:

$$\epsilon_{e\lambda} : p_t(i) = \mathbb{I}\left(e_t(i) < e_t^* + \lambda(\mathbf{e}_t)\right) \quad (5)$$

$$\epsilon_{y\lambda} : p_t(i) = \mathbb{I}\left(e_t(i) < \lambda(\mathbf{e}_t)\right) \quad (6)$$

An important consideration in parent selection is the time complexity of the selection procedure. Lexicase selection has a theoretical worst-case time complexity of $O(|P|^2 N)$, compared to a time complexity of $O(|P|N)$ for tournament selection. Although clearly undesirable, this worst-case complexity is only reached if every individual passes every test case during selection; in practice [7], lexicase selection normally uses a small number of cases for each selection and therefore incurs only a small amount of overhead. We quantify the wall clock times for our variants of lexicase compared to other methods in §4.4.

3. RELATED WORK

Although to an extent the ideas of multiobjective optimization apply to multiple test cases, they are qualitatively different: objectives are the defined goals of a task, whereas test cases are tools for estimating progress towards those objectives. Objectives and test cases therefore commonly exist at different scales: symbolic regression often involves one or two objectives (e.g. accuracy and model conciseness) and hundreds or thousands of test cases. One example of using

test cases explicitly as objectives occurs in Langdon's work on data structures [16] in which small numbers of test cases (in this case 6) are used as multiple objectives in a Pareto selection scheme. Other multi-objective approaches such as NSGA-II [2], SPEA2 [33] and ParetoGP [26] are used commonly with a small set of objectives in symbolic regression. The "curse of dimensionality" prevents the use of objectives at the scale of typical test case sizes, since most individuals become nondominated[1], leading to selection based mostly on expensive diversity measures rather than performance. Scaling issues in many-objective optimization are reviewed in [9]. In lexicase selection, parents are guaranteed to be nondominated with respect to the fitness cases. Pareto strength in SPEA2 promotes individuals based on how many individuals they dominate, and similarly lexicase selection increases the probability of selection for individuals who solve *more* cases and *harder* cases (i.e. cases that are not solved by other individuals) and decreases for individuals who solve *fewer* or *easier* cases.

A number of GP methods attempt to affect selection by weighting test cases based on population performance. In non-binary Implicit Fitness Sharing (IFS) [13], the fitness proportion of a case is scaled by the performance of other individuals on that case. Similarly, historically assessed hardness scales error on each test case by the success rate of the population [10]. Discovery of objectives by clustering (DOC) [12] clusters test cases by population performance, and thereby reduces test cases into a set of objectives for search. Both IFS and DOC were outperformed by lexicase selection on program synthesis and boolean problems in previous studies [6, 17]. Other methods attempt to sample a subset of $\mathcal{T}$ to reduce computation time or improve performance, such as dynamic subset selection [3], interleaved sampling [4], and co-evolved fitness predictors [22]. Unlike these methods, lexicase selection begins each selection with the full set of training cases, and allows selection to adapt to program performance on them.

The conversion of a model's real-valued fitness into discrete values based on an ϵ threshold has been explored in other research; for example, Novelty Search GP [18] uses a reduced error vector to define behavioral representation of individuals in the population. This paper proposes it for the first time as a solution to applying lexicase selection effectively to regression.

As a behavioral-based search driver, lexicase selection belongs to a class of GP systems that attempt to incorporate a program's behavior explicitly into the search process, and as such shares a general motivation with recently proposed methods such as Semantic GP [20] and Behavioral GP [14], despite differing strongly in approach. Although lexicase is designed with behavioral diversity in mind, recent studies suggest that structural diversity can also significantly affect GP performance [1].

4. EXPERIMENTAL ANALYSIS

We define the problems used to assess ϵ-lexicase selection here, as well as a set of existing GP methods used for comparison. We then analyze and tune the value of ϵ_e and ϵ_y on an example problem and discuss the results. Finally we test all of the methods on each problem and summarize the findings.

