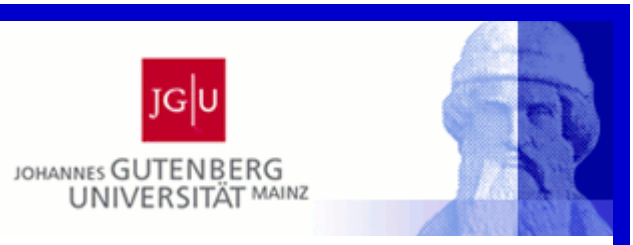

Gutenberg School of Management and Economics

& Research Unit “Interdisciplinary Public Policy”

Discussion Paper Series

A Comprehensive Survey on Lexicase Selection

Alina Geiger, Dirk Schweim, Martin Briesch, David Wittenberg, Franz Rothlauf, Dominik Sobania

June 2026

Discussion paper number 2605

Johannes Gutenberg University Mainz
Gutenberg School of Management and Economics
Jakob-Welder-Weg 9
55128 Mainz
Germany
https://wiwi.uni-mainz.de/

Contact details

Alina Geiger
Information Systems Group
Johannes Gutenberg University
55099 Mainz
Germany
geiger@uni-mainz.de

Dirk Schweim
School of Business
Mainz University of Applied Sciences
55128 Mainz
Germany
schweim@hs-mainz.de

Martin Briesch
Information Systems Group
Johannes Gutenberg University
55099 Mainz
Germany
briesch@uni-mainz.de

David Wittenberg
Former member of the Information Systems Group
Johannes Gutenberg University
55099 Mainz
Germany
wittenberg@uni-mainz.de

Franz Rothlauf
Information Systems Group
Johannes Gutenberg University
55099 Mainz
Germany
rothlauf@uni-mainz.de

Dominik Sobania
Generative Artificial Intelligence in Software Engineering Group
University of Duisburg-Essen
45127 Essen
Germany
Dominik.sobania@uni-due.de

A Comprehensive Survey on Lexicase Selection

Alina Geiger, Dirk Schweim, Martin Briesch, David Wittenberg, Franz Rothlauf, Dominik Sobania

***Abstract*—In recent years, rapid progress has been made in the field of lexicase selection with the development of many new variants, expanding its applicability across various domains. In order to consolidate this progress, we conduct a systematic survey on the current state of research on lexicase selection. Our analysis includes about 300 in-scope papers, making it a comprehensive overview of current literature in this field. We identify five major research streams, namely lexicase, ϵ-lexicase, batch-lexicase, down-sampled lexicase as well as hybrids and approximations. For each research stream, we highlight and discuss the main empirical and theoretical contributions. Further, we identify the major application domains of lexicase, ranging from program synthesis and symbolic regression to classification, Boolean logic, and even robot control. Based on our findings, we discuss current challenges and identify opportunities for future research directions in evolutionary computation and beyond.**

***Index Terms*—Lexicase, Selection, Survey.**

I. Introduction

Selection plays a crucial role in evolutionary computation [1] and many selection operators have been proposed over the years, like fitness-proportionate selection, tournament selection, and, last but not least, lexicase selection [2]. Classical methods like tournament selection operate on an aggregated fitness value, which is calculated by averaging the performance of a candidate solution across all training cases (input-output pairs). However, this creates an evaluation bottleneck where all the information about a candidate solution is compressed into a single value [3], [4]. This poses a dilemma, as problems of interest often exhibit structures that require different modular capabilities from a solution [2]. Spector [2] discusses the example of a computer program that should serve as a mathematical calculator and, therefore, must be able to perform very different operations such as addition, subtraction, and multiplication correctly. Discovering all those operations simultaneously during evolution is highly unlikely. In contrast, evolving a candidate solution that solves one operation at a time may be possible. With an evaluation bottleneck, however, such candidate solutions that excel in a particular modular part (e.g., addition) but perform poorly on all other modular parts will be filtered out by selection and their genetic material might be lost [5].

In contrast to those selection operators, lexicase selection does not operate on an aggregated fitness value but instead evaluates candidate solutions based on their performance on individual training cases [2]. More specifically, for each selection event, all training cases are first shuffled in random order. All candidate solutions are then evaluated on a single training case, only preserving those individuals with the best performance on this case. This is repeated for the next training case until only one candidate solution remains. If more than one candidate solution remains after all training cases have been used for evaluation, one of the remaining candidates is chosen either at random or according to some other metric [6].

This approach not only increases diversity [7], [8] but also promotes the selection of *specialist* solutions that perform exceptionally well on training cases where others perform poorly while performing worse on average. These specialists would rarely be selected with methods based on aggregated fitness values. Specialists stand in contrast to *generalist* solutions, which perform well on average but do not excel on any particular training case compared to the rest of the population [9], [10]. Specialists can serve as stepping stones for the evolutionary search and may contribute genetic material that leads to candidate solutions capable of solving all parts of the problem [7], [10]–[13]. In the mathematical calculator example, lexicase selection may discover specialists for individual mathematical operations, which can guide the search towards a generalist solution that successfully performs all required operations [2].

Lexicase selection has demonstrated impressive performance across a variety of application domains [14]–[18], leading to the development of many different variations and extensions in the literature. Nevertheless, many researchers only use outdated or inconsistent versions of lexicase selection without accounting for recent advancements. As a result, there is a need for a comprehensive overview that brings together current knowledge, clarifies the differences among lexicase variants, and highlights its practical applications.

Therefore, in this work, we conduct a systematic literature review to identify, consolidate, and structure the current state of research on lexicase selection. We identify almost 300 papers published between 2012 and 2025 that either investigated lexicase selection or used it as part of their experiments. From these papers, we derive five major research streams: First, lexicase as a selection method for modular problem settings (Section III). Second, ϵ-lexicase selection for continuous-valued problems (Section IV). Next, the combination of lexicase with batching (Section V) and down-sampling techniques (Section VI). Lastly, methods that approximate lexicase selection as well as hybrid approaches (Section VII). Moreover, we find that the main application domains of lexicase selection are program synthesis and symbolic regression. Additional applications include classification, Boolean logic, robotics, and others. We also identify applications of lexicase selection

Alina Geiger, Martin Briesch, and Franz Rothlauf are with the Information Systems Group, Johannes Gutenberg University, 55099 Mainz, Germany (e-mail: geiger@uni-mainz.de; briesch@uni-mainz.de; rothlauf@uni-mainz.de).

Dirk Schweim is with the School of Business, Mainz University of Applied Sciences, 55128 Mainz, Germany (e-mail: schweim@hs-mainz.de).

David Wittenberg is a former member of the Information Systems Group, Johannes Gutenberg University, 55099 Mainz, Germany (e-mail: wittenberg@uni-mainz.de).

Dominik Sobania is with the Generative Artificial Intelligence in Software Engineering Group, University of Duisburg-Essen, 45127 Essen, Germany (e-mail: dominik.sobania@uni-due.de).

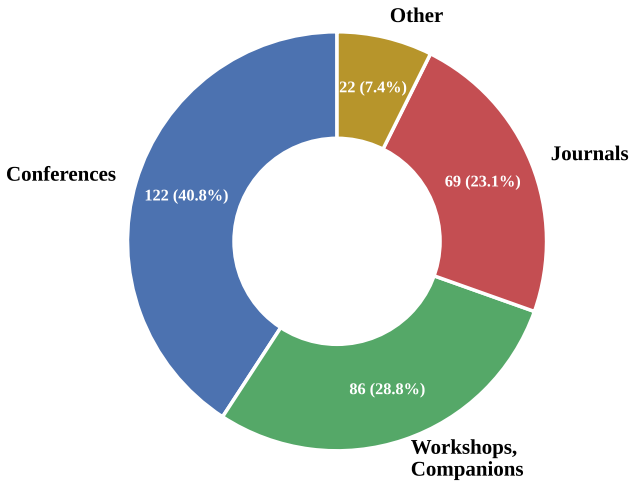

Fig. 1. Number of publications by publication type.

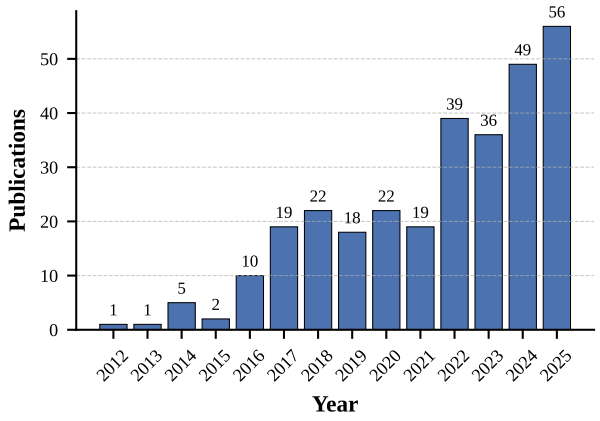

Fig. 2. Number of publications per year.

beyond the scope of evolutionary computation. Based on our findings, we discuss ongoing challenges of lexicase selection and open questions in the current body of research.

The rest of the paper is structured as follows: In Section II, we present the method and results of our systematic literature review. Sections III–VII give a detailed overview of lexicase selection and its variants. Section VIII provides an overview of the application domains of lexicase selection, followed by a discussion of challenges and open questions in Section IX. Finally, Section X concludes this survey.

II. Methodology

To systematically identify the relevant literature on lexicase selection, we performed a search on Google Scholar using the keyword *"lexicase"*. As lexicase selection was first introduced in 2012 [2], we limited our search to papers published between 2012 and 2025. At the time of data collection, the query returned 907 publications. From these, we identified papers that are written in English and used lexicase as part of their experiments or theoretical analysis. We excluded papers that only cite lexicase selection as part of their related work or use the term lexicase in a different context (e.g., linguistics). From the remaining papers, we included peer-reviewed articles in journals and conference proceedings, technical reports, and preprints, while excluding bachelor's, master's, and PhD theses.[1] In total, this resulted in 299 in-scope publications for this survey. We categorize those papers by application domain in Table III[2] and further discuss their application areas in Section VIII.

Figure 1 shows the number of publications by publication type. We notice that most in-scope publications on lexicase selection appear in conference proceedings ($\sim$ 41%) with *Genetic and Evolutionary Computation Conference* (45) and *European Conference on Genetic Programming* (19) being the most frequent venues. Workshop proceedings and conference companions account for approximately 29% of in-scope publications, with 55 *Genetic and Evolutionary Computation Conference Companion* and 26 *Genetic Programming Theory and Practice* publications. Around 23% of in-scope papers are published in journals such as *Genetic Programming and Evolvable Machines* and *IEEE Transactions of Evolutionary Computation*. Approximately 7% of the in-scope publications do not belong to any of the mentioned categories and are primarily preprints.

Additionally, as displayed in Figure 2, we observe a steady increase in publications over the years since the inception of lexicase selection. While initially only a few papers were published on the topic, in 2016 lexicase began to gain traction and from 2017 to 2021 around 20 papers were published every year. By 2022, this number doubled again to almost 40 papers and increased to over 50 in 2025. This trend indicates growing interest in lexicase selection and motivates a systematic review of the rapidly expanding body of literature.

Finally, we identified those papers that not only applied lexicase selection in their experiments or used it as a baseline, but also significantly contributed to the field of lexicase selection by either improving the performance of lexicase selection or contributing to the understanding of lexicase selection and its applications. We refer to these papers as *"core papers"* in the remainder of this survey and list them in Table I.[2] In total, we identified 156 core papers. Based on the core papers listed in Table I, we identified five major research streams of lexicase selection: Lexicase selection in general, ϵ-lexicase selection as a special variant of lexicase for continuous-valued problems like symbolic regression, combinations of lexicase with batching or down-sampling approaches, and approximations as well as hybridization of lexicase selection with other methods.

In the following sections, we will discuss the advancements and contributions to each research stream in detail. Although not included in the systematic review or quantitative analyses, we discuss relevant recent papers published in 2026 where appropriate.

[1]For most PhD theses, the key contributions have already been published in peer-reviewed venues and are therefore already represented in this survey.
[2]Note that some papers appear multiple times.

TABLE I
CORE PAPERS BY DIFFERENT RESEARCH STREAMS IN CHRONOLOGICAL ORDER. WE DEFINE CORE PAPERS AS THOSE THAT ACTIVELY CONTRIBUTE TO THE UNDERSTANDING AND PROGRESS OF LEXICASE SELECTION, RATHER THAN MERELY APPLYING IT AS A SELECTION METHOD IN EC OR AS A BASELINE FOR COMPARISON.

Research Stream	**References**	**#**
Lexicase	[2], [19], [20], [21], [6], [22], [23], [7], [24], [8], [25], [12], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [9], [46], [47], [48], [10], [1], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63], [64], [65], [66], [67], [68], [69], [70], [71], [72], [73], [74], [75], [76], [77], [78], [79], [80], [81], [82], [83], [84], [85], [86], [87], [88]	78
ϵ-Lexicase	[15], [16], [89], [27], [90], [91], [92], [93], [94], [95], [96], [97], [44], [98], [99], [100], [101], [1], [102], [103], [104], [105], [106], [107], [108], [109], [110], [56], [58], [111], [112], [113], [69], [70], [114], [115], [76], [116], [117], [86], [118], [119], [120], [121]	44
Batched Lexicase	[17], [1], [107], [122], [115], [119]	6
Down-Sampling	[123], [124], [125], [1], [126], [127], [128], [129], [130], [131], [56], [132], [133], [134], [135], [136], [137], [138], [139], [140], [141], [142], [143], [144], [119], [145]	26
Hybrids & Approximations	[22], [146], [147], [148], [95], [149], [150], [151], [152], [153], [1], [154], [155], [18], [156], [157], [56], [158], [159], [160], [161], [162], [163]	23

Algorithm 1 Lexicase selection (based on [6])

1: $P' :=$ all candidate solutions $p \in P$
2: $C' :=$ all training cases $c \in C$ in random order
3: **while** $|P'| > 1$ **and** $|C'| > 0$ **do**
4: $\quad c :=$ first case in C'
5: $\quad \mathbf{e}_c :=$ **evaluate** all $p \in P'$ on c
6: $\quad P' :=$ all candidate solutions with $e_c = e_c^*$
7: $\quad$ **remove** c from C'
8: **end while**
9: **return** random candidate solution from P'

III. LEXICASE SELECTION

In contrast to other selection methods, lexicase selection does not use a single aggregated fitness value for selection. Instead, lexicase selection operates on a case-by-case basis, selecting candidate solutions that perform particularly well on individual training cases more frequently, even if their aggregated performance is worse than that of other candidate solutions [2].

