Enhancing Evolutionary Computation through Phylogenetic Analysis

Chenfei Peng
peng0368@morris.umn.edu
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA

Abstract

In this paper, we will provide an overview of the paper "Phylogeny-informed fitness estimation for test-based parent selection" by Lalejini, et al. [7] Phylogenies, or ancestry trees, provide a detailed look into the evolutionary journey of a population. In evolutionary computation, a phylogeny can represent the progress of an evolutionary algorithm through a search space. Although phylogenetic analysis is mainly used to deepen the understanding of evolutionary algorithms after they have been run, this study explores its potential use in real-time to enhance parent selection during evolutionary searches. The research by Lalejini, et al. introduces the concept of phylogeny-informed fitness estimation, leveraging a population's phylogeny to predict fitness values. This method is tested using both down-sampled lexicase and cohort lexicase selection algorithms across four genetic programming (GP) problems. The findings suggest that using phylogenies to estimate fitness values can improve the performance of down-sampled lexicase selection, fostering better diversity and search space exploration.

Keywords: genetic programming · parent selection · phylogeny · lexicase selection

1 Introduction

This paper explores evolutionary computation and genetic programming – two subfields within artificial intelligence that solve complex problems by simulating natural evolutionary processes. In this context, we focus on a novel idea introduced by Lalejini et al. [7], which is the use of phylogeny-informed fitness estimation. This concept uses the evolutionary history of a population to enhance the the process of selecting parents during genetic programming.

The traditional methods of selecting parents in genetic programming often face challenges such as premature convergence, where the algorithm settles on suboptimal solutions too early, which limits exploration of the possible solutions. Lalejini et al. suggest that by understanding the ancestry or phylogeny of the population, we can better estimate the fitness of individuals, which can potentially lead to more diverse and effective solutions.

By integrating the evolutionary history into the fitness assessment, the proposed method aims to improve the exploration of the search space and maintain diversity within the population. This could help prevent the algorithm from getting stuck in less optimal solutions and enhance the discovery of more innovative and high-quality solutions.

In the next section we will provide background materials about Evolutionary Computation (EC) and Genetic Programming (GP) . Then we will cover important subtopics such as phylogeny (ancestry tree) and parent selection, as well as lexicase selection algorithms. In Section 3 we will describe the main method used by Lalejini, et al. [7], phylogeny-informed fitness estimation, following with their experimental design in Section 4. Finally we will go through the results of their experiments and their conclusions.

2 Background

The exploration of genetic and evolutionary biology is not only about understanding the lineage relationships among organisms but also involves investigating the mechanisms driving their evolution and the methods used for selecting traits that enhance adaptation to environments. This field offers a wealth of methodologies and concepts focused on tracing the ancestry of species and illustrating the interconnected web of life that unites all living beings.

To bridge the gap between biology and evolutionary computation (EC), the subsequent sections will cover several key concepts crucial to both fields: evolutionary computation, genetic programming, phylogeny (ancestry tree), parent selection, and lexicase selection algorithms such as down-sampled lexicase selection and cohort lexicase selection.

Furthermore, the exploration of these concepts extends well beyond biological scope, offering profound insights into the field of evolutionary computation. By understanding and applying these fundamental biological concepts, researchers are equipped to simulate and enhance processes similar to natural selection and evolution within computational environments, which advances people's capabilities in solving complex real-world computational problems.

2.1 Evolutionary Computation

Evolutionary computation (EC) stands as a cornerstone in the realm of bio-inspired algorithms, drawing from the fundamental processes of natural selection and genetics. This computational approach simulates the evolutionary process to iteratively improve solutions to complex problems. EC embodies a suite of algorithms, including evolutionary strategies, evolutionary programming, genetic algorithms, and genetic programming. The versatility of EC allows for its application across a myriad of fields, from optimizing engineering designs to solving intricate optimization problems in finance and logistics. By simulating the adaptive processes of natural evolution, evolutionary computation facilitates the discovery of high-quality solutions that conventional approaches might overlook.

