Probabilistic Lexicase Selection

Li Ding
University of Massachusetts Amherst
Amherst, MA
liding@umass.edu

Edward Pantridge
Swoop
Cambridge, MA
ed@swoop.com

Lee Spector
Amherst College
University of Massachusetts Amherst
Amherst, MA
lspector@amherst.edu

ABSTRACT

Lexicase selection is a widely used parent selection algorithm in genetic programming, known for its success in various task domains such as program synthesis, symbolic regression, and machine learning. Due to its non-parametric and recursive nature, calculating the probability of each individual being selected by lexicase selection has been proven to be an NP-hard problem, which discourages deeper theoretical understanding and practical improvements to the algorithm. In this work, we introduce probabilistic lexicase selection (plexicase selection), a novel parent selection algorithm that efficiently approximates the probability distribution of lexicase selection. Our method not only demonstrates superior problem-solving capabilities as a semantic-aware selection method, but also benefits from having a probabilistic representation of the selection process for enhanced efficiency and flexibility. Experiments are conducted in two prevalent domains in genetic programming: program synthesis and symbolic regression, using standard benchmarks including PSB and SRBench. The empirical results show that plexicase selection achieves state-of-the-art problem-solving performance that is competitive to the lexicase selection, and significantly outperforms lexicase selection in computation efficiency.

CCS CONCEPTS

• **Mathematics of computing** → **Genetic programming**; *Probabilistic algorithms*; • **Computing methodologies** → *Supervised learning*.

KEYWORDS

genetic programming, evolutionary algorithms, parent selection, program synthesis, machine learning, symbolic regression

ACM Reference Format:
Li Ding, Edward Pantridge, and Lee Spector. 2023. Probabilistic Lexicase Selection. In *Genetic and Evolutionary Computation Conference (GECCO '23), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3583131.3590375

1 INTRODUCTION

Parent selection is an essential component in genetic and evolutionary algorithms, which determines a set of individuals to use as the source to create off-sprints for the next generation. Among many parametric and non-parametric selection methods, lexicase selection [28, 47] has shown state-of-the-art performance in genetic programming, and has been successfully extended to other problem domains such as symbolic regression [37, 38], machine learning [34], and deep learning [10].

As a semantic selection method, the lexicase selection algorithm[1] evaluates individuals on each single training case in random orders. One major drawback of lexicase selection is that the selection process remains a black-box due to its non-parametric and recursive nature. As a result, numerous repeated selection events are usually required when using lexicase selection, as opposed to more efficient probabilistic methods, *e.g.*, fitness proportionate selection [17, 39], that directly sampling the individuals from a probability distribution calculated from fitness measures.

While repetitions of lexicase selection events eventually produce a probability distribution of individuals to be selected, the underlying recursion introduces step-wise dependency and thus prevents a straightforward way to numerically calculate the probability distribution of individuals to be selected. Nonetheless, there has been rising interest in analyzing lexicase selection from a probabilistic point of view [32], which aims to develop a deeper theoretical understanding of lexicase selection. In fact, recent work [12] has proved that the exact calculation of the probabilities of individuals being selected in lexicase selection is nevertheless an NP-Hard problem, which implies the potential need for an approximation solution that would help develop better theoretical interpretation and practical improvement to lexicase selection.

With the above motivations in mind, we propose probabilistic lexicase selection (plexicase selection), which is, to our knowledge, the first semantic-aware selection method that has a probabilistic representation of selection events. Our method efficiently calculates an approximation to the probability distribution of individuals to be selected by lexicase selection, and samples individuals from this distribution instead of actually performing selection. There are two main advantages of the proposed method. Firstly, plexicase selection directly calculates the probability of individuals being selected instead of performing repeated selection events, which can reduce the algorithm runtime significantly. Secondly, having the probability distribution of individuals allow us to perform parametric optimization on the selection process. We introduce a probability manipulation process to plexicase selection, which uses a hyperparameter to control the kurtosis of the distribution of individuals, and thus manages to further enhance the problem-solving performance.

In order to illustrate the advantages of plexicase selection, we conduct experiments in two prevalent domains: program synthesis

GECCO '23, July 15–19, 2023, Lisbon, Portugal
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0119-1/23/07…$15.00
https://doi.org/10.1145/3583131.3590375

[1]For comprehensive insights into the lexicase selection algorithm, detailed descriptions can be found in Appendix Sec. A.

and symbolic regression. Results on standard benchmarks including the General Program Synthesis Benchmark Suite (PSB) [25] and Symbolic Regression Benchmark (SRBench) [36] suggest that plexicase selection significantly outperforms lexicase selection in terms of efficiency, and at the same time demonstrates superior or competitive problem-solving capabilities.

The paper is organized as follows. In Section 2, we summarize the background of lexicase selection and the challenges that motivate our work. Section 3 explains the theories and methodologies of the proposed algorithm in detail, and Section 4 describes the experiments and results in two domains where genetic programming has been extensively adopted. The results include various measures of problem-solving performance, algorithm runtime, and ablation studies on comparisons and hyperparameters. We conclude with a discussion of limitations and future work.

2 RELATED WORK

Lexicase selection [28, 47] assesses performance on individual training cases instead of employing aggregated accuracy or fitness metrics. It is a semantic-aware selection method [40] with the benefit of including semantic information regarding the population. In other words, lexicase selection tends to select specialist individuals for being elite on a subset of cases [23, 24], which enhances population diversity [21, 43]. Lexicase selection has been demonstrated more reliable performance than other contemporary genetic programming methods [14, 16, 31]. Recent work has empirically studied and extended lexicase selection to other supervised and unsupervised learning domains, *e.g.*, symbolic regression [38], machine learning [34, 35], rule-based learning [1], deep learning [8, 10], and evolutionary robotics [30, 33], which illustrates its capability of improving model performance and generalization.

Lexicase selection involves an essential procedure of gradually eliminating individuals by evaluating the population on each case. Recent work started to investigate the probability distribution of individuals selected by lexicase selection. Helmuth et al. [20] explored the correlation between the probability of selecting an individual and its rank in a population. La Cava et al. [32] derived the expected probabilities of lexicase selection and studied the effect of varying population and training set sizes. Dolson [12] further proved that the problem of calculating the exact probabilities of lexicase selection is NP-Hard, which inspired this work to alternatively design an approximation algorithm to obtain the probability distribution.

