# Untangling the Effects of Down-Sampling and Selection in Genetic Programming

Ryan Boldi[1], Ashley Bao[2], Martin Briesch[3], Thomas Helmuth[4], Dominik Sobania[3], Lee Spector[2,1] and Alexander Lalejini[5]

[1]University of Massachusetts Amherst, Amherst MA, USA
[2]Amherst College, Amherst MA, USA
[3]Johannes Gutenberg University, Mainz, Germany
[4]Hamilton College, Clinton NY, USA
[5]Grand Valley State University, Allendale MI, USA
rbahlousbold@umass.edu



## Abstract

Genetic programming systems often use large training sets to evaluate the quality of candidate solutions for selection, which is often computationally expensive. Down-sampling training sets has long been used to decrease the computational cost of evaluation in a wide range of application domains. More specifically, recent studies have shown that both random and informed down-sampling can substantially improve problem-solving success for GP systems that use the lexicase parent selection algorithm. We test whether these down-sampling techniques can also improve problem-solving success in the context of three other commonly used selection methods, fitness-proportionate, tournament, implicit fitness sharing plus tournament selection, across six program synthesis GP problems. We verified that down-sampling can significantly improve the problem-solving success for all three of these other selection schemes, demonstrating its general efficacy. We discern that the selection pressure imposed by the selection scheme does not interact with the down-sampling method. However, we find that informed down-sampling can improve problem solving success significantly over random down-sampling when the selection scheme has a mechanism for diversity maintenance like lexicase or implicit fitness sharing. Overall, our results suggest that down-sampling should be considered more often when solving test-based problems, regardless of the selection scheme in use.

## Introduction

Genetic programming (GP) applies the principles of Darwinian evolution to automatically synthesize programs instead of writing them by hand. GP systems are commonly used in the context of artificial life for both applied problem-solving (Cava et al., 2021) and studying general evolutionary dynamics (Dolson and Ofria, 2021), as evolved programs can express complex traits while still allowing researchers to fully disentangle the genetic mechanisms implementing those traits (Lenski et al., 2003; Lalejini et al., 2021). GP systems often use large training sets to evaluate the quality of candidate solutions (individuals). These training sets comprise examples of input and output pairs that describe the correct behavior of a program for a given problem. Each generation, individuals are evaluated on these pairs in order to determine whether or not they exhibit this desired behavior, such as returning the correct value for a program synthesis or regression problem. A parent selection algorithm then chooses the "best" individuals to contribute genetic material to the next generation.

To thoroughly assess the quality of individuals in a population, most GP systems evaluate all individuals on every input-output example in the training set. This process can be computationally expensive when using large population sizes on large training sets or when individual evaluations are slow to compute. Reducing these computational costs can increase the scale at which we apply GP, allowing us to solve problems or conduct experiments that would otherwise not be possible. Down-sampling has been shown to be effective for reducing the per-generation cost of evaluating programs when using lexicase selection (Hernandez et al., 2019; Ferguson et al., 2019). Here, we show that these benefits apply to other selection methods, including tournament selection and fitness-proportionate selection.

Previous work demonstrated that using random down-sampling in the context of lexicase selection can substantially improve problem-solving success when the per-generation computational savings are reallocated to other aspects of evolutionary search, such as running for more generations (Hernandez et al., 2019; Ferguson et al., 2019; Helmuth and Spector, 2021; Schweim et al., 2022; Geiger et al., 2023). However, naively constructing random down-samples has the drawback of leaving out potentially important training cases or over-representing redundant training cases, which can slow or even impede problem-solving success (Hernandez et al., 2022b; Boldi et al., 2022; Helmuth and Spector, 2021; Boldi et al., 2023b). Informed down-sampling (IDS) (Boldi et al., 2024) addresses this drawback by using runtime population statistics to construct down-samples with distinct, more informative training cases. Informed down-sampling was found to significantly improve success rates over random down-sampling for program synthesis runs using the PushGP system (Spector et al., 2005). In each of these previous studies, down-sampling is ap-

plied in the context of standard lexicase selection alone. To our knowledge, these down-sampling techniques have yet to be rigorously evaluated in combination with other commonly used parent selection methods in GP, like tournament or fitness-proportionate selection, or other diversity-focused selection methods like implicit fitness sharing.