2.2 Genetic Programming

Genetic programming (GP) is a specialized branch of evolutionary computation that focuses on the evolution of computer programs or symbolic expressions to solve specific tasks [1]. Unlike other branches of EC that evolve fixed-size strings or vectors, GP evolves tree-like structures or variable-length vectors representing programs or mathematical models. This approach enables GP to automatically discover algorithms, mathematical expressions, or software programs that perform well against predefined fitness criteria. GP has been successfully applied in diverse domains such as automated software engineering, symbolic regression, and even the evolution of control algorithms for robotics and artificial intelligence tasks. By evolving programmatic structures, genetic programming transcends traditional optimization, offering a powerful tool for automatic programming, problem-solving, and knowledge discovery.

Figure 1 is a flowchart that demonstrates the rough idea of how genetic programming works in a real-world scenario when applied to a specific task. In genetic programming, the process begins by generating an initial random population, marking the start of the first generation (Gen = 0). Each individual within this population is then evaluated to determine its fitness. Based on these fitness evaluations, individuals are selected for further genetic operations such as mutation. After an individual undergoes mutation, the resultant mutant is added back into the population.

The loop of evaluating fitness, selecting individuals, mutating, and reintegrating continues until the population reaches the desired size (M). If the population size is achieved, but the termination criteria – such as a maximum number of generations – are not yet met, the generation counter is increased by 1 (Gen = Gen + 1), and it will restart the process of the evaluation of fitness. This loop repeats until the termination criteria are satisfied, at which point the process concludes and the results are designated.

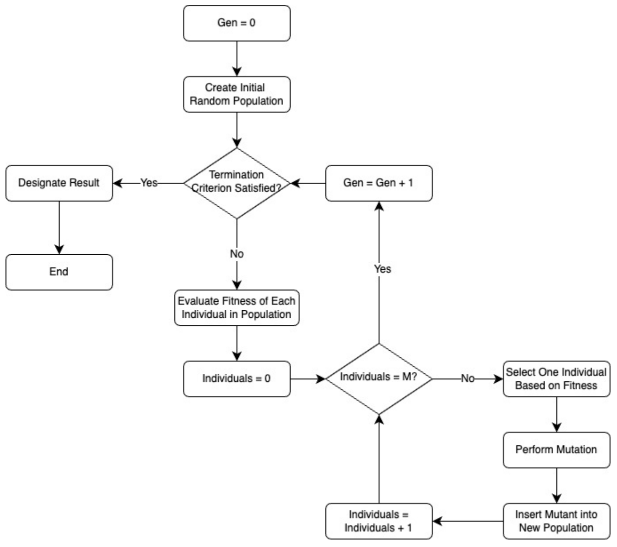

Gen = 0
Create Initial Random Population
Termination Criterion Satisfied?
Yes
Designate Result
End
Gen = Gen + 1
No
Evaluate Fitness of Each Individual in Population
Yes
Individuals = 0
Individuals = M?
No
Select One Individual Based on Fitness
Perform Mutation
Insert Mutant into New Population
Individuals = Individuals + 1

Figure 1. Flowchart for Genetic Programming [5]

2.3 Phylogeny (Ancestry Tree)

Phylogeny refers to the evolutionary history and lineage relationships among species or groups of organisms, which can be traced back through time to reveal the branching patterns of evolution. Phylogenetic trees, which graphically represent these relationships, are fundamental tools in biology for understanding biodiversity, evolution, ecology, and genomes. They depict how species diverge from common ancestors over time.

Similarly, phylogenetic trees in evolutionary computation can also demonstrate the lineage relationships between different individuals. Figure 2 is an example of phylogenetic tree in EC. As shown in Figure 2, node A produces node B; node B produces node C and D; node C produces node E; node D produces node F and node G. There are many details in this figure, and we will cover them in subsequent sections.

2.4 Parent Selection

Parent selection is a process in genetic algorithms and evolutionary computation where individual solutions are chosen, based on their performance, to contribute to the next generation [2]. This step is crucial for guiding the evolutionary process towards optimal solutions, as it influences the genetic diversity and quality of subsequent populations. Techniques for parent selection vary, including roulette wheel selection, tournament selection, and rank selection [6], each with its advantages and disadvantages in maintaining diversity and pressure towards optimal solutions.