More specifically, Algorithm 1 (adopted from [6]) describes how lexicase selects a single parent in detail. First, the candidate pool P' is initialized by copying the whole population of candidate solutions $p \in P$ (line 1). Similarly, the case pool C' is created, containing all training cases $c \in C$ in random order (line 2). Afterwards, all candidate solutions in P' are evaluated on the first training case $c \in C'$ (line 4–5) to obtain a vector of error values for that case, denoted by $\mathbf{e}_c$.[3]

Only those candidate solutions with the lowest error $e_c = e_c^*$ on that training case are retained, while all other candidates (i.e., those that do not achieve the best performance on that case) are removed from P' (line 6). The first training case c is then removed from C' (line 7) and the process is repeated with the next training case in C'. If only a single candidate solution remains in P', that candidate solution is selected and returned. Otherwise, if C' contains no more training cases, one of the remaining candidate solutions in P' is selected uniformly at random (line 9). To construct the parent population, the selection procedure is repeated using a newly shuffled ordering of C' each time until the desired number of parents has been selected.

[3]In practice, the error of each candidate solution on each training case is usually only calculated once per generation and then cached for all selection events.

TABLE II
EXAMPLE POPULATION OF FIVE CANDIDATE SOLUTIONS EVALUATED ON FOUR TRAINING CASES WITH THEIR CORRESPONDING ELITE CASES, AGGREGATED PERFORMANCE IN TERMS OF THE MAE, AND SELECTION PROBABILITY USING LEXICASE A_{lex} (BASED ON [2], [44]).

	Error per case						
Candidates	e_1	e_2	e_3	e_4	Elite cases	MAE	A_{lex}
p_1	0	2	2	3	$\{c_1, c_3\}$	1.75	0.167
p_2	0	3	2	2	$\{c_1, c_3\}$	1.75	0.25
p_3	0	0	6	3	$\{c_1, c_2\}$	2.25	0.333
p_4	0	1	3	3	$\{c_1\}$	1.75	0
p_5	2	5	5	1	$\{c_4\}$	3.25	0.25

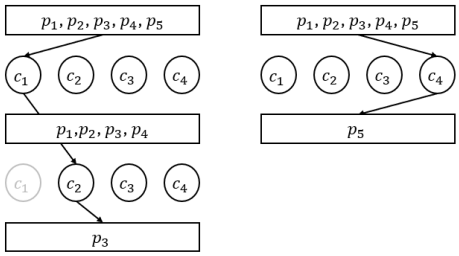

Fig. 3. Two example parent selection events with lexicase selection for the example population described in Table II (based on [44])

To further illustrate lexicase selection, consider the example displayed in Table II (based on [2], [44]), which shows an example population consisting of five candidate solutions $p \in P$ and the error vectors of four training cases $c \in C$. For each candidate solution p_i, we indicate the elite cases (i.e., the cases where the individual achieves the lowest error e_c^*) and report its aggregated error in terms of the mean absolute error (MAE).

Figure 3 (based on [44]) illustrates two selection events

with the example population described in Table II. In the first selection event (left), the case c_1 is drawn first, where four candidate solutions are elite. The candidate solution p_5 is excluded from the candidate pool, because it is outperformed by the other candidates. Next, c_2 is drawn, where p_3 performs best among the remaining candidate solutions. Hence, p_3 is selected as a parent in this selection event. In the second example (right), the case c_4 is drawn first, where only p_5 is elite, so this candidate is selected immediately, even though it would never be selected based on aggregated fitness scores because it has the highest average error.

Overall, the case-by-case approach of lexicase selection alters the selection probabilities and shifts selection pressure from global average performance to excellence in specific training cases [36], effectively negating the evaluation bottleneck of other selection methods [5]. In Table II, the selection probabilities A_{lex} indicate the probability of an individual p_i being selected using lexicase selection, which can be calculated using the following recursive function taken from La Cava et al. [44]:

$$\mathrm{A}_{\text{lex}}(p|P,C) = \begin{cases} 1 & \text{if } |P| = 1, \\ \frac{1}{|P|} & \text{if } |C| = 0, \\ \frac{1}{|C|}\sum_{k_s \in K_p} \mathrm{A}_{\text{lex}}\Big(p \mid P\big(q \mid k_s \in K_q\big), C \setminus \{k_s\}\Big) & \text{otherwise.} \end{cases} \tag{1}$$

where C denotes the set of training cases, P the set of candidate solutions, and K_p the subset of cases from C for which candidate p is elite. If only one candidate remains ($|P| = 1$), this candidate is selected. If there are no training cases left ($|C| = 0$), all remaining candidates are selected with equal probability.

If multiple candidates and cases remain, the probability of selecting candidate p depends on the probability of drawing a case $k_s \in K_p$ (meaning a case where p is elite), which is $\frac{1}{|C|}$ for each case. If a case $k_s \in K_p$ is drawn, the probability of selecting candidate p further depends on the subset of candidates that are also elite on case k_s, $P\big(q \mid k_s \in K_q\big)$, as well as the remaining cases $C \setminus \{k_s\}$ [44]. Calculating these probabilities is NP-hard [70].

For our illustrative example (Table II), the selection probability of p_5 is 0.25, as p_5 is always selected whenever c_4 is drawn first and never otherwise. The selection probability of p_4 is 0, because it performs best on case c_1, but so do p_1, p_2, and p_3. These candidates are also elite on additional training cases, whereas p_4 is not. Consequently, p_4 is always eliminated during subsequent filtering steps. For candidate p_3, the selection probability is 0.333, because it will be selected if case c_2 is drawn first (with probability 0.25) or if case c_1 is drawn first and subsequently c_2 (with probability 0.083). Thus, selection pressure is shifted away from aggregate performance measures such as MAE and toward case-wise elite performance. This shift in selection pressure has several important consequences.

For example, lexicase selection has a significant impact on population diversity, which is typically measured as the number of distinct error vectors in a population, commonly referred to as behavioral diversity [164]. Multiple studies show that lexicase selection increases population diversity compared to other selection methods [7], [8]. Moreover, lexicase selection benefits from a high population diversity to effectively explore the search space, while tournament selection often gets stuck in local optima [56].

However, diversity sometimes drops significantly during runs with lexicase selection due to hyperselection events. As described in [25], hyperselection events occur when a single candidate is selected in a disproportionately large number of selection events because it is elite on a large number of training cases relative to the rest of the population. However, hyperselection events do not influence the performance of lexicase selection [25] because lexicase selection is able to re-diversify populations from a low diversity starting point or after a diversity crash, an advantage that is not observed with tournament selection [8].

While these studies suggest that this increase in diversity positively affects the performance of lexicase selection, it is not the sole reason why lexicase selection performs better [10]. Lexicase selection is also more likely to select specialists, meaning candidates that perform exceptionally well on difficult training cases despite having a comparatively high average error. In our example, candidate p_5 has a higher selection probability compared to p_1 and p_4 because it is the only one that is elite on case c_4 even though its average error is higher. In contrast, with tournament selection, the selection probability would only depend on the average error, so generalists would be preferred (in our example p_1, p_2, and p_4). It has been found that the selection of specialists is crucial for the problem-solving success of lexicase selection [9], [10], [36] as these specialists can be stepping stones to find a solution that solves all training cases [7], [10]–[13]. Moreover, it is assumed that lexicase selection produces higher population diversity because specialists with different behavior can coexist within a population, whereas candidates exhibiting similar behavior compete directly with each other [7], [8].

Despite all these positive effects, lexicase comes at the cost of increased runtime. For a population P and a set of training cases C, the worst-case time complexity of generating a parent population using lexicase selection is $\mathcal{O}(|P|^2|C|)$. This is worse than aggregated selection methods like tournament selection, which have a worst-case complexity of $\mathcal{O}(|P||C|)$ [44]. Fortunately, in practice, the runtime for lexicase tends to be lower than its worst-case complexity suggests, as the pool of candidate solutions is often filtered down to a single candidate solution before all training cases are used [6], [44].

Moreover, Helmuth et al. [55] showed that the expected runtime of lexicase selection depends on behavioral diversity. In highly diverse populations, the expected runtime reduces to $\mathcal{O}(|P| \cdot (|P| + |C|))$, and empirical results indicate that populations using lexicase selection are often sufficiently diverse for runtimes to be much lower than the worst-case complexity. Nonetheless, lexicase is slower than aggregated methods, especially for large populations.

The base lexicase algorithm can be adjusted in multiple ways. For example, if more than one candidate solution remains after all training cases have been considered, the selection within the remaining candidate pool does not have

Algorithm 2 Dynamic ϵ-lexicase selection (based on [44])

1: $P' :=$ all candidate solutions $p \in P$
2: $C' :=$ all training cases $c \in C$ in random order
3: **while** $|P'| > 1$ **and** $|C'| > 0$ **do**
4: $\quad c :=$ first case in C'
5: $\quad \mathbf{e}_c :=$ **evaluate** all $p \in P'$ on c
6: $\quad$ **calculate** ϵ_c from $\mathbf{e_c}$
7: $\quad P' :=$ all candidate solutions with $e_c \leq e_c^* + \epsilon_c$
8: $\quad$ **remove** c from C'
9: **end while**
10: **return** random candidate solution from P'

to be random (line 9 in Algorithm 1). Alternatively, a second objective like program size could break ties between candidate solutions by applying parsimony pressure as discussed in [157]. This favors smaller solutions, effectively reducing unnecessary code growth. However, other tie-breaking metrics could be applied here as well.

Further, Helmuth et al. [10] describe a pre-selection step for lexicase selection that groups candidate solutions with identical behavior and only initializes P' with one representative of each group. This does not change the selection probabilities, but reduces the computational time. Otherwise, if only candidates with identical behavior are left, the algorithm will continue to compare them on all remaining cases without reducing the candidate pool any further.

Additionally, the ordering of C' (line 2) does not have to be random but can instead be biased in different ways. So far, no case ordering has been shown to significantly improve performance [148]. However, ordering the training cases by their discriminatory power (so that those cases with more variation in their corresponding error vectors are more likely to appear earlier in the sequence) reduces runtime while keeping the same performance, which is especially beneficial for large data sets [156], [158]. Others describe variants of lexicase selection that only use a subset of the available training cases to initialize C' [123], [124], [138], which we discuss further in Section VI.

Likewise, instead of evaluating individuals iteratively on a single training case at a time (line 4-5), one can use a batch of training cases for each filtering event [17], [122] (see Section V). Lastly, relaxing the elitism in line 6 can be beneficial, particularly in continuous or noisy application domains [15] (see Section IV).

IV. Epsilon Lexicase Selection

So far, we have only considered lexicase selection with a strictly elitist filtering approach. This works particularly well for application domains with discrete fitness values. However, in application domains with continuous (potentially noisy) fitness values per training case, such as symbolic regression, strict elitism can become a major drawback since ties in performance are very rare [15], [44]. Therefore, selecting only the best-performing candidate on each case typically results in only a single training case being used per selection event. This can lead to a significant decrease in performance [15], [44].

One solution to this problem is relaxing the condition for candidate solutions to be kept in P'. In particular, ϵ-lexicase selection [15] introduces a threshold value: Let e_c denote the error of a candidate solution for a given case c (lower is better) and e_c^* the best error on this case of all candidates in the candidate pool P', then all candidate solutions with an error $e_c \leq e_c^* + \epsilon_c$ are retained in P' [15], [44]. ϵ_c is calculated for each training case as the median absolute deviation [165]

$$\epsilon_c = \text{median}(|\mathbf{e}_c - \text{median}(\mathbf{e}_c)|),$$

where $\mathbf{e}_c$ is a vector of error values of the remaining candidate solutions in P' with respect to c. The rationale of calculating ϵ_c per training case is that the selective pressure is adapted based on the variability of error values in the candidate pool P' per training case. Usually, the variance is high at the beginning of the search, and thus ϵ_c is relatively high. As the search converges, the epsilon values decrease [44].

Algorithm 2 shows the pseudocode for ϵ-lexicase selection. In comparison to standard lexicase selection, an extra line for calculating ϵ_c is added in line 6 and the condition in line 7 is relaxed using ϵ_c. This specific version, where both e_c^* and ϵ_c are determined based on the remaining candidate pool P', is called *dynamic* ϵ-lexicase selection [44].

Alternatively, the threshold values ϵ_c and the corresponding elite errors e_c^* can also be calculated only once per generation before the filtering step using the error values of the whole population for each case, which is called *static* ϵ-lexicase selection [15], [44]. *Semi-dynamic* ϵ-lexicase selection only calculates e_c^* dynamically from the remaining candidate solutions in P' while keeping ϵ_c fixed within each generation for each case [44]. All described variations of ϵ-lexicase selection perform similarly [44].

Recently, a new method for calculating the threshold has been proposed that divides the error values for a case into two partitions so that the sum of the variances within the two groups is minimized. This has been found to be beneficial for real-world regression problems [116].

Regarding other properties of lexicase selection (such as diversity maintenance, hyperselection, and specialist selection), ϵ-lexicase selection behaves similarly to standard lexicase selection, except that the average number of cases used during selection is higher [133].

Other relaxed forms of lexicase selection for symbolic regression problems are discussed in [95]. However, ϵ-lexicase selection generally outperformed the other approaches in the reported experiments and is the most commonly used lexicase variant for symbolic regression. The concept of adding a threshold to lexicase selection has also been applied in other problem domains such as evolutionary robotics [92], [146] and learning classifier systems [17].