In this work, we expand on a previously published short-form communication (Boldi et al., 2023a) to investigate whether the benefits of random and informed down-sampling extend beyond lexicase selection. By doing this, we hope to motivate more artificial life practitioners to incorporate down-sampling into their evolutionary frameworks with a variety of selection schemes. To do so, we compare problem-solving success when using different combinations of selection scheme and down-sampling method across six program synthesis benchmark problems (Helmuth and Spector, 2015; Helmuth and Kelly, 2021). In addition, we test whether the level of selection pressure imposed by a selection scheme or whether the presence of diversity maintenance mechanisms influence the efficacy of random versus informed down-sampling.

## Selection

The process of selection is a fundamental feature of evolutionary search. Parent selection algorithms steer evolving populations through a search space by determining which individuals should contribute genetic material to the next generation. Many selection algorithms have been developed, each targeting different problem domains and search space topologies (e.g., (Holland, 1992; Brindle, 1980; Ross, 1999; Spector, 2012; Helmuth et al., 2015)). In this subsection, we overview the four selection strategies studied in this paper: fitness-proportionate, tournament, implicit fitness sharing and lexicase selection. The implementation details and specific parameters of each of these strategies are then discussed in their respective subsection.

We acknowledge that there are many different selection schemes that were not included in this study, including those based on rankings (Blickle and Thiele, 1996), elitism, quality diversity (Mouret and Clune, 2015), or other techniques. However, we believe that this set of selection schemes captures a significant portion of what GP practitioners use.

### Fitness-Proportionate Selection

Fitness-proportionate selection (FPS) is one of the earliest proposed selection strategies in evolutionary computation (Holland, 1992). Fitness-proportionate selection assigns each parent a selection probability based on its aggregate fitness relative to that of the other population members. Although individuals with higher fitness have a higher chance of being selected, those with lower fitness can still be selected. The probability $p_i$ that an individual $i$ is selected is

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$$

where $f_i$ is the $i^{\text{th}}$ individual's fitness, and $N$ is the population size. Since our genetic programming system evaluates individuals with *errors* instead of fitness values, we compute the fitness of the individual as $\frac{1}{1+e_i}$ where $e_i$ is the aggregate error the individual achieved on the training set.

On its own, fitness-proportionate selection can impose low selection pressure on a population relative to other commonly used selection algorithms, such as tournament selection (Blickle and Thiele, 1996; Zhong et al., 2005). Fitness-proportionate selection is also simple to implement and computationally efficient with a time complexity of $\mathcal{O}(N)$. As such, fitness-proportionate selection is still commonly used for evolutionary search (Dang et al., 2019; Arabas and Opara, 2020), often as one component of more sophisticated selection procedures (Yan et al., 2019).

### Tournament Selection

Tournament selection requires there to be a comparison metric between individuals. This can either be represented as a total ordering (ranking) of the individuals, or an assignment of a fitness value for each individual. To select a single individual with tournament selection, $t$ individuals are chosen from the population at random. Then, the individual with the best fitness (or lowest error) "wins" the tournament and is selected as a parent. Tournament size ($t$) controls the strength of selection; larger tournament sizes impose stronger selection, and smaller tournament sizes impose weaker selection. In this work, we use a variety of tournament sizes.

Tournament selection has been found to be more stable (Butz et al., 2003) than fitness-proportionate selection, as it is not affected by fitness scaling (Goldberg and Deb, 1991). Tournament selection also has a time complexity of $\mathcal{O}(N)$, which makes it attractive when using large population sizes (Goldberg and Deb, 1991). Due to its simplicity and efficiency, it is widely used as the standard selection strategy for evolutionary computation (Fang and Li, 2010).

