ϵ-Lexicase selection: a probabilistic and multi-objective analysis of lexicase selection in continuous domains

William La Cava   lacava@upenn.edu
Institute for Bioinformatics, University of Pennsylvania, Philadelphia, PA, 19104, USA

Thomas Helmuth   helmutht@wlu.edu
Department of Computer Science, Washington and Lee University, Lexington, VA, USA

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

Jason H. Moore   jhmoore@upenn.edu
Institute for Bioinformatics, University of Pennsylvania, Philadelphia, PA, 19104, USA

Abstract

Lexicase selection is a parent selection method that considers training cases individually, rather than in aggregate, when performing parent selection. As opposed to previous work that has demonstrated the ability of lexicase selection to solve difficult problems, the goal of this paper is to develop the theoretical underpinnings that explain its performance. To this end, we derive an analytical formula that gives the expected probabilities of selection under lexicase selection, given a population and its behavior. In addition, we expand upon the relation of lexicase selection to many-objective optimization methods to show the behavior of lexicase, which is to select individuals on the boundaries of Pareto fronts in high-dimensional space. We show analytically why lexicase selection performs more poorly for certain sizes of population and training cases, and why it has been shown to perform more poorly in continuous error spaces. To addres this last concern, we introduce ϵ-lexicase selection, which modifies the pass condition defined in lexicase selection to allow near-elite individuals to pass cases, thereby improving selection performance. We show that ϵ-lexicase outperforms several diversity-maintenance strategies on a number of real-world and synthetic regression problems.

1 Introduction

Evolutionary computation (EC) traditionally assigns scalar fitness values to candidate solutions to determine how to guide search. In the case of genetic programming (GP), this fitness value attempts to capture how closely, on average, the behavior of the candidate programs match the desired behavior. Take for example the task of symbolic regression, in which we attempt to find a model using a set of training examples, i.e. cases or tests. A typical fitness measure is the mean squared error (MSE), which averages the squared differences between the model's outputs, $\hat{y}$, and the target output, y.

The effect of this averaging is to reduce a rich set of information comparing the model's output and the desired output to a single value. As noted by Krawiec (2016), the relationship of $\hat{y}$ to y can only be represented crudely by the fitness value. The fitness score thereby restricts the information conveyed to the search process about candidate programs relative to the description of their behavior available in the raw comparisons of the output to the target (i.e., the squared error), information which could help guide the search (Krawiec and Liskowski, 2015). This observation has led to increased interest in the development of methods that can leverage the program outputs directly to drive search more effectively (Vanneschi et al., 2014).

In addition to reducing information, averaging test performance assumes all tests are equally informative, leading to the potential loss of individuals who perform poorly *on average* even if they are the best on a training case that is difficult for most of the population to solve. This is particularly relevant for problems that require different modes of behavior to produce an adequate solution to the problem (Spector, 2012). The underlying assumption of GP in this regard is that selection pressure should be applied evenly with respect to training cases, ignoring the fact that cases that comprise the problem are unlikely to be uniformly difficult. As a result, the search is likely to benefit if it can take into account the difficulty of specific cases by recognizing individuals that perform well on harder parts of the problem. Underlying this last point is the assumption that GP solves problems by identifying, propagating and recombining partial solutions (i.e. building blocks) to the task at hand(cite schema Poli). As a result, a program that performs well on unique subsets of the problem may imply a partial solution to our task.

Several methods have been proposed to reward individuals with uniquely good test performance, such as implicit fitness sharing (IFS) (McKay, 2001), historically assessed hardness (Klein and Spector, 2008), and co-solvability (Krawiec and Lichocki, 2010), 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 designed to account for case hardness is lexicase selection (Spector, 2012). In particular, "global pool, uniform random sequence, elitist lexicase selection" (Spector, 2012), which we refer to simply as lexicase selection, has outperformed other similarly-motivated methods in recent studies (Helmuth et al., 2014; Helmuth and Spector, 2015; Liskowski et al., 2015). 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 training case elitism. For this reason we recently proposed (La Cava et al., 2016b) modulating the case pass conditions in lexicase selection using an automatically defined ϵ threshold, allowing the benefits of lexicase selection to be achieved in continuous domains.

To date, lexicase selection and ϵ-lexicase selection have mostly been analyzed via empirical studies, rather than algorithmic analysis. In particular, previous work has not explicitly described the probabilities of selection under lexicase selection compared to other selection methods, nor how lexicase selection relates to the multi-objective literature. Therefore, the foremost purpose of this paper is to describe analytically how lexicase selection and ϵ-lexicase selection operate on a given population compared to other approaches. With this in mind, in §6 we derive an equation that describes the expected probability of selection for individuals in a given population based on their behavior on the training cases, for all variants of lexicase selection. Then in §7, we analyze lexicase and ϵ lexicase selection from a multi-objective viewpoint, in which we

imagine each training case to be an objective. We prove that individuals selected by lexicase selection exist at the boundaries of the Pareto front in the high-dimensional space defined by the program error vectors. We show via an illustrative example population in §8 how the probabilities of selection differ under tournament, lexicase, and ϵ-lexicase selection.

The second purpose of this paper is to empirically assess the use of ϵ-lexicase selection in continuous domains. In §4, we define two new automated pass conditions in ϵ-lexicase which are shown to improve the method compared to the original implementation. A set of experiments compares variants of ϵ-lexicase selection to several existing selection techniques on a set of real world benchmark problems. The results show that ability of ϵ-lexicase selection to improve the predictive accuracy of models on these problems. We examine in detail the diversity of programs during these runs, as well as the number of cases used in selection events to validate our hypothesis that ϵ-lexicase selection allows for more cases to be used in selecting individuals compared to lexicase selection.

2   Preliminaries

In symbolic regression, we attempt to find a model $\hat{y}(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$ that maps variables to a target output using a set of training examples $\mathcal{T} = \{t_i = (y_i, \mathbf{x}_i)\}_{i=1}^T$, where $\mathbf{x}$ is a d-dimensional vector of variables, i.e. features, and y is the desired output. We refer to elements of $\mathcal{T}$ as "cases". GP poses the problem as

$$\text{minimize } f(n, \mathcal{T}) \quad \text{subject to } n \in \mathfrak{S} \tag{1}$$

where $\mathfrak{S}$ is the space of possible programs n and f denotes a minimized fitness function. GP attempts to solve the symbolic regression task by optimizing a population of programs $\mathcal{N} = \{n_i\}_{i=1}^N$, each of which encodes a model of the process and produces an estimate $\hat{y}_t(n, \mathbf{x}_t) : \mathbb{R}^d \to \mathbb{R}$ when evaluated on case t. We refer to $\hat{y}(n)$ as the *semantics* of program n, omitting $\mathbf{x}$ for brevity. We denote the squared differences between $\hat{y}$ and y, (i.e., the errors) as $e_t(n) = (y_t - \hat{y}_t(n))^2$. We use $\mathbf{e}_t \in \mathbb{R}^N$ to refer to the errors of all programs in the population on training case t.

A typical fitness measure (f) is the mean squared error $\text{MSE}(n, \mathcal{T}) = \frac{1}{N} \sum_{t \in \mathcal{T}} e_t(n)$ which we use to compare our results in §10. For the purposes of our discussion, it is irrelevant whether the MSE or the mean absolute error, i.e. $\text{MAE}(n, \mathcal{T}) = \frac{1}{N} \sum_{t \in \mathcal{T}} |y_t - \hat{y}_t(n)|$, is used, and so we use MAE to simplify a few examples throughout the paper. With lexicase selection and its variants, $e(n)$ is used directly during selection rather than averaging over cases. Nevertheless, in keeping with Eqn. 1 in our experiments, the final program returned is that which minimizes the MSE.

3   Lexicase Selection

Lexicase selection is a parent selection technique based on lexicographic ordering of test (i.e. fitness) cases. The lexicase selection algorithm for a single selection event is presented below:

Algorithm 2.1: Lexicase Selection

```
GetParent(𝒩, 𝒯):
    𝒯' ← 𝒯                                  training cases
    𝒮 ← 𝒩                                  initial selection pool is the population
    while |𝒯'| > 0 and |𝒮| > 1:            main loop
        case ← random choice from 𝒯'        consider a random case
        elite ← best fitness in 𝒮 on case   determine elite fitness
        𝒮 ← n ∈ 𝒮 if fitness(n) = elite     reduce selection pool to elites
        𝒯' ← 𝒯' − case                      reduce remaining cases
    return random choice from 𝒮             return parent
```

Algorithm 2.1 is very simple to implement because it consists of just a few steps: 1) choosing a case, 2) filtering the selection pool based on that case, and 3) repeating until the cases are exhausted or the selection pool is reduced to one individual. If the selection pool is not reduced by the time each case has been considered, an individual is chosen randomly from the remaining pool, $\mathcal{S}$.

Under lexicase selection, cases in $\mathcal{T}$ can be thought of as filters that reduce the selection pool to the individuals in the pool that are best on that case. It's important to note that each parent selection event constructs a new path through these filters. The filtering strength of the case is affected by two main factors: 1) its difficulty as defined by the number of individuals that pass it, and 2) its order in the selection event, which varies from selection to selection. Regarding 1), consider two extreme examples: if a case is passed by the whole population, then it will perform no filtering, resulting in no selection pressure; if a case is passed by a single individual, that individual will be selected every time the case is considered for a selection pool containing the individual that passes it. This mechanism allows selective pressure to continually shift to individuals that are elite on cases that are not widely solved in $\mathcal{N}$. Because of 2), cases appear in various orderings during selection, resulting in selective pressure for individuals that solve difficult *subsets* of cases. Lexicase selection thereby accounts not only for the difficulty of individual cases but the difficulty of solving arbitrarily-sized subsets of cases. This selection pressure leads to the preservation of high behavioral diversity during evolution (Helmuth et al., 2014; La Cava et al., 2016b).

The worst-case complexity of selecting N parents per generation is $O(TN^2)$, which occurs only if every parent passes or fails every case in $\mathcal{T}$. Normally, due to differential performance across the population and due to lexicase selection's tendency to promote diversity, the overall case depth per selection event is much less than T, and the selection poool typically winnows below N as well, reducing the run-time complexity (Helmuth et al., 2014; La Cava et al., 2016b).