V. Batched Lexicase Variants

With lexicase selection, candidate solutions are usually evaluated on only a few training cases before a parent is selected, which can negatively affect generalization [17], [52], [122]. To solve this problem, Aenugu and Spector [17] proposed using batches of training cases instead of single training cases

Algorithm 3 Batch-lexicase selection (based on [122])

1: $P' :=$ all candidate solutions $p \in P$
2: $C' :=$ all training cases $c \in C$ in random order
3: **while** $|P'| > 1$ **and** $|C'| > 0$ **do**
4: $\quad B :=$ first b cases in C'
5: $\quad \mathbf{e}_B :=$ **evaluate** all $p \in P'$ on B
6: $\quad P' :=$ all candidate solutions with $e_B = e^*_B$
7: $\quad$ **remove** B from C'
8: **end while**
9: **return** random candidate solution from P'

to evaluate candidate solutions in each filtering step in the lexicase selection procedure as shown in Algorithm 3. In each selection event, the training cases are combined to form batches of equal size b (line 4). Then, the average performance of each candidate solution $p \in P'$ is calculated for the first batch B (line 5). Candidates that are elite on batch B proceed to the next batch (line 6). This is repeated until a single candidate remains or there are no more training cases left. Batch-lexicase selection has been found to improve generalization compared to lexicase selection in learning classifier systems [17] and for program synthesis problems [122].

This approach can be combined with a threshold depending on the problem domain. A straightforward way to incorporate a threshold is to require a candidate's average performance on a batch to exceed a manually set value before it can pass the filtering step, e.g., for classification problems [17]. Geiger et al. [136] proposed batch-ϵ-lexicase selection for symbolic regression problems, where training cases are combined into batches but the filtering step is performed using ϵ-lexicase selection. A recent study has found that batch-ϵ-lexicase selection outperforms ϵ-lexicase selection for symbolic regression problems when the search is conducted for only a short period of time [143]. The authors assume that this is because the use of batches enables the evaluation of individuals on more training cases early on, reducing the impact of case ordering [17]. However, in longer search runs, batch-free variants outperform those that use batches [143].

VI. Down-Sampling Techniques

From a computational perspective, the evaluation of candidate solutions is usually the most expensive part of evolutionary search [166]. Consequently, large training sets limit the number of candidate solutions that can be evaluated during search for a given budget [123], [124], [128]. A simple solution would be to reduce the training set beforehand in order to increase the number of generations or the population size while keeping the number of evaluations constant. However, if the training set is reduced too much, the problem space might not be adequately covered anymore [123], [124].

Therefore, down-sampling techniques can be used to sample different subsets of training cases in each generation. This not only reduces computational costs, but also improves performance and reduces overfitting and bloat [21], [167]. Using different subsets in each generation has the benefit that candidate solutions are evaluated on a large fraction of the original training set over the course of evolution [123].

Algorithm 4 Random down-sampled lexicase selection (based on [123])

1: $P' :=$ all candidate solutions $p \in P$
2: $S :=$ random subset of C of size $r * |C|$
3: $S' :=$ all training cases $c \in S$ in random order
4: **while** $|P'| > 1$ **and** $|S'| > 0$ **do**
5: $\quad c :=$ first case in S'
6: $\quad \mathbf{e}_c :=$ **evaluate** all $p \in P'$ on c
7: $\quad P' :=$ all candidate solutions with $e_c = e^*_c$
8: $\quad$ **remove** c from S'
9: **end while**
10: **return** random candidate solution from P'

Hernandez et al. [123] analyzed the combination of lexicase selection with random down-sampling (RDS) [167] as shown in Algorithm 4. In each generation, a different random subset S of size $r \cdot |C|$ (with $r < 1$) is created from the original training set C (line 2). This subset S is shuffled (line 3) and then used for parent selection with lexicase selection (line 4-10). Lexicase selection with RDS has previously been used implicitly in the domain of evolutionary robotics, where evaluations are very expensive, and therefore only a few random training cases are used for evaluation in each generation [16], [92], [97].

Furthermore, Hernandez et al. [123] proposed cohort lexicase selection, which splits the population P and the training set C into subgroups, where each candidate subgroup is evaluated only on a subgroup of C. During lexicase selection, each candidate solution competes only with solutions in the same subgroup.

Cohort sampling and RDS significantly increased the success rates of lexicase selection [123], [124]. Both reduce the number of evaluations by a factor of $D = \frac{1}{r}$ per generation. Thus, using the same number of evaluations, you can explore D times more candidate solutions (either by increasing the number of generations or by increasing the population size) [123]. As RDS and cohort lexicase selection perform similarly, while RDS is considerably simpler [123], [124], subsequent research mainly focused on analyzing RDS [125], [128]. These studies attribute the higher performance of lexicase selection with RDS to the fact that it evaluates a larger number of candidate solutions, rather than other factors, such as extending the search to more generations, changing the environment, or improving generalization. Furthermore, they found that the performance improvements are consistent across a wide range of down-sampling rates r [125], [128], as long as r is not extremely high or low [129].

In addition, RDS has been studied in combination with ϵ-lexicase selection [133] and other lexicase variants for symbolic regression problems [136], [143], where it significantly improved performance. However, RDS slightly reduced population diversity for ϵ-lexicase [133].

A downside of RDS is that it reduces the likelihood of selecting specialists, because it may exclude those training cases on which candidate solutions are specialized [124]. Consequently, selection pressure shifts toward generalists and away from specialists, which can lead to performance drops on

certain problems [56], [124]. Hernandez et al. [56] argue that RDS reduces the exploration capabilities of lexicase selection.

Moreover, RDS selects training cases with equal probability, although not all training cases are equally important. Some are redundant and provide little additional information, while others are rare and highly-informative edge cases or provide critical information about the current population. RDS may discard these critical cases for several generations, potentially harming the search [138]. To overcome this problem, Boldi et al. [138] proposed informed down-sampling (IDS).

In particular, IDS first constructs error vectors $\mathbf{e}_c$ for each training case by evaluating the population on all training cases. Each entry in the error vector corresponds to the performance of a single candidate solution on case c. Thus, distances between error vectors measure how differently the current population performs on distinct training cases. For example, if all candidate solutions show identical performance on two training cases, then the distance between their error vectors is zero. Consequently, larger differences in performance between candidate solutions of a population for two training cases lead to larger distances between the corresponding error vectors [138]. These distances are then used to create a diverse subset of training cases with regard to their error vectors, for example, by using the farthest-first-traversal algorithm [168]. To achieve similar computational savings as RDS, the error vectors are constructed using only a fraction of the parent population and are only updated every few generations. As a result, IDS leads to the creation of diverse and meaningful subsets, including highly informative edge cases, instead of randomly drawing training cases.

IDS combined with lexicase selection outperforms RDS for program synthesis problems [132], [138]. Consequently, IDS has been studied in combination with other selection methods as well [134], [141]–[143], [169]. For program synthesis, IDS is particularly effective for diversity-preserving selection methods such as lexicase [141]. In the domain of symbolic regression, IDS closes the performance and behavior gap between tournament and lexicase selection [142], [143]. However, it does not consistently improve performance [143]. Recent work indicates that this might be due to the tendency of IDS to over-sample outliers when applied to real-world problems [170]. Unlike edge cases, outliers deviate from the underlying pattern [171], and oversampling them potentially harms the search process and might lead to overfitting. Therefore, Geiger et al. [170] proposed ROIDS, an outlier-aware variant of IDS that excludes potential outliers by removing a small percentage of cases with the worst aggregated error value before creating subsets with IDS.

Moore and Stanton [135] proposed an alternative approach to generating more meaningful subsets by selecting training cases more frequently based on their effectiveness in filtering during the selection process. This sampling approach achieved a significant reduction in computational cost within the domain of evolutionary robotics [135], [145].

VII. Hybrids and Approximations

As discussed in Section III, lexicase selection has improved performance in many problem domains due to characteristics such as mitigating the evaluation bottleneck and preserving specialists. However, other selection methods exhibit strengths that complement those of lexicase selection. Consequently, several hybrid approaches have been proposed to combine the strengths of lexicase selection with those of other selection methods.

For example, novelty search [172] maintains a high population diversity by selecting parents with novel behavior. Knobelty selection [149] aims to combine novelty and lexicase selection by randomly choosing either lexicase selection or novelty selection for each selection event to obtain a parent population that is both diverse and contains high-performing individuals. Similarly, novelty-lexicase selection [151] calculates the novelty scores for an individual on each training case and appends them to its list of error values. Therefore, during lexicase selection, the focus is sometimes on individuals with low errors on certain cases and sometimes on individuals exhibiting novel behavior on certain cases. Novelty-lexicase selection increased population diversity for program synthesis problems and found more solutions.

Double lexicase selection [160] reduces bloat in regression problems by performing a two-step selection process. First, multiple candidates are sampled using ϵ-lexicase selection. Then, one of them is selected as a parent based on its size with roulette wheel selection. Consequently, smaller individuals have a higher probability of being selected.

However, the major drawback of lexicase selection is that its high-quality solutions come at the cost of increased runtime. Therefore, recent approaches focus on reducing the runtime of lexicase selection by approximating its behavior. For example, Ding et al. [159] introduce probabilistic lexicase (plexicase) selection, which samples parents from an approximate probability distribution that estimates the probability that an individual would be selected by lexicase selection. The approximation is necessary because computing the exact probabilities is an NP-hard problem [70]. The approximation method is based on the observation that an individual's selection probability depends on whether it ranks among the best on a given case and on the performance of the other individuals on that same case (see Section III). The advantage of plexicase selection is that once the probability distribution is calculated, parents can be sampled from the distribution directly instead of performing multiple selection events. This significantly reduces the runtime compared to standard lexicase selection.

DALex [162] approximates lexicase selection by assigning random weights to training cases and multiplying an individual's error vector by these weights before producing an aggregated error. Because the weights are assigned randomly for each selection event, the intuition behind lexicase selection is preserved, as more information from an individual's error vector is retained than with traditional aggregated fitness measures. The standard deviation of the weight distribution can be adjusted to easily approximate relaxed variants of lexicase selection. The performance of DALex is comparable to standard lexicase selection across several domains while being significantly faster.

TABLE III
REFERENCES BY DIFFERENT APPLICATION DOMAINS IN CHRONOLOGICAL ORDER

Application Domain	References	#
Program Synthesis	[2], [19], [20], [6], [173], [22], [14], [23], [7], [24], [8], [25], [12], [26], [28], [31], [174], [175], [13], [176], [177], [178], [179], [180], [148], [42], [32], [95], [181], [35], [182], [36], [37], [38], [39], [183], [40], [184], [41], [149], [123], [151], [43], [185], [9], [186], [187], [124], [47], [125], [188], [48], [1], [10], [189], [190], [191], [192], [52], [51], [193], [50], [194], [122], [128], [155], [195], [196], [129], [130], [131], [156], [55], [197], [198], [158], [199], [132], [159], [200], [113], [134], [201], [202], [203], [162], [138], [204], [139], [137], [140], [141], [205], [206], [207], [208], [209], [210], [85], [81], [211], [212], [144], [213], [214], [215]	106
Symbolic Regression	[21], [216], [15], [89], [217], [27], [30], [218], [177], [219], [147], [94], [95], [35], [96], [150], [44], [220], [221], [222], [223], [98], [152], [99], [100], [101], [224], [189], [225], [103], [104], [106], [107], [108], [226], [227], [228], [229], [230], [197], [231], [232], [233], [234], [235], [112], [64], [159], [133], [134], [236], [237], [160], [114], [162], [136], [116], [238], [117], [207], [239], [240], [241], [242], [243], [244], [245], [246], [247], [248], [249], [250], [251], [142], [143], [120], [118], [252], [253], [254], [255], [256], [119], [257], [258], [259], [260], [261], [262], [263], [264], [265], [266], [267], [268], [269], [270], [271], [272], [273], [274]	101
Classification	[275], [29], [30], [90], [177], [17], [46], [153], [276], [107], [49], [154], [157], [230], [62], [60], [277], [278], [231], [18], [156], [158], [112], [279], [66], [161], [162], [74], [78], [115], [163], [280], [281], [87], [121], [282], [283]	37
Boolean Logic/ Hardware Circuit Design	[22], [45], [284], [192], [285], [157], [286], [231], [287], [288], [63], [161], [78], [80], [82]	15
Robot Control	[146], [16], [91], [92], [97], [102], [289], [126], [127], [110], [135], [290], [73], [145]	14
Other/ Not Applicable	[93], [33], [34], [44], [105], [291], [109], [292], [53], [293], [61], [54], [56], [57], [58], [59], [294], [111], [67], [65], [295], [68], [69], [296], [297], [70], [72], [75], [71], [77], [298], [299], [300], [76], [79], [301], [302], [303], [304], [305], [84], [306], [86], [307], [308], [309], [310], [311], [83], [88], [312]	51

VIII. APPLICATION DOMAINS

Early applications of lexicase selection were mainly in the domains of program synthesis [6] and symbolic regression [15]. From 2016 onward, there was a noticeable expansion in both the number of papers and the variety of applications. We identified five major application domains and categorized all in-scope publications accordingly in Table III. Program synthesis and symbolic regression are still the most prominent categories with 106 and 101 publications, respectively, followed by classification with 37. Boolean logic and hardware circuit design, along with robot control, represent more niche application areas, with 15 and 14 publications, respectively. This diversification has intensified over the past two years: nearly half of the papers grouped in the category "others" were published in 2024 and 2025.

For program synthesis, lexicase selection significantly increased problem-solving success [6], [14], potentially due to its ability to preserve diversity [7], [8] and to maintain specialists [9], [10], [36]. In the domain of symbolic regression, ϵ-lexicase selection showed superior performance compared to traditional selection methods [15], [44]. In learning classifier systems, a batched lexicase variant has been found to be more robust to data errors, thereby improving the generalization in classification problems [17]. Furthermore, lexicase selection has been successfully applied to Boolean logic problems using genetic programming [22] and Boolean constraint satisfaction problems using genetic algorithms, where it found more valid solutions than traditional selection methods [45]. This is likely due to the ability of lexicase to preserve a diverse set of partial solutions that each satisfy specific constraints, rather than relying on an aggregated fitness score [45]. In evolutionary robotics, lexicase selection with down-sampling has been shown to outperform specialized strategies [16], possibly due to its effectiveness in preserving high population diversity [92].

Other lexicase applications are combinatorial optimization [79], [311], generative art [65], [76], [86], and dynamic job scheduling [67], [295].

Furthermore, lexicase selection has been analyzed in areas related to machine learning. These include evolving artificial neural networks [275], [290], optimizing deep neural networks [18], [156], [158], clustering [291], [292], [294], computer vision [113], feature representation [96], feature construction [64], [66], [160], [232]–[234], [237], [244], [247], [256], [260], [262], [263], [272], feature selection [46], [89], [90], [264], [279], and even evolving machine learning pipelines [112].