4.1 Problems

Three synthetic and three real-world problems were chosen for benchmarking different GP methods. The first problem is the housing data set [5] that seeks a model to estimate Boston housing prices. The second problem is the Tower problem[2] that consists of 15-minute averaged time series data taken from a chemical distillation tower, with the goal of predicting propelyne concentration. The third problem, referred to as the Wind problem [15], features data collected from the Controls and Advanced Research Turbine, a 600 kW wind turbine operated by the National Wind Technology Center. The data set consists of time-series measurements of wind speed, control actions, and acceleration measurements that are used to predict the bending moment measured at the base of the wind turbine. In this case solutions are formulated as first-order discrete-time dynamic models of the form $\hat{y} = f(\mathbf{x}, \mathbf{x}_{t-1}, \hat{y}_{t-1})$. The fourth and fifth problem tasks are to estimate the energy efficiency of heating (ENH) and cooling (ENC) requirements for various simulated buildings [30]. The last problem is the UBall5D problem[3] which has the form

$$y = \frac{10}{5 + \sum_{i=1}^{5} (x_i - 3)^2}$$

The Tower problem and UBall5D were chosen from the benchmark suite suggested by White et. al. [32]. The dimensions of all data sets are shown in Table 1. Aside from UBall5D which has a pre-defined test set [31], the problems were divided 70/30 into training and testing sets. These sets were normalized to zero mean, unit variance and randomly partitioned for each trial.

4.2 Compared Methods

Our definitions of ϵ in §2 yield four methods which we analyze in our experiments, abbreviated as Lex ϵ_e, Lex ϵ_y, Lex $\epsilon_{e\lambda}$, Lex $\epsilon_{y\lambda}$. We compare these variants to standard lexicase selection (denoted as simply Lex) and standard tournament selection of size 2 (denoted Tourn). To control for the effect of selection in GP, we also compare these methods to random parent selection, denoted Rand Sel.

In addition to these methods, many state-of-the-art symbolic regression tools leverage Pareto optimization [26, 23, 1] and/or age layering [8] to improve symbolic regression performance. With this in mind, we also compare ϵ-lexicase selection to age-fitness Pareto survival (AFP) [24], in which each individual is assigned an age equal to the number of generations since its oldest ancestor was created. Each generation, a new individual is introduced to the population as a means of random restart. Selection for breeding is random, and during breeding a number of children are created equal to the overall population size. Survival is conducted according to the environmental selection algorithm in SPEA2 [33], as in [25].

Every method uses sub-tree crossover and point mutation as search operators. For each method, we include a parameter hill climbing step each generation that perturbs the constants in each equation with Gaussian noise and saves those

[1]Program i_1 dominates i_2 if $f_j(i_1) \leq f_j(i_2)\ \forall j$ and $f_j(i_1) < f_j(i_2)$ for at least one j (f is minimized).

[2]http://symbolicregression.com/?q=towerProblem

[3]UBall5D is also known as Vladislavleva-4.

Table 1: Symbolic regression problem settings.

Setting			Value
Population size			1000
Crossover / mutation			80/20%
Program length limits			[3, 50]
ERC range			[-1,1]
Generation limit			1000
Trials			30
Terminal Set			{**x**, ERC, +, −, ∗, /, sin, cos, exp, log}
Elitism			keep best
Problem	**Dimension**	**Training Cases**	**Test Cases**
Housing	14	354	152
Tower	25	2195	940
Wind	6	4200	1800
ENH	8	538	230
ENC	8	538	230
UBall5D	5	1024	5000

changes that improve the model's MAE (Eq. (1)). The complete code for these tests is available online[4].

4.3 Parameter tuning

In the cases of ϵ_e and ϵ_y, the user must specify fixed parameter values. For both cases we tested the set of parameter values {0.01, 0.05, 0.10, 0.50, 1.0, 5.0, 10.0} over 15 trials. For ϵ_e, these values mean that an individual's $e_t(i)$ must be within 1% to 1000% of e_t^* to pass test t. For ϵ_y, $\hat{y}_t(i)$ must be within that range of y_t. The parameter study was conducted on the Tower symbolic regression problem, the details of which are shown in Table 1. It is important to note that the optimal value of these parameters is problem-dependent, although the best values from the parameter tuning experiment were used for all problems in the subsequent sections.

The test fitness results for different values of ϵ_e are shown in Figure 1. The best results are obtained for $\epsilon_e = 5.0$. The number of cases used during selection are shown in Figure 2. This figure matches our intuition about the sensitivity of case usage to ϵ_e: larger tolerances for error use more cases in each selection event. For $\epsilon_e = 1.0$, we observe steady growth in case usage, suggesting population convergence. The diversity of the population's behavior also grows with ϵ_e, as shown by the unique output vectors, i.e. unique $\hat{y}$, plot in Figure 3. For subsequent experiments we use $\epsilon_e = 5.0$ which corresponds to the lowest median test fitness for the Tower problem.

The test fitness results for different values of ϵ_y are shown in Figure 4. From $\epsilon_y = 1.0$ and upward, we note that Lex ϵ_y uses all fitness cases for nearly every selection event, causing long run-times and suggesting that selection has become random. As we show in §4.4, Rand Sel performs similarly to $\epsilon_y > 0.50$ on this problem, further supporting the idea of selection pressure loss. We set $\epsilon_y = 0.10$ for the subsequent experiments, again corresponding to the lowest median test fitness.