We use an example population originally presented in (Spector, 2012) to illustrate some aspects of standard lexicase selection in the following sections. The population, shown in Table 1, consists of five individuals and four training cases with discrete errors. A graphical example of the filtering mechanism of selection is presented for this example in Figure 1. Each lexicase selection event can be visualized as a randomized depth-first pass through the training cases. Figure 1 shows four example selection events represented by different line types. The populations are winnowed at each case to the elites until single individuals, shown with diamond-shaped nodes, are selected.

Table 1: Example population from original lexicase paper (Spector 2013).

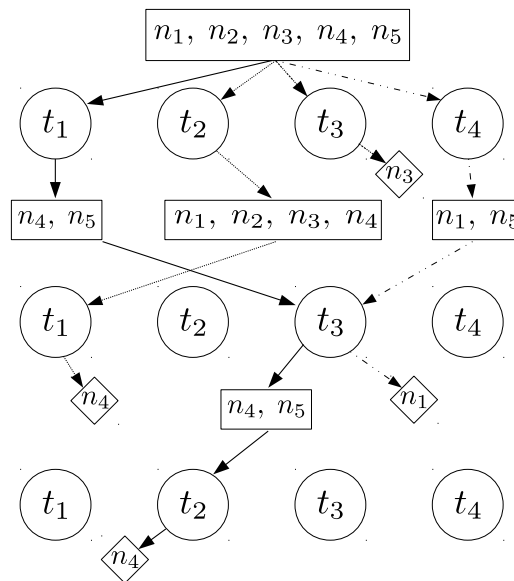| Program	Case Error				Elite Cases	MAE	P_{lex}	P_t
	e_1	e_2	e_3	e_4				
n_1	2	2	4	2	$\{t_2, t_4\}$	2.5	0.25	0.28
n_2	1	2	4	3	$\{t_2\}$	2.5	0.00	0.28
n_3	2	2	3	4	$\{t_2, t_3\}$	2.75	0.33	0.12
n_4	0	2	5	5	$\{t_1, t_2\}$	3.0	0.208	0.04
n_5	0	3	5	2	$\{t_1, t_4\}$	2.5	0.208	0.28

Figure 1: A graphical representation of four example parent selections using lexicase selection on the population in Table 1. The bold, dashed, bold-dashed and dot-dashed lines indicate different selection paths through the training cases in circles. The boxes indicate the selection pool at each step in the process. The diamonds show the individual selected by each selection event.

4 ϵ-Lexicase Selection

Lexicase selection has been shown to be effective in discrete error spaces, both for multi-modal problems (Spector, 2012) and for problems for which every case must be solved exactly to be considered a solution (Helmuth et al., 2014). In continuous error spaces, however, the requirement for individuals to be *exactly* equal to the elite error in the selection pool to pass a case turns out to be overly stringent (La Cava et al., 2016b). In continuous error spaces and especially for symbolic regression with noisy datasets, it is unlikely for two individuals to have exactly the same error on any training case unless they are (or reduce to) equivalent models. As a result, lexicase selection is prone to conducting selection based on single cases, for which the selected individual satisfies $e_t \equiv e_t^*$. Here, e_t^* is the best error on t among $\mathcal{N}$. Selecting on single cases limits the ability of lexicase to leverage case information effectively, and can lead to poorer performance than traditional selection methods (La Cava et al., 2016b).

These observations led to the development of ϵ-lexicase selection (La Cava et al., 2016b), which modifies lexicase selection by modulating case filtering conditions using an ϵ threshold criteria. Hand-tuned and automatic variants of ϵ were proposed and tested. The best performance was achieved by a 'parameter-less' version that defines ϵ according to the dispersion of errors in the population on each training case using the median absolute deviation statistic:

$$\epsilon_t = \text{median}(\mathbf{e}_t - \text{median}(\mathbf{e}_t)) = MAD(\mathbf{e}_t) \tag{2}$$

Defining ϵ according to Eqn. 2 allows the threshold to adapt to changing performance of the population on each training case. As performance across the population improves for a training case, ϵ shrinks, thereby modulating the selectivity of a case based on how difficult it is. We choose MAD in lieu of the standard deviation statistic for calculating ϵ because it is more robust to outliers (Pham-Gia and Hung, 2001).

We study three implementations of ϵ-lexicase selection in this paper: static, which is the version originally proposed (La Cava et al., 2016a); semi-dynamic, in which the elite error is defined relative to the pool; and dynamic, in which both the elite error and ϵ are defined relative to the current selection pool.

Static ϵ-lexicase selection can be viewed as a preprocessing step added to lexicase selection in which the program errors are converted to pass/fail based on an ϵ threshold. This threshold is defined relative to e_t^*. Static ϵ-lexicase is defined below:

Algorithm 3.1: Static ϵ-Lexicase Selection

```
GetParent(𝒩,𝒯):
    𝒯′ ← 𝒯                                              training cases
    𝒮 ← 𝒩                                              initial selection pool is the population
    ε ← MAD(eₜ) for t ∈ 𝒯                              get ε for each case across population
    fitness(n) ← 1 − 𝕀(eₜ(n) ≤ eₜ*(n) + εₜ) for t ∈ 𝒯 and n ∈ 𝒩   convert fitness using within-ε pass
                                                        condition
    while |𝒯′| > 0 and |𝒮| > 1:                        main loop
        case ← random choice from 𝒯′                   consider a random case
        elite ← best fitness in 𝒮 on case             determine elite fitness
        𝒮 ← n ∈ 𝒮 if fitness(n) ≤ elite                reduce selection pool to elites
        𝒯′ ← 𝒯′ − case                                 reduce remaining cases
    return random choice from 𝒮                        return parent
```

Semi-dynamic ϵ-lexicase selection differs from static ϵ-lexicase selection in that the pass condition is defined relative to the best error *among the pool* rather than among $\mathcal{N}$. In this way it behaves more similarly to standard lexicase selection (Algorithm 2.1). It is defined below:

Algorithm 3.2: Semi-dynamic ϵ-Lexicase Selection

```
GetParent(N,T):
    T' ← T                                      training cases
    S ← N                                       initial selection pool is the population
    ε ← MAD(e_t) for t ∈ T                      get ε for each case across population
    while |T'| > 0 and |S| > 1:                 main loop
        case ← random choice from T'               consider a random case
        elite ← best fitness in S on case          determine elite fitness
        S ← n ∈ S if fitness(n) ≤ elite+ε_case     reduce selection pool to elites
        T' ← T' − case                             reduce remaining cases
    return random choice from S                 return parent
```

The final variant of ϵ-lexicase selection is dynamic ϵ-lexicase selection, in which both the error and ϵ are defined among the current selection pool. In this case, ϵ is defined as

$$\epsilon_t(\mathcal{S}) = \text{median}(\mathbf{e}_t(\mathcal{S}) - \text{median}(\mathbf{e}_t(\mathcal{S}))) = MAD(\mathbf{e}_t(\mathcal{S})) \tag{3}$$

where $\mathbf{e}_t(\mathcal{S})$ is the error for case t among the current selection pool $\mathcal{S}$. The dynamic ϵ-lexicase selection algorithm is presented below:

Algorithm 3.3: Dynamic ϵ-Lexicase Selection

```
GetParent(N,T):
    T' ← T                                      training cases
    S ← N                                       initial selection pool is the population
    while |T'| > 0 and |S| > 1:                 main loop
        case ← random choice from T'               consider a random case
        elite ← best fitness in S on case          determine elite fitness
        ε ← MAD(fitness(S)) on case                determine ε for this case
        S ← n ∈ S if fitness(n) ≤ elite+ε_case     reduce selection pool to elites
        T' ← T' − case                             reduce remaining cases
    return random choice from S                 return parent
```

Since calculating ϵ according to Eqn. 2 is $O(TN)$, static and semi-dynamic ϵ-lexicase share a worst-case complexity with lexicase selection of $O(TN^2)$ to select N parents. Dynamic ϵ-lexicase, on the other hand, requires the computation of ϵ with the consideration of each case, rendering a worst-case complexity of $O(TN^3)$. As discussed in §3, these worst-case time complexities are rare, and empirical results have confirmed ϵ-lexicase to run with approximately the same time as tournament selection, which has a time complexity of $O(TN)$.

5 Related Work

Lexicase selection belongs to a class of GP systems that incorporate a program's full semantics directly into the search process, and as such shares a general motivation with recently proposed methods such as Geometric Semantic GP (Moraglio et al., 2012) and Behavioral GP (Krawiec and O'Reilly, 2014), despite differing strongly in approach. Instead of incorporating the full semantics, a number of GP methods alter the fitness

metric by weighting training cases based on population performance. In non-binary Implicit Fitness Sharing (IFS) (Krawiec and Nawrocki, 2013), for example, 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 training case by the success rate of the population (Klein and Spector, 2008). Discovery of objectives by clustering (DOC) (Krawiec and Liskowski, 2015) clusters training cases by population performance, and thereby reduces training cases into a set of objectives used in multi-objective optimization. Both IFS and DOC were outperformed by lexicase selection on program synthesis and boolean problems in previous studies (Helmuth and Spector, 2015; Liskowski et al., 2015). Other methods attempt to sample a subset of $\mathcal{T}$ to reduce computation time or improve performance, such as dynamic subset selection (Gathercole and Ross, 1994), interleaved sampling (Gonalves and Silva, 2013), and co-evolved fitness predictors (Schmidt and Lipson, 2008). 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.

Although to an extent the ideas of multiobjective optimization apply to multiple training cases, they are qualitatively different: objectives are the defined goals of a task, whereas training cases are tools for estimating progress towards those objectives. Objectives and training 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 training cases. One example of using training cases explicitly as objectives occurs in Langdon (1995) in which small numbers of training cases (in this case 6) are used as multiple objectives in a Pareto selection scheme. Other multi-objective approaches such as NSGA-II (Deb et al., 2000), SPEA2 (Zitzler et al., 2001) and ParetoGP (Smits and Kotanchek, 2005) 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 training case sizes, since most individuals become nondominated, leading to selection based mostly on expensive diversity measures rather than performance. Scaling issues in many-objective optimization are reviewed by Ishibuchi et al. (2008). The connection between lexicase selection and multi-objective methods is discussed in depth in §7.