2.5 Lexicase Selection

Lexicase selection is a parent selection algorithm tailored for test-based problem-solving environments, particularly where candidates are evaluated based on a series of input-

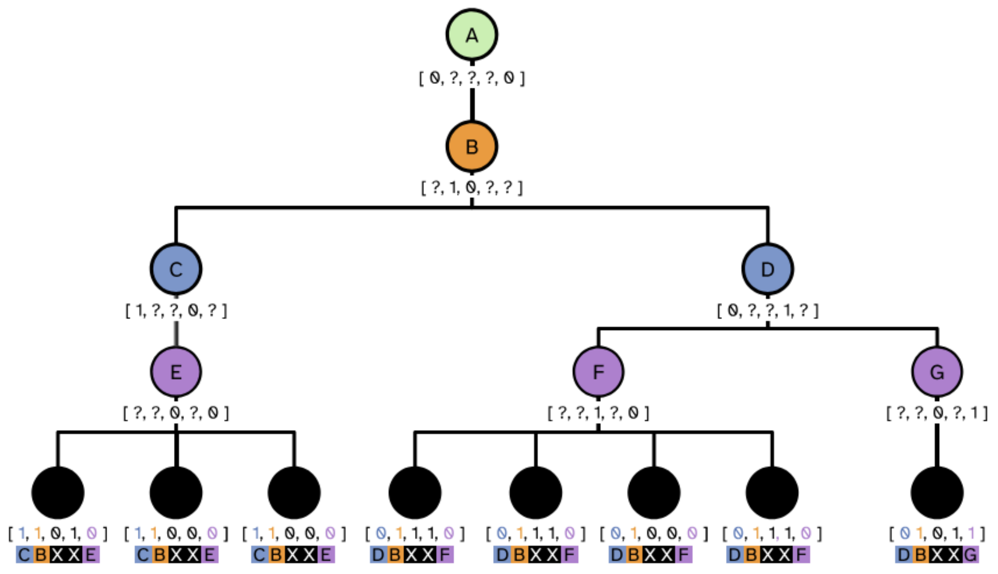


Figure 2. Example diagram of ancestor-based phylogeny-informed fitness estimation [7]

Algorithm 1 Lexicase Selection [9]

```
1:  Result: Individual to be used as a parent
2:  candidates ← entire population
3:  cases ← list of all test cases in a random order
4:  while True do
5:      candidates ← candidates who perform best on cases[0]
6:      if only one candidate exists in candidates then
7:          return candidate
8:      end if
9:      if cases is empty then
10:         return a randomly selected candidate from candidates
11:     end if
12:     delete cases[0]
13: end while
```

output training cases. The vector under each individual in Figure 2 represents the scores of each individual on 5 separate training cases. Here “0” means that the individual fails the training case, while “1” means that it passes. And “?” means that the training case is not evaluated. These training cases set the foundation for a selection process that starts by randomizing the order of the training cases. At the beginning of the selection process, the entire population is viewed as potential candidates.

The selection process iteratively narrows down this pool of candidates by retaining only those performing best on the current training case being considered, as detailed in Algorithm 1. This step-by-step filtering, following the randomized sequence of training cases, ensures that only the most promising candidates for this particular ordering of the training cases remain. If, at any point, only one candidate remains, that candidate is immediately selected. If, after evaluating all training cases, multiple candidates still qualify, the final parent is chosen randomly among them. In addition, shuffling the training cases will help preserve the diversity of lexicase selection. A series of training cases in different order sometimes may result in different selection results.

This approach, while thorough and effective, is also known for its computational demands because it requires each individual to be evaluated against every training case.

2.5.1 Down-sampled Lexicase Selection.

To reduce the computational demands in standard lexicase selection, the down-sampled variant introduces a strategy of random subsampling of the training set for each generation. This approach limits the evaluation to a subset of the original training cases, largely reducing the computational demands [4].

So how does down-sampled lexicase selection actually work? Let’s go back to Figure 2 and focus on the blue and purple generations (from node C to node G). The blue generation cares about the 1st and 4th training cases, while the purple one cares about the 3rd and 5th. This shows that the subsampled training sets are usually different from each generation. However, the individuals under a certain generation will always share the same subsampled training sets.