### Implicit Fitness Sharing

Implicit fitness sharing (IFS) changes how errors are aggregated to incorporate an estimate of the difficulty of a training example (Smith et al., 1993; McKay, 2000). This technique has been used as a diversity preservation method as individuals that solve hard or rare test cases have a higher fitness than otherwise. IFS was later adapted to enable it to be used in cases where errors are non binary (Krawiec and Nawrocki, 2013), which is the framework we use in this work. Implicit fitness sharing is not a selection scheme on its own, but a fitness augmentation scheme that can be applied before using

a different selection scheme, such as tournament or fitness-proportionate selection to make the selection.

The idea for implicit fitness sharing is that individual test case error is weighted with respect to how the rest of the population performs on that test case. Specifically, the corrected fitness for each individual is given by the formula

$$F_{NBIFS}(i) = \sum_{t \in T} \frac{f(i,t)}{\sum_{i' \in P} f(i',t)}$$

Where $T$ is the set of training cases, $f(i,t) \in [0,1]$ is the raw fitness of an individual $i$ on training case $t$ (higher is better). We calculate this value from the errors of the individual in the same manner as we did for fitness-proportionate Selection. After this re-weighting scheme, the population is selected from using tournament selection. In this work, we use a tournament size of $t = 30$ for the IFS experiments, as this tournament size was empirically determined to perform the best from our plain tournament selection runs. When we refer to IFS as a selection scheme, we are referring to IFS with tournament selection with $t = 30$.

### Lexicase Selection

Unlike fitness-proportionate and tournament selection, lexicase selection does not aggregate error values across training cases to choose parents. Instead, lexicase considers each training case individually. To select a single parent, lexicase selection first shuffles the set of training cases into a random order, and all individuals in the population are included in a pool eligible for selection. Each training case is then applied in sequence (in the shuffled order) to filter down the pool of eligible candidates. At each step, the next pool of eligible individuals is set to include only individuals with elite performance on the current training case. This filtering continues until one individual remains in the eligible pool to be selected or until all training cases have been exhausted, where one of the remaining eligible individuals is then selected at random. Because each parent selection event uses a random permutation of training cases, lexicase selection prioritizes high performance on different sets of training cases across parent selection events, improving its capacity for diversity maintenance (Helmuth et al., 2016a; Dolson et al., 2018).

Lexicase selection was initially designed for multi-modal test-based program synthesis problems (Spector, 2012; Helmuth et al., 2015) and has frequently been found to outperform other selection methods in this domain (Helmuth and Kelly, 2022; Sobania et al., 2022). Lexicase selection has also been shown to be effective in domains beyond GP, including evolutionary robotics (Moore and Stanton, 2017; Stanton and Moore, 2022), deep learning (Ding and Spector, 2021), genetic algorithms (Metevier et al., 2019), learning classifier systems (Aenugu and Spector, 2019) and even in the directed evolution of microbes (Lalejini et al., 2022). Lexicase selection's success is often attributed to its capacity to preserve diversity (Helmuth et al., 2016a; Dolson et al., 2018) and maintain specialists (Helmuth et al., 2020). However, the worst case time complexity for lexicase selection is $\mathcal{O}(N * C)$ where $N$ is the population size, and $C$ is the number of training cases. In practice, this number is often closer to $\mathcal{O}(N + C)$ when population diversity is high (Helmuth et al., 2022), and there are strategies that can be used to reduce this even further (Ding et al., 2022; Lalejini et al., 2023). Despite this larger time complexity, in practice, the computational cost of genetic programming is more often dominated by program evaluation instead of selection. In this work, we focus on training set down-sampling, a strategy that can be used to reduce how expensive the evaluation step of evolutionary runs are by reducing the effective size of the training set.

## Down-sampling

In test-based program synthesis, candidate solutions are evaluated on a set of training cases in order to assess quality or correctness. Down-sampling techniques reduce the total number of training cases used for assessing candidate solution quality, which in turn, reduces the total number of program evaluations needed each generation. Down-sampling has been studied in evolutionary computation as a means to reduce computational loads (Langdon, 2011; Ross, 1999) and reduce overfitting (Gonçalves et al., 2012; Liu and Khoshgoftaar, 2004; Goncalves and Silva, 2013).