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 (Martnez et al., 2013) 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.

6 Expected Probabilities of Selection

What is the probability of an individual being selected by lexicase selection, given its performance in a population on a set of training cases? Put into words, the probability of n being selected is the probability that a case n passes is selected and: 1) no more cases remain and n is selected among the set of individuals that pass the selected case; or 2) n is the only individual that passes the case; or 3) n is selected via the selection of another case that n passes (repeating the process).

Formally, let $P_{lex}(n|\mathcal{N}, \mathcal{T})$ be the probability of $n \in \mathcal{N}$ being selected. Let $\mathcal{K}_n(\mathcal{T}, \mathcal{N}) = \{k_i\}_{i=1}^{K} \subseteq \mathcal{T}$ be the training cases from $\mathcal{T}$ for which individual n is elite

among $\mathcal{N}$. We will use $\mathcal{K}_n$ for brevity. Then the probability of selection under lexicase can be represented as a piece-wise recursive function:

$$P_{lex}(n|\mathcal{N}, \mathcal{T}) = \tag{4}$$
$$\begin{cases} 1 & : & |\mathcal{T}| > 0, |\mathcal{N}| = 1; \\ 1/|\mathcal{N}| & : & |\mathcal{T}| = 0; \\ \frac{1}{|\mathcal{T}|}\sum_{k_s \in K_n} P_{lex}\left(n|\mathcal{N}(m|k_s \in K_m), \mathcal{T}(t|t \neq k_s)\right) & : & \text{otherwise} \end{cases}$$

The first two elements of P_{lex} follow from the lexicase algorithm: if there is one individual in $\mathcal{N}$, then it is selected; otherwise if there no more cases in in $\mathcal{T}$, then n has a probability of selection split among the individuals in $\mathcal{N}$, i.e., $1/|\mathcal{N}|$. If neither of these conditions are met, the remaining probability of selection is $1/|\mathcal{T}|$ times the summation of P_{lex} over n's elite cases. Each case in K_n has a probability of $1/|\mathcal{T}|$ of being selected. For each potential selection k_s, the probability of n being selected as a result of this case being chosen is dependent on the number of individuals that are also elite on these cases, represented by $\mathcal{N}(m|k_s \in K_m(\mathcal{T}))$, and the cases that are left to be traversed, represented by $\mathcal{T}(t|t \neq k_s)$.

Eqn. 4 also describes the probability of selection under ϵ-lexicase selection, with the condition that *elitism* on a case is defined as being within ϵ of the best error on that case, where the best error is defined among the whole population (static) or among the current selection pool (semi-dynamic and dynamic) and ϵ is defined according to Eqn. 2 or Eqn. 3.

According to Eqn. 4, when fitness values across the population are unique, selection probability is $P_{lex}(n) = \frac{1}{|\mathcal{T}|}\sum_{k_s \in \mathcal{K}_n} 1 = \frac{|\mathcal{K}_n|}{|\mathcal{T}|} \; \forall \; n$, since filtering the population according to any case for which n is elite will result in n being selected. Conversely, if the population semantics are completely homogeneous such that every individual is elite on every case, the selection will be uniformly random, giving the selection probability $P_{lex}(n) = \frac{1}{N} \; \forall \; n$. This property of uniform random selection for identical performance holds true for individual cases as well: a case t will have no impact on the probability of selection of n if $t \notin K_n \; \forall \; n, t \; \in \mathcal{N}, \mathcal{T}$ or $t \in K_n \; \forall \; n, t \; \in \mathcal{N}, \mathcal{T}$. This intuition is also gleaned from Algorithm 2.1: any case that every individual passes provides no selective pressure because the selection pool does not change when it is selected.

Although it is tempting to pair Eqn. 4 with roulette wheel selection as an alternative to lexicase selection, an analysis of its complexity discourages such use. Eqn. 4 has a worst-case complexity of $O(T^N)$ when all individuals are elite on $\mathcal{T}$.

6.1 Effect of Population and Training Set Size

Previous studies have suggested that the performance of lexicase selection is sensitive to the number of training cases (Liskowski et al., 2015). In this section we develop the relation of population size and number of training cases to the performance of lexicase selection as a search driver. In part, this behavior is inherent to the design of the algorithm. However, this behavior is also linked to the fidelity with which lexicase selection samples the expected probabilities of selection for each individual in the population.

Concerning the number of training cases, the effectiveness of lexicase selection is expected to suffer when there are few training cases. When T is small, there are very few ways in which individuals can be selected. For example, if $T = 2$, an individual

must be elite on one of these two cases to be selected. For continuous errors in which few individuals are elite, this means that very few individuals are likely to produce all of the children for the subsequent generation, leading to hyperselection (Helmuth et al., 2016) and diversity loss. On the other hand, if many individuals solve both cases, selection becomes uniformly random. Because ϵ-lexicase modulates the threshold for passing a case, it is likely to be more robust to fewer cases than lexicase selection since the case depth per selection event is likely to be higher.

The population size is tied to selection behavior because lexicase selection conducts N depth-first searches of the case orderings to choose N parents. This implies that the value of N determines the fidelity with which P_{lex} is approximated via the sampling. Smaller populations will therefore produce poorer approximations of P_{lex}. We illustrate this in our example in §8. Of course, this problem is not unique to lexicase selection; tournament selection also samples from an expected distribution and is affected by the number of tournaments (Xie et al., 2007).

Both N and T affect how well the expected probabilities of selection derived from Eqn. 4 are approximated by lexicase selection. Consider the probability of a case being in at least one selection event in a generation. The probability of a case being in at least one selection event in a generation is one minus the probability of it not appearing, yielding

$$1 - \prod_{i=1}^{N} \frac{(T-1)!}{T!(T-1-d_i)!}$$

Here, the case depth d_i is the number of cases used to select a parent for selection event i. Because the case depth varies from selection to selection based on population semantics, this case probability is difficult to analyze. However, it can be simplified to consider the scenario in which a case appears *first* in selection. In fact, Eqn. 4 implies that a case in $\mathcal{K}_n$ influences the probability of selection of n most heavily when it occurs first in a selection event. There are two reasons: first, the case has the potential to filter $N-1$ individuals, which is the strongest selection pressure it can apply. Second, a case's effect size is highest when selected first because it is not conditioned on the probability of selection of any other cases. Each subsequent case selection has a reduced effect on P_{lex} of $\prod_{i=1}^{d} \frac{1}{T-i}$, where d is the recursion depth. These observations also highlight the importance of the relative sizes of N and T because they affect the probability that a case will be observed at the top of a selection event in a given generation, which affects how closely Eqn. 4 is approximated. The probability that a case will come first in a selection event *at least once* in a generation is

$$Pr = 1 - \left(\frac{(T-1)}{T} \right)^N \tag{5}$$

assuming N selection events. This function is plotted for various values of N and T in Figure 2, and illustrates that the probability of a case appearing first in selection drops as T grows and as N shrinks. For example, $Pr \approx 0.5$ when $N = 1000$ and $T = 1433$. We therefore expect the observed probabilities of selection for $n \in \mathcal{N}$ to differ from $P_{lex}(n)$ when $T >> N$, due to insufficient sampling of the cases. In the case of $N >> T$, we expect most cases to appear first and therefore the probability predictions made by Eqn. 4 to be more robust.
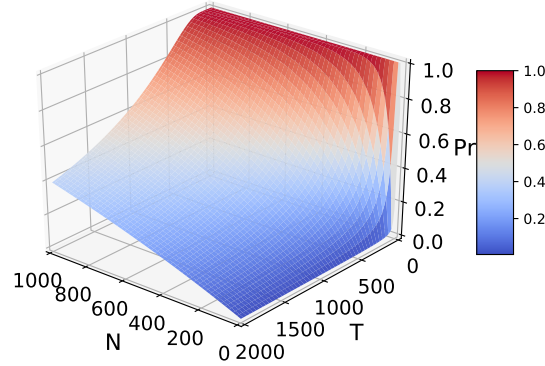
Figure 2: The probability of a case occuring first in a selection event given T training cases and N selections.

6.2 Probabilities under tournament selection

We compare the probability of selection under lexicase selection to that using tournament selection with an indentical population and fitness structure. To do so we must first formulate the probability of selection for an individual undergoing tournament selection with r-size tournaments. The fitness ranks of $\mathcal{N}$ can be calculated, for example using MAE as fitness, with lower rank indicating better fitness. Let S_i be the individuals in $\mathcal{N}$ with a fitness rank of i, and let Q be the number of unique fitness ranks. Xie et al. (2007) showed that the probability of selecting an individual with rank j in a single tournament is

$$P_t = \frac{1}{|S_j|} \left(\left(\frac{\sum_{i=j}^{Q} |S_i|}{N} \right)^r - \left(\frac{\sum_{i=j+1}^{Q} |S_i|}{N} \right)^r \right) \tag{6}$$

In Table 1, the selection probabilities for the example population are shown according to lexicase selection (Eqn. 4) and tournament selection (Eqn. 6). Note that the tournament probabilities are proportional to the aggregate fitness, whereas lexicase probabilities reflect more subtle but intuitive performance differences as discussed by Spector (2012). In §8 we present a more detailed population example with continuous errors and compare probabilities of selection using lexicase, ϵ lexicase and tournament selection.

7 Multi-objective Interpretation of Lexicase Selection

Objectives and training cases are fundamentally different entities: objectives define the goals of the task being learned, whereas cases are the units by which progress towards those objectives is measured. By this criteria, lexicase selection and multi-objective optimization have historically been differentiated (Helmuth, 2015), although there is clearly a "multi-objective" interpretation of the behavior of lexicase selection with respect to the training cases. Let us assume for the remainder of this section that individual fitness cases are objectives to solve. The symbolic regression task then becomes a

high-dimensional, many-objective optimization problem. At this scale, the most popular multi-objective methods (e.g. NSGA-II and SPEA-2) tend to perform poorly, a behavior that has been explained in literature (Wagner et al., 2007; Farina and Amato, 2002). Farina and Amato (2002) point out two short-comings of these multi-objective methods when many objectives are considered:

> the Pareto definition of optimality in a multi-criteria decision making problem can be unsatisfactory due to essentially two reasons: the number of improved or equal objective values is not taken into account, the (normalized) size of improvements is not taken into account.