Probability-based selection methods have been studied for decades in genetic and evolutionary computation. The basic strategy is to assign higher selection probabilities to better-fitted individuals based on their aggregated fitness. The most common method in the category is fitness proportionate selection [17, 39], also known as roulette wheel selection, which assumes the probability of selection is proportional to the fitness of an individual. Extensions [2] to fitness proportionate explored using the ranking or ordering of individuals rather than their raw fitness. There have been some other methods with selection probabilities depending on fitness, *e.g.*, remainder stochastic independent sampling [3] and stochastic universal selection [18]. While these methods have slightly different statistical properties, none of them takes into account the program semantics [40], *i.e.*, the vector of outputs returned by a given program for a given set of fitness evaluations. To our knowledge, the proposed plexicase selection is the first semantic-aware selection method that has a probabilistic representation of the population, which has substantial advantages in both problem-solving performance and efficiency, as described in detail in Section 3.4.

From the perspective of algorithm efficiency, there have been a few attempts to reduce the runtime of lexicase selection. Helmuth et al. [19] theoretically analyzed the runtime of lexicase selection and found that the expected runtime depended on population diversity. De Melo et al. [5] introduced a hybrid selection method combining the idea of tournament and lexicase selection to improve both efficiency and quality of solutions. Aenugu and Spector [1] introduces a batched variant of lexicase selection, which assesses batched data samples instead of individual cases during selection events. Most recently, Ding et al. [6, 7] proposed a partial evaluation method based on weighted shuffle [48] to reduce the total number of evaluations. In general, most of the prior work focuses on optimizing the single selection event, but still requires repetitions of selection events to complete parent selection of the whole population. Our work takes a different direction: first, we calculate the probability distribution of individuals to be selected, and then we perform all the selection events at once by sampling from the distribution. Such an approach produces significant improvement in efficiency for up to 10 times speed-up, as shown in our empirical results on the program synthesis task.

Another popular trend in developing practical improvements to lexicase selection is to utilize the downsampling of training cases. Hernandez et al. [29] proposed downsampled lexicase selection and [13] studied it further. Recent work [26, 27] investigated the reasons behind the effectiveness of downsampling, and proposed a few hypotheses. Moore and Stanton [42] also uses downsampling with lexicase selection, but with different terminology. Boldi et al. [4] recently introduce informed downsampled lexicase selection, which leverages population statistics to build downsamples that contain more distinct and informative training cases. In this work, we demonstrate that the proposed plexicase selection method works well with existing downsampling terminologies, and the benefits of adding downsampling to plexicase selection are more significant compared to lexicase selection.

3 METHODS

In this section, we describe the proposed method in detail. The plexicase selection method has two main components: finding Pareto set boundaries and assigning probabilities to individuals. We start by reviewing the preliminaries on Pareto set boundaries and their connection with lexicase selection, followed by proposing an algorithm to efficiently obtain Pareto set boundaries through pairwise comparisons. We then introduce a method to assign probabilities to individuals, and finally present plexicase selection and its extension with ϵ-relaxation.

3.1 Preliminaries on Pareto Set Boundaries

The behavior of lexicase selection can be alternative interpreted as multi-objective optimization with respect to the training cases, where each training case is considered as a stand-alone objective. La Cava et al. [32] has proved that lexicase selection only selects

Pareto set boundaries, *i.e.*, individuals that are in the boundaries of the Pareto front. We first review the definitions related to Pareto set boundaries in the context of evolutionary computing.

Given a population $\mathcal{Y}$, $f_j(y_i)$ denotes the fitness (objective) function for the ith individual y_i on the jth case x_j in the training set $\mathbb{X}$ of size N. In this work, we assume the goal of optimization is to maximize the fitness function.

Definition 3.1. For individuals $y_1, y_2 \in \mathcal{Y}$, if $f_j(y_1) \geq f_j(y_2)\ \forall j \in \{1, \cdots, N\}$, we state $y_1 \succeq y_2$ (y_1 dominates y_2).

Definition 3.2. If $y_1 \succeq y_2$ and $\exists j \in \{1, \cdots, N\}$ for which $f_j(y_1) > f_j(y_2)$, we state $y_1 \succ y_2$ (y_1 strictly dominates y_2).

In parent selection, if two individuals have the exact same fitness on every training case, the selection algorithm can not distinguish them. In this work, we only consider the situation where the individuals have distinct overall fitness evaluations, *i.e.*, $\exists j \in \{1, \cdots, N\}$ for which $f_j(y_i) \neq f_j(y_k)\ \forall y_i, y_k \in \mathcal{Y}$. We show that the domination in this case can be replaced by its strict form.

Lemma 3.3. *If an individual is dominated by another individual, and they have distinct case-wise fitness, the domination is strict.*

Proof. Let $y_1 \succeq y_2$, we have $f_j(y_1) \geq f_j(y_2)\ \forall j \in \{1, \cdots, N\}$. If y_1 and y_2 have distinct case-wise fitness, we have $\exists k \in \{1, \cdots, N\}$ for which $f_k(y_1) \neq f_k(y_2)$. Thus, we have $f_k(y_1) > f_k(y_2)$. According to Def. 3.2, we can obtain $y_1 \succ y_2$. □

The Pareto set $\mathcal{P}$ of $\mathcal{N}$ is a subset ($\mathcal{P} \subseteq \mathcal{N}$) where each individual is non-dominated. We have the following definitions regarding Pareto set with respect to parent selection.

Definition 3.4. Given $y_i \in \mathcal{N}$, if $y_i \nprec y_k\ \forall y_k \in \mathcal{Y}$, $y_i \in \mathcal{P}$ where $\mathcal{P}$ is the Pareto set.

Definition 3.5. Given $y_i \in \mathcal{P}$, if $\exists j \in \{1, \cdots, N\}$ for which $f_j(y_i) \geq \max_{y_k \in \mathcal{Y}} f_j(y_k)$, y_i is a Pareto set *boundary*.[2]

La Cava et al. [32] proved the following, which appears as Theorem 3.4 in their paper:

Theorem 3.6. *If individuals from a population $\mathcal{Y}$ are selected by lexicase selection, those individuals are Pareto set boundaries of $\mathcal{Y}$ with respect to the training set $\mathbb{X}$.*

3.2 Obtaining Pareto Set Boundaries through Pairwise Comparisons

Suppose we have the global best fitness of x_j over population $\mathcal{Y}$ as $f^*_j(\mathcal{Y}) = \max_{y_i \in \mathcal{Y}} f_j(y_i)$, we show that an individual must achieve global best fitness on at least one case in order to be selected by lexicase selection.