*Historical subset selection* is one simple down-sampling method that maintains a single static subset of the training cases for an entire evolutionary run (Gathercole and Ross, 1994). In contrast, *random subset selection* (Gathercole and Ross, 1994) randomly chooses to include or not include each training case each generation, resulting in different down-sample sizes from generation to generation. *Stochastic subset sampling* (Nordin and Banzhaf, 1997; Lasarczyk et al., 2004) chooses a new fixed-size down-sample each generation. More sophisticated methods of down-sampling have also been developed. *Dynamic subset selection* creates down-samples that are biased toward including harder cases and cases not seen for many generations (Gathercole and Ross, 1994). Other work introduces a topology-based selection that takes problem structure into account by selecting cases that individuals perform differently in a problem domain (Lasarczyk et al., 2004). Topology-based selection is very similar to the later proposed informed down-sampling (Boldi et al., 2024); although the latter has only been tested on program synthesis problems, and the former only on symbolic regression and classification. In this work, we compare the performance of informed down-sampling to that of a method similar to the earlier proposed *stochastic subset sampling*, also known as random down-sampling (Hernandez et al., 2019), and the standard non-down-sampled selection strategy.

Given the demonstrated value of down-sampling for evolutionary search, it is important to understand how differ-

ent down-sampling methods interact with different selection algorithms to benefit (or hinder) problem-solving success. Here, we focus on two down-sampling techniques that have been demonstrated to be effective in combination with lexicase selection for GP: random down-sampling and informed down-sampling, each described in detail below. We ask whether the benefits of these down-sampling methods might extend to other commonly used selection procedures in GP.

### Random Down-sampling

Random down-sampling (in this context) constructs a random fixed-size subset of the training set each generation. This smaller subset of training cases is then used to evaluate the quality of the population for selection in the current generation. By reducing the number of training cases used to evaluate programs each generation, random down-sampling reduces the per-generation computational costs of population evaluation and parent selection. Previous work demonstrated that reallocating these computational savings to other aspects of evolutionary search can lead to substantial improvements in problem-solving success in the context of lexicase selection (Hernandez et al., 2019; Ferguson et al., 2019; Helmuth and Spector, 2021; Schweim et al., 2022; Geiger et al., 2023).

However, random down-sampling results in less thorough program evaluations, which can lead to misleading assessments of program quality. For example, a random down-sample might omit important training cases (e.g., cases that test input edge cases), as the down-sample is created randomly with no consideration for the program behavior that each training case might be assessing. Prior work explored the extent to which random down-sampling resulted in discontinuities between training sets used to select successive generations (Boldi et al., 2022). They found that the commonality of synonymous training cases usually prevented discontinuities; that is, most training sets contain many training cases that measure the same behavior are thus passed by the same groups of individuals. In these circumstances, down-samples are less likely to entirely omit an entire class of training cases.

### Informed Down-sampling

Informed down-sampling addresses random down-sampling's drawback of potentially omitting informative training cases by minimizing the number of synonymous training cases included in the down-sample (Boldi et al., 2024). To estimate differences among training cases, informed down-sampling fully evaluates a random subset of the population on the complete set of training cases. If an individual has 0 error on a training case, we refer to this individual as "solving" that training case. To select an individual, we evaluate them on a subset of the training cases. Two training cases solved by the exact same subset of the population are less informative than having two training cases that are solved by very different sub-populations. This is because if both cases are solved by the same set of the population, they exert very similar selective pressure to if there was only one of those cases. Following prior work, we call these cases "synonymous" due to the fact that they measure same behavior in the context of the current population. With informed down-sampling, down-samples are constructed to include cases as far from being synonymous with each other as possible.