As we describe in §6, lexicase selection takes into account the number of improved or equal objectives (i.e. cases) by increasing the probability of selection for individuals who solve more cases (consider the summation in the third part of Eqn. 4). The increase per case is proportional to the difficulty of that case, as defined by the selection pool's performance. Regarding Farina and Amato's second point, the *size* of the improvements are taken into account by ϵ-lexicase selection. They are taken into account by the automated thresholding performed by ϵ which rewards individuals for being within an acceptable range of the best performance on the case. We develop the relationship between lexicase selection and Pareto optimization in the remainder of this section.

It has been noted lexicase selection guarantees the return of individuals that are on the Pareto front with respect to the fitness cases (La Cava et al., 2016b). However, this is a necessary but not sufficient condition for selection. As we show below, lexicase selection only selects those individuals in the "corners" or boundaries of the Pareto front, meaning they are on the front *and* elite, globally, with respect to at least one fitness case. Below, we define these Pareto relations with respect to the training cases.

Definition 7.1. n_1 *dominates* n_2, *i.e.,* $n_1 \prec n_2$, *if* $e_j(n_1) \leq e_j(n_2) \ \forall j \in \{1, \ldots, T\}$ *and* $\exists j \in \{1, \ldots, T\}$ *for which* $e_j(n_1) < e_j(n_2)$.

Definition 7.2. *The Pareto set of* $\mathcal{N}$ *is the subset of* $\mathcal{N}$ *that is non-dominated with respect to* $\mathcal{N}$; *i.e.,* $n \in \mathcal{N}$ *is in the Pareto set if* $m \nprec n \ \forall \ m \in \mathcal{N}$.

Definition 7.3. $n \in \mathcal{N}$ *is a Pareto set boundary if* $n \in$ *Pareto set of* $\mathcal{N}$ *and* $\exists j \in \{1, \ldots, T\}$ *for which* $e_j(n) \leq e_j(m) \ \forall \ m \in \mathcal{N}$.

With these definitions in mind, we show that individuals selected by lexicase are on the Pareto set boundaries.

Theorem 7.4. *If individuals from a population* $\mathcal{N}$ *are selected by lexicase selection, those individuals are Pareto set boundaries of* $\mathcal{N}$ *with respect to* $\mathcal{T}$.

Proof: First, we prove (by supposing a contradiction) that individuals selected by lexicase are in the Pareto set. Second, we prove these individuals to be Pareto set boundaries.

First: Let $n_1 in \mathcal{N}$ be any individual and let $n_2 \in \mathcal{N}$ be an individual selected from $\mathcal{N}$ by lexicase selection. Suppose $n_1 \prec n_2$. Then $e_j(n_1) \leq e_j(n_2) \ \forall j \in \{1, \ldots, T\}$ and $\exists j \in \{1, \ldots, T\}$ for which $e_j(n_1) < e_j(n_2)$. Therefore n_1 remains in the selection pool for every case that n_2 does, yet $\exists t \in \mathcal{T}$ for which n_2 is removed from every selection event due to n_1. Hence, n_2 cannot be selected by lexicase selection and the supposition is false. Therefore n_1 and n_2 must be in the Pareto set of $\mathcal{N}$.

Second: let $n \in \mathcal{N}$ be an individual selected from $\mathcal{N}$ by lexicase selection. Then by definition of Algorithm 2.1, $\exists j \in \{1, \ldots, T\}$ for which $e_j(n) \le e_j(m) \; \forall \; m \in \mathcal{N}$. Therefore, since n is in the Pareto set of $\mathcal{N}$, according to Definition 7.3, n is a Pareto set boundary of $\mathcal{N}$.

7.1   Extension to ϵ-lexicase selection

We can extend our multi-objective analysis to ϵ-lexicase selection for conditions in which ϵ is pre-defined for each fitness case (Eqn. 2), which is true for static and semi-dynamic ϵ-lexicase selection. However when ϵ is recalculated for each selection pool, the theorem is not as easily extended due to the need to account for the dependency of ϵ on the current selection pool. We first define ϵ elitism in terms of a relaxed dominance relation and a relaxed Pareto set. We define the dominance relation with respect to ϵ as follows:

Definition 7.5. n_1 ϵ-dominates n_2, i.e., $n_1 \prec_\epsilon n_2$, if $e_j(n_1) + \epsilon_j \le e_j(n_2) \; \forall j \in \{1, \ldots, T\}$ and $\exists j \in \{1, \ldots, T\}$ for which $e_j(n_1) + \epsilon_j < e_j(n_2)$, where $\epsilon_j > 0$ is defined according to Eqn. 2.

This definition of ϵ-dominance differs from a previous ϵ-dominance definition used by Laumanns et al. (2002) (cf. Eqn. (6)) in which $n_1 \prec_\epsilon n_2$ if

$$e_j(n_1) + \epsilon_j \le e_j(n_2) \; \forall \; j \in \{1, \ldots, T\}$$

Definition 7.5 is more strict, requiring $e_j(n_1) + \epsilon_j < e_j(n_2)$ for at least one j in analogous fashion to Definition 7.1. In order to extend Theorem 7.4, this definition must be more strict since a useful ϵ-dominance relation needs to capture the ability of an individual to preclude the selection of another individual under ϵ-lexicase selection.

Definition 7.6. The ϵ-Pareto set of $\mathcal{N}$ is the subset of $\mathcal{N}$ that is non-ϵ-dominated with respect to $\mathcal{N}$; i.e., $n \in \mathcal{N}$ is in the ϵ-Pareto set if $m \nprec_\epsilon n \; \forall \; m \in \mathcal{N}$.

Definition 7.7. $n \in \mathcal{N}$ is an ϵ-Pareto set boundary if n is in the ϵ-Pareto set of $\mathcal{N}$ and $\exists j \in \{1, \ldots, T\}$ for which $e_j(n_1) \le e_j(m) + \epsilon_j \; \forall \; m \in \mathcal{N}$, where ϵ_j is defined according to Eqn 2.

Theorem 7.8. If ϵ is defined according to Eqn. 2, and if individuals are selected from a population $\mathcal{N}$ by ϵ-lexicase selection, then those individuals are ϵ-Pareto set boundaries of $\mathcal{N}$.

Proof. First: Let $n_1 in \mathcal{N}$ be any individual and let $n_2 \in \mathcal{N}$ be an individual selected from $\mathcal{N}$ by static or semi-dynamic ϵ-lexicase selection. Suppose $n_1 \prec_\epsilon n_2$. Therefore n_1 remains in the selection pool for every case that n_2 does, yet $\exists t \in \mathcal{T}$ for which n_2 is removed from every selection event due to n_1. Hence, n_2 cannot be selected by lexicase selection and the supposition $n_1 \prec_\epsilon n_2$ is false. Therefore n_1 and n_2 must be in the ϵ-Pareto set of $\mathcal{N}$ to be selected.

Second: let n be an individual selected from population $\mathcal{N}$ by static or semi-dynamic ϵ-lexicase selection. Then by definition of Algorithm 3.1 or 3.2, $\exists j \in \{1, \ldots, T\}$ for which $e_j(n) \le e_j(m) \; \forall \; m \in \mathcal{N}$. Since n is in the ϵ-Pareto set of $\mathcal{N}$ and according to Definition 7.7, n must be a ϵ-Pareto set boundary of $\mathcal{N}$. $\square$

The selection of Pareto set boundaries by lexicase selection is illustrated in left plot of Figure 3 for a scenario with two fitness cases. Each point in the plot represents an

individual, and the squares are the Pareto set. Under a lexicase selection event with case sequence $\{t_1, t_2\}$, individuals would first be filtered to the two left-most individuals that are elite on e_1, and then to the individual among those two that is best on e_2, i.e. the selected square individual. Note that the selected individual is a Pareto set boundary. The individual on the other end of the Pareto set shown as a black square would be selected using the opposite order of cases.

Consider the analogous case for semi-dynamic ϵ-lexicase selection illustrated in the right plot of Figure 3. In this case the Pareto set is indicated by squares that also belong to the ϵ-Pareto set, indicated by circles with dotted lines. Under a semi-dynamic ϵ-lexicase selection event with case order $\{t_1, t_2\}$, the population would first be filtered to the four left-most individuals that are within ϵ_1 of the elite error on case t_1, and then the indicated square would be selected by being the only individual within ϵ_2 of the elite error on t_2 among the current pool. Note that that although the selected individual is an ϵ-Pareto set boundary by Definition 7.7, it is *not* a boundary of the Pareto set. Theorem 7.8 only guarantees that the selected program is within ϵ of the best error for at least one case, which in this scenario is t_1.

Comparing the left and right plots of Figure 3 illustrates the effect of introducing ϵ to the filter conditions in lexicase: it reduces the selectivity of each case, ultimately resulting in the selection of individuals that are not as extremely positioned in objective space. Regarding the position of solutions in this space, it's worth noting the significance of boundary solutions (and near boundary solutions) in the context of multi-objective optimization. Boundary solutions are known to contribute significantly to hypervolume measures (Deb et al., 2005), where the hypervolume is a measure of how well-covered the objective space is by the current set of solutions. The boundary solutions have an infinite score according to NSGA-II's crowding measure (Deb et al., 2000), with higher being better, meaning they are the first non-dominated solutions to be preserved by selection when the population size is reduced. Nonetheless, multi-objective literature appears divided on how these boundary solutions drive search when the goal of the algorithm is to approximate the optimal Pareto front (Wagner et al., 2007). The goal of GP, in contrast, is to preserve points in the search space that, when combined and varied, yield a single best solution. So while the descriptions above lend insight to the function of lexicase and ϵ-lexicase selection, the different goals of search and the high dimensionality of training cases must be remembered when comparing to multi-objective optimization.