Several papers address multi-objective optimization [34], [72], [75], [77], [93], [111], [297], [300], [310], as lexicase selection can be interpreted as a multi-objective strategy. La Cava et al. [44] found that lexicase selection only selects individuals located on the boundaries of the Pareto front with respect to their performance on training cases (which can be viewed as different objectives). Thus, lexicase selection performs well for multi-objective problems, while also being significantly faster than traditional approaches [93]. Shahbandegan and Dolson [72] also observed this behavior for problems with many contradictory objectives.

This overview shows that lexicase selection is beneficial in a variety of problem domains, particularly when dealing with modular problem structures, where it is important to preserve specialists for behavioral niches [2]. For discrete problems, such as in program synthesis or Boolean logic, the standard form of lexicase selection is well-suited [6]. In contrast, for continuous domains such as symbolic regression, ϵ-lexicase

selection is generally recommended [15]. If overfitting is a concern, batched variants of lexicase selection offer improved generalization [122]. Additionally, down-sampling strategies are often beneficial [123], [138], especially if evaluations are very expensive, as is often the case in robot control tasks [145]. If computational efficiency is very important, approximations may be preferable [159], [162]. Lastly, for specific requirements such as bloat control or novelty promotion, hybrid approaches exist that combine the advantages of lexicase selection with those of more traditional selection methods [149], [151], [160].

IX. Challenges, Open Questions, and Beyond

Our literature review demonstrates that lexicase selection has proven successful across many application domains largely because it maintains high population diversity [7], [8] and effectively promotes and preserves specialists [9], [10], [36], which can serve as stepping stones during search [11]–[13].

Besides its many advantages, the original formulation of lexicase selection suffers from high computational costs [44]. This challenge has been approached by several approximation strategies. In particular, DALex [162] significantly reduces runtime and even enables GPU-based implementations of lexicase selection. In addition, down-sampling techniques [123], [138] reduce the number of evaluations per generation, thus saving runtime while maintaining or even improving performance [142].

Furthermore, it has been observed that lexicase selection sometimes generalizes poorly to unseen test cases [52], [125]. Evaluating solutions on batches of training cases rather than single cases mitigates this shortcoming and has been shown to improve generalization [17], [122]. In symbolic regression, down-sampling techniques have also helped to improve generalization [142].

This demonstrates that many of the original limitations of lexicase selection have been effectively mitigated, making it a compelling option for a wide range of problems. Despite these advancements, lexicase selection has not yet become the default selection method in genetic programming.

We identified four key challenges that, if addressed, could significantly advance the field and promote the broader adoption of lexicase selection in the community.

1) We strongly recommend integrating lexicase selection and especially new lexicase variants into genetic programming frameworks to make them more accessible to researchers and practitioners.
2) Many new variants introduce additional parameters that require costly hyperparameter tuning and consequently raise the barrier to adoption for practitioners. Introducing adaptive parameters, for example, a dynamic adjustment of parameters based on convergence speed, would reduce the need for manual tuning and make new methods more accessible.
3) Despite its empirical success, lexicase selection variants lack a strong theoretical foundation. Only a few theoretical studies have been published so far [44], [69]–[71], leaving our understanding of lexicase variants incomplete. A deeper theoretical grounding would not only improve our understanding of their behavior but also enable a more systematic development of new variants.
4) The growing variety of lexicase variants is currently difficult to compare. Most studies evaluate new methods on a single problem domain and a limited set of benchmarks, making it difficult to assess their relative strengths and weaknesses. To remedy this, we advocate for a unified evaluation framework that includes all relevant lexicase variants and benchmark problems for domains such as symbolic regression and classification from SRBench [103] and PMLB [313], program synthesis tasks from PSB1 [14] and PSB2 [194], as well as problems from other domains. Such a framework would allow researchers to compare methodological advances rather than implementation-specific differences across domains.

Addressing these challenges would not only deepen our understanding of lexicase selection, but also accelerate its adoption as a standard selection method in genetic programming.

Beyond genetic programming, the properties of lexicase selection are also interesting when using large language models as variation operators in evolutionary search [314]–[318]. As the output diversity of large language models is often limited [206], [319], evolutionary search converges too early and tends to generate mediocre rather than rare but high-quality outputs [320]. This poses a problem because large language models need to be used in an open-ended manner to constantly produce new and interesting outputs to make novel discoveries [321]. This diversity problem has been successfully approached using Quality-Diversity methods [320], [322]. Nevertheless, lexicase selection remains underexplored in this domain, despite its potential to serve as a general diversity-preserving optimization algorithm, making it a promising direction for future work.

X. Conclusions

Lexicase selection led to significant performance improvements across a variety of application domains due to its ability to preserve diversity and select specialists. This success inspired the development of variants that address original limitations of lexicase, thereby expanding its applicability.

Despite these advances, we observed that many researchers use outdated versions of lexicase or continue to rely on traditional selection methods. To address this problem, we provided the first comprehensive survey of the field to consolidate current research.

We identified five major research streams that emerged in recent years and discussed their contributions. Further, we identified the most relevant application domains, ranging from program synthesis and symbolic regression to classification, Boolean logic, and robot control. Based on these findings, we identified critical challenges for future work such as the need for a stronger theoretical foundation, a unified evaluation framework for developing new methods, and the integration of lexicase variants in existing frameworks to accelerate the

adoption of lexicase selection in genetic programming. Lastly, we identified several promising research directions, including the use of lexicase selection beyond genetic programming.

REFERENCES

[1] T. Helmuth and A. Abdelhady, "Benchmarking parent selection for program synthesis by genetic programming," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, 2020, pp. 237–238.

[2] L. Spector, "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*, 2012, pp. 401–408.

[3] K. Krawiec and P. Liskowski, "Automatic derivation of search objectives for test-based genetic programming," in *European Conference on Genetic Programming*. Springer, 2015, pp. 53–65.

[4] K. Krawiec, J. Swan, and U.-M. O'Reilly, "Behavioral program synthesis: Insights and prospects," *Genetic programming theory and practice XIII*, pp. 169–183, 2016.

[5] K. Krawiec and U.-M. O'Reilly, "Behavioral programming: a broader and more detailed take on semantic gp," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, 2014, pp. 935–942.

[6] T. Helmuth, L. Spector, and J. Matheson, "Solving uncompromising problems with lexicase selection," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 5, pp. 630–643, 2014.

[7] T. Helmuth, N. F. McPhee, and L. Spector, "Lexicase selection for program synthesis: a diversity analysis," in *Genetic Programming Theory and Practice XIII*. Springer, 2016, pp. 151–167.

[8] ——, "Effects of lexicase and tournament selection on diversity recovery and maintenance," in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, 2016, pp. 983–990.

[9] T. Helmuth, E. Pantridge, and L. Spector, "Lexicase selection of specialists," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 1030–1038.

[10] ——, "On the importance of specialists for lexicase selection," *Genetic Programming and Evolvable Machines*, vol. 21, no. 3, pp. 349–373, 2020.

[11] N. F. McPhee, D. Donatucci, and T. Helmuth, "Using graph databases to explore the dynamics of genetic programming runs," in *Genetic programming theory and practice XIII*. Springer, 2016, pp. 185–201.

[12] N. F. McPhee, M. M. Casale, M. Finzel, T. Helmuth, and L. Spector, "Visualizing genetic programming ancestries," in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, 2016, pp. 1419–1426.

[13] ——, "Visualizing genetic programming ancestries using graph databases," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2017, pp. 245–246.

[14] T. Helmuth and L. Spector, "General program synthesis benchmark suite," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, 2015, pp. 1039–1046.

[15] W. La Cava, L. Spector, and K. Danai, "Epsilon-lexicase selection for regression," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, 2016, pp. 741–748.

[16] J. M. Moore and A. Stanton, "Lexicase selection outperforms previous strategies for incremental evolution of virtual creature controllers," in *ECAL 2017, the Fourteenth European Conference on Artificial Life*. MIT Press, 2017, pp. 290–297.

[17] S. Aenugu and L. Spector, "Lexicase selection in learning classifier systems," in *Proceedings of the genetic and evolutionary computation conference*, 2019, pp. 356–364.

[18] L. Ding and L. Spector, "Optimizing neural networks with gradient lexicase selection," in *International Conference on Learning Representations*, 2022.

[19] T. Helmuth and L. Spector, "Evolving a digital multiplier with the pushgp genetic programming system," in *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*, 2013, pp. 1627–1634.

[20] ——, "Word count as a traditional programming benchmark problem for genetic programming," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, 2014, pp. 919–926.

[21] Y. Martínez, L. Trujillo, E. Naredo, and P. Legrand, "A comparison of fitness-case sampling methods for symbolic regression with genetic programming," in *EVOLVE—A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation V*. Springer International Publishing, 2014, pp. 201–212.

[22] P. Liskowski, K. Krawiec, T. Helmuth, and L. Spector, "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*, 2015, pp. 1301–1307.

[23] N. F. McPhee, D. Donatucci, and T. Helmuth, "Using graph databases to explore the dynamics of genetic programming runs," *Genetic programming theory and practice XIII*, pp. 185–201, 2016.

[24] E. Moscovici, "Adding program length bias to the lexicase and tournament selection algorithms," in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, 2016, pp. 1035–1038.

[25] T. Helmuth, N. F. McPhee, and L. Spector, "The impact of hyperselection on lexicase selection," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, 2016, pp. 717–724.

[26] L. Spector, N. F. McPhee, T. Helmuth, M. M. Casale, and J. Oks, "Evolution evolves with autoconstruction," in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, 2016, pp. 1349–1356.

[27] U. López, L. Trujillo, Y. Martinez, P. Legrand, E. Naredo, and S. Silva, "Ransac-gp: Dealing with outliers in symbolic regression with genetic programming," in *European Conference on Genetic Programming*. Springer, 2017, pp. 114–130.

[28] S. Forstenlechner, D. Fagan, M. Nicolau, and M. O'Neill, "A grammar design pattern for arbitrary program synthesis problems in genetic programming," in *European Conference on Genetic Programming*. Springer, 2017, pp. 262–277.

[29] W. La Cava, S. Silva, L. Vanneschi, L. Spector, and J. Moore, "Genetic programming representations for multi-dimensional feature learning in biomedical classification," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2017, pp. 158–173.

[30] K. Oksanen and T. Hu, "Lexicase selection promotes effective search and behavioural diversity of solutions in linear genetic programming," in *2017 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2017, pp. 169–176.

[31] K. Krawiec, I. Błądek, and J. Swan, "Counterexample-driven genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2017, pp. 953–960.

[32] I. Błądek, K. Krawiec, and J. Swan, "Counterexample-driven genetic programming: heuristic program synthesis from formal specifications," *Evolutionary computation*, vol. 26, no. 3, pp. 441–469, 2018.

[33] E. Dolson, W. Banzhaf, and C. Ofria, "Applying ecological principles to genetic programming," in *Genetic programming theory and practice XV*. Springer, 2018, pp. 73–88.

[34] T. Jansen and C. Zarges, "Theoretical analysis of lexicase selection in multi-objective optimization," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2018, pp. 153–164.

[35] E. Dolson and C. Ofria, "Ecological theory provides insights about evolutionary computation," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2018, pp. 105–106.

[36] E. Pantridge, T. Helmuth, N. F. McPhee, and L. Spector, "Specialization and elitism in lexicase and tournament selection," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2018, pp. 1914–1917.

[37] P. Liskowski, I. Błądek, and K. Krawiec, "Neuro-guided genetic programming: prioritizing evolutionary search with neural networks," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018, pp. 1143–1150.

[38] T. Helmuth, N. F. McPhee, and L. Spector, "Program synthesis using uniform mutation by addition and deletion," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018, pp. 1127–1134.

[39] K. Krawiec, I. Bladek, J. Swan, and J. H. Drake, "Counterexample-driven genetic programming: Stochastic synthesis of provably correct programs." in *IJCAI*, 2018, pp. 5304–5308.

[40] D. C. Le, S. Khanchi, A. N. Zincir-Heywood, and M. I. Heywood, "Benchmarking evolutionary computation approaches to insider threat detection," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018, pp. 1286–1293.

[41] E. Schulte, J. Ruchti, M. Noonan, D. Ciarletta, and A. Loginov, "Evolving exact decompilation," in *Workshop on Binary Analysis Research (BAR)*, 2018.

[42] H. Ahmad and T. Helmuth, "A comparison of semantic-based initialization methods for genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2018, pp. 1878–1881.

[43] E. Dolson, A. Lalejini, and C. Ofria, "Exploring genetic programming systems with map-elites," in *Genetic Programming Theory and Practice XVI*. Springer, 2019, pp. 1–16.

[44] W. La Cava, T. Helmuth, L. Spector, and J. H. Moore, "A probabilistic and multi-objective analysis of lexicase selection and ε-lexicase selection," *Evolutionary Computation*, vol. 27, no. 3, pp. 377–402, 2019.

[45] B. Metevier, A. K. Saini, and L. Spector, "Lexicase selection beyond genetic programming," in *Genetic programming theory and practice XVI*. Springer, 2019, pp. 123–136.

[46] W. La Cava, S. Silva, K. Danai, L. Spector, L. Vanneschi, and J. H. Moore, "Multidimensional genetic programming for multiclass classification," *Swarm and evolutionary computation*, vol. 44, pp. 260–272, 2019.

[47] D. Lynch, J. McDermott, and M. O'Neill, "Program synthesis in a continuous space using grammars and variational autoencoders," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2020, pp. 33–47.

[48] A. K. Saini and L. Spector, "Effect of parent selection methods on modularity," in *European Conference on Genetic Programming (Part of EvoStar)*. Springer, 2020, pp. 184–194.

[49] R. J. Smith, R. Amaral, and M. I. Heywood, "Evolving simple solutions to the cifar-10 benchmark using tangled program graphs," in *2021 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2021, pp. 2061–2068.

[50] D. Sobania and F. Rothlauf, "A generalizability measure for program synthesis with genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2021, pp. 822–829.

[51] A. K. Saini and L. Spector, "Relationships between parent selection methods, looping constructs, and success rate in genetic programming," *Genetic Programming and Evolvable Machines*, vol. 22, pp. 495–509, 2021.