Lemma 3.7. *If an individual $y_i \in \mathcal{Y}$ is selected by lexicase selection, then $\exists j \in \{1, \cdots, N\}$ for which $f_j(y_i) = f^*_j(\mathcal{Y})$.*

Proof. According to Theorem 3.6, y_i must be a Pareto set boundary, *i.e.*, $\exists j \in \{1, \cdots, N\}$ for which $f_j(y_i) \geq \max_{y_k \in \mathcal{Y}} = f^*_j(\mathcal{Y})$. □

[2]We follow the definition of Pareto set *boundary* in La Cava et al. [32]. Note that the Pareto set boundary in this work refers to the 'boundary' of the Pareto set, which is different from Pareto set (or Pareto boundary used in other work).

Algorithm 1: Find Pareto Set Boundaries through Pairwise Comparisons

```
Data:
    • $f(y_i)$: the fitness vector of individuals $y_i \in \mathcal{Y}$
    • $\mathcal{Y}$: the population of individuals
Result:
    • $\mathcal{P}$: a set of individuals that are Pareto set boundaries
$f^*(\mathcal{Y}) \leftarrow \max_i f(y_i)$
$\mathcal{P} \leftarrow \mathcal{Y}$ sorted by number of elitism $E(y_i)$
for $y_i$ in $\mathcal{P}$ do
    if $E(y_i) == 0$ then
        Remove $y_i$ from $\mathcal{P}$
    end
    else
        individuals_to_compare $\leftarrow \{y_k\}\ \forall E(y_k) \leq E(y_i)$.
        for $y_k$ in individuals_to_compare do
            if $\sum f(y_i) < f(y_k) = 0$ then
                Remove $y_k$ from $\mathcal{P}$
            end
        end
    end
end
return $\mathcal{P}$
```

Besides, we show that the number of elitism $E(y_i) = |\{f_j(y_i) = f^*_j(\mathcal{Y})\ \forall j \in \{1, \cdots, N\}\}|$, *i.e.*, the number of cases that an individual achieves global best fitness on, can be useful in determining dominance.

Theorem 3.8. *For two individuals $y_1, y_2 \in \mathcal{Y}$, if $E(y_1) > E(y_2)$, $y_1 \nprec y_2$.*

Proof. Given that $E(y_1) > E(y_2)$, there must $\exists j \in \{1, \cdots, N\}$ for which $f_j(y_1) = f^*_j(\mathcal{Y}) > f_j(y_2)$. According to Definition 3.1, we have $y_1 \nprec y_2$. □

We can also extend Theorem 3.6 to the case of pairwise comparison of two individuals.

Lemma 3.9. *Given two individuals $y_1, y_2 \in \mathcal{Y}$, y_2 can not be selected by lexicase selection if $y_1 \succ y_2$.*

Proof. If $y_1 \succ y_2$, y_2 is not in the Pareto set according to Definition 3.4. So y_2 can not be selected by lexicase selection according to Theorem 3.6. □

Combining Lemma 3.7 and Lemma 3.9, we show a way of determining whether an individual can be selected by lexicase selection through global best fitness and pairwise comparisons.

Theorem 3.10. *An individual $y_i \in \mathcal{Y}$ is a Pareto set boundary,* i.e., *can be selected by lexicase selection, only if the following two conditions are met:*

(1) $\exists j \in \{1, \cdots, N\}$ *for which* $f_j(y_i) = f^*_j(\mathcal{Y})$.
(2) $y_i \nprec y_k\ \forall y_k \in \mathcal{Y}$.

We hereby propose an efficient algorithm to find Pareto set boundaries in a population through vectorized pairwise comparisons. This step helps us identify individuals with zero and non-zero probabilities to be selected by lexicase selection.

The basic idea is to check each individual in the population if Theorem 3.10 holds. It is clear that if the $E(y_i) = 0$, y_i can not be selected by lexicase selection (due to the second condition). For the pairwise comparisons, since one individual may dominate multiple individuals, and we can remove the individual from the comparison pool once it is dominated (due to the first condition), it is better to start with the best individual and compare it against others. So we start by sorting the individuals by their number of elitism $E(y_i)$, as individuals with high elitism are more likely to dominate others. Another benefit of sorting is to reduce the number of comparisons. As shown in Theorem 3.8, if $E(y_1) > E(y_2)$, $y_1 \nprec y_2$. So starting from the first individual in the list sorted by $E(y_i)$ from high to low, we only need to compare the current individual with individuals that have equal or lower $E(y_i)$.

The comparisons are performed in a vectorized fashion. Given the fitness vector $f(y_i) = [f_1(y_i), f_2(y_i), \cdots, f_N(y_i)]^T$, if for any case j we have $f_j(y_1) \geq f_j(y_2)$, then we can infer that $y_1 \succeq y_2$. We use vectorized comparison to check if $\sum f(y_1) < f(y_2) = 0$. Since the fitness vectors are all distinct after preselection, according to Lemma 3.3, we have $y_1 \succ y_2$.

During the iterations, if an individual is found to be dominated, we record that and also remove it from the sorted list. This is because dominance follows transitive property, *i.e.*, if y_i is dominated by y_k, any of its dominants are also dominated by y_k, so removing y_i will avoid unnecessary comparisons.

3.3 Assigning Probabilities to Individuals

After getting the Pareto set boundaries, we already know which individuals will and will not be selected by lexicase selection, and we can assign a selection probability of zero to individuals that are not in the Pareto set boundaries. The next step is to determine the selection probabilities for Pareto set boundaries.

It has been proved that calculating the exact probabilities is an NP-Hard problem [12]. So here we propose an efficient algorithm to alternatively approximate the probabilities. Our hypothesis for the approximation is: for any training case, suppose there are k individuals being elites in that case, the probability of which individual to be selected is based on their total elitism, *i.e.*, the number of best fitness achieved. The intuition is that when we look at the training first case in the sequence of a lexicase selection event, suppose k individuals tie on this training case, then instead of going to the next case like normal lexicase selection, we assume that whichever case has a higher chance to be the best on the next case is more likely to be finally selected. After computing the distribution for each individual on each training case, we average over all the cases to get the final distribution.