As a last note, when considered as objectives, the worst-case complexity of lexicase selection matches that of NSGA-II: $O(TN^2)$. Interestingly, the worst case complexity of the crowding distance assignment portion of NSGA-II, $O(TN\log(N))$, occurs when all individuals are non-dominated, which is expected in high dimensions (Farina and Amato, 2002; Wagner et al., 2007). Under lexicase selection, a non-dominated population *that is semantically unique* yields the minimum complexity scenario of $O(N^2)$.

8 Illustrative Example

An example population is presented in Table 2 featuring floating point errors, in contrast to Table 1. In this case, the population semantics are completely unique, although they result in the same mean error across the training cases, as shown in the "Mean" column. As a result, tournament selection picks uniformly from among these individ-
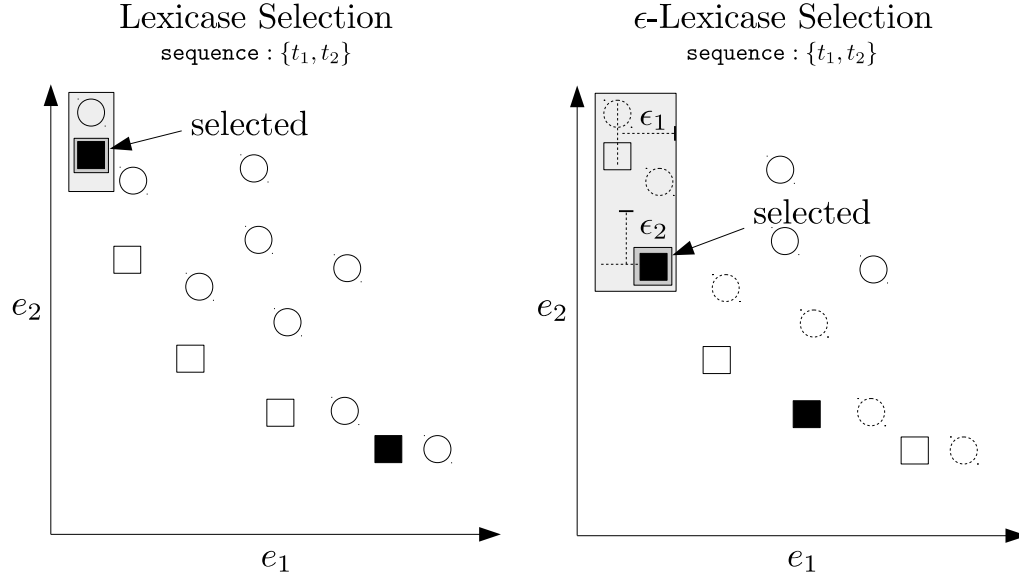
Lexicase Selection
sequence : $\{t_1, t_2\}$

ϵ-Lexicase Selection
sequence : $\{t_1, t_2\}$

Figure 3: An illustration of the performance of lexicase selection (left) semi-dynamic ϵ-lexicase selection (right) in a scenario involving two cases. Each point represents and individual in the population. The squares are individuals in the Pareto set, and the circles with dotted lines are individuals in the ϵ-Pareto set (the ϵ-Pareto set also includes the squares). A selection event for case sequence $\{t_1, t_2\}$ is shown by the gray rectangles. The black points are individuals that could be selected by any case ordering.

uals, resulting in equivalent probabilities of selection. As mentioned in §6, with unique populations, lexicase selection is proportional to the number of cases for which an individual is elite. This leads lexicase selection to pick from among the four individuals that are elite on cases, i.e. n_1, n_4, n_5, and n_9, with respective probabilities 0.2, 0.2, 0.2, and 0.4.

Due to its strict definition of elitism, lexicase selection does not account for the fact other individuals are very close to being elite on these cases as well. The ϵ-lexicase variants address this as noted by the smoother distribution of selection probabilities among this population. We focus first on static ϵ-lexicase selection. Applying the ϵ threshold to the errors yields the following discrete fitnesses:

	e_1	e_2	e_3	e_4	e_5
n_1	0	1	1	1	1
n_2	0	1	1	0	1
n_3	0	1	1	0	1
n_4	1	1	0	0	1
n_5	1	1	0	1	1
n_6	1	1	0	1	1
n_7	1	0	1	1	0
n_8	1	0	1	1	0
n_9	1	0	1	1	0

Table 2: Example population with training case performances and selection probabilities according to the different algorithms.

$\mathcal{N}$	Cases					Mean	Probability of Selection				
	e_1	e_2	e_3	e_4	e_5		tourn	lex	ϵ lex static	ϵ lex semi	ϵ lex dyn
n_1	0.0	1.1	2.2	3.0	5.0	2.26	0.111	0.200	0.000	0.067	0.033
n_2	0.1	1.2	2.0	2.0	6.0	2.26	0.111	0.000	0.150	0.117	0.200
n_3	0.2	1.0	2.1	1.0	7.0	2.26	0.111	0.000	0.150	0.117	0.117
n_4	1.0	2.1	0.2	0.0	8.0	2.26	0.111	0.200	0.300	0.200	0.167
n_5	1.1	2.2	0.0	4.0	4.0	2.26	0.111	0.200	0.000	0.050	0.050
n_6	1.2	2.0	0.1	5.0	3.0	2.26	0.111	0.000	0.000	0.050	0.033
n_7	2.0	0.1	1.2	6.0	2.0	2.26	0.111	0.000	0.133	0.133	0.133
n_8	2.1	0.2	1.0	7.0	1.0	2.26	0.111	0.000	0.133	0.133	0.217
n_9	2.2	0.0	1.1	8.0	0.0	2.26	0.111	0.400	0.133	0.133	0.050
ϵ	0.9	0.9	0.9	2.0	2.0						

The selection probabilities for static ϵ-lexicase selection are equivalent to the selection probabilities of lexicase selection on this converted error matrix. Despite elitism on case t_1, n_1 is not selected since n_2 and n_3 are ϵ-elite on this case *in addition to* t_4. Consider n_4, which has a higher probability of selection under static ϵ-lex than lexicase selection. This is due to n_4 being ϵ-elite on a unique combination of cases: t_3 and t_4. Lastly n_8 is selected in equal proportions to n_7 and n_6 since all are within ϵ of the elite error on the same cases.

Semi-dynamic ϵ-lexicase selection allows for all nine individuals to be selected with varying proportions that are similar to those derived for static ϵ lexicase selection. Selection probabilities for n_1 illustrate the differences in the static and semi-dynamic variants: n_1 has a chance for selection in the semi-dynamic case because when t_1 is selected as the first case, n_1 is within ϵ of the best case errors *among the pool*, i.e. $\{n_1, n_2, n_3\}$, for any subsequent order of cases. The probability of selection for n_5 and n_6 follow the same pattern.

Dynamic ϵ-lexicase selection produces the most differentiated selection pressure for this example. Consider individual n_8 which is the most likely to be selected for this example. It is selected more often than n_7 or n_9 due to the adaptations to ϵ as the selection pool is winnowed. For example, n_8 is selected by case sequence $\{t_2, t_1, t_3\}$, for which the selection pool takes the following form after each case: $\{n_7, n_8, n_9\}, \{n_7, n_8\}, \{n_8\}$. Under the other semi-dynamic ϵ lexicase selection, n_7 and n_8 would not be removed by these cases due to the fixed nature of ϵ for those cases.

The expected probabilities of selection for the population in Table 2 is shown graphically in Figure 4 and visually confirms the similarities in probabilities of selection for the ϵ-lexicase variants, especially semi-dynamic and dynamic variants. The expected probability of selection for semi-dynamic ϵ-lexicase and tournament selection are compared to measured probabilities of selection in Figures 5 and 6, respectively. 100 trials of a series of 500 selections are conducted, and the probabilities of selection
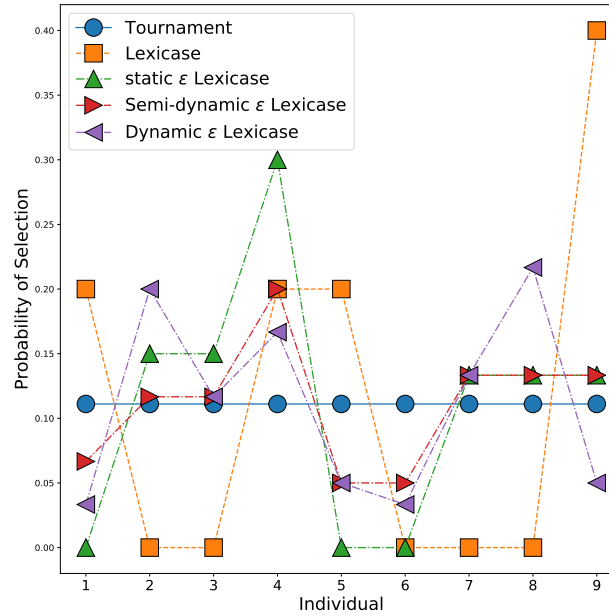
Figure 4: A graphical representation of the probabilities of selection for the population in Table 2 according to different selection methods.

are reported for each individual after each selection event. These figures illustrate that the probabilities of selection converge on the analytical predictions provided by Eqn. 4 and Eqn. 6, but that the observed probability of selection is sensitive to the number of selection events for both selection methods. We use population sizes of 1000 in our experiments in §9, which should provide adequate sampling to converge to approximately the expected probabilities of selection.

9 Experimental Analysis

The problems studied in this section are listed in Table 4. We benchmark nine methods using eight different datasets. Six of the problems are available from the UCI repository (Lichman, 2013). The UBall5D problem is a simulated equation[1] 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. (2012).