4.4 Results

We summarize the experimental results in Table 2. The median best fitness of all runs on the test sets, the mean ranking of each method across problems, and the total runtime to conduct the trials for each method is shown. Figure 6 shows the distributions of MAE (Eq. (1)) on the test sets for

[4]https://www.github.com/lacava/ellen

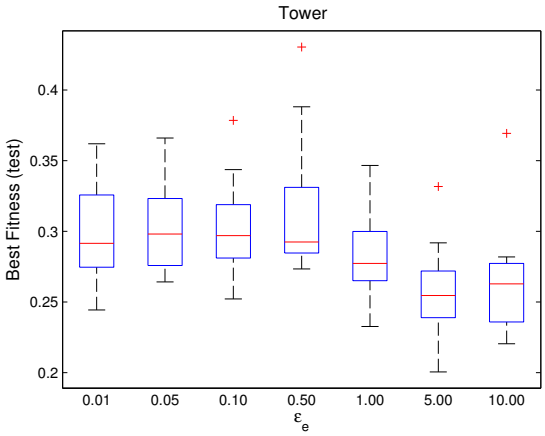

Figure 1: Best-of-run fitness on the test set for various levels of ϵ_e.

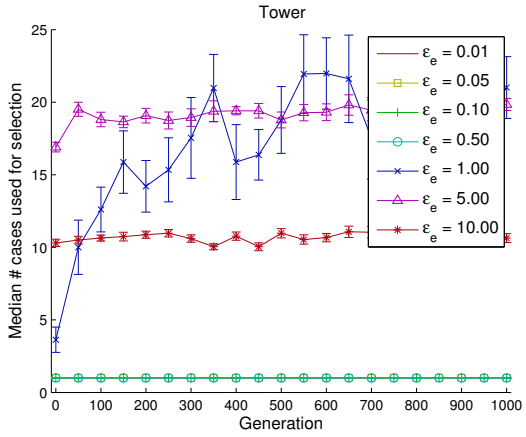

Figure 2: Number of test cases used for selection for various levels of ϵ_e. All cases of $\epsilon_e \leq 0.50$ have a median of one case each generation.

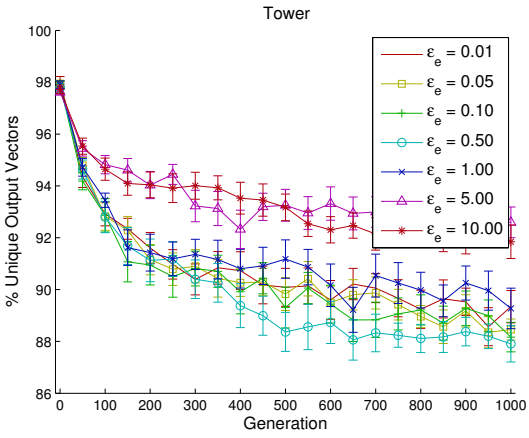

Figure 3: Unique output vectors for various levels of ϵ_e.

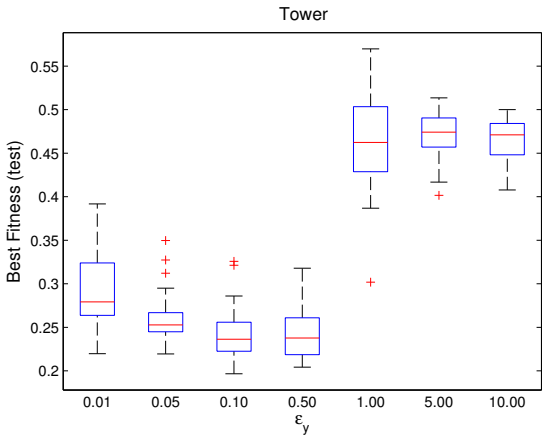

Figure 4: Best-of-run fitness the test sets for various levels of ϵ_y.

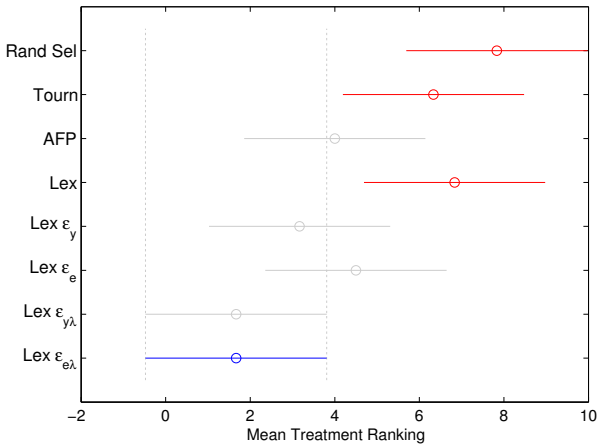

Figure 5: Multiple comparison of the Friedman test of mean rankings across all problems, with approximate intervals of significance. Two treatments being compared are significantly different if their intervals do not overlap. Signficant differences from Lex $\epsilon_{e\lambda}$ (shown in blue) are denoted in red.

the best-of-run models. We also visualize the best fitness on the training set each generation in Figure 7.