[52] D. Sobania, "On the generalizability of programs synthesized by grammar-guided genetic programming," in *European Conference on Genetic Programming (Part of EvoStar)*. Springer, 2021, pp. 130–145.

[53] J. G. Hernandez, A. Lalejini, and E. Dolson, "What can phylogenetic metrics tell us about useful diversity in evolutionary algorithms?" in *Genetic programming theory and practice XVIII*. Springer, 2022, pp. 63–82.

[54] A. Lalejini, E. Dolson, A. E. Vostinar, and L. Zaman, "Selection schemes from evolutionary computing show promise for directed evolution of microbes," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2022, pp. 723–726.

[55] T. Helmuth, J. Lengler, and W. La Cava, "Population diversity leads to short running times of lexicase selection," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2022, pp. 485–498.

[56] J. G. Hernandez, A. Lalejini, and C. Ofria, "An exploration of exploration: Measuring the ability of lexicase selection to find obscure pathways to optimality," *Genetic Programming Theory and Practice XVIII*, p. 83, 2022.

[57] ——, "A suite of diagnostic metrics for characterizing selection schemes," *arXiv preprint arXiv:2204.13839*, 2022.

[58] S. Shahbandegan, J. G. Hernandez, A. Lalejini, and E. Dolson, "Untangling phylogenetic diversity's role in evolutionary computation using a suite of diagnostic fitness landscapes," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2022, pp. 2322–2325.

[59] J. G. Hernandez, A. Lalejini, and C. Ofria, "Measuring the ability of lexicase selection to find obscure pathways to optimality," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2022, pp. 21–22.

[60] B. Portman and M. I. Heywood, "On the interaction between lexicase selection, modularity and data subsets," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2022, pp. 586–589.

[61] A. Lalejini, E. Dolson, A. E. Vostinar, and L. Zaman, "Artificial selection methods from evolutionary computing show promise for directed evolution of microbes," *Elife*, vol. 11, p. e79665, 2022.

[62] T. Pleshkova and V. Stanovov, "Classification algorithm with lexicase selection," *European Proceedings of Computers and Technology*, 2023.

[63] B. Majeed, S. Carvalho, D. M. Dias, A. Youssef, A. Murphy, and C. Ryan, "Performance upgrade of sequence detector evolution using grammatical evolution and lexicase parent selection method," in *International Conference on Complex Computational Ecosystems*. Springer, 2023, pp. 90–103.

[64] H. Zhang, Q. Chen, B. Xue, W. Banzhaf, and M. Zhang, "Automatically choosing selection operator based on semantic information in evolutionary feature construction," in *Pacific Rim International Conference on Artificial Intelligence*. Springer, 2023, pp. 385–397.

[65] E. M. Fredericks, A. C. Diller, and J. M. Moore, "Generative art via grammatical evolution," in *2023 IEEE/ACM International Workshop on Genetic Improvement (GI)*. IEEE, 2023, pp. 1–8.

[66] H. Zhang, Q. Chen, A. Zhou, B. Xue, Y. Wang, and M. Zhang, "Beyond deep learning: An evolutionary feature engineering approach to tabular data classification," 2023.

[67] M. Xu, Y. Mei, F. Zhang, and M. Zhang, "Genetic programming with lexicase selection for large-scale dynamic flexible job shop scheduling," *IEEE Transactions on Evolutionary Computation*, 2023.

[68] R. Boldi, A. Lokhandwala, E. Annatone, Y. Schechter, A. Lavrenenko, and C. Sigrist, "Improving recommendation system serendipity through lexicase selection," *arXiv preprint arXiv:2305.11044*, 2023.

[69] S. Shahbandegan and E. Dolson, "Theoretical limits on the success of lexicase selection under contradictory objectives," in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, 2023, pp. 827–830.

[70] E. Dolson, "Calculating lexicase selection probabilities is np-hard," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2023, pp. 1575–1583.

[71] E. Dolson and A. Lalejini, "Reachability analysis for lexicase selection via community assembly graphs," in *Genetic Programming Theory and Practice XX*. Springer, 2024, pp. 283–301.

[72] S. Shahbandegan and E. Dolson, "On the robustness of lexicase selection to contradictory objectives," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2024, pp. 594–602.

[73] R. Bahlous-Boldi, L. Ding, L. Spector, and S. Niekum, "Pareto-optimal learning from preferences with hidden context," *arXiv preprint arXiv:2406.15599*, 2024.

[74] J. G. Hernandez, A. K. Saini, and J. H. Moore, "Lexidate: Model evaluation and selection with lexicase," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2024, pp. 279–282.

[75] T. Fukase, Y. Kobayashi, and C. Aranha, "Extending lexicase de for a multi-niche constraint satisfaction industrial design problem," in *2024 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2024, pp. 01–07.

[76] E. M. Fredericks, J. M. Moore, and A. C. Diller, "Generativegi: creating generative art with genetic improvement," *Automated Software Engineering*, vol. 31, no. 1, p. 23, 2024.

[77] R. Boldi, L. Ding, and L. Spector, "Solving deceptive problems without explicit diversity maintenance," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2024, pp. 171–174.

[78] D. Reyes Fernández De Bulnes, A. De Lima, A. Murphy, D. Mota Dias, and C. Ryan, "Feature encapsulation by stages using grammatical evolution," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2024, pp. 531–534.

[79] P. Barbosa, R. Savisaar, and A. Fonseca, "Semantically rich local dataset generation for explainable ai in genomics," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2024, pp. 267–276.

[80] D. Reyes Fernández de Bulnes, E. Naredo, and L. Trujillo, "Boolean circuits design by means of feature encapsulation by stages," *Available at SSRN 5206460*.

[81] A. Grishina, V. Liventsev, A. Härmä, and L. Moonen, "Fully autonomous programming using iterative multi-agent debugging with large language models," *ACM Transactions on Evolutionary Learning*, vol. 5, no. 1, pp. 1–37, 2025.

[82] B. Majeed, R. Sarma, A. Youssef, D. M. Dias, and C. Ryan, "Automatic generation of synthesisable hardware description language code of multi-sequence detector using grammatical evolution," *Algorithms*, vol. 18, no. 6, p. 345, 2025.

[83] C. Sigrist, A. Li, P. Lertsaroj, R. Boldi, A. Lechowicz, N. Bashir, and M. Hajiesmaili, "Learning to site: A multi-objective optimization of rooftop solar installations using evolutionary neural networks," *ACM SIGMETRICS Performance Evaluation Review*, vol. 53, no. 2, pp. 51–56, 2025.

[84] J. G. Hernandez, A. K. Saini, and J. H. Moore, "Lexicase selection parameter analysis: Varying population size and test case redundancy with diagnostic metrics," in *Genetic Programming Theory and Practice XXI*. Springer, 2025, pp. 375–393.

[85] K. Suzue, C. Ofria, and A. Lalejini, "Using lineage age to augment search space exploration in lexicase selection," in *Genetic Programming Theory and Practice XXI*. Springer, 2025, pp. 395–411.

[86] E. M. Fredericks, D. Bobeldyk, and J. M. Moore, "Crafting generative art through genetic improvement: Managing creative outputs in diverse fitness landscapes," in *Genetic Programming Theory and Practice XXI*. Springer, 2025, pp. 321–335.

[87] J. G. Hernandez, A. K. Saini, A. Gupta, and J. Moore, "Evaluating the generalizability of machine learning pipelines when using lexicase or tournament selection," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2025, pp. 283–286.

[88] R. G. S. Alphonse Raj and B. Sandesh, "Innovative dual-objective framework for image captioning: Harmonizing visual analysis and linguistic precision," *SN Computer Science*, vol. 6, no. 6, p. 583, 2025.

[89] W. La Cava and J. Moore, "A general feature engineering wrapper for machine learning using-lexicase survival," in *European Conference on Genetic Programming*. Springer, 2017, pp. 80–95.

[90] W. La Cava and J. H. Moore, "Ensemble representation learning: an analysis of fitness and survival for wrapper-based genetic programming methods," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2017, pp. 961–968.

[91] W. La Cava and J. Moore, "Behavioral search drivers and the role of elitism in soft robotics," in *ALIFE 2018: The 2018 Conference on Artificial Life*. MIT Press, 2018, pp. 206–213.

[92] J. M. Moore and A. Stanton, "Tiebreaks and diversity: Isolating effects in lexicase selection," in *Artificial life conference proceedings*. MIT Press, 2018, pp. 590–597.

[93] W. La Cava and J. H. Moore, "An analysis of ϵ-lexicase selection for large-scale many-objective optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2018, pp. 185–186.

[94] P. Orzechowski, W. La Cava, and J. H. Moore, "Where are we now? a large benchmark study of recent symbolic regression methods," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018, pp. 1183–1190.

[95] L. Spector, W. La Cava, S. Shanabrook, T. Helmuth, and E. Pantridge, "Relaxations of lexicase parent selection," in *Genetic Programming Theory and Practice XV*. Springer, 2018, pp. 105–120.

[96] W. L. Cava, T. R. Singh, J. Taggart, S. Suri, and J. Moore, "Learning concise representations for regression by evolving networks of trees," in *International Conference on Learning Representations*, 2019.

[97] J. M. Moore and A. Stanton, "The limits of lexicase selection in an evolutionary robotics task," in *Artificial Life Conference Proceedings*. MIT Press, 2019, pp. 551–558.

[98] S. Ruberto, V. Terragni, and J. H. Moore, "Sgp-dt: Semantic genetic programming based on dynamic targets," in *European Conference on Genetic Programming (Part of EvoStar)*. Springer, 2020, pp. 167–183.

[99] A. Hara, J.-i. Kushida, and T. Takahama, "Maintaining population diversity in deterministic geometric semantic genetic programming by ϵ-lexicase selection," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2020, pp. 205–210.

[100] A. K. Saini and L. Spector, "Using modularity metrics as design features to guide evolution in genetic programming," *Genetic Programming Theory and Practice XVII*, pp. 165–180, 2020.

[101] W. La Cava and J. H. Moore, "Learning feature spaces for regression with genetic programming," *Genetic Programming and Evolvable Machines*, vol. 21, no. 3, pp. 433–467, 2020.

[102] J. M. Moore and A. Stanton, "When specialists transition to generalists: Evolutionary pressure in lexicase selection," in *Artificial Life Conference Proceedings*. MIT Press, 2020, pp. 719–726.

[103] W. L. Cava, P. Orzechowski, B. Burlacu, F. O. de Francca, M. Virgolin, Y. Jin, M. Kommenda, and J. H. Moore, "Contemporary symbolic regression methods and their relative performance," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1 (NeurIPS Datasets and Benchmarks 2021)*, 2021, pp. 1–16.

[104] P. Moscato, H. Sun, and M. N. Haque, "Analytic continued fractions for regression: A memetic algorithm approach," *Expert Systems with Applications*, vol. 179, pp. 1–17, 2021.

[105] L. Planinić, M. Đurasević, and D. Jakobović, "On the application of ε-lexicase selection in the generation of dispatching rules," in *2021 IEEE congress on evolutionary computation (CEC)*. IEEE, 2021, pp. 2125–2132.

[106] S. Ruberto, V. Terragni, and J. H. Moore, "A semantic genetic programming framework based on dynamic targets," *Genetic Programming and Evolvable Machines*, vol. 22, no. 4, pp. 463–493, 2021.

[107] A. R. Wagner and A. Stein, "Adopting lexicase selection for michigan-style learning classifier systems with continuous-valued inputs," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2021, pp. 171–172.

[108] I. Bakurov, M. Castelli, O. Gau, F. Fontanella, and L. Vanneschi, "Genetic programming for stacked generalization," *Swarm and Evolutionary Computation*, vol. 65, p. 100913, 2021.

[109] V. Stanovov, S. Akhmedova, and E. Semenkin, "Neuroevolution for parameter adaptation in differential evolution," *Algorithms*, vol. 15, no. 4, p. 122, 2022.

[110] J. Huizinga and J. Clune, "Evolving multimodal robot behavior via many stepping stones with the combinatorial multiobjective evolutionary algorithm," *Evolutionary computation*, vol. 30, no. 2, pp. 131–164, 2022.

[111] Y. He, C. Aranha, A. Hallam, and R. Chassagne, "Optimization of subsurface models with multiple criteria using lexicase selection," *Operations Research Perspectives*, vol. 9, p. 100237, 2022.

[112] N. Matsumoto, A. K. Saini, P. Ribeiro, H. Choi, A. Orlenko, L.-P. Lyytikäinen, J. O. Laurikka, T. Lehtimäki, S. Batista, and J. H. Moore, "Faster convergence with lexicase selection in tree-based automated machine learning," in *European Conference on Genetic Programming (Part of EvoStar)*. Springer, 2023, pp. 165–181.

[113] Y. Lavinas, K. Cortacero, and S. Cussat-Blanc, "Evolving graphs with cartesian genetic programming with lexicase selection," in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, 2023, pp. 1920–1924.

[114] L. Ingelse, J.-I. Hidalgo, J. M. Colmenar, N. Lourenço, and A. Fonseca, "Comparing individual representations in grammar-guided genetic programming for glucose prediction in people with diabetes," in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, 2023, pp. 2013–2021.

[115] A. de Lima, S. Carvalho, D. M. Dias, J. Amaral, J. P. Sullivan, and C. Ryan, "Fuzzy pattern trees for classification problems using genetic programming," in *European Conference on Genetic Programming (Part of EvoStar)*. Springer, 2024, pp. 3–20.

[116] G. S. Imai Aldeia, F. O. De França, and W. G. La Cava, "Minimum variance threshold for epsilon-lexicase selection," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2024, pp. 905–913.

[117] D. Reyes Fernández de Bulnes, A. de Lima, E. Galván, and C. Ryan, "Feature encapsulation by stages in the regression domain using grammatical evolution," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2024, pp. 105–120.

[118] M. Kotanchek, N. Haut, and K. Kotanchek, "Representation and reachability: Assumption impact in data modeling," in *Genetic Programming Theory and Practice XXI*. Springer, 2025, pp. 1–25.

[119] I. Bakurov, A. Murphy, C. Ofria, and W. Banzhaf, "A comparison of tournament and lexicase selection paradigms in regression problems: error-based fitness versus correlation fitness," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2025, pp. 970–979.