2.5.2 Cohort Lexicase Selection.

Cohort lexicase selection further refines the process by dividing both the training set and the population into equally sized cohorts, based on a specified subsample levels. For instance, a subsample level of 20% divides the population and training set into 5 evenly sized cohorts. With cohort memberships randomly assigned each generation, this method facilitates a focused evaluation where each group of candidates is tested against only its matched set of training cases.

Despite the reduced per-generation evaluation scope, the integrity of utilizing the full set of training cases for selection purposes is maintained. The selection of a parent involves first randomly choosing a cohort and then applying standard lexicase selection within it [4].

3 Phylogeny-informed Fitness Estimation

Phylogeny-informed fitness estimation is a strategy for enhancing the efficiency and efficacy of evolutionary algorithms. The method involves dynamically subsampling a set of discrete fitness test cases, which enhances the efficiency by reducing the number of evaluations each individual undergoes. However, this efficiency comes with a restriction: it limits the selection process to a subset of available information, potentially overlooking vital data that could influence the selection of the fittest candidates.

Phylogeny-informed fitness estimation addresses this challenge by using the lineage information of the population. While the actual evaluation is confined to the subsampled training cases, this approach leverages the phylogenetic data to infer an individual's likely performance on the non-evaluated portions of the training set. This strategy ensures that the selection process remains robust, allowing Lalejini et al. to consider an individual's capability more comprehensively, without directly testing every single case [7].

3.1 Ancestor-based Estimation

The first method under the scope of phylogeny-informed fitness estimation is ancestor-based estimation. This technique confines the fitness estimation to the direct line of ancestry of the individual whose performance we're currently estimating, tapping into the ancestral lineage to glean insights into performance on unevaluated training cases. Specifically, it involves tracing the lineage of an individual backward through its ancestors – parent, grandparent, and so on – until it identifies the nearest ancestor that has been evaluated against the training case we care about. The performance score of this ancestor on the training case then serves as a proxy for the individual's own score. This method not only preserves the integrity of the evolutionary process by relying on genetically linked performance indicators but also streamlines the computational demands by limiting the search to direct ancestors, making it an efficient option for integrating phylogenetic insights into the fitness estimation process.

So how does ancestor-based estimation work in evolutionary computation? Let's go back to Figure 2 again and focus on the right-most black node. Cross marks here indicate the evaluated training cases, while the colorful numbers mean the estimated ones. But where do these colorful numbers come from? The estimation of the 1st training case is 0, which comes from D. Since there is a "?" on the 1st training case of G, D becomes the nearest ancestor that has been evaluated against the 1st training case. Using the same logic, we can also figure out why the estimation of the 2nd training case comes from B and why the 5th comes from G.

3.2 Relative-based Estimation

In contrast, relative-based estimation expands the horizon of the search for fitness estimates beyond direct ancestors to include any genetically related individuals within the population's phylogeny. Employing a breadth-first search strategy starting from the focal individual, this method seeks out the nearest relative that has been evaluated against the specific training case, regardless of their direct ancestral relation. This could include "cousin" or even more distantly related individuals, provided they offer relevant performance data for the training case at hand.

Despite its potential for drawing estimates from a wider subset of the population in estimating fitness, the relative-based approach necessitates a more complex and potentially computationally intensive search process, highlighting the trade-offs between breadth of search and efficiency in the context of phylogeny-informed fitness estimation.

4 Genetic Programming Experiments

In the exploration of the efficacy of phylogeny-informed fitness estimation within genetic programming (GP), Lalejini, et al. conducted a series of experiments focusing on the impact of ancestor-based estimation, relative-based estimation, and a control condition without estimation [7]. These experiments were designed to assess problem-solving success across 4 genetic programming problems – *Median*, *Small or Large*, *Grade*, and *Fizz Buzz*, sourced from PSB1 and PSB2 benchmark suites [3]. We will cover the details of these problems in Section 4.2.

Besides, the experiments were structured to evaluate the performance of these phylogeny-informed strategies using both down-sampled and cohort lexicase selection methods across four subsampling levels: 1%, 5%, 10%, and 50%. A critical modification in these experiments was that the search depth was limited to 5, which basically means that the estimation of an individual can only based on the performance of related ancestors or relatives in the former 5 generations.