We compare eight different selection methods: random selection, tournament selection, lexicase selection, age-fitness pareto optimization (Schmidt and Lipson, 2011), deterministic crowding Mahfoud (1995), and the three ϵ-lexicase methods presented in §4. In addition to the selection methods that are benchmarked, we include a comparison to regularized linear regression using Lasso (Tibshirani, 1996). These methods are

---

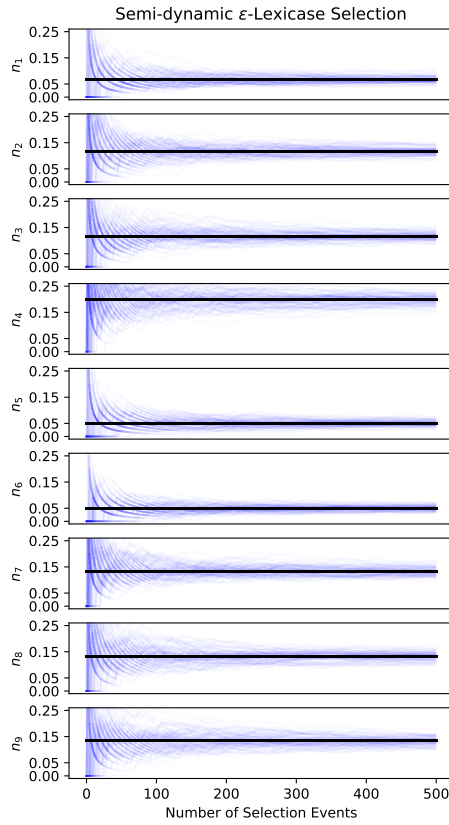[1]UBall5D is also known as Vladislavleva-4.

W. La Cava

Figure 5: Selection probabilities for semi-dynamic ϵ-lexicase selection on the example population in Table 2. The black line indicates the expected probability of selection according to Eqn. 4. The blue lines are measured probabilities of selection after several selection events. 100 trials of 500 selections are shown.
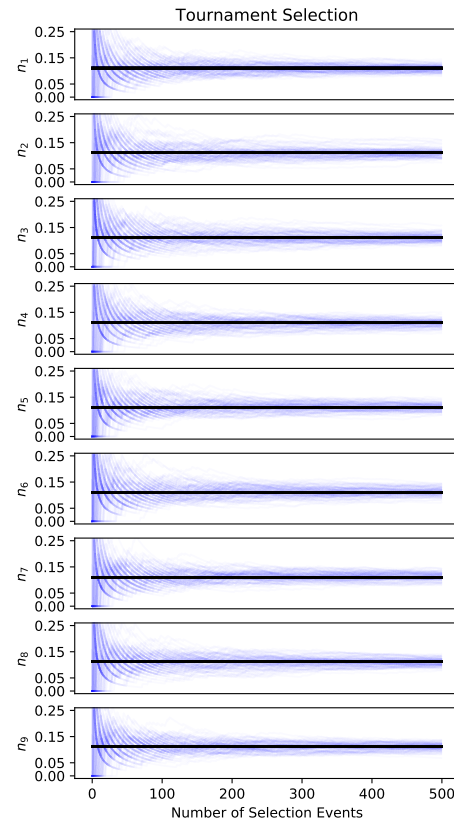
Figure 6: Selection probabilities for tournament selection on the example population in Table 2. The black line indicates the expected probability of selection according to Eqn. 6. The blue lines are measured probabilities of selection after several selection events. 100 trials of 500 selections are shown.

described briefly below, along with their abbreviations used in the results.

- Random Selection (rand): selection for parents is random.

- Tournament Selection (tourn): size two tournaments are conducted for choosing parents.

- Lexicase Selection (lex): see Algorithm 2.1.

- Age-fitness Pareto optimization (afp): this method introduces a new individual each generation with an age of 0. Each generation, individuals are assigned an age equal to the number of generations since their oldest ancestor entered the population. Parents are selected randomly to create N children. The children and parents then compete in survival tournaments of size two, in which an individual is culled from the population if it is dominated in terms of age and fitness by its competitor.

- Deterministic crowding (dc): A generational form of this niching method is used in which parents are selected randomly for variation and the child competes to replace the parent with which it is most similar. Similarity is determined based on the Levenshtein distance of the parent's equation forms, using a universal symbol for coefficients. A child replaces its parent in the population only if it has a better fitness.

- Static ϵ-lexicase selection (ep-lex-s): See Algorithm 3.1.

- Semi-dynamic ϵ-lexicase selection (ep-lex-sd): See Algorithm 3.2.

- Dynamic ϵ-lexicase selection (ep-lex-d): See Algorithm 3.3.

- Lasso (lasso): this method incorporates a regularization penalty into least squares regression using an ℓ_1 measure of the model coefficients and uses a tuning parameter, λ, to specify the weight of this regularization. We use a least angle regression (Efron et al., 2004) implementation of Lasso that automatically chooses λ using cross validation.

We conduct 50 trials of each method by training on a random partition of 70% of the dataset and comparing the prediction error of the best model from each method on the other 30% of the dataset. In addition to test error, we compare the training convergence of the GP-based methods, the semantic diversity of the populations during the run, and the number of cases used for selection for the lexicase methods. We calculate population diversity as the fraction of unique semantics in the population, i.e.

$$\text{diversity}(\mathcal{N}) = \frac{1}{N^2} \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} \mathbb{I}[\hat{y}(n_i) \neq \hat{y}(n_j)] \qquad (7)$$

To compare the number of cases used in selection for the lexicase methods, we save the median number of cases used in selection events, i.e. the case depth, each generation.

Table 3: GP settings.

Setting	Value
Population size	1000
Crossover / mutation	60/40%
Program length limits	[3, 50]
ERC range	[-1,1]
Generation limit	1000
Trials	50
Terminal Set	$\{\mathbf{x}, \text{ERC}, +, -, *, /, \sin, \cos, \exp, \log\}$
Elitism	keep best

Table 4: Regression problems used for method comparisons.

Problem	Dimension	Samples
Airfoil	5	1503
Concrete	8	1030
ENC	8	768
ENH	8	768
Housing	14	506
Tower	25	3135
UBall5D	5	6024
Yacht	6	309

10   Results

The boxplots in Figure 7 show the test set MSE for each method on each problem. In the final subplot, we summarize the mean rankings of the methods on each trial of each problem to give a general comparison of performance. In general we find that the ϵ-lexicase methods produce models with the best generalization performance across the tested problems. Random selection and Lasso tend to perform the worst on these problems. It is interesting to note the performance of Lasso on the Tower problem, which is better than on other datasets; ep-lex-sd and ep-lex-d are the only GP variants to significantly outperform it. For every problem, a variant of ϵ-lexicase selection performs the best, and the three variants of it tend to perform similarly. In accordance with previous results (La Cava et al., 2016b), lexicase selection performs worse than tournament selection for these continuous valued problems. In contrast with previous findings (Schmidt and Lipson, 2011), dc tends to outperform afp, although both methods perform better than tournament selection.

The ϵ-lexicase methods show a marked advantage in converging on a low training error in fewer generations compared to all other methods, as evidenced in Figure 8. Note Figure 8 reports the normalized MSE values on the training set for the best individual in the population each generation. Again we observe very little difference between the ϵ-lexicase variants.

We analyze the statistical significance of the test MSE results in Tables 5 and 6. Table 5 shows pair-wise Wilcoxon ranksum tests for each method in comparison to ep-lex-d. There are significant differences in performance for all problems between ep-lex-d and all non-ϵ-lexicase methods, with the sole exception of the comparison to dc on the housing dataset. Analysis of variance of the method rankings across all problems indicates significant differences ($p < $ 2e-16). A post-hoc statistical analysis shown in Table 6 indicates that this difference is due to significant differences in rankings across all problems for ep-lex-sd and ep-lex-d in pairwise comparison to all other non-ϵ-lexicase

methods. The three variants of ϵ-lexicase do not differ significantly from each other according to this test.

Figure 9 shows the semantic diversity of the populations for each generation using different selection methods. ϵ-lexicase variants, dc, and lexicase selection all produce the highest population diversity, as expected due to their diversity maintenance design. Interestingly, they all produce more diverse semantics than random selection, suggesting that the preservation of *useful* diversity is an important feature of the observed performance improvements. Surprisingly, afp is found to produce low semantic diversity, despite its incorporation of age and random restarts each generation. Given that afp has no explicit semantic diversity mechanism, it's possible that age is not an adequate surrogate for behavioral diversity on these problems.

One of the motivations for introducing an ϵ threshold into lexicase selection is to allow selection to make use of more cases in continuous domains when selecting parents. Figure 10 demonstrates that ϵ-lexicase methods achieve this goal. As we noted at the beginning of §4, lexicase selection likely only uses one case per selection event in continuous domains, leading to poor performance. We observe this phenonemon in the median case depth measurements. Among the ϵ-lexicase variants, ep-lex-sd uses the most cases in general, followed by ep-lex-s and ep-lex-d. Intuitively this result makes sense: ϵ is likely to be largest when computed across the population, and because ep-lex-sd uses the global ϵ (Eqn. 2) and a local error threshold, it is likely to keep the most individuals at each case filtering. These results also suggest that ϵ shrinks substantially when calculated among the pool after each case (Eqn. 3) in ep-lex-d.

Table 5: Significance test p-values comparing test MSE using the pair-wise Wilcoxon rank-sum test with Holm correction for multiple comparisons. All significance tests are conducted relative to dynamic ϵ-lexicase (ep-lex-d). Bold indicates $p < 0.05$.

	lasso	rand	tourn	lex	afp	dc	ep-lex-s	ep-lex-sd
airfoil	**2.54e-16**	**2.54e-16**	**3.99e-16**	**2.54e-16**	**2.95e-14**	**1.04e-12**	0.17	0.57
concrete	**2.54e-16**	**2.54e-16**	**1.14e-11**	**5.60e-16**	**2.46e-06**	**2.09e-02**	0.67	0.32
enc	**2.54e-16**	**2.54e-16**	**6.96e-16**	**2.77e-16**	**4.99e-15**	**1.18e-09**	0.23	**3.02e-02**
enh	**2.54e-16**	**2.54e-16**	**3.99e-16**	**2.54e-16**	**2.03e-15**	**1.27e-14**	**1.93e-05**	0.48
housing	**4.76e-05**	**1.46e-12**	**6.99e-04**	**5.73e-07**	**1.42e-02**	0.11	1	1
tower	**4.26e-02**	**2.54e-16**	**1.14e-15**	**1.81e-15**	**2.10e-15**	**8.54e-15**	**3.20e-02**	0.56
uball5d	**2.54e-16**	**2.54e-16**	**2.55e-15**	**1.57e-13**	**4.33e-16**	**8.70e-13**	1	1
yacht	**2.54e-16**	**3.23e-16**	**6.20e-11**	**1.06e-10**	**6.20e-11**	**5.92e-03**	**1.56e-02**	**4.55e-03**

Table 6: Post-hoc pairwise statistical tests of rankings across problems according to Tukey's Honest Significant Difference test. Bold values indicate $p < 0.05$ with adjustment for multiple comparisons.