[120] L. Ingelse, J. I. Hidalgo, J. M. Colmenar, N. Lourenço, and A. Fonseca, "A comparison of representations in grammar-guided genetic programming in the context of glucose prediction in people with diabetes," *Genetic Programming and Evolvable Machines*, vol. 26, no. 1, pp. 1–25, 2025.

[121] A. de Lima, J. F. Albarracín, D. M. Dias, J. Amaral, and C. Ryan, "On fitting numerical features into probabilistic distributions to represent data for fuzzy pattern trees," *Genetic Programming and Evolvable Machines*, vol. 26, no. 2, p. 25, 2025.

[122] D. Sobania and F. Rothlauf, "Program synthesis with genetic programming: the influence of batch sizes," in *European Conference on Genetic Programming (Part of EvoStar)*. Springer, 2022, pp. 118–129.

[123] J. G. Hernandez, A. Lalejini, E. Dolson, and C. Ofria, "Random subsampling improves performance in lexicase selection," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2019, pp. 2028–2031.

[124] A. J. Ferguson, J. G. Hernandez, D. Junghans, A. Lalejini, E. Dolson, and C. Ofria, "Characterizing the effects of random subsampling on lexicase selection," in *Genetic Programming Theory and Practice XVII*. Springer, 2020, pp. 1–23.

[125] T. Helmuth and L. Spector, "Explaining and exploiting the advantages of down-sampled lexicase selection," in *Artificial Life Conference Proceedings*. MIT Press, 2020, pp. 341–349.

[126] J. M. Moore and A. Stanton, "Objective sampling strategies for generalized locomotion behavior with lexicase selection," in *Artificial*

Life Conference Proceedings 33, vol. 2021, no. 1. MIT Press, 2021, p. 73.

[127] A. Stanton and J. M. Moore, "Lexicase selection for multi-task evolutionary robotics," *Artificial Life*, vol. 28, no. 4, pp. 479–498, 2022.

[128] T. Helmuth and L. Spector, "Problem-solving benefits of down-sampled lexicase selection," *Artificial life*, vol. 27, no. 3–4, pp. 183–203, 2022.

[129] D. Schweim, D. Sobania, and F. Rothlauf, "Effects of the training set size: a comparison of standard and down-sampled lexicase selection in program synthesis," in *2022 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2022, pp. 1–8.

[130] T. Helmuth and P. Kelly, "Applying genetic programming to psb2: the next generation program synthesis benchmark suite," *Genetic Programming and Evolvable Machines*, vol. 23, no. 3, pp. 375–404, 2022.

[131] R. Boldi, T. Helmuth, and L. Spector, "The environmental discontinuity hypothesis for down-sampled lexicase selection," *arXiv preprint arXiv:2205.15931*, 2022.

[132] R. Boldi, A. Lalejini, T. Helmuth, and L. Spector, "A static analysis of informed down-samples," in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, 2023, pp. 531–534.

[133] A. Geiger, D. Sobania, and F. Rothlauf, "Down-sampled epsilon-lexicase selection for real-world symbolic regression problems," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2023, pp. 1109–1117.

[134] R. Boldi, A. Bao, M. Briesch, T. Helmuth, D. Sobania, L. Spector, and A. Lalejini, "The problem solving benefits of down-sampling vary by selection scheme," in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, 2023, pp. 527–530.

[135] J. M. Moore and A. Stanton, "Fitness agnostic adaptive sampling lexicase selection," in *ALIFE 2023: Ghost in the Machine: Proceedings of the 2023 Artificial Life Conference*. MIT Press, 2023.

[136] A. Geiger, D. Sobania, and F. Rothlauf, "A comprehensive comparison of lexicase-based selection methods for symbolic regression problems," in *European Conference on Genetic Programming (Part of EvoStar)*. Springer, 2024, pp. 192–208.

[137] A. Lalejini, M. A. Moreno, J. G. Hernandez, and E. Dolson, "Phylogeny-informed fitness estimation for test-based parent selection," in *Genetic Programming Theory and Practice XX*. Springer, 2024, pp. 241–261.

[138] R. Boldi, M. Briesch, D. Sobania, A. Lalejini, T. Helmuth, F. Rothlauf, C. Ofria, and L. Spector, "Informed down-sampled lexicase selection: Identifying productive training cases for efficient problem solving," *Evolutionary computation*, pp. 1–31, 2024.

[139] R. Boldi, A. Bao, M. Briesch, T. Helmuth, D. Sobania, L. Spector, and A. Lalejini, "A comprehensive analysis of down-sampling for genetic programming-based program synthesis," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2024, pp. 487–490.

[140] A. Lalejini, M. Sanson, J. Garbus, M. A. Moreno, and E. Dolson, "Runtime phylogenetic analysis enables extreme subsampling for test-based problems," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2024, pp. 511–514.

[141] R. Boldi, A. Bao, M. Briesch, T. Helmuth, D. Sobania, L. Spector, and A. Lalejini, "Untangling the effects of down-sampling and selection in genetic programming," in *ALIFE 2024: Proceedings of the 2024 Artificial Life Conference*. MIT Press, 2024.

[142] A. Geiger, M. Briesch, D. Sobania, and F. Rothlauf, "Was tournament selection all we ever needed? a critical reflection on lexicase selection," in *European Conference on Genetic Programming (Part of EvoStar)*. Springer, 2025, pp. 207–223.

[143] A. Geiger, D. Sobania, and F. Rothlauf, "A performance analysis of lexicase-based and traditional selection methods in gp for symbolic regression," *ACM Transactions on Evolutionary Learning*, 2025.

[144] G. Pinna, D. Ravalico, L. Rovito, L. Manzoni, and A. De Lorenzo, "Exploring the effect of genetic improvement for large language models-generated code," *SN Computer Science*, vol. 6, no. 7, p. 760, 2025.

[145] J. M. Moore and A. Stanton, "Efficient adaptive sampling lexicase selection in evolutionary robotics," in *Artificial Life Conference Proceedings 37*, vol. 2025, no. 1. MIT Press, 2025, p. 33.

[146] J. M. Moore and P. K. McKinley, "A comparison of multiobjective algorithms in evolving quadrupedal gaits," in *International Conference on Simulation of Adaptive Behavior*. Springer, 2016, pp. 157–169.

[147] A. R. Burks and W. F. Punch, "An investigation of hybrid structural and behavioral diversity methods in genetic programming," *Genetic Programming Theory and Practice XIV*, pp. 19–34, 2018.

[148] S. A. Troise and T. Helmuth, "Lexicase selection with weighted shuffle," in *Genetic Programming Theory and Practice XV*. Springer, 2018, pp. 89–104.

[149] J. Kelly, E. Hemberg, and U.-M. O'Reilly, "Improving genetic programming with novel exploration-exploitation control," in *European Conference on Genetic Programming*. Springer, 2019, pp. 64–80.

[150] V. V. De Melo, D. V. Vargas, and W. Banzhaf, "Batch tournament selection for genetic programming: the quality of lexicase, the speed of tournament," in *Proceedings of the genetic and evolutionary computation conference*, 2019, pp. 994–1002.

[151] L. Jundt and T. Helmuth, "Comparing and combining lexicase selection and novelty search," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 1047–1055.

[152] H. Zhang, H. Zhang, and A. Zhou, "A multi-metric selection strategy for evolutionary symbolic regression," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2020, pp. 585–591.

[153] W. La Cava and J. H. Moore, "Genetic programming approaches to learning fair classifiers," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, 2020, pp. 967–975.

[154] M. Sipper and J. H. Moore, "Conservation machine learning: a case study of random forests," *Scientific Reports*, vol. 11, no. 1, p. 3629, 2021.

[155] D. Schweim, E. Hemberg, D. Sobania, and U.-M. O'Reilly, "Exploiting knowledge from code to guide program search," in *European Conference on Genetic Programming (Part of EvoStar)*. Springer, 2022, pp. 262–277.

[156] L. Ding, R. Boldi, T. Helmuth, and L. Spector, "Going faster and hence further with lexicase selection," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2022, pp. 538–541.

[157] A. de Lima, S. Carvalho, D. M. Dias, E. Naredo, J. P. Sullivan, and C. Ryan, "Lexi2: lexicase selection with lexicographic parsimony pressure," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2022, pp. 929–937.

[158] L. Ding, R. Boldi, T. Helmuth, and L. Spector, "Lexicase selection at scale," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2022, pp. 2054–2062.

[159] L. Ding, E. Pantridge, and L. Spector, "Probabilistic lexicase selection," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2023, pp. 1073–1081.

[160] H. Zhang, Q. Chen, B. Xue, W. Banzhaf, and M. Zhang, "A double lexicase selection operator for bloat control in evolutionary feature construction for regression," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2023, pp. 1194–1202.

[161] A. de Lima, S. Carvalho, D. M. Dias, J. P. Sullivan, and C. Ryan, "On switching selection methods to increase parsimony pressure," in *IJCCI*, 2023, pp. 96–107.

[162] A. Ni, L. Ding, and L. Spector, "Dalex: Lexicase-like selection via diverse aggregation," in *European Conference on Genetic Programming (Part of EvoStar)*. Springer, 2024, pp. 90–107.

[163] Q. Fan, Y. Bi, B. Xue, and M. Zhang, "Multi-tree genetic programming for learning color and multi-scale features in image classification," *IEEE Transactions on Evolutionary Computation*, 2024.

[164] D. Jackson, "Promoting phenotypic diversity in genetic programming," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2010, pp. 472–481.

[165] T. Pham-Gia and T. L. Hung, "The mean and median absolute deviations," *Mathematical and computer Modelling*, vol. 34, no. 7-8, pp. 921–936, 2001.

[166] Y. Martínez, E. Naredo, L. Trujillo, P. Legrand, and U. Lopez, "A comparison of fitness-case sampling methods for genetic programming," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 29, no. 6, pp. 1203–1224, 2017.

[167] I. Gonçalves, S. Silva, J. B. Melo, and J. M. B. Carreiras, "Random sampling technique for overfitting control in genetic programming," in *Genetic Programming*. Springer Berlin Heidelberg, 2012, pp. 218–229.

[168] D. S. Hochbaum and D. B. Shmoys, "A best possible heuristic for the k-center problem," *Mathematics of operations research*, vol. 10, no. 2, pp. 180–184, 1985.

[169] M. Briesch, "On the effects of down-sampling for tournament and lexicase selection in program synthesis," in *European Conference on Genetic Programming (Part of EvoStar)*. Springer, 2026, pp. 3–18.

[170] A. Geiger, M. Briesch, D. Sobania, and F. Rothlauf, "Roids: Robust outlier-aware informed down-sampling," *arXiv preprint arXiv:2601.19477*, 2026.

[171] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[172] J. Lehman, K. O. Stanley *et al.*, "Exploiting open-endedness to solve problems through the search for novelty." in *ALIFE*, 2008, pp. 329–336.

[173] H. Zhan, "A quantitative analysis of the simplification genetic operator," in *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*, 2014, pp. 1077–1080.

[174] S. Forstenlechner, D. Fagan, M. Nicolau, and M. O'Neill, "Semantics-based crossover for program synthesis in genetic programming," in *International Conference on Artificial Evolution (Evolution Artificielle)*. Springer, 2017, pp. 58–71.

[175] V. Kashyap, R. Swords, E. Schulte, and D. Melski, "Musynth: Program synthesis via code reuse and code manipulation," in *International Symposium on Search Based Software Engineering*. Springer, 2017, pp. 117–123.

[176] N. F. McPhee, T. Helmuth, and L. Spector, "Using algorithm configuration tools to optimize genetic programming parameters: A case study," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2017, pp. 243–244.

[177] E. Pantridge and L. Spector, "Pyshgp: Pushgp in python," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2017, pp. 1255–1262.

[178] E. Pantridge, T. Helmuth, N. F. McPhee, and L. Spector, "On the difficulty of benchmarking inductive program synthesis methods," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2017, pp. 1589–1596.

[179] L. Spector and E. Moscovici, "Recent developments in autoconstructive evolution," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2017, pp. 1154–1156.

[180] T. Helmuth, N. F. McPhee, E. Pantridge, and L. Spector, "Improving generalization of evolved programs through automatic simplification," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2017, pp. 937–944.

[181] S. Forstenlechner, D. Fagan, M. Nicolau, and M. O'Neill, "Towards understanding and refining the general program synthesis benchmark suite with genetic programming," in *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2018.

[182] N. F. McPhee, M. D. Finzel, M. M. Casale, T. Helmuth, and L. Spector, "A detailed analysis of a pushgp run," *Genetic Programming Theory and Practice XIV*, pp. 65–83, 2018.

[183] S. Forstenlechner, D. Fagan, M. Nicolau, and M. O'Neill, "Extending program synthesis grammars for grammar-guided genetic programming," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2018, pp. 197–208.

[184] S. Forstenlechner, D. Fagan, M. Nicolau, and M. O'Neill, "Towards effective semantic operators for program synthesis in genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018, pp. 1119–1126.

[185] J. Swan, K. Krawiec, and Z. A. Kocsis, "Stochastic synthesis of recursive functions made easy with bananas, lenses, envelopes and barbed wire," *Genetic Programming and Evolvable Machines*, vol. 20, no. 3, pp. 327–350, 2019.

[186] A. Lalejini and C. Ofria, "Tag-accessed memory for genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2019, pp. 346–347.

[187] E. Hemberg, J. Kelly, and U.-M. O'Reilly, "On domain knowledge and novelty to improve program synthesis performance with grammatical evolution," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 1039–1046.

[188] T. Helmuth, E. Pantridge, G. Woolson, and L. Spector, "Genetic source sensitivity and transfer learning in genetic programming," in *Artificial Life Conference Proceedings 32*. MIT Press, 2020, pp. 303–311.

[189] A. K. Saini and L. Spector, "Why and when are loops useful in genetic programming?" in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, 2020, pp. 247–248.

[190] T. Welsch and V. Kurlin, "Synthesis through unification genetic programming," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, 2020, pp. 1029–1036.

[191] E. Pantridge and L. Spector, "Code building genetic programming," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, 2020, pp. 994–1002.

[192] A. Lalejini, M. A. Moreno, and C. Ofria, "Tag-based regulation of modules in genetic programming improves context-dependent problem solving," *Genetic Programming and Evolvable Machines*, vol. 22, pp. 325–355, 2021.