4.1 GP System

The experimental setup involved running 30 replicates for each condition, utilizing a population of 1,000 linear genetic programs developed using the SignalGP representation [8]. This setup aimed to simulate a controlled environment that allowed for the asexual reproduction of programs, with mutations introduced to offspring through single-instruction insertions, deletions, and substitutions.

4.2 Program Synthesis Problems

The selected program synthesis problems involved a range of introductory programming challenges, each with specific requirements and constraints. The *Median* problem tasked programs with determining the median value from three integer inputs, while the *Small or Large* problem required programs to classify an integer as “small”, “large” or “neither” based on predefined thresholds. The *Grade* problem involved assigning letter grades based on input scores and predefined thresholds, and the *Fizz Buzz* problem challenged programs to output “Fizz”, “Buzz”, “FizzBuzz”, or the input integer based on divisibility criteria.

These problems were chosen not only for their diversity but also for their ability to test the GP system’s problem-solving capabilities under various conditions. Each problem was associated with a set of 100 training cases for program evaluation and selection, and 1,000 test cases for determining problem-solving success, with the aim of including input-output edge cases in both sets.

Through this experimental framework, the study sought to uncover the potential benefits and limitations of incorporating phylogeny-informed fitness estimation techniques in genetic programming. By comparing the outcomes of ancestor-based and relative-based estimation against a control condition without estimation, the experiments aimed to provide insights into the strategies that could enhance the problem-solving success of GP systems across a spectrum of programming challenges.

4.3 Statistical Analyses

In this study, the analyses were strictly confined to 3 specific fitness estimation treatments without cross-comparison across different problems, subsampling methods (either down-sampling or cohorts), or levels of down-sampling (1%, 5%, 10%, and 50%). Only pairwise comparisons were made between methods that were consistent in problem type, subsampling technique, and down-sampling percentage.

Statistical assessments of the distributions from various treatments involved the use of Kruskal–Wallis tests to identify significant differences across independent conditions. Subsequent to identifying significant differences with the Kruskal–Wallis test ($p < 0.05$), pairwise distinctions were explored through Wilcoxon rank-sum tests. For evaluating differences in problem-solving success rates, pairwise Fisher’s exact tests were applied, also at a significance threshold of 0.05. Multiple comparison corrections were applied using the Holm–Bonferroni method as necessary.

5 Results

Based on Lalejini, et al.’s research [7], there are 3 main results from their experiments. The first one is that phylogeny-informed fitness estimation reduces diversity loss caused by subsampling. Secondly, it improves poor exploration caused by down-sampling. Both results are largely based on two diagnostic experiments which are not covered here due to lack of space. As a result we will mainly focus on the third result: Phylogeny-informed fitness estimation can enable extreme subsampling for some genetic programming problems.

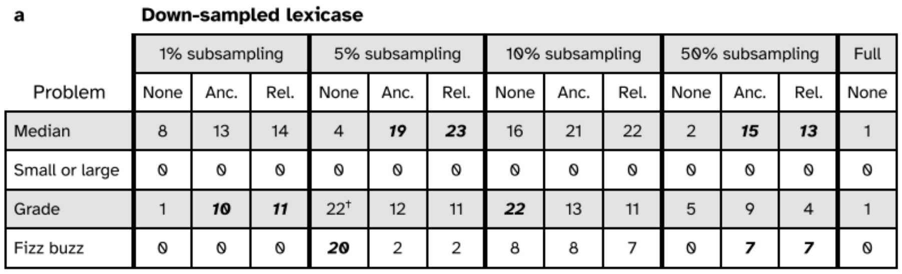
Table 1 presents the success rates of different phylogeny-informed estimation methods across a variety of genetic programming problems under both down-sampled and cohort lexicase. The table presents problem-solving success, defined as the production of a program that passes all tests in the testing set, across various subsampling levels and problems, with a total of 32 distinct combinations evaluated.

The majority of comparisons – 20 out of 32 – showed no statistically significant difference in success rates among the different fitness estimation treatments. However, phylogeny-informed methods did result in statistically significantly higher success rates in 4 out of 16 comparisons for down-sampled lexicase and 5 out of 16 for cohort lexicase. In a few cases under down-sampled lexicase, the control without estimation outperformed phylogeny-informed methods. No such instances were detected under cohort lexicase.