Figure 7 shows that most of the differences in learning on the training set occurs in the first 250 generations, although in all cases Lex $\epsilon_{e\lambda}$ or $\epsilon_{y\lambda}$ maintains the lowest final training set error. Across all problems, the median best fitness on the test sets is obtained by either Lex $\epsilon_{e\lambda}$ or Lex $\epsilon_{y\lambda}$. According to the pairwise tests annotated in Table 2 , Lex $\epsilon_{e\lambda}$ or Lex $\epsilon_{y\lambda}$ perform significantly better than Rand Sel, Tourn, Lex and Lex ϵ_e on 6/6 problems, better than AFP on 5/6 problems, and better than Lex ϵ_y on one problem. In terms of the mean ranking across tests (Table 2), Lex $\epsilon_{e\lambda}$ and Lex $\epsilon_{y\lambda}$ rank the best, followed by Lex ϵ_y, AFP, Lex ϵ_e, Tourn, Lex, and Rand Sel, in that order. We conduct a Friedman's test of the mean rankings across problems, the intervals of which are shown in Figure 5. This comparison indicates that the performance improvement of Lex $\epsilon_{e\lambda}$ and Lex $\epsilon_{y\lambda}$ relative to Tourn, Lex, and Rand Sel is significant across all tested problems. The intervals show partial overlap with respect to AFP and Lex ϵ_e that may warrant further experiments.

The median total trial times reported in Table 2 indicate that ϵ-lexicase selection takes nearly the same time to finish as tournament selection in practice, despite its higher theoretical worst-case time complexity. On average, Lex ϵ_y, ϵ_e, $\epsilon_{y\lambda}$ and $\epsilon_{e\lambda}$ report wall clock times that are 96%, 82%, 120%, and 120% the duration of tournament selection, respectively, giving a negligible average of 105%. Rand Sel finishes the fastest due to no selection, and AFP finishes the slowest, most likely due to the overhead of computing dominance relations for P and, in the case of non-dominated populations, densities in objective space [33]. It is possible that the tournament selection-based version of AFP [24] would have a lower run-time, although it may have difficulty scaling to more than two objectives [25].

Lex takes only 41% of the time of tournament selection to finish, which is explained by its case usage. The number of fitness cases used by lexicase selection variants for four of the problems is shown in Figure 8. Note that Lex uses only one test case during parent selection due to the rarity of elitism in continuous error space; the fact that parents are chosen based on single cases also explains its poor performance. On the Tower and Wind problems, ϵ-lexicase variants show small increases in case usage over the course of evolution. Lex ϵ_e uses the highest number of cases on Tower and Wind, but the lowest number of cases for ENH and ENC among ϵ-lexicase variants. On ENH and ENC, a higher percent of total cases are used during selection compared to Tower and Wind, indicating the problem-dependent nature of GP population structures and performance. Lex $\epsilon_{y\lambda}$ and Lex $\epsilon_{e\lambda}$ use nearly the same numbers of cases for selection on each problem, which suggests that the performance of λ-based ϵ is robust to being defined relative to e or y (Eq. (5) and (6)). Lex ϵ_e and ϵ_y, on the other hand, vary strongly across problems in terms of their case usage, again indicating their parametric sensitivity.

We observe exceptionally high population diversity for the lexicase methods, which supports observations in [7]. We measure diversity by the percent of unique $\hat{y}$ among programs in P, plotted as an average across problems in Figure 9. Interestingly, the diversity is higher using lexicase than random selection, which indicates lexicase selection's ability to exploit behavioral difference to increase diversity beyond the search operator effects. The differential performance between Rand Sel and ϵ-lexicase selection shown in Table 2 demonstrates that the gains afforded by ϵ-lexicase selection are not due to simply increased randomization, but rather the promotion of individuals with exceptionally good performance on diverse orderings of test cases.