More formally, the unnormalized probability density function for an individual y_i in the population $\mathcal{Y}$ on the jth training case is given by

$$h_j(y_i) = \begin{cases} E(y_i) & \text{if } f_j(y_i) = f_j^*(\mathcal{Y}) \\ 0 & \text{otherwise} \end{cases}. \tag{1}$$

Algorithm 2: Probabilistic Lexicase Selection

```
Data:
    • $f(y_i)$: the fitness vector of individuals $y_i \in \mathcal{Y}$
    • $\mathcal{Y}$: the population of individuals
Result:
    • $Pa$: individuals that are selected as parents
$\mathcal{P} \leftarrow$ Find Pareto Set Boundaries (Algorithm 1)
for $y_i$ in $\mathcal{Y}$ do
    if $y_i \in \mathcal{P}$ then
        Calculate $P(y_i)$ (Equation 1,2,3)
    end
    else
        $P(y_i) = 0$
    end
end
$Pa \leftarrow$ sample parents from the calculated distribution $P(\mathcal{Y})$
return $Pa$
```

The probability distribution of selecting individuals on the jth training case is

$$P_j(y_i) = \frac{h_j(y_i)}{\sum_{y_k \in \mathcal{Y}} h_j(y_k)}. \tag{2}$$

Finally, the probability distribution of selecting individuals is averaged over all the training cases,

$$P(y_i) = \frac{\sum_{j=1}^{N} P_j(y_i)}{N}, \tag{3}$$

which is used to assign probabilities in the proposed plexicase selection.

3.4 Probabilistic Lexicase Selection

With aforementioned definitions in mind, we introduce probabilistic lexicase selection (plexicase selection). In general, plexicase selection first find the Pareto set boundaries through pairwise comparisons, then assign probabilities to each individual to form the probability distribution of selection. Finally, the parent selection is performed by sampling from the generated distribution. The complete algorithm is outlined in Algorithm 2.

There are two major benefits of our method, which samples parents from a calculated distribution instead of running repeated selection events like the original lexicase selection. First, once the probabilities are calculated, sampling can be efficiently performed to obtain numerous parents from the distribution. The worst-case runtime of our method is the same as running lexicase selection for just one selection event. In practice, we observe that our method is significantly faster than lexicase selection in various tasks. Secondly, having a probabilistic representation of the individuals being selected can provide us with flexibility to further apply parametric operations to improve the performance.

In particular, we propose a probability manipulation strategy to control the kurtosis (tailedness) of the selection distribution, *i.e.*, the randomness in multiple selection events. We introduce a hyperparameter $\alpha \geq 0$, which acts similarly to a temperature parameter in the Softmax function. After calculating the probability of selecting each individual $P(y_i)$, we perform the following probability

Table 1: Results on program synthesis benchmark problems. We report the number of successes over 100 runs. With fixed $\alpha = 1$, plexicase selection is competitive to lexicase selection on all the problems. With probability manipulation (selecting the best $\alpha \in \{0.5, 1, 2\}$), plexicase selection outperforms lexicase selection on some problems. We also perform Pearson's Chi-squared tests to compare the number of successes of plexicase selection against lexicase selection, and the results are marked with significance levels ($P \leq 0.05^{*}$, 0.01^{**}, 0.001^{***}).

Problem	Regular			Downsampled		
	Lexicase	Plexicase ($\alpha = 1$)	Plexicase	Lexicase	Plexicase ($\alpha = 1$)	Plexicase
compare-string-lengths	0	0	0	0	0	0
median	89	83	**96**	70	88^{**}	$\mathbf{98}^{***}$
negative-to-zero	79	80	**85**	76	90^{*}	$\mathbf{94}^{**}$
number-io	99	**100**	**100**	99	**100**	**100**
replace-space-with-newline	10	6	**13**	11	7	**12**
smallest	100	100	100	**100**	99	**100**
vector-average	**100**	99	**100**	100	100	100

manipulation:

$$P'(y_i) = \frac{P(y_i)^{\alpha}}{\sum_{y_i \in \mathcal{Y}} P(y_i)^{\alpha}} \tag{4}$$

where $P'(y_i)$ is the new probability distribution to be used in plexicase selection.

Intuitively, α controls the difference in probabilities of each individual to be selected. When $\alpha = 1$, there is no change in the distribution. When α is small, the probabilities are more evenly distributed, (*e.g.*, at the extreme case when $\alpha = 0$, all the individuals that are Pareto set boundaries will be selected uniformly at random), whereas when α is large, the probabilities are more skewed towards the elites.

Note that such manipulation does not change the probability of individuals with an initial probability of zero, so it keeps the basic terminology of selection that only Pareto set boundaries will be selected, as stated in Theorem 3.6.

3.5 Extension with ϵ-Relaxation

The proposed plexicase selection can be easily extended to its ϵ-relaxed form, namely ϵ-plexicase selection, to handle tasks where the fitness measure is in the continuous space. In particular, ϵ refers to the dynamic relaxation on elitism proposed in prior work [38]. We briefly describe the key modifications needed for the extension to ϵ-plexicase selection.

Firstly, most of the definitions in Section 3.2 can be extended to their ϵ-relaxed forms. In general, we have the following definition regarding ϵ-domination:

Definition 3.11. For individuals $y_1, y_2 \in \mathcal{Y}$, if $f_j(y_1) - \epsilon \geq f_j(y_2)\ \forall j \in \{1, \cdots, N\}$, we state $y_1 \succeq y_2$ (y_1 ϵ-dominates y_2).

Similarly, Algorithm 1 can be extended to find ϵ-relaxed Pareto set boundaries if we use ϵ-domination to address elitism and domination. By replacing Algorithm 1 with its ϵ-relaxed form in Algorithm 2, we get the ϵ-relaxed form of plexicase selection, *i.e.*, ϵ-plexicase selection.

4 EXPERIMENTS

We conduct experiments to validate the proposed method in two domains: program synthesis and symbolic regression, in which lexicase selection has been demonstrated to be a state-of-the-art approach. In this section, we describe the implementation details and experimental results.

4.1 Datasets and Benchmarks

For program synthesis, we use a sample of problems taken from the General Program Synthesis Benchmark Suite [25], which is a standard benchmark for testing GP systems for program synthesis. The problems contained in this suite were sourced from introductory computer science textbooks and programming competitions. This aligns the complexity of the problems with the average skill of a beginner human programmer. Solutions to these problems require the use of various data types, control flow, and basic data structures. We have selected the same subset of 7 problems used in [45] out of the full PSB suite in order to compare the change in performance when using plexicase. State-of-the-art solution rates to these problems for PushGP [27], Grammar Guided Genetic Programming [15], and Code Building GP [44] show that this subset of the PSB covers a range of difficulties regardless of the GP method.