Algorithm 1 specifies the full informed down-sampling procedure. We modified the algorithm from (Boldi et al., 2024) to allow for an arbitrary selection scheme, $\mathcal{S}$. To construct a down-sample, we use a random subset of the population to estimate the "distance" between all pairs of training cases. The distance between two training cases is the Hamming distance between their "solve vectors", which are vectors of binary value that specify which individuals in the population subset (or "parent sample") solved the training case. We specify the size of the population subset (and thus the maximum distance between cases) using the $\rho$ parameter. With $\rho = 0.01$, we include $1\%$ of the population in the population subset used for calculating each training case's solve vector. We used $\rho = 0.01$ for all experiments in this work. Next, a single training case is added to the down-sample at random, and training cases are added in sequence such that each additional training case is maximally far away from the current down-sample through a process known as farthest first traversal (Hochbaum and Shmoys, 1985).

Whilst what we outlined above is one specific specification of informed (or non-random) down-sampling, there are several other methods that can fall under this umbrella term. We believe the above outlined informed down-sampling sufficiently captures the main ideas from semantic aware down-sampling. It does not use information regarding inputs and outputs to form a down-sample. Instead, it uses population statistics, which exist regardless of the problem being solved, which makes it applicable in all problem domains without problem specific augmentations needed. For this reason, we think it is the most general approach to semantic aware down-sampling and we include it as the representative algorithm in this work.

## Methods

In this work, we study random and informed down-sampling in the context of four selection schemes commonly used in GP: tournament selection, fitness-proportionate selection, implicit fitness sharing and lexicase selection.

Our first series of experiments analyzes the performance of each specific selection and down-sampling combination. The second set of experiments focuses on determining how varying the strength of selection pressure influences the efficacy of down-sampling. For each of these experiments, we compare problem-solving success on six program synthesis problems as detailed in the following section.

**Algorithm 1** Informed Down-Sampling. Adapted from (Boldi et al., 2024).

**Data:**
$\mathcal{P}$ : population, **cases**: set of all training cases,
$\mathcal{S}$ : selection scheme, ▷ $\mathcal{S}$ picks a new pop. given an old pop. and a set of cases
$k$ : scheduled case distance computation parameter,
$\rho$ : parent sampling rate, ▷ $\rho, k$ are parameters to reduce the distance computation cost
$\mathcal{G}$ : current generation counter,
$\mathcal{D}$ : case distance matrix. ▷ all distances are initialized to be maximally far
**Result:** A list of selected parents
1: **if** $\mathcal{G}\%k == 0$ **then**
2: $\hat{\mathcal{P}} \leftarrow$ sample $\rho \times |\mathcal{P}|$ parents from $\mathcal{P}$
3: evaluate $\hat{\mathcal{P}}$ on **cases** ▷ parent sample, purely used for distance calculations
4: calculate $\mathcal{D}$ from solve vectors from solutions in $\hat{\mathcal{P}}$ on **cases**
5: **end if**
6: **ds** $\leftarrow$ create down-sample using farthest first traversal ▷ picks cases that are of high distance to each other in a greedy fashion
7: $\mathcal{P} \leftarrow$ select $|\mathcal{P}|$ new parents using $\mathcal{S}$ from $\mathcal{P}$ using **ds** as cases ▷ selecting new population
8: **return** $\mathcal{P}$

## Program Synthesis Problems

The goal of program synthesis problems is to achieve zero error on each of a set of training cases, where each training case encodes what the program should output given a certain input. For this work, we chose six program synthesis problems from the first and second program synthesis benchmark suites (Helmuth and Spector, 2015; Helmuth and Kelly, 2021, 2022). These problems have been explored in previous work on informed down-sampling (Boldi et al., 2024) and are therefore a good basis for this investigation. We included problems where informed down-sampling has been shown to improve problem-solving success (Count Odds, Fizz Buzz and Scrabble Score), reduce problem-solving success (Small or Large), and have no significant effect on problem-solving success (Fuel Cost). We also include a new problem that has not been investigated in prior work on informed down-sampling (Middle Character).