[193] A. K. Saini and L. Spector, "Gleam: genetic learning by extraction and absorption of modules," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2021, pp. 263–264.

[194] T. Helmuth and P. Kelly, "Psb2: the second program synthesis benchmark suite," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2021, pp. 785–794.

[195] A. K. Saini and L. Spector, "Evolving and analyzing modularity with gleam (genetic learning by extraction and absorption of modules)," in *Genetic Programming Theory and Practice XVIII*. Springer, 2022, pp. 181–195.

[196] Y. He, C. Aranha, and T. Sakurai, "Knowledge-driven program synthesis via adaptive replacement mutation and auto-constructed subprogram archives," in *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2022, pp. 14–21.

[197] A. K. Saini, L. Spector, and T. Helmuth, "Environments with local scopes for modules in genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2022, pp. 598–601.

[198] E. Pantridge, T. Helmuth, and L. Spector, "Functional code building genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2022, pp. 1000–1008.

[199] D. Sobania, M. Briesch, P. Röchner, and F. Rothlauf, "Mtgp: Combining metamorphic testing and genetic programming," in *European Conference on Genetic Programming (Part of EvoStar)*. Springer, 2023, pp. 324–338.

[200] V. Liventsev, A. Grishina, A. Härmä, and L. Moonen, "Fully autonomous programming with large language models," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2023, pp. 1146–1155.

[201] E. Pantridge and T. Helmuth, "Solving novel program synthesis problems with genetic programming using parametric polymorphism," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2023, pp. 1175–1183.

[202] M. Briesch, D. Sobania, and F. Rothlauf, "On the trade-off between population size and number of generations in gp for program synthesis," in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, 2023, pp. 535–538.

[203] T. Helmuth, J. G. Frazier, Y. Shi, and A. F. Abdelrehim, "Human-driven genetic programming for program synthesis: a prototype," in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, 2023, pp. 1981–1989.

[204] M. Briesch, R. Boldi, D. Sobania, A. Lalejini, T. Helmuth, F. Rothlauf, C. Ofria, and L. Spector, "Improving lexicase selection with informed down-sampling," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2024, pp. 25–26.

[205] T. Helmuth, E. Pantridge, J. G. Frazier, and L. Spector, "Generational computation reduction in informal counterexample-driven genetic programming," in *European Conference on Genetic Programming (Part of EvoStar)*. Springer, 2024, pp. 21–37.

[206] D. Sobania, J. Petke, M. Briesch, and F. Rothlauf, "A comparison of large language models and genetic programming for program synthesis," *IEEE Transactions on Evolutionary Computation*, 2024.

[207] A. Ni and L. Spector, "Effective adaptive mutation rates for program synthesis," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2024, pp. 952–960.

[208] T. Helmuth, J. Fedoroff, E. Pantridge, and L. Spector, "Facilitating function application in code building genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2024, pp. 887–895.

[209] M. C. Fernandes, F. O. de Franca, and E. Francesquini, "Origami:(un) folding the abstraction of recursion schemes for program synthesis," in *Genetic Programming Theory and Practice XX*. Springer, 2024, pp. 263–281.

[210] N. Freitag McPhee and R. Lussier, "The impact of step limits on generalization and stability in software synthesis," in *Genetic Programming Theory and Practice XX*. Springer, 2024, pp. 87–104.

[211] E. Pantridge and T. Helmuth, "Code building genetic programming is faster than pushgp," in *Genetic Programming Theory and Practice XXI*. Springer, 2025, pp. 133–150.

[212] G. Pinna, D. Ravalico, L. Rovito, L. Manzoni, A. De Lorenzo *et al.*, "Improving llm-generated code via genetic improvement: A summary of recent advances." CEUR Workshop Proceedings, 2025.

[213] J. G. Hernandez, A. K. Saini, G. Ketron, and J. H. Moore, "Gp and llms for program synthesis: No clear winners," *arXiv preprint arXiv:2508.03966*, 2025.

[214] J. G. Hernandez, A. K. Saini, G. Ketron, and J. Moore, "Comparing pushgp and gpt-4o on program synthesis with only input-output examples," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2025, pp. 619–622.

[215] G. S. I. Aldeia, D. S. Herman, and W. G. La Cava, "Iterative learning of computable phenotypes for treatment resistant hypertension using large language models," *Proceedings of machine learning research*, vol. 298, 2025.

[216] L. Spector and T. Helmuth, "Uniform linear transformation with repair and alternation in genetic programming," *Genetic Programming Theory and Practice XI*, pp. 137–153, 2014.

[217] T. P. Pawlak and K. Krawiec, "Synthesis of mathematical programming constraints with genetic programming," in *European Conference on Genetic Programming*. Springer, 2017, pp. 178–193.

[218] A. R. Burks and W. F. Punch, "An analysis of the genetic marker diversity algorithm for genetic programming," *Genetic Programming and Evolvable Machines*, vol. 18, pp. 213–245, 2017.

[219] P. Liskowski and K. Krawiec, "Discovery of search objectives in continuous domains," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2017, pp. 969–976.

[220] M. A. Lones, "Instruction-level design of local optimisers using push gp," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2019, pp. 1487–1494.

[221] W. La Cava and J. H. Moore, "Semantic variation operators for multidimensional genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 1056–1064.

[222] I. Błądek and K. Krawiec, "Solving symbolic regression problems with formal constraints," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 977–984.

[223] O. Dugan, R. Dangovski, A. Costa, S. Kim, P. Goyal, J. Jacobson, and M. Soljačić, "Occamnet: A fast neural model for symbolic regression at scale," *arXiv preprint arXiv:2007.10784*, 2020.

[224] S. Ruberto, V. Terragni, and J. H. Moore, "Sgp-dt: Towards effective symbolic regression with a semantic gp approach based on dynamic targets," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, 2020, pp. 25–26.

[225] Q. Chen, B. Xue, and M. Zhang, "Preserving population diversity based on transformed semantics in genetic programming for symbolic regression," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 3, pp. 433–447, 2020.

[226] F. O. de Franca and M. Z. de Lima, "Interaction-transformation symbolic regression with extreme learning machine," *Neurocomputing*, vol. 423, pp. 609–619, 2021.

[227] S. Powers, J. Smith, and C. Pinciroli, "Extracting symbolic models of collective behaviors with graph neural networks and macro-micro evolution," in *International Conference on Swarm Intelligence*. Springer, 2022, pp. 142–154.

[228] Z. Huang, Y. Mei, and J. Zhong, "Semantic linear genetic programming for symbolic regression," *IEEE Transactions on Cybernetics*, vol. 54, no. 2, pp. 1321–1334, 2022.

[229] R. Nieto-Fuentes and C. Segura, "Gp-dmd: A genetic programming variant with dynamic management of diversity," *Genetic Programming and Evolvable Machines*, vol. 23, no. 2, pp. 279–304, 2022.

[230] C. Kuranga and N. Pillay, "A comparative study of genetic programming variants," in *International Conference on Artificial Intelligence and Soft Computing*. Springer, 2022, pp. 377–386.

[231] A. de Lima, S. Carvalho, D. M. Dias, E. Naredo, J. P. Sullivan, and C. Ryan, "Grape: grammatical algorithms in python for evolution," *Signals*, vol. 3, no. 3, pp. 642–663, 2022.

[232] H. Zhang, A. Zhou, Q. Chen, B. Xue, and M. Zhang, "Sr-forest: A genetic programming based heterogeneous ensemble learning method," *IEEE Transactions on Evolutionary Computation*, 2023.

[233] H. Zhang, Q. Chen, B. Xue, W. Banzhaf, and M. Zhang, "A semantic-based hoist mutation operator for evolutionary feature construction in regression," *IEEE Transactions on Evolutionary Computation*, 2023.

[234] ——, "Modular multi-tree genetic programming for evolutionary feature construction for regression," *IEEE Transactions on Evolutionary Computation*, 2023.

[235] W. G. La Cava, P. C. Lee, I. Ajmal, X. Ding, P. Solanki, J. B. Cohen, J. H. Moore, and D. S. Herman, "A flexible symbolic regression method for constructing interpretable clinical prediction models," *NPJ Digital Medicine*, vol. 6, no. 1, p. 107, 2023.

[236] F. O. de Franca and G. Kronberger, "Reducing overparameterization of symbolic regression models with equality saturation," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2023, pp. 1064–1072.

[237] H. Zhang, A. Zhou, Q. Chen, B. Xue, and M. Zhang, "Genetic programming-based evolutionary feature construction for heterogeneous ensemble learning [hot of the press]," in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, 2023, pp. 49–50.

[238] S. Jha, S. Gleyzer, E. Reinhardt, V. Baules, N. Okada, and F. Charton, "Evolutionary and transformer based methods for symbolic regression," in *Proceedings of the NeurIPS 2024 Workshop on Machine Learning and the Physical Sciences.→(Symbolic Regression with Transformers)*, 2024.

[239] H. Zhang, Q. Chen, B. Xue, W. Banzhaf, and M. Zhang, "Sharpness-aware minimization for evolutionary feature construction in regression," *arXiv preprint arXiv:2405.06869*, 2024.

[240] D. Parra, D. Joedicke, J. M. Velasco, G. Kronberger, and J. I. Hidalgo, "Learning difference equations with structured grammatical evolution for postprandial glycaemia prediction," *IEEE Journal of Biomedical and Health Informatics*, 2024.

[241] H. Zhang, Q. Chen, B. Xue, W. Banzhaf, and M. Zhang, "A semantic-based hoist mutation operator for evolutionary feature construction in regression [hot off the press]," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2024, pp. 65–66.

[242] T.-C. Chiang, C.-H. Chang, and T.-L. Yu, "A novel symbolic regressor enhancer using genetic programming," in *2024 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2024, pp. 1–8.

[243] H. Zhang, Q. Chen, B. Xue, W. Banzhaf, and M. Zhang, "A geometric semantic macro-crossover operator for evolutionary feature construction in regression," *Genetic Programming and Evolvable Machines*, vol. 25, no. 1, p. 2, 2024.

[244] ——, "Bias-variance decomposition: An effective tool to improve generalization of genetic programming-based evolutionary feature construction for regression," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2024, pp. 998–1006.

[245] L. Cárdenas Florido, L. Trujillo, D. E. Hernandez, and J. M. Muñoz Contreras, "M5gp: Parallel multidimensional genetic programming with multidimensional populations for symbolic regression," *Mathematical and Computational Applications*, vol. 29, no. 2, p. 25, 2024.

[246] L. Kammerer, G. Kronberger, and S. Winkler, "Bias and variance analysis of contemporary symbolic regression methods," *Applied Sciences*, vol. 14, no. 23, p. 11061, 2024.

[247] H. Zhang, Q. Chen, B. Xue, W. Banzhaf, and M. Zhang, "P-mixup: Improving generalization performance of evolutionary feature construction with pessimistic vicinal risk minimization," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2024, pp. 201–220.

[248] P. Ma, T.-H. Wang, M. Guo, Z. Sun, J. B. Tenenbaum, D. Rus, C. Gan, and W. Matusik, "Llm and simulation as bilevel optimizers: a new paradigm to advance physical scientific discovery," in *Proceedings of the 41st International Conference on Machine Learning*, ser. ICML'24, 2024.

[249] A. Ni and L. Spector, "Leveraging symbolic regression for heuristic design in the traveling thief problem," *arXiv preprint arXiv:2404.12750*, 2024.

[250] D. Li, J. Yin, J. Xu, X. Li, and J. Zhang, "Visymre: Vision-guided multimodal symbolic regression," *arXiv preprint arXiv:2412.11139*, 2024.

[251] P. Kahlmeyer, J. Giesen, M. Habeck, and H. Voigt, "Scaling up unbiased search-based symbolic regression," in *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, 2024, pp. 4264–4272.

[252] S. Powers and C. Pinciroli, "Automatic extraction of symbolic models of collective behaviors with graph neural networks and macro-micro evolution," *Swarm Intelligence*, vol. 19, no. 2, pp. 147–171, 2025.

[253] S. Ye and M. Li, "Ab initio nonparametric variable selection for scalable symbolic regression with large p," in *Forty-second International Conference on Machine Learning*, 2025.

[254] J. M. Muñoz Contreras, L. Trujillo, D. E. Hernandez, and L. A. Cardenas Florido, "Benchmarking gsgp: Still competitive 10 years later?" *Genetic Programming and Evolvable Machines*, vol. 26, no. 1, p. 6, 2025.

[255] Z. Ma and J. Zhong, "Gesr: A geometric evolution model for symbolic regression." *Evolutionary Computation*, pp. 1–27, 2025.

[256] H. Zhang, Q. Chen, W. Banzhaf, M. Zhang *et al.*, "Rag-sr: Retrieval-augmented generation for neural symbolic regression," in *The Thirteenth International Conference on Learning Representations*, 2025.

[257] A. Geiger, D. Sobania, and F. Rothlauf, "Selection methods in genetic programming: A performance analysis," in *Proceedings of the Genetic*

and Evolutionary Computation Conference Companion, 2025, pp. 31–32.

[258] J. M. Simões, N. Lourenço, and P. Machado, "Desire-driven selection: An epigenetic experiment in genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2025, pp. 1044–1052.

[259] W.-Z. Fang, C.-H. Chang, J.-C. Liu, and T.-L. Yu, "Genetic programming with ranging-binding mechanism for symbolic regression," in *International Joint Conference on Computational Intelligence*. Springer, 2025, pp. 37–51.

[260] H. Zhang, Q. Chen, B. Xue, W. Banzhaf, and M. Zhang, "Las-gp: Harnessing large semantic libraries for evolutionary feature construction in symbolic regression," *IEEE Transactions on Evolutionary Computation*, 2025.

[261] ——, "Llm-meta-sr: In-context learning for evolving selection operators in symbolic regression," *arXiv preprint arXiv:2505.18602*, 2025.

[262] D. Yang, S. Ding, L. Pan, and Y. Xu, "Vsg-fc: A combined virtual sample generation and feature construction model for effective prediction of surface roughness in polishing processes," *Micromachines*, vol. 16, no. 6, p. 622, 2025.