Another domain that receives rising interest from the GP and ML community is symbolic regression. We adopt the recently proposed SRBench [36], which is a large-scale, open-source, reproducible benchmarking platform for symbolic regression. In particular, we use a subset of 20 black-box regression problems with sample sizes varying from 40 to 200. These regression problems consist of both real-world problems (data obtained from physical observations) and synthetic problems (data generated from static functions or simulations), covering diverse domains such as health informatics, business, environmental science, and economics.

4.2 Implementation Details

For program synthesis, we implement our algorithm in the Code Building Genetic Programming (CBGP) framework [45]. CBGP is a general program synthesis system that works with arbitrary data

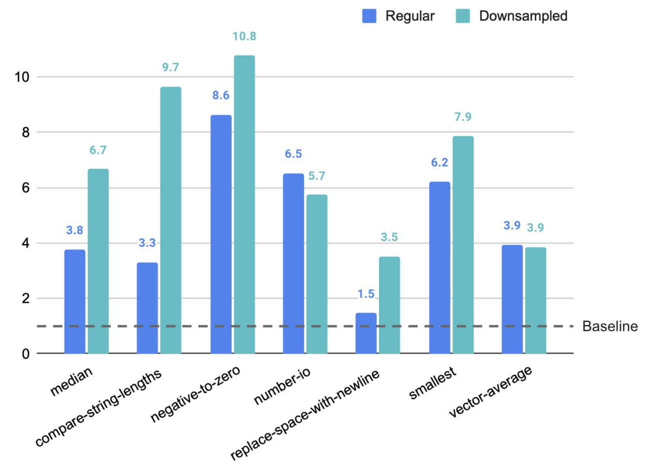

Figure 1: Average runtime speed-up of plexicase selection compared to lexicase selection in paired comparisons. A speed-up of n means the plexicase selection runtime is $1/n$ of the lexicase selection runtime. The comparisons are conducted in both regular and downsampled circumstances.

types and structures, and shows competitive problem-solving performance. The original CBGP system uses lexicase selection as the default parent selection method, and we compare it with plexicase selection. We follow the original CBGP paper [45] to set the system configurations: for each problem, we perform 100 runs of the system with a population size of 1000 for 300 generations; the variation operator used in all the runs is UMAD [22]. For plexicase selection, we use three different $\alpha = 0.5, 1, 2$ for probability manipulation and report the best result for each problem out of the three configurations.

For symbolic regression, we start with the *gplearn*[1] framework, which originally uses tournament selection. Since symbolic regression problems need to optimize with continuous values, we implement both ϵ-lexicase selection and ϵ-plexicase selection, and compare them against the tournament selection baseline. For tournament and ϵ-lexicase selection, we adopt the same configurations as in SRBench [36], which are 6 combinations of hyperparameters with 3 population sizes of 1000, 500, 100 (the according numbers of generations are 500, 1000, 5000) and 2 function sets. The tournament size is 20 for all the problems. For ϵ-plexicase selection, we include different α values (0.5, 1, and 2) as hyperparameters, but we limit the number of combinations to six for fair comparisons with other methods. Each method is tested for 10 repeated trials on each of the 20 problem (200 trials in total) with different random seeds controlling both the train/test split and the initialization of the algorithm.

4.3 Results on Program Synthesis

4.3.1 Problem-solving performance. Table 1 shows the problem-solving performance of plexicase selection compared to the regular lexicase selection. We report the number of successes over 100 runs. With fixed $\alpha = 1$, plexicase is competitive to lexicase on all the problems. With probability manipulation (selecting the best $\alpha \in \{0.5, 1, 2\}$), our method is able to outperform lexicase on some problems. We also perform Pearson's Chi-squared tests [46] to compare plexicase selection against lexicase selection, showing that there is no significant difference in problem-solving performance between regular lexicase and plexicase selection.

[1]https://github.com/trevorstephens/gplearn

Table 2: Ablation study on different α values for probability manipulation of plexicase selection on program synthesis benchmark problems. We report the number of successes over 100 runs.

Problem	Plexicase (fixed α)		
	$\alpha = 0.5$	$\alpha = 1$	$\alpha = 2$
compare-string-lengths	0	0	0
median	61	83	**96**
negative-to-zero	**85**	80	70
number-io	99	**100**	**100**
replace-space-with-newline	4	6	**13**
smallest	98	**100**	**100**
vector-average	**100**	99	**100**

Recent work [27, 29] has demonstrated that the advantages of lexicase selection can be potentially amplified by downsampling. Following this trend, we also test the algorithms with a downsampling rate of 0.25. The downsampling is performed before each generation to use only a subset of training cases. We observe that plexicase selection benefits more from downsampling, and is able to significantly outperform lexicase selection on 2 problems with downsampling. Such results indicate that plexicase is likely to work well on larger-scale problems with downsampling.

4.3.2 Runtime analysis. Another important factor we care about is efficiency. In order to fairly compare the runtime of plexicase and lexicase selection, we design a paired-comparison experiment. During each generation, we run both lexicase selection and plexicase selection on the same population with the same training data, and record the runtime just for each selection algorithm.

Figure 1 shows the average runtime speed-up of plexicase selection to lexicase selection in paired comparisons. A speed-up of n means the runtime is $1/n$ of lexicase runtime. We can see that plexicase achieves significant speed-up on all the problems. With downsampling, the speed-up is further enhanced on most of the problems. Such results indicate that our method is more efficient than lexicase selection on the benchmark program synthesis problems, and the advantage is further enhanced with downsampling.