5. DISCUSSION

ϵ-lexicase selection is a global pool, uniform random sequence, non-elitist version of lexicase selection [27] that performs well on symbolic regression problems according to the experimental analysis presented in the last section. "Global pool" refers to the fact that each selection event begins with the whole population (step 1 in §2). Smaller pool sizes have yet to be tried, but could potentially improve performance on certain problems that historically respond well to relaxed selection pressure. Pools could also be defined geographically [28]. "Uniform random sequence" refers to the shuf-

Table 2: Comparison of median best-of-run MAE on the test sets and total trial time. The best fitness results are highlighted. Significant improvements with respect to each method are denoted by $a-h$ according to the method labels. Significance is defined as $p < 0.05$ according to a pairwise Wilcoxon rank-sum test with Holm correction. The median total time to run 30 trials of each algorithm is shown on the right.

Method	Housing	Tower	Wind	ENH	ENC	UBall5D	Mean Rank	Median Total Trial Time (hr:min:s)
[a]Rand Sel	0.469	0.458	0.463	0.288	0.272	[d] 0.128	7.83	00:07:25
[b]Tourn	[a] 0.408	[a] 0.402	[ad] 0.397	[a] 0.207	[a] 0.236	[ad] 0.113	6.33	00:24:37
[c]AFP	[abd] 0.354	[abd] 0.319	[abd] 0.381	[abd] f 0.138	[abd] f 0.171	[abd] 0.094	4.00	01:09:18
[d]Lex	[a] 0.402	[ab] 0.355	[a] 0.419	[a] 0.210	[a] 0.237	0.142	6.83	00:10:11
[e]Lex ϵ_y	[abd] f 0.325	[abcd] 0.260	[ad] 0.386	[abcd] f 0.113	[abcd] f 0.150	[abcd] f 0.079	3.17	00:23:44
[f]Lex ϵ_e	[a] 0.386	[abcd] 0.263	[ad] 0.386	[abd] 0.165	[abd] 0.193	[abcd] 0.082	4.50	00:20:24
[g]Lex $\epsilon_{y\lambda}$	[abcd] f 0.321	[abcd] f 0.239	[abd] 0.378	[abcd] ef 0.101	[abcd] f 0.137	[abcd] 0.080	1.67	00:29:26
[h]Lex $\epsilon_{e\lambda}$	[abcd] f 0.309	[abcd] f 0.233	[abd] ef 0.381	[abcd] f 0.106	[abcd] f 0.141	[abcd] fg 0.078	1.67	00:29:37

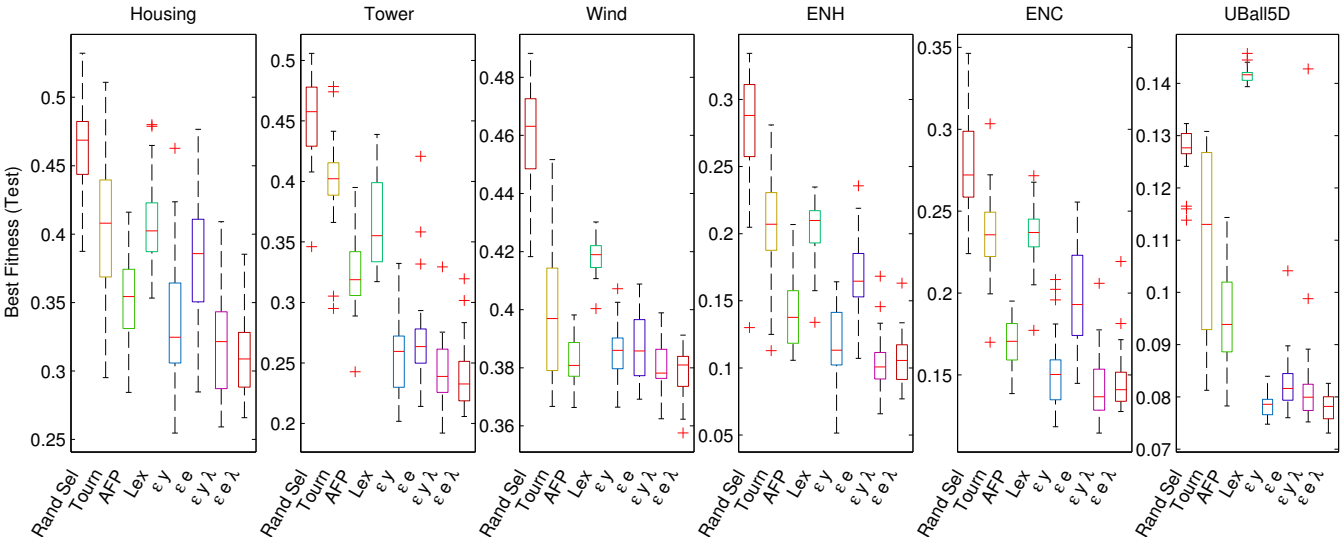

Figure 6: Best-of-run fitness statistics on the test sets for all problems.