The benefits of phylogeny-informed estimation varied depending on the subsampling method and the problem at hand. Interestingly, no significant differences were found between the success rates of ancestor-based and relative-based estimation methods. Cohort lexicase showed a more consistently neutral or positive effect with phylogeny-informed fitness estimation compared to down-sampled lexicase. In some scenarios, such as with the Fizz Buzz problem at 5% down-sampling, not using phylogeny-informed fitness estimation yielded better results, which deviates from diagnostic results where phylogeny-informed estimation was beneficial more consistently.

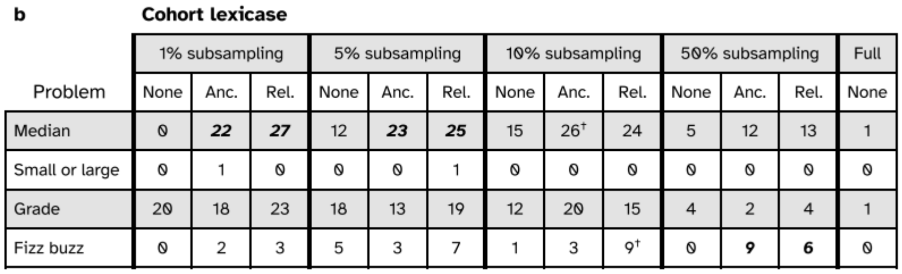
The discrepancies observed suggest that while phylogeny-informed estimation can enhance certain aspects of problem-solving, such as diversity maintenance, it may negatively impact other facets. For instance, inaccuracies in fitness estimation can potentially lead to the selection of suboptimal candidate solutions. These findings highlight the need for further research to refine phylogeny-informed estimation methods and to better understand their effects on evolutionary search processes.

Notably, problem-solving success at 1% subsampling was highlighted as especially significant. Even with such extreme subsampling, resulting in elite selection based on very limited information (a single training case per individual), the no-estimation control still managed to produce successful solutions for the median and grade problems. This indicates that effective problem-solving can sometimes occur even under conditions of extreme subsampling.



a **Down-sampled lexicase**

Problem	1% subsampling			5% subsampling			10% subsampling			50% subsampling			Full
	None	Anc.	Rel.	None	Anc.	Rel.	None	Anc.	Rel.	None	Anc.	Rel.	None
Median	8	13	14	4	***19***	***23***	16	21	22	2	***15***	***13***	1
Small or large	0	0	0	0	0	0	0	0	0	0	0	0	0
Grade	1	***10***	***11***	22†	12	11	***22***	13	11	5	9	4	1
Fizz buzz	0	0	0	***20***	2	2	8	8	7	0	***7***	***7***	0



b **Cohort lexicase**

Problem	1% subsampling			5% subsampling			10% subsampling			50% subsampling			Full
	None	Anc.	Rel.	None	Anc.	Rel.	None	Anc.	Rel.	None	Anc.	Rel.	None
Median	0	***22***	***27***	12	***23***	***25***	15	26†	24	5	12	13	1
Small or large	0	1	0	0	0	1	0	0	0	0	0	0	0
Grade	20	18	23	18	13	19	12	20	15	4	2	4	1
Fizz buzz	0	2	3	5	3	7	1	3	9†	0	***9***	***6***	0

Table 1. Problem-solving success for 2 lexicase selection methods (down-sampled lexicase selection, cohort lexicase selection) on 4 GP problems (*Median*, *Small or Large*, *Grade*, and *Fizz Buzz*), in terms of 3 fitness estimation treatments (ancestor-based estimation, relative-based estimation, and a control condition without estimation) and 4 subsampling levels (1%, 5%, 10%, 50%), out of 30 replicates. Bolded and italicized numbers mean both phylogeny-informed estimation results significantly outperformed the no-estimation control, or the opposite. "†" means only one phylogeny-informed estimation result significantly outperformed the control, or the opposite. [7]

6 Conclusion

This study explores two phylogeny-informed fitness estimation strategies: ancestor-based for quick reference and relative-based for a more comprehensive search of close relatives. The research provides evidence that these methods can offset some of the limitations of lexicase selection by enhancing diversity retention and search space exploration. However, the impact on problem-solving success in genetic programming is not uniform, varying with the problem, subsampling method, and level, as detailed in Table 1.