4.3.3 Effect of probability manipulation. As an ablation study, we report the detailed results of plexicase selection with different α values for probability manipulation. As shown in Table 2, for some problems, *e.g.*, `median` and `replace-space-with-newline`, larger α value gives better results, but for other problems like `negative-to-zero`, small α works better. These observations indicate that different problems may favor different levels of kurtosis of the individual distribution. In other words, some problems require

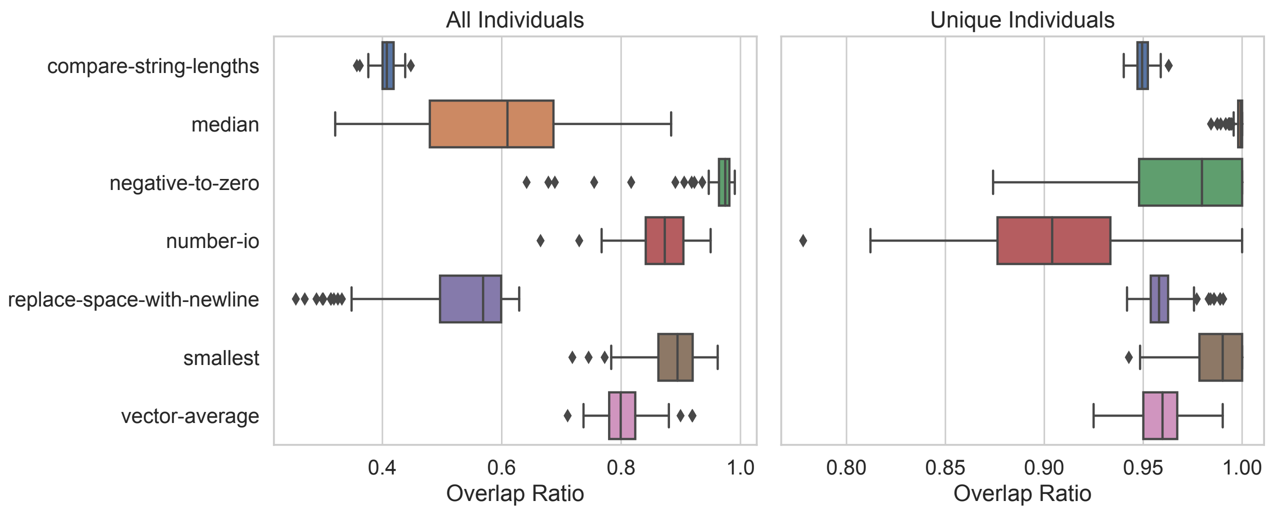

Figure 2: Overlap between individuals selected by plexicase selection and those selected by lexicase selection. The ratios are calculated for all individuals as well as unique individuals.

more exploitation with good solutions, others need more exploration with a large diversity of solutions. While lexicase selection does not have control over the trade-off between exploration and exploitation, our method manages to overcome this disadvantage and demonstrates better performance.

4.3.4 Selection similarity. In order to validate how well plexicase selection is approximating the probabilities of selecting individuals with lexicase selection, we perform an ablation study on the similarity of individuals selected by two algorithms. The similarity is measured as the overlap ratio of individuals selected by both algorithms over those selected by lexicase selection. The overlap is separately calculated over all individuals as well as just unique individuals. We report the distribution of mean overlap ratios over generations for all runs, as summarized in Figure 2. We can see that for 4 problems, the overlap ratios of all individuals are above 0.8, indicating that plexicase is approximating the distribution with considerable overlap. There also exists some variances of overlap ratios among different problems, which suggests that the approximation performance may be problem-dependent.

Similar observations have been found in recent work [19] that lexicase selection produces varying population diversity on different problems. However, the overlap ratios of unique individuals are almost always above 0.9, indicating that most of the individuals selected by lexicase selection have been selected at least once by plexicase selection. This result validates the theoretical correctness of plexicase selection in obtaining the Pareto set boundaries as individuals that can be selected by lexicase selection.

4.4 Results on Symbolic Regression

4.4.1 Performance on black-box regression problems. We assess the performance of our method on symbolic regression problems in terms of both accuracy and complexity. For accuracy, we use the coefficient of determination, *i.e.*, R^2, which is defined as

$$R^2 = \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}. \tag{5}$$

For complexity, we follow SRBench and calculate the number of mathematical operators, features, and constants in the model.

The median performance over all 20 black-box benchmark problems is summarized in Figure 3. For each problem, we take the median performance over 10 repeated trials with different random seeds, following SRBench's recommended evaluation setting. We can see that using probability manipulation, ϵ-plexicase selection slightly outperforms ϵ-lexicase selection in regression accuracy, with significantly less model size and training time.

We also perform a Mood's median test [41] comparing the R^2 values of ϵ-plexicase and ϵ-lexicase selection. The resulting p-value is 0.8256, indicating that there is no significant difference in performance between the two algorithms, *i.e.*, ϵ-plexicase well approximates the problem-solving capability of ϵ-lexicase selection. In addition, the performance of ϵ-plexicase selection may be further improved with more budget on hyperparameter tuning to compensate for the reduction in training time.

The results also show that ϵ-plexicase selection has less variance in performance across different problems, compared to ϵ-lexicase selection, indicating that our method is more stable and robust in solving the symbolic regression task in general. It is worth noting that the variances are inevitable since SRBench aggregates many ML problems and the problems are with varying difficulties.

4.4.2 Runtime analysis. Figure 3 shows a significant reduction in the overall training time of ϵ-plexicase selection compared to ϵ-lexicase selection. It should be noted that training time includes other processes, such as evaluations of individuals, which is equal for both algorithms. This suggests that the improvement in the runtime of the selection process alone is substantial.

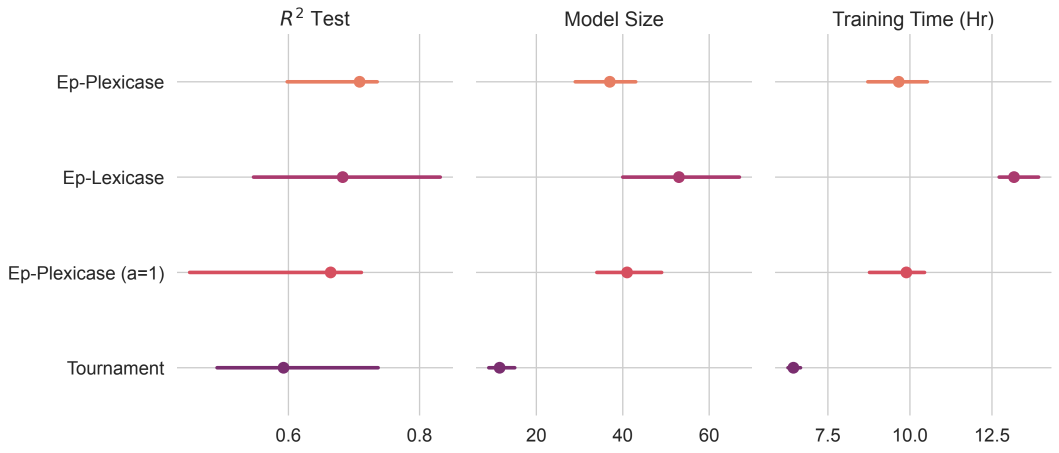

Figure 3: Testing performance on the black-box benchmark problems. Each algorithm is evaluated for 200 trials in total. Points indicate the median values of each metric among all 20 problems, where for each problem we take the median value over 10 repeated trials. The bars show the 0.95 confidence interval obtained from bootstrapping.