[263] H. Zhang, Q. Chen, B. Xue, W. Banzhaf, and M. Zhang, "Enhancing generalization in evolutionary feature construction for symbolic regression through vicinal jensen gap minimization," *IEEE transactions on evolutionary computation*, 2025.

[264] ——, "A general feature-informed crossover for two-stage feature selection in symbolic regression." in *CEC*, 2025, pp. 1–8.

[265] W. Ying, H. Bai, N. Gong, X. Wang, S. Dong, H. Chen, and Y. Fu, "Bridging the domain gap in equation distillation with reinforcement feedback," *arXiv preprint arXiv:2505.15572*, 2025.

[266] G. S. Imai Aldeia, H. Zhang, G. Bomarito, M. Cranmer, A. Fonseca, B. Burlacu, W. G. La Cava, and F. O. de França, "Call for action: towards the next generation of symbolic regression benchmark," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2025, pp. 2529–2538.

[267] K. S. Fong and M. Motani, "Pareto-optimal fronts for benchmarking symbolic regression algorithms," in *Forty-second International Conference on Machine Learning*, 2025.

[268] G. S. Imai Aldeia, J. D. Romano, F. Olivetti de França, D. S. Herman, and W. G. La Cava, "Towards symbolic regression for interpretable clinical decision scores," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 384, no. 2317, 2026.

[269] W. Ying, J. Zhang, H. Bai, N. Gong, X. Wang, K. Liu, C. K. Reddy, and Y. Fu, "Data-efficient symbolic regression via foundation model distillation," *arXiv preprint arXiv:2508.19487*, 2025.

[270] H. Zhao, H. Zhang, C. Tian, Z. Wei, and A. Zhou, "Srlinear: Lightweight long-term time series forecasting via symbolic regression," *IEEE Transactions on Artificial Intelligence*, 2025.

[271] G. K. R. Lau, Y. R. Kang, Z.-Y. Khoo, A. Hemachandra, R. W. T. Chew, and B. K. H. Low, "Readme: Rapid equation discovery with multimodal encoders," in *2nd AI for Math Workshop@ ICML 2025*, 2025.

[272] H. Zhang, A. Tonda, Q. Chen, B. Xue, E. Lutton, and M. Zhang, "Micro-step time-series regression: Insights from system identification using symbolic regression," in *European Conference on Genetic Programming (Part of EvoStar)*. Springer, 2025, pp. 240–256.

[273] O. Boussif, M. Mahfoud, Y. Kaddar, M. Jain, S. Li, D. Fornasiere, X. Chen, Y. Bengio, and N. Malkin, "Bayesian symbolic regression with entropic reinforcement learning," 2025.

[274] Z. Yu, G. Wang, J. Ding, H. Wang, and Y. Li, "Beyond formula complexity: Effective information criterion improves performance and interpretability for symbolic regression," *arXiv preprint arXiv:2509.21780*, 2025.

[275] E. R. Pantridge and L. Spector, "Evolution of layer based neural networks: Preliminary report," in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, 2016, pp. 1015–1022.

[276] W. L. Cava, P. C. Lee, I. Ajmal, X. Ding, P. Solanki, J. B. Cohen, J. H. Moore, and D. S. Herman, "Application of concise machine learning to construct accurate and interpretable ehr computable phenotypes," *medRxiv*, 2020.

[277] M. Sipper, J. H. Moore, and R. J. Urbanowicz, "Automatically balancing model accuracy and complexity using solution and fitness evolution (safe)," *arXiv preprint arXiv:2206.15409*, 2022.

[278] P. Orzechowski, P. Renc, W. La Cava, J. H. Moore, A. Sitek, J. Wąs, and J. Wagenaar, "A comparative study of gp-based and state-of-the-art classifiers on a synthetic machine learning benchmark," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2022, pp. 276–279.

[279] P. Carvalho, B. Ribeiro, N. M. Rodrigues, J. E. Batista, L. Vanneschi, and S. Silva, "Feature selection on epistatic problems using genetic algorithms with nested classifiers," in *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*. Springer, 2023, pp. 656–671.

[280] C. Xue, Y. Jin, A. C. Revilla, J. Chakraborty, and M. Chen, "Autogets: Automated generation of text synthetics for improving text classification," 2025.

[281] H. Zhang, Q. Chen, B. Xue, Y. Wang, A. Zhou, and M. Zhang, "Evofeat: Genetic programming-based feature engineering approach to tabular data classification," in *Genetic Programming Theory and Practice XXI*. Springer, 2025, pp. 27–49.

[282] Q. Fan, Y. Zhang, X. Yao, Y. Bi, B. Xue, and M. Zhang, "Es-gp: An ensemble surrogate-assisted genetic programming approach to image classification," *IEEE Transactions on Evolutionary Computation*, 2025.

[283] D. Reyes Fernández de Bulnes and P. Rosati, "Grammatical evolution-based customer behaviour prediction with feature encapsulation by stages," *Available at SSRN 5913172*.

[284] C. Ryan, M. K. Tetteh, and D. M. Dias, "Behavioural modelling of digital circuits in system verilog using grammatical evolution," in *International Joint Conference on Computational Intelligence*, 2020, pp. 28–39.

[285] M. K. Tetteh, D. Mota Dias, and C. Ryan, "Evolution of complex combinational logic circuits using grammatical evolution with systemverilog," in *European Conference on Genetic Programming (Part of EvoStar)*. Springer, 2021, pp. 146–161.

[286] C. Ryan, M. Tetteh, J. McEllin, D. Mota-Dias, R. Conway, and E. Naredo, "Addc: Automatic design of digital circuit," in *Genetic Algorithms*. IntechOpen, 2022.

[287] M. Tetteh, D. M. Dias, and C. Ryan, "Grammatical evolution of complex digital circuits in systemverilog," *SN Computer Science*, vol. 3, no. 3, p. 188, 2022.

[288] M. Tetteh, A. de Lima, J. McEllin, A. Murphy, D. M. Dias, and C. Ryan, "Evolving multi-output digital circuits using multi-genome grammatical evolution," *Algorithms*, vol. 16, no. 8, p. 365, 2023.

[289] J. M. Moore and A. J. Clark, "Supervision and evolution: Pretraining neural networks for quadrupedal locomotion," in *ALIFE 2021: The 2021 Conference on Artificial Life*. MIT Press, 2021.

[290] K. J. A. Ahumada, J. M. Moore, and A. J. Clark, "Does kinematic-based pretraining improve evolution of quadrupedal gaits?" in *ALIFE 2023: Ghost in the Machine: Proceedings of the 2023 Artificial Life Conference*. MIT Press, 2023.

[291] P. Jiménez, J. C. Roldán, and R. Corchuelo, "A clustering approach to extract data from html tables," *Information Processing & Management*, vol. 58, no. 6, p. 102683, 2021.

[292] ——, "A coral-reef approach to extract information from html tables," *Applied Soft Computing*, vol. 115, p. 107980, 2022.

[293] J. Kim and S. Ha, "Energy-aware scenario-based mapping of deep learning applications onto heterogeneous processors under real-time constraints," *IEEE Transactions on Computers*, vol. 72, no. 6, pp. 1666–1680, 2022.

[294] P. Jiménez, J. C. Roldán, and R. Corchuelo, "On exploring data lakes by finding compact, isolated clusters," *Information Sciences*, vol. 591, pp. 103–127, 2022.

[295] M. Xu, Y. Mei, F. Zhang, and M. Zhang, "Genetic programming for dynamic flexible job shop scheduling: Evolution with single individuals and ensembles," *IEEE Transactions on Evolutionary Computation*, 2023.

[296] M. Foreback, S. Leither, and E. Dolson, "Using evolutionary computation to find parameters that promote egalitarian major evolutionary transitions," in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, 2023, pp. 135–138.

[297] D. Parra, A. Gutiérrez-Gallego, J. M. Velasco, R.-J. Villanueva, and J. I. Hidalgo, "Modelling the transmission dynamics of obesity: A multi-objective approach using dynamic structured grammatical evolution," in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, 2023, pp. 73–74.

[298] J. Garbus, T. Willkens, A. Lalejini, and J. Pollack, "Accelerating co-evolutionary learning through phylogeny-informed interaction estimation," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2024, pp. 427–430.

[299] L. Spector, L. Ding, and R. Boldi, "Particularity," in *Genetic Programming Theory and Practice XX*. Springer, 2024, pp. 159–176.

[300] M. Manavi, "Multi-objective genetic algorithm for materialized view optimization in data warehouses," in *2024 4th Interdisciplinary Conference on Electrics and Computer (INTCEC)*. IEEE, 2024, pp. 1–6.

[301] J. Garbus, T. Willkens, A. Lalejini, and J. Pollack, "Phylogeny-informed interaction estimation accelerates co-evolutionary learning," in *Artificial Life Conference Proceedings 36*, vol. 2024, no. 1. MIT Press, 2024, p. 66.

[302] M. L. López, J.-C. Cortés, M. T. Guillén, and R.-J. Villanueva, "Mathematical modeling for the window to the brain: application of the hybrid digital twins to enhance precision," in *Mathematical Modelling in Engineering Human Behaviour*. Springer, 2024, pp. 156–161.

[303] T. Guo, Y. Mei, M. Zhang, R. Sun, Y. Zhu, and W. Du, "Genetic programming with multi-fidelity surrogates for large-scale dynamic air traffic flow management," *IEEE Transactions on Evolutionary Computation*, 2024.

[304] J. G. Frazier and T. Helmuth, "Explaining automatically designed software defined perimeters with a two phase evolutionary computation system," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2024, pp. 1527–1535.

[305] I. M. Putrama and P. Martinek, "Self-supervised data lakes discovery through unsupervised metadata-driven weighted similarity." *Inf. Sci.*, vol. 662, p. 120242, 2024.

[306] X. Xue and J. C.-W. Lin, "Two-phase similarity feature construction for enhancing sensor knowledge graph alignment via genetic programmings," *IEEE Internet of Things Journal*, 2025.

[307] B. Palamutçuoğlu, S. Çavuşoğlu, A. Çamlı, F. Virlanuta, S. Bacalum, D. Züngün, and F. Moisescu, "Solution of the capacity-constrained vehicle routing problem considering carbon footprint within the scope of sustainable logistics with genetic algorithm," *Sustainability (Switzerland)*, vol. 17, no. 2, 2025.

[308] M. L. López, M. T. Guillén, J.-C. Cortés, and R.-J. Villanueva, "Enhancing precision in window to the brain modeling: Methodology and implementation of hybrid digital twins," *Journal of Industrial Information Integration*, vol. 48, p. 100973, 2025.

[309] V. Stanovov and E. Semenkin, "Genetic improvement of dynamic optimization algorithms using pushgp," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2025, pp. 1976–1983.

[310] C. Sigrist, A. Li, A. Zhang, A. Lechowicz, N. Bashir, P. Lertsaroj, R. Bahlous-Boldi, and M. Hajiesmaili, "Multi-objective evolutionary learning for near pareto-optimal optimization of solar deployment," in *Proceedings of the 12th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, 2025, pp. 213–223.

[311] C. Yu, X. Cao, B. Zhang, W. Du, and T. Guo, "Coevolutionary genetic programming for large-scale dynamic multi-aircraft task allocation," *Frontiers of Information Technology & Electronic Engineering*, vol. 26, no. 12, pp. 2440–2454, 2025.

[312] M. Llamazares and R.-J. Villanueva, "Analysis of rsv hospitalizations in infants under 1 year: Application of hybrid digital twins," *Modelling for Engineering & Human Behaviour*, 2025.

[313] R. S. Olson, W. La Cava, P. Orzechowski, R. J. Urbanowicz, and J. H. Moore, "Pmlb: a large benchmark suite for machine learning evaluation and comparison," *BioData mining*, vol. 10, no. 1, p. 36, 2017.

[314] J. Lehman, J. Gordon, S. Jain, K. Ndousse, C. Yeh, and K. O. Stanley, "Evolution through large models," in *Handbook of evolutionary machine learning*. Springer, 2023, pp. 331–366.

[315] A. E. Brownlee, J. Callan, K. Even-Mendoza, A. Geiger, C. Hanna, J. Petke, F. Sarro, and D. Sobania, "Enhancing genetic improvement mutations using large language models," in *International symposium on search based software engineering*. Springer, 2023, pp. 153–159.

[316] K. Even-Mendoza, A. Brownlee, A. Geiger, C. Hanna, J. Petke, F. Sarro, and D. Sobania, "Llm-guided genetic improvement: Envisioning semantic aware automated software evolution," in *2025 40th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2025, pp. 3902–3906.

[317] A. E. Brownlee, J. Callan, K. Even-Mendoza, A. Geiger, C. Hanna, J. Petke, F. Sarro, and D. Sobania, "Large language model based mutations in genetic improvement," *Automated Software Engineering*, vol. 32, no. 1, p. 15, 2025.

[318] P. Anthes, D. Sobania, and F. Rothlauf, "Transformer semantic genetic programming for symbolic regression," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2025, pp. 952–960.

[319] L. Jiang, Y. Chai, M. Li, M. Liu, R. Fok, N. Dziri, Y. Tsvetkov, M. Sap, and Y. Choi, "Artificial hivemind: The open-ended homogeneity of language models (and beyond)," in *Advances in Neural Information Processing Systems*, vol. 38, 2025.

[320] H. Bradley, A. Dai, H. Teufel, J. Zhang, K. Oostermeijer, M. Bellagente, J. Clune, K. Stanley, G. Schott, and J. Lehman, "Quality-diversity through ai feedback," in *International Conference on Learning Representations*, vol. 2024, 2024, pp. 21 036–21 147.

[321] E. Hughes, M. Dennis, J. Parker-Holder, F. Behbahani, A. Mavalankar, Y. Shi, T. Schaul, and T. Rocktäschel, "Position: open-endedness is essential for artificial superhuman intelligence," in *Proceedings of the 41st International Conference on Machine Learning*, 2024, pp. 20 597–20 616.

[322] A. Novikov, N. Vũ, M. Eisenberger, E. Dupont, P.-S. Huang, A. Z. Wagner, S. Shirobokov, B. Kozlovskii, F. J. Ruiz, A. Mehrabian *et al.*, "Alphaevolve: A coding agent for scientific and algorithmic discovery," *arXiv preprint arXiv:2506.13131*, 2025.