Moreover, no significant differences were found between the two phylogeny-informed methods, prompting further investigation into their specific impacts on evolutionary search. Currently, ancestor-based estimation is preferred due to its potential for more efficient optimization compared to relative-based estimation.

Lalejini, et al.'s study applied these estimation methods to both down-sampled and cohort lexicase [7]. By doing so, it enabled every selection event in lexicase to consider the entire training set through estimated performances on unevaluated cases, thereby allowing population members to be assessed on varying training subsets.

Future enhancements could involve systematic subsampling to minimize the search depth, thus improving estimation accuracy and problem-solving effectiveness. Additionally, incorporating mutation data into subsampling decisions could further improve evaluations, especially for offspring with significant mutations.

Beyond the scope of this research, Lalejini, et al. see potential for using runtime phylogeny tracking to enhance evolutionary search in various ways [7]. This could be particularly valuable in quality-diversity algorithms, which are a new type of evolutionary algorithms aiming to find a set of high-performing and diverse solutions, with applications in machine learning and robotics [10].

7 Acknowledgements

I would like to thank Nicholas F. McPhee, my senior seminar adviser, for the important feedback and all the help throughout the whole process of composing this paper. I would also like to thank Wenkai Guan, our senior seminar coordinator, who is supervising the progress of each student and always willing to help. Lastly, I would like to thank my alumni reviewer for his detailed feedback.

References

[1] Genetic programming. In M. Giacobini, B. Xue, and L. Manzoni, editors, *Proceedings of EuroGP 2024*, Lecture Notes in Computer Science, pages X, 227, Aberystwyth, UK, April 3-5 2024. Springer Cham. https://doi.org/10.1007/978-3-031-56957-9.

[2] 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*,

pages 237–238, Cancún, Mexico, 2020. ACM. https://doi.org/10.1145/3377929.3389987.

[3] T. Helmuth and P. Kelly. Psb2: The second program synthesis benchmark suite. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 785–794, Lille, France, 2021. ACM. https://doi.org/10.1145/3449639.3459285.

[4] 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 on - GECCO '19*, pages 2028–2031, Prague, Czech Republic, 2019. ACM Press. https://doi.org/10.1145/3319619.3326900.

[5] Imperial College London. Diagram of genetic programming process. https://www.doc.ic.ac.uk/project/examples/2005/163/g0516312/Programming/Process.html, 2024. Accessed: 2024-04-03.

[6] T. T. T. Khuat and M. H. Le. A genetic algorithm with multi-parent crossover using quaternion representation for numerical function optimization. *Applied Intelligence*, 46:810–826, 2017. https://doi.org/10.1007/s10489-016-0867-y.

[7] A. Lalejini, M. A. Moreno, J. G. Hernandez, and E. Dolson. Phylogeny-informed fitness estimation for test-based parent selection. In S. Winkler, L. Trujillo, C. Ofria, and T. Hu, editors, *Genetic Programming Theory and Practice XX*, Genetic and Evolutionary Computation, pages 241–261, Michigan State University, USA, June 1-3 2023. Springer. https://link.springer.com/chapter/10.1007/978-981-99-8413-8_13.

[8] A. Lalejini and C. Ofria. Evolving event-driven programs with signalgp. In *Proceedings of the Genetic and Evolutionary Computation Conference on - GECCO '18*, pages 1135–1142, Kyoto, Japan, 2018. ACM Press. https://doi.org/10.1145/3205455.3205523.

[9] B. Metevier, A. K. Saini, and L. Spector. Lexicase selection beyond genetic programming. In W. Banzhaf, L. Spector, and L. Sheneman, editors, *Genetic Programming Theory and Practice XVI*, Genetic and Evolutionary Computation, pages 123–136. Springer International Publishing, Cham, 2019. https://doi.org/10.1007/978-3-030-04735-1_7.

[10] C. Qian, K. Xue, and R.-J. Wang. Quality-diversity algorithms can provably be helpful for optimization. *arXiv preprint arXiv:2401.10539*, January 2024. https://arxiv.org/abs/2401.10539.