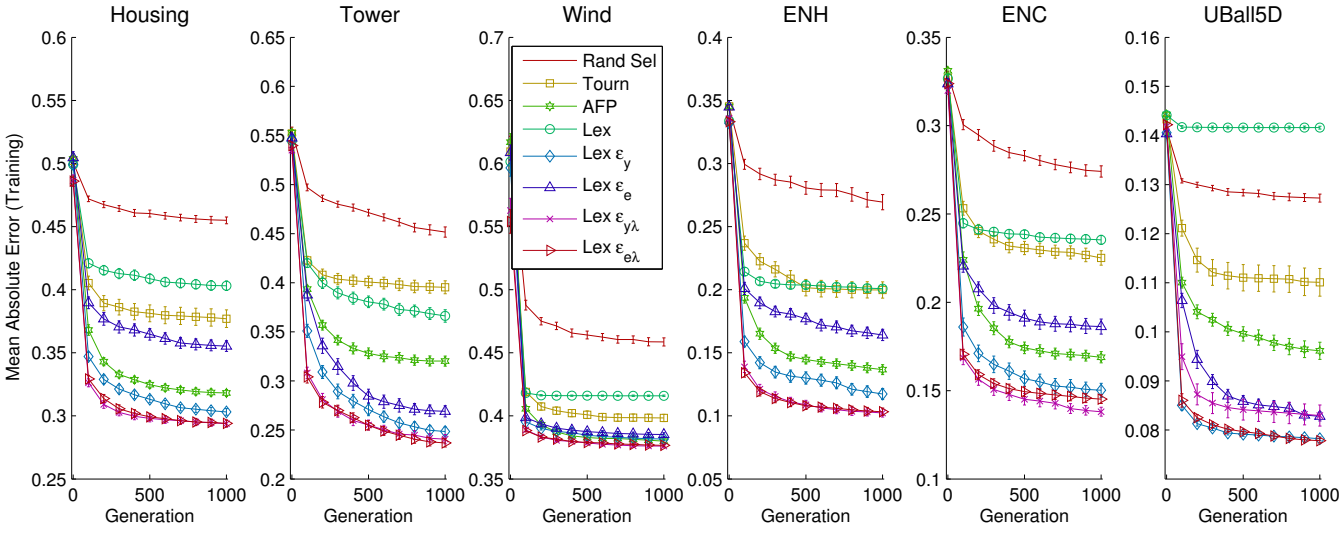

Figure 7: Best-of-run fitness each generation on the training sets for all problems. The bars indicate the standard error across all runs.

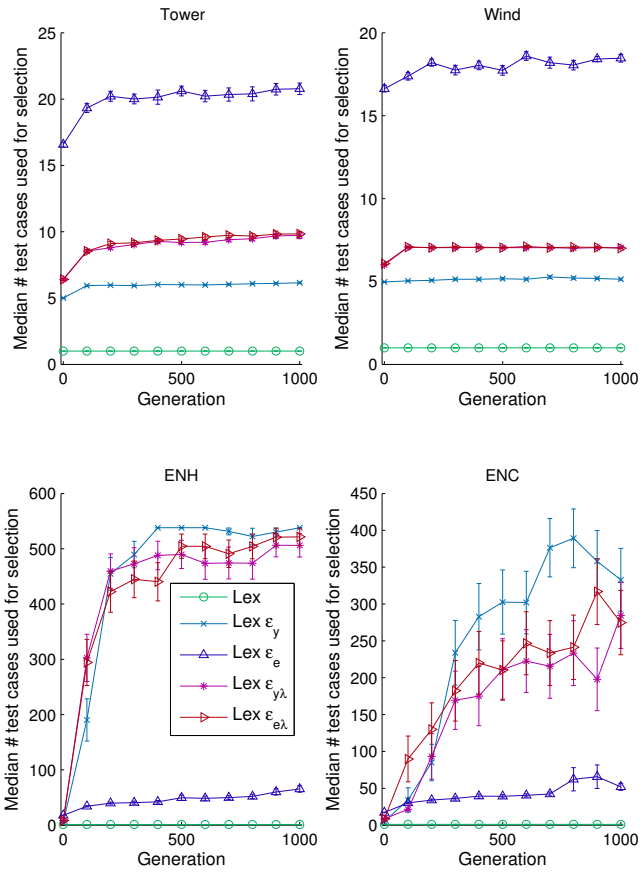

Figure 8: Number of fitness cases used in selection for different lexicase methods. The bars indicate the standard error.