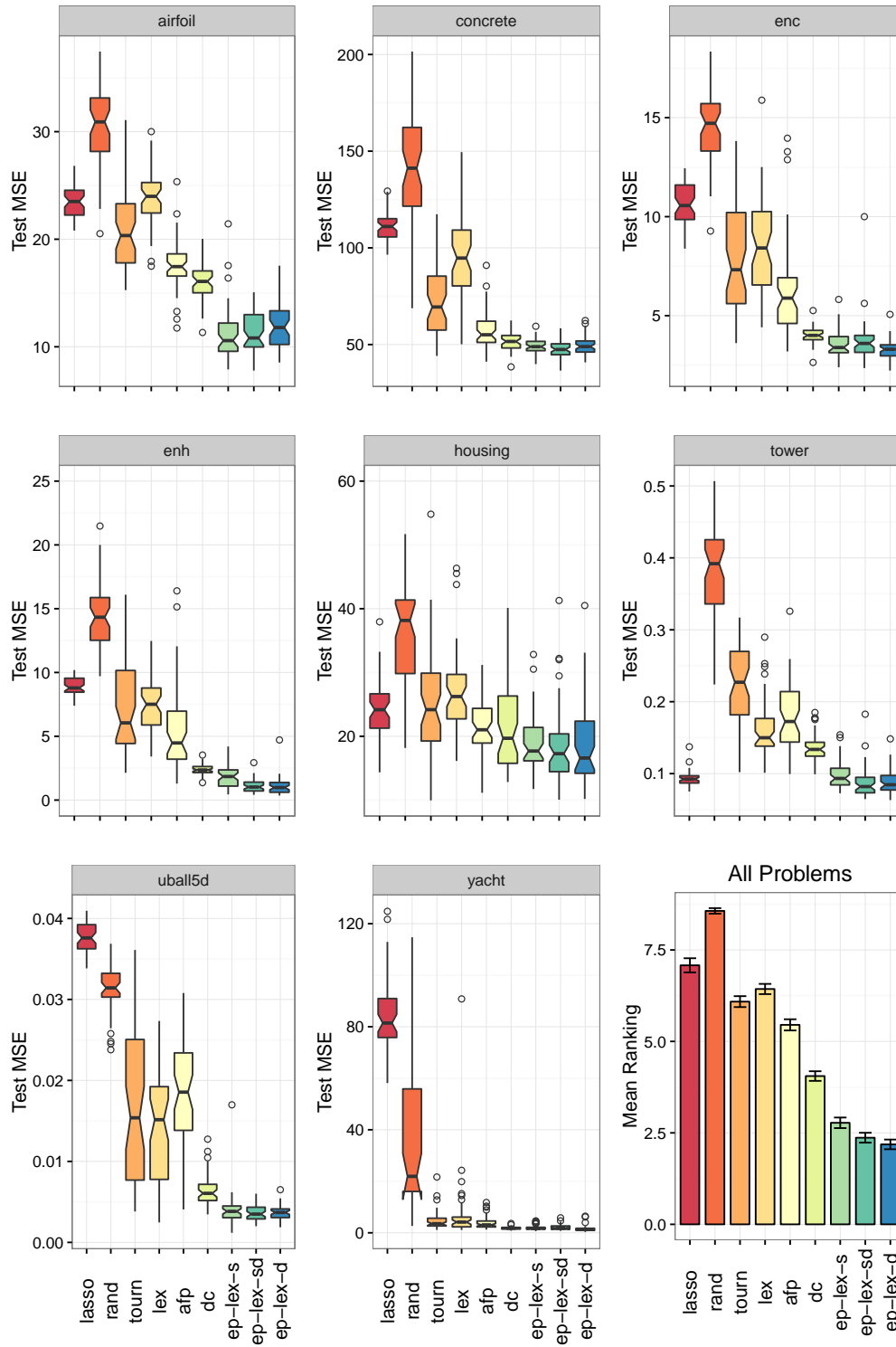|	lasso	rand	tourn	lex	afp	dc	ep-lex-s	ep-lex-sd
ep-lex-s	**1.56e-11**	**1.53e-11**	**2.04e-09**	**9.04e-11**	**8.54e-07**	0.077		
ep-lex-sd	**1.54e-11**	**1.53e-11**	**5.68e-11**	**1.70e-11**	**1.79e-08**	**4.82e-03**	0.99	
ep-lex-d	**1.54e-11**	**1.53e-11**	**2.28e-11**	**1.57e-11**	**3.13e-09**	**1.14e-03**	0.89	1

Figure 7: Boxplots of the mean squared error on the test set for 50 randomized trials of each algorithm on the regression benchmark datasets.
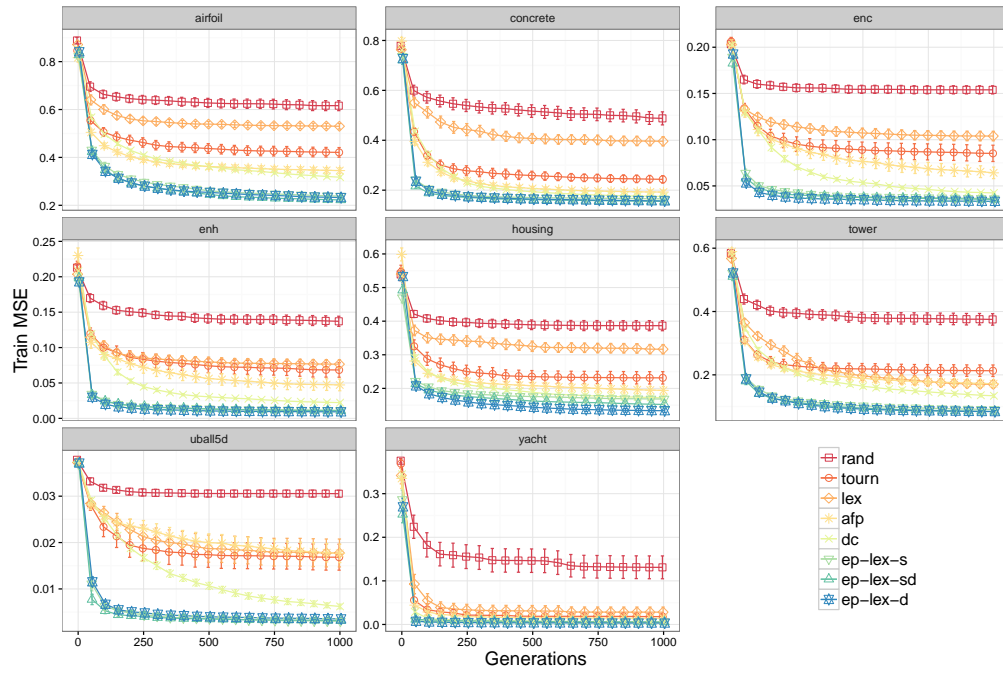
Figure 8: Training error convergence for the best individual using different selection methods. The error bars represent 95% confidence intervals over 50 trials.
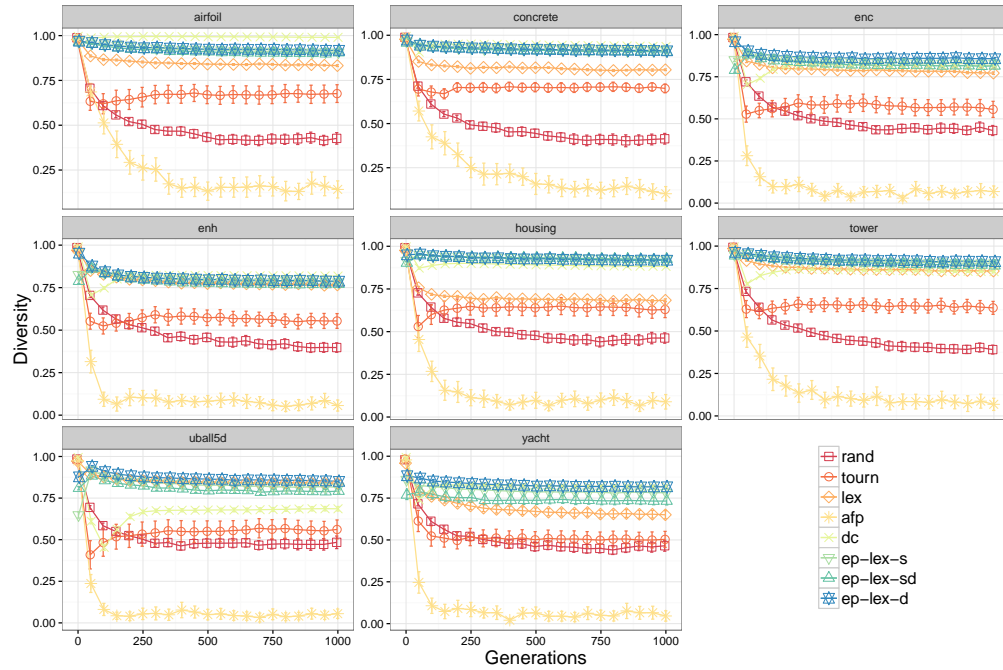