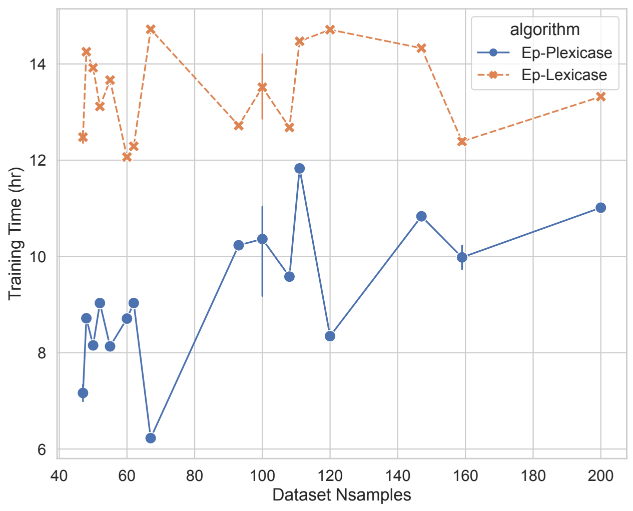

Figure 4: Runtime comparison between ϵ-plexicase selection and ϵ-lexicase selection on symbolic regression problems.

To further investigate the patterns of runtime improvement with ϵ-plexicase selection, we plot the runtime of both methods separately for each problem, characterized by the total number of samples, as shown in Figure 4. The results indicate that ϵ-plexicase selection consistently outperforms lexicase selection in terms of runtime across all problems, irrespective of the dataset size. This finding highlights the efficiency of our approach in solving the symbolic regression tasks.

5 CONCLUSION AND FUTURE WORK

In this work, we introduce a novel parent selection method called Probabilistic Lexicase Selection, which efficiently approximates the probability distribution of lexicase selection. The proposed method not only demonstrates superior problem-solving capabilities as a semantic-aware selection method, but also benefits from having a probabilistic representation of selection for enhanced efficiency and flexibility. Specifically, the probability distribution enables us to efficiently sample numerous parents at once instead of performing repeated selection events. We also introduce a probability manipulation method to further enhance the problem-solving performance of plexicase selection by controlling the kurtosis of distribution.

To validate the proposed method, we conducted experiments in two prevalent task domains: program synthesis and symbolic regression. The empirical results on standard benchmarks demonstrate that our plexicase selection algorithm achieves state-of-the-art problem-solving performance comparable to lexicase selection while also surpassing lexicase selection in terms of computation efficiency. Ablation studies further investigate the effect of hyperparameters and validate the correctness of the proposed method.

One limitation of this work is that although we tested our method on two popular domains for genetic programming, the benchmark problems used in the experiments are relatively small-scale compared to modern machine learning tasks. In future work, we plan to extend plexicase selection to solve larger-scale tasks such as evolutionary optimization for deep neural networks [10] and parameterized quantum computing models [9, 11].

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 2117377. 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 was performed in part using high performance computing equipment obtained from the Collaborative R&D Fund managed by the Massachusetts Technology Collaborative. The authors would like to thank Thomas Helmuth, Ryan Boldi, and Anil Saini for their helpful comments and suggestions.

REFERENCES

[1] Sneha Aenugu and Lee Spector. 2019. Lexicase selection in learning classifier systems. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 356–364.

[2] James Edward Baker. 2014. Adaptive selection methods for genetic algorithms. In *Proceedings of the first international conference on genetic algorithms and their applications*. Psychology Press, 101–106.

[3] James E Baker et al. 1987. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the second international conference on genetic algorithms*, Vol. 206. 14–21.

[4] Ryan Boldi, Martin Briesch, Dominik Sobania, Alexander Lalejini, Thomas Helmuth, Franz Rothlauf, Charles Ofria, and Lee Spector. 2023. Informed Down-Sampled Lexicase Selection: Identifying productive training cases for efficient problem solving. *arXiv preprint arXiv:2301.01488* (2023).

[5] Vinícius V De Melo, Danilo Vasconcellos Vargas, and Wolfgang Banzhaf. 2019. Batch tournament selection for genetic programming: the quality of lexicase, the speed of tournament. In *Proceedings of the genetic and evolutionary computation conference*. 994–1002.

[6] Li Ding, Ryan Boldi, Thomas Helmuth, and Lee Spector. 2022. Going faster and hence further with lexicase selection. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 538–541.

[7] Li Ding, Ryan Boldi, Thomas Helmuth, and Lee Spector. 2022. Lexicase selection at scale. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 2054–2062.

[8] Li Ding and Lee Spector. 2021. Evolving neural selection with adaptive regularization. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 1717–1725.

[9] Li Ding and Lee Spector. 2022. Evolutionary quantum architecture search for parametrized quantum circuits. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 2190–2195.

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

[11] Li Ding and Lee Spector. 2023. Multi-Objective Evolutionary Architecture Search for Parameterized Quantum Circuits. *Entropy* 25, 1 (2023), 93.

[12] Emily Dolson. 2023. Calculating lexicase selection probabilities is NP-Hard. *arXiv preprint arXiv:2301.06724* (2023).

[13] Austin J. Ferguson, Jose Guadalupe Hernandez, Daniel Junghans, Alexander Lalejini, Emily Dolson, and Charles Ofria. 2019. Characterizing the effects of random subsampling and dilution on Lexicase selection, In Genetic Programming Theory and Practice XVII, Wolfgang Banzhaf, Erik Goodman, Leigh Sheneman, Leonardo Trujillo, and Bill Worzel (Eds.). *Genetic Programming Theory and Practice XVII*.

[14] Jonathan E Fieldsend and Alberto Moraglio. 2015. Strength through diversity: Disaggregation and multi-objectivisation approaches for genetic programming. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. 1031–1038.