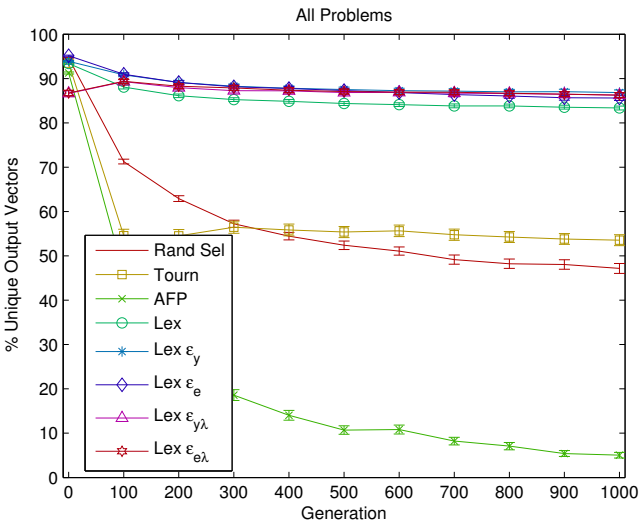

Figure 9: Mean population diversity as a function of generation for each method over all problems. The bars indicate the standard error.

fling procedure for cases in step 2 (§2), and, as is the case with pool size, other orderings of test cases have yet to be reported in literature. One could consider biasing the ordering of cases in some ways that could select parents with certain desired properties. In [17], Liskowski attempted to use derived objective clusters as cases in lexicase selection, but found that this actually decreased performance. Still, there may be a form of ordering or case reduction that improves lexicase selection's performance over random shuffling.

The ordering of the test cases that produce a given parent also contains potentially useful information that could be used by the search operators in GP. Helmuth [6] observed that lexicase selection creates large numbers of distinct behavioral clusters in the population (an observation supported by Figure 9). In that regard, it may be advantageous, for instance, to perform crossover on individuals selected by differing orders of cases such that their offspring are more likely to inherit subprograms with unique partial solutions to a given task. On the other hand, one could argue for pairing individuals based on similar selection cases, to promote niching and minimize the destructive nature of subtree crossover.

6. CONCLUSIONS

We find that ϵ-lexicase selection, especially with automatic threshold adaptation ($\epsilon_{e\lambda}$ and $\epsilon_{y\lambda}$), performs the best on the regression problems studied here in comparison to the other GP methods studied. The performance in terms of test fitness is promising, as well as the measured wall clock times, which are comparable to tournament selection. In addition to introducing a non-elitist version of lexicase that defines test case pass conditions using an ϵ threshold, we demonstrated that ϵ can be set automatically based on the dispersion of error across the population on a test case. We observed that the definition of this threshold is insensitive to the elite error offset e_t^*. The results should motivate the use of ϵ-lexicase selection as a parent selection technique for symbolic regression, and should motivate further research using non-elitist lexicase selection methods for continuous-valued problems in GP.

7. ACKNOWLEDGMENTS

The authors would like to thank Thomas Helmuth, Nic McPhee and Bill Tozier for their feedback as well as members of the Computational Intelligence Laboratory at Hampshire College. This work is partially supported by NSF Grant Nos. 1068864, 1129139 and 1331283. 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. This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by NSF grant number ACI-1053575 [29].

8. REFERENCES

[1] A. R. Burks and W. F. Punch. An Efficient Structural Diversity Technique for Genetic Programming. In *GECCO*, pages 991–998. ACM Press, 2015.

[2] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II. In *PPSN*

VI, volume 1917, pages 849–858. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[3] C. Gathercole and P. Ross. Dynamic training subset selection for supervised learning in Genetic Programming. In *PPSN III*, number 866 in Lecture Notes in Computer Science, pages 312–321. Springer Berlin Heidelberg, Oct. 1994.

[4] I. Gonçalves and S. Silva. Balancing learning and overfitting in genetic programming with interleaved sampling of training data. In *EuroGP 2013*, pages 73–84, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[5] D. Harrison and D. L. Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of environmental economics and management*, 5(1):81–102, 1978.

[6] T. Helmuth. *General Program Synthesis from Examples Using Genetic Programming with Parent Selection Based on Random Lexicographic Orderings of Test Cases*. PhD thesis, UMass Amherst, Jan. 2015.

[7] T. Helmuth, L. Spector, and J. Matheson. Solving Uncompromising Problems with Lexicase Selection. *IEEE Transactions on Evolutionary Computation*, PP(99):1–1, 2014.

[8] G. S. Hornby. ALPS: The Age-layered Population Structure for Reducing the Problem of Premature Convergence. In *GECCO*, pages 815–822, New York, NY, USA, 2006. ACM.