Figure 9: Behavioral diversity (Eqn. 7) of the population using different selection methods. The error bars represent 95% confidence intervals over 50 trials.
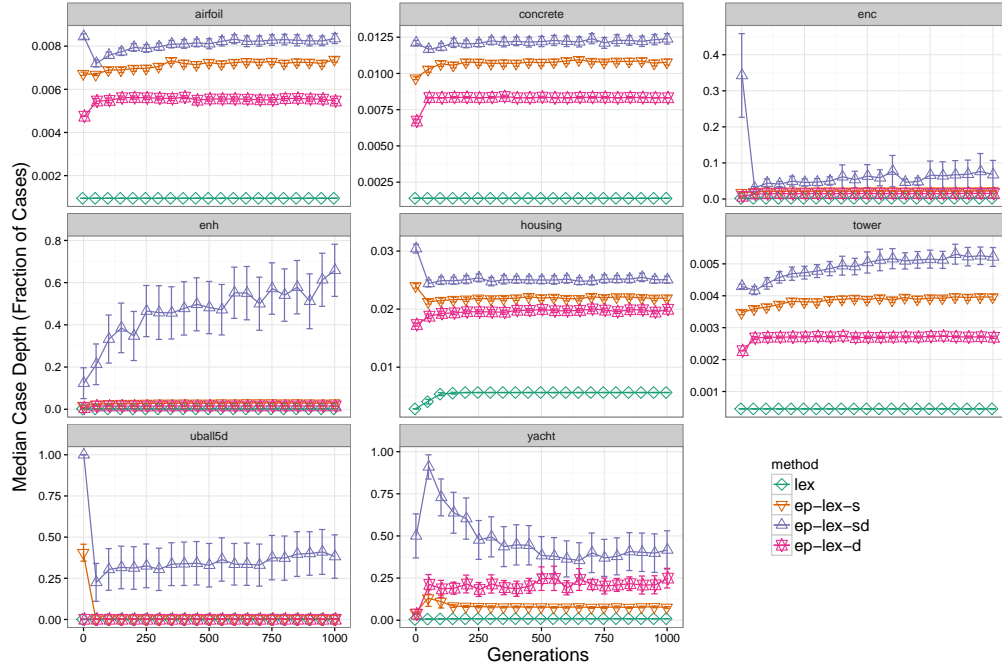
Figure 10: Median case depths of selection each generation for the lexicase selection variants on the regression problems.

11   Discussion

The experimental results show that ϵ-lexicase selection performs well on the symbolic regression problems studied compared to other methods. ϵ-lexicase leads to quicker learning on the training set (Figure 8), better test set performance (Figure 7), and high semantic diversity throughout training (Figure 9). The variants of ϵ-lexicase do not vary significantly from each other over all tested problems, which suggests that the foundations of the method are robust to different definitions of ϵ as long as they result in higher leverage of case information during selection. With this in mind, we suggest semi-dynamic ϵ-lexicase (ep-lex-sd, Algorithm 3.2) as the default implementation since it performs similarly to dynamic ϵ-lexicase (ep-lex-d), has a lower worst-case complexity (see §4), and appears to utilize the most case information according to Figure 10.

ϵ-lexicase selection is a global pool, uniform random sequence, non-elitist version of lexicase selection (Spector, 2012). "Global pool" means that each selection event begins with the entire population. It is possible that smaller pools, defined geographically for example (Spector and Klein, 2006), could improve performance on certain problems that respond well to relaxed selection pressure. "Uniform random sequence" refers to the order of the cases, and unlike pool size, other orderings of training 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. Liskowski et al. (2015) attempted to use derived objective clusters as cases in lexicase selection, but found that this actually decreased performance, possibly due to the small number of resultant objectives. Still, there may be a form of ordering or case reduction that improves lexicase

selection's performance over random shuffling.

The ordering of the training cases that produce a given parent also contains potentially useful information that could be used by the search operators in GP. Helmuth and Spector (2015) 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.

Based on the observations in §6, when the training set is much larger than the population size, some cases are likely to go unused. In these scenarios it may be advantageous to reduce program evaluations by lazily evaluating programs on cases as they appear in selection events. Indeed, Eqn. 5 could be used as a guide for determining when a lazy evaluation strategy would lead to significant computational savings.

12   Conclusions

In this paper we present a probabilistic and multi-objective analysis of lexicase selection and ϵ-lexicase selection. We develop the expected probabilities of selection under lexicase selection variants, and show the impact of population size and training set size on probabilities of selection. For the first time, the connection between lexicase selection and multi-objective optimization is analyzed, showing that individuals selected by lexicase selection occupy the boundaries or near boundaries of the Pareto front in the high-dimensional space spanned by the population errors.

In addition, we experimentally validate ϵ-lexicase selection, including two new variants, on a set of real-world and synthetic symbolic regression problems. The results suggest that ϵ-lexicase selection strongly improves the ability of GP to find accurate models. Further analysis of these runs show that lexicase variants maintain exceptionally high diversity during evolution, and that ϵ-lexicase variants consider more cases per selection event than standard lexicase selection. The results validate our motivation for creating this variant of lexicase for continuous domains, and suggest the adoption of lexicase selection and variants of it in similar domains.

13   Acknowledgments

References

Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. (2000). A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II.

In Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J. J., and Schwe-fel, H.-P., editors, *Parallel Problem Solving from Nature PPSN VI*, volume 1917, pages 849–858. Springer Berlin Heidelberg, Berlin, Heidelberg.

Deb, K., Mohan, M., and Mishra, S. (2005). Evaluating the -Domination Based Multi-Objective Evolutionary Algorithm for a Quick Computation of Pareto-Optimal Solutions. *Evolutionary Computation*, 13(4):501–525.

Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., and others (2004). Least angle regression. *The Annals of statistics*, 32(2):407–499.

Farina, M. and Amato, P. (2002). On the optimal solution definition for many-criteria optimization problems. In *Fuzzy Information Processing Society, 2002. Proceedings. NAFIPS. 2002 Annual Meeting of the North American*, pages 233–238. IEEE.

Gathercole, C. and Ross, P. (1994). Dynamic training subset selection for supervised learning in Genetic Programming. In Davidor, Y., Schwefel, H.-P., and Mnner, R., editors, *Parallel Problem Solving from Nature PPSN III*, number 866 in Lecture Notes in Computer Science, pages 312–321. Springer Berlin Heidelberg. DOI: 10.1007/3-540-58484-6_275.

Gonalves, I. and Silva, S. (2013). *Balancing learning and overfitting in genetic programming with interleaved sampling of training data*. Springer.

Helmuth, T. (2015). General Program Synthesis from Examples Using Genetic Programming with Parent Selection Based on Random Lexicographic Orderings of Test Cases. *Doctoral Dissertations May 2014 - current*.

Helmuth, T., McPhee, N. F., and Spector, L. (2016). The impact of hyperselection on lexicase selection. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, pages 717–724. ACM.

Helmuth, T. and Spector, L. (2015). General Program Synthesis Benchmark Suite. pages 1039–1046. ACM Press.

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

Ishibuchi, H., Tsukamoto, N., and Nojima, Y. (2008). Evolutionary many-objective optimization: A short review. In *IEEE congress on evolutionary computation*, pages 2419–2426. Citeseer.

Klein, J. and Spector, L. (2008). Genetic programming with historically assessed hardness. *Genetic Programming Theory and Practice VI*, pages 61–75.

Krawiec, K. (2016). *Behavioral program synthesis with genetic programming*, volume 618. Springer.

Krawiec, K. and Lichocki, P. (2010). Using Co-solvability to Model and Exploit Synergetic Effects in Evolution. In Schaefer, R., Cotta, C., Koodziej, J., and Rudolph, G., editors, *Parallel Problem Solving from Nature, PPSN XI*, pages 492–501. Springer Berlin Heidelberg, Berlin, Heidelberg.

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

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

Krawiec, K. and O'Reilly, U.-M. (2014). Behavioral programming: a broader and more detailed take on semantic GP. In *Proceedings of the 2014 conference on Genetic and evolutionary computation*, pages 935–942. ACM Press.

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

La Cava, W., Spector, L., and Danai, K. (2016b). Epsilon-Lexicase Selection for Regression. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, pages 741–748, New York, NY, USA. ACM.

Langdon, W. B. (1995). Evolving Data Structures with Genetic Programming. In *ICGA*, pages 295–302.

Laumanns, M., Thiele, L., Zitzler, E., and Deb, K. (2002). Archiving with guaranteed convergence and diversity in multi-objective optimization. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pages 439–447. Morgan Kaufmann Publishers Inc.

Lichman, M. (2013). *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences.

Liskowski, P., Krawiec, K., Helmuth, T., and Spector, L. (2015). Comparison of Semantic-aware Selection Methods in Genetic Programming. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO Companion '15, pages 1301–1307, New York, NY, USA. ACM.

Mahfoud, S. W. (1995). *Niching methods for genetic algorithms*. PhD thesis.

Martnez, Y., Naredo, E., Trujillo, L., and Galvn-Lpez, E. (2013). Searching for novel regression functions. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 16–23. IEEE.

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

Moraglio, A., Krawiec, K., and Johnson, C. G. (2012). Geometric semantic genetic programming. In *Parallel Problem Solving from Nature-PPSN XII*, pages 21–31. Springer.

Pham-Gia, T. and Hung, T. L. (2001). The mean and median absolute deviations. *Mathematical and Computer Modelling*, 34(78):921–936.

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

Schmidt, M. and Lipson, H. (2011). Age-fitness pareto optimization. In *Genetic Programming Theory and Practice VIII*, pages 129–146. Springer.

Smits, G. F. and Kotanchek, M. (2005). Pareto-front exploitation in symbolic regression. In *Genetic Programming Theory and Practice II*, pages 283–299. Springer.

Spector, L. (2012). Assessment of problem modality by differential performance of lexicase selection in genetic programming: a preliminary report. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion*, pages 401–408.

Spector, L. and Klein, J. (2006). Trivial geography in genetic programming. In *Genetic programming theory and practice III*, pages 109–123. Springer.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

Vanneschi, L., Castelli, M., and Silva, S. (2014). A survey of semantic methods in genetic programming. *Genetic Programming and Evolvable Machines*, 15(2):195–214.

Wagner, T., Beume, N., and Naujoks, B. (2007). Pareto-, Aggregation-, and Indicator-Based Methods in Many-Objective Optimization. In *Evolutionary Multi-Criterion Optimization*, pages 742–756. Springer, Berlin, Heidelberg. DOI: 10.1007/978-3-540-70928-2_56.

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

Xie, H., Zhang, M., and Andreae, P. (2007). Another investigation on tournament selection: modelling and visualisation. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1468–1475. ACM.

Zitzler, E., Laumanns, M., and Thiele, L. (2001). *SPEA2: Improving the strength Pareto evolutionary algorithm*. Eidgenssische Technische Hochschule Zrich (ETH), Institut fr Technische Informatik und Kommunikationsnetze (TIK).