[15] Stefan Forstenlechner, David Fagan, Miguel Nicolau, and Michael O'Neill. 2018. Extending program synthesis grammars for grammar-guided genetic programming. In *Parallel Problem Solving from Nature–PPSN XV: 15th International Conference, Coimbra, Portugal, September 8–12, 2018, Proceedings, Part I 15*. Springer, 197–208.

[16] Edgar Galvan-Lopez, Brendan Cody-Kenny, Leonardo Trujillo, and Ahmed Kattan. 2013. Using semantics in the selection mechanism in genetic programming: a simple method for promoting semantic diversity. In *2013 IEEE Congress on Evolutionary Computation*. IEEE, 2972–2979.

[17] David E Golberg. 1989. Genetic algorithms in search, optimization, and machine learning. *Addion wesley* 1989, 102 (1989), 36.

[18] David E Goldberg and Kalyanmoy Deb. 1991. A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of genetic algorithms*. Vol. 1. Elsevier, 69–93.

[19] Thomas Helmuth, Johannes Lengler, and William La Cava. 2022. Population Diversity Leads to Short Running Times of Lexicase Selection. In *Parallel Problem Solving from Nature–PPSN XVII: 17th International Conference, PPSN 2022, Dortmund, Germany, September 10–14, 2022, Proceedings, Part II*. Springer, 485–498.

[20] Thomas Helmuth, Nicholas Freitag McPhee, and Lee Spector. 2016. The impact of hyperselection on lexicase selection. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*. 717–724.

[21] Thomas Helmuth, Nicholas Freitag McPhee, and Lee Spector. 2016. Lexicase selection for program synthesis: a diversity analysis. In *Genetic Programming Theory and Practice XIII*. Springer, 151–167.

[22] Thomas Helmuth, Nicholas Freitag McPhee, and Lee Spector. 2018. Program synthesis using uniform mutation by addition and deletion. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 1127–1134.

[23] Thomas Helmuth, Edward Pantridge, and Lee Spector. 2019. Lexicase Selection of Specialists. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Prague, Czech Republic) *(GECCO '19)*. Association for Computing Machinery, New York, NY, USA, 1030–1038. https://doi.org/10.1145/3321707.3321875

[24] Thomas Helmuth, Edward Pantridge, and Lee Spector. 2020. On the Importance of Specialists for Lexicase Selection. *Genetic Programming and Evolvable Machines* 21, 3 (sep 2020), 349–373. https://doi.org/10.1007/s10710-020-09377-2

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

[26] Thomas Helmuth and Lee Spector. 2020. Explaining and exploiting the advantages of down-sampled lexicase selection. In *ALIFE 2020: The 2020 Conference on Artificial Life*. MIT Press, 341–349.

[27] Thomas Helmuth and Lee Spector. 2022. Problem-solving benefits of down-sampled lexicase selection. *Artificial Life* 27, 3–4 (2022), 183–203.

[28] Thomas Helmuth, Lee Spector, and James Matheson. 2014. Solving uncompromising problems with lexicase selection. *IEEE Transactions on Evolutionary Computation* 19, 5 (2014), 630–643.

[29] Jose Guadalupe Hernandez, Alexander Lalejini, Emily Dolson, and Charles Ofria. 2019. Random subsampling improves performance in lexicase selection. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 2028–2031.

[30] Joost Huizinga and Jeff Clune. 2018. Evolving multimodal robot behavior via many stepping stones with the combinatorial multi-objective evolutionary algorithm. *arXiv preprint arXiv:1807.03392* (2018).

[31] Krzysztof Krawiec and Paweł Liskowski. 2015. Automatic derivation of search objectives for test-based genetic programming. In *European Conference on Genetic Programming*. Springer, 53–65.

[32] William La Cava, Thomas Helmuth, Lee Spector, and Jason H Moore. 2019. A probabilistic and multi-objective analysis of lexicase selection and ε-lexicase selection. *Evolutionary Computation* 27, 3 (2019), 377–402.

[33] William La Cava and Jason Moore. 2018. Behavioral search drivers and the role of elitism in soft robotics. In *ALIFE 2018: The 2018 Conference on Artificial Life*. MIT Press, 206–213.

[34] William La Cava and Jason H Moore. 2020. Genetic programming approaches to learning fair classifiers. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. 967–975.

[35] William La Cava and Jason H Moore. 2020. Learning feature spaces for regression with genetic programming. *Genetic Programming and Evolvable Machines* 21, 3 (2020), 433–467.

[36] William La Cava, Patryk Orzechowski, Bogdan Burlacu, Fabrício Olivetti de França, Marco Virgolin, Ying Jin, Michael Kommenda, and Jason H Moore. 2021. Contemporary symbolic regression methods and their relative performance. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, Vol. 1.

[37] William La Cava, Tilak Raj Singh, James Taggart, Srinivas Suri, and Jason H Moore. 2018. Learning concise representations for regression by evolving networks of trees. *arXiv preprint arXiv:1807.00981* (2018).

[38] William La Cava, Lee Spector, and Kourosh Danai. 2016. Epsilon-lexicase selection for regression. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*. 741–748.

[39] Adam Lipowski and Dorota Lipowska. 2012. Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications* 391, 6 (2012), 2193–2196.

[40] Pawel Liskowski, Krzysztof Krawiec, Thomas Helmuth, and Lee Spector. 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*. 1301–1307.

[41] Alexander McFarlane Mood. 1950. Introduction to the Theory of Statistics. (1950).

[42] Jared M Moore and Adam Stanton. 2017. Lexicase selection outperforms previous strategies for incremental evolution of virtual creature controllers. In *ECAL 2017, the Fourteenth European Conference on Artificial Life*. MIT Press, 290–297.

[43] Jared M Moore and Adam Stanton. 2018. Tiebreaks and Diversity: Isolating Effects in Lexicase Selection. , 590–597 pages. https://doi.org/10.1162/isal_a_00109

[44] Edward Pantridge, Thomas Helmuth, and Lee Spector. 2022. Functional code building genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 1000–1008.

[45] Edward Pantridge and Lee Spector. 2020. Code building genetic programming. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. 994–1002.

[46] Karl Pearson. 1900. X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 50, 302 (1900), 157–175.

[47] Lee Spector. 2012. Assessment of problem modality by differential performance of lexicase selection in genetic programming: a preliminary report. In *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*. 401–408.

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