The specific parameters used for our program synthesis experiments can be found in Table 1. We performed 50 evolutionary runs for each program synthesis problem configuration, each with a population size of 1000. Each of these runs were performed at 5% down-sampling, meaning $r = 0.05$. Since we have 200 training cases in the entire training set, each individual is evaluated on 10 of the cases every generation. We chose $r = 0.05$ based on previous work (Boldi et al., 2024). The down-sampling strategy being used determines which cases are selected out of the 200 to make up the down-sample.

To make the comparisons fair, we ensure that all methods (regardless of down-sampling) use the same number of program executions. We limited all runs to 60,000,000 program executions, which is equivalent to running a full GP run (with no down-sampling) for 300 generations.

Table 1: System parameters used for the program synthesis runs. We seperate the parameters by general GP parameters as well as parameters used specifically when down-sampling is enabled (DS).

| **GP Parameter** | **Value** |
|---|---|
| runs per problem | 50 |
| population size | 1000 |
| initial training set size | 200 |
| testing set size | 1000 |
| maximum program executions | 60,000,000 |
| variation operator | UMAD |
| **DS Parameter** | **Value** |
| down-sample rate $r$ | 0.05 |
| parent sample rate $\rho$ | 0.01 |
| generational interval $k$ | 100 |

## Experimental Design

We used the PushGP framework to conduct our experiments. PushGP is a genetic programming framework for evolving Push programs. The Push programming language uses a set of typed memory stacks to allow programs to handle different data types (e.g., strings, numbers, etc.) and includes a Turing complete instruction set that supports basic computations as well as complex control flow, such as looping and conditional execution (Spector, 2001; Spector and Robinson, 2002; Spector et al., 2005). For these experiments, we use the same instruction sets as those used by Boldi et al. (2024), and used the propeller[1] implementation of PushGP.

For each configuration, we report the number of generalizing runs, which is the number of runs that produce a program that passes all test cases in the held out testing set. Since we are not evaluating the individuals on the entire

[1] https://github.com/lspector/propeller

Table 2: The effect that varying the selection schemes has on problem solving success when in conjunction with down-sampling. We report the number of generalizing solutions out of 50 program synthesis runs achieved by PushGP on the test set. In **bold** font are the down-sampling runs that perform significantly better than the respective runs with no down-sampling. Significant differences (according to a pairwise Chi-Squared significance test) between informed and random down-sampling are denoted by an asterisk (*).

| | Selection Scheme | Fitness Prop. | | | Tourn. ($t = 30$) | | | IFS ($t = 30$) | | | Lexicase | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Down-sample Type | No | Rnd | IDS | No | Rnd | IDS | No | Rnd | IDS | No | Rnd | IDS |
| Problem | Count Odds | 0 | 1 | 0 | 1 | **31** | **34** | 2 | **22** | **42*** | 11 | 10 | **49*** |
| | Fizz Buzz | 0 | 5 | 5 | 0 | 4 | **7** | 0 | 2 | 4 | 5 | **32** | **45*** |
| | Fuel Cost | 0 | **19** | **14** | 0 | **28** | **37** | 0 | **26** | **40*** | 22 | **40** | **41** |
| | Small or Large | 0 | 0 | 0 | 11 | **37** | **41** | 13 | 18 | **47*** | 16 | **42** | **38** |
| | Middle Character | 0 | 0 | 0 | 4 | **17** | 10 | 1 | **15** | **16** | 16 | **30** | 26 |
| | Scrabble Score | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 2 |

training set when we down-sample, checking if an individual passes the *entire* training set happens when an individual passes all the cases in the down-sample. If an individual passes the entire training set, the evolutionary run ends. This individual is then evaluated on the held out testing set. If this individual passes the testing set, the run is marked as a generalizing run. If the individual passes the down-sample, but not the entire *training* set, the evolutionary run continues. Contrary to previous work on down-sampling with lexicase selection (Hernandez et al., 2019; Helmuth and Spector, 2021; Boldi et al., 2024), the extra program executions required to verify if an individual passes the entire training set are added to our program execution tally used to limit our runs.

### Diversity preservation properties

A specific quality of selection schemes that could have an effect when used in conjunction with down-sampling is the diversity preservation qualities