[9] H. Ishibuchi, N. Tsukamoto, and Y. Nojima. Evolutionary many-objective optimization: A short review. In *IEEE CEC 2008*, pages 2419–2426. Citeseer, 2008.

[10] J. Klein and L. Spector. Genetic programming with historically assessed hardness. *GPTP VI*, pages 61–75, 2008.

[11] K. Krawiec and P. Lichocki. Using Co-solvability to Model and Exploit Synergetic Effects in Evolution. In *PPSN XI*, pages 492–501. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[12] K. Krawiec and P. Liskowski. Automatic derivation of search objectives for test-based genetic programming. In *Genetic Programming*, pages 53–65. Springer, 2015.

[13] K. Krawiec and M. Nawrocki. *Implicit fitness sharing for evolutionary synthesis of license plate detectors*. Springer, 2013.

[14] K. Krawiec and U.-M. O'Reilly. Behavioral programming: a broader and more detailed take on semantic GP. In *GECCO*, pages 935–942. ACM Press, 2014.

[15] W. La Cava, K. Danai, L. Spector, P. Fleming, A. Wright, and M. Lackner. Automatic identification of wind turbine models using evolutionary multiobjective optimization. *Renewable Energy*, 87, Part 2:892–902, Mar. 2016.

[16] W. B. Langdon. Evolving Data Structures with Genetic Programming. In *ICGA*, pages 295–302, 1995.

[17] P. Liskowski, K. Krawiec, T. Helmuth, and L. Spector. Comparison of Semantic-aware Selection Methods in Genetic Programming. In *GECCO Companion*, pages 1301–1307, New York, NY, USA, 2015. ACM.

[18] Y. Martínez, E. Naredo, L. Trujillo, and E. Galván-López. Searching for novel regression functions. In *IEEE CEC 2013*, pages 16–23. IEEE, 2013.

[19] R. I. B. McKay. An Investigation of Fitness Sharing in Genetic Programming. *The Australian Journal of Intelligent Information Processing Systems*, 7(1/2):43–51, July 2001.

[20] A. Moraglio, K. Krawiec, and C. G. Johnson. Geometric semantic genetic programming. In *PPSN XII*, pages 21–31. Springer, 2012.

[21] T. Pham-Gia and T. L. Hung. The mean and median absolute deviations. *Mathematical and Computer Modelling*, 34(7-8):921–936, Oct. 2001.

[22] M. Schmidt and H. Lipson. Coevolution of Fitness Predictors. *IEEE Transactions on Evolutionary Computation*, 12(6):736–749, Dec. 2008.

[23] M. Schmidt and H. Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.

[24] M. Schmidt and H. Lipson. Age-fitness pareto optimization. In *GPTP VIII*, pages 129–146. Springer, 2011.

[25] M. D. Schmidt. *Machine Science: Automated Modeling of Deterministic and Stochastic Dynamical Systems*. PhD thesis, Cornell University, Ithaca, NY, USA, 2011. AAI3484909.

[26] G. F. Smits and M. Kotanchek. Pareto-front exploitation in symbolic regression. In *GPTP II*, pages 283–299. Springer, 2005.

[27] L. Spector. Assessment of problem modality by differential performance of lexicase selection in genetic programming: a preliminary report. In *GECCO*, pages 401–408, 2012.

[28] L. Spector and J. Klein. Trivial geography in genetic programming. In *GPTP III*, pages 109–123. Springer, 2006.

[29] J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, R. Roskies, J. R. Scott, and N. Wilkens-Diehr. XSEDE: Accelerating Scientific Discovery. *Computing in Science and Engineering*, 16(5):62–74, 2014.

[30] A. Tsanas and A. Xifara. Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy and Buildings*, 49:560–567, 2012.

[31] E. Vladislavleva, G. Smits, and D. den Hertog. Order of Nonlinearity as a Complexity Measure for Models Generated by Symbolic Regression via Pareto Genetic Programming. *IEEE Transactions on Evolutionary Computation*, 13(2):333–349, 2009.

[32] D. R. White, J. McDermott, M. Castelli, L. Manzoni, B. W. Goldman, G. Kronberger, W. Jaśkowski, U.-M. O'Reilly, and S. Luke. Better GP benchmarks: community survey results and proposals. *Genetic Programming and Evolvable Machines*, 14(1):3–29, Dec. 2012.

[33] E. Zitzler, M. Laumanns, and L. Thiele. *SPEA2: Improving the strength Pareto evolutionary algorithm*. ETH Zürich, Institut für Technische Informatik und Kommunikationsnetze (TIK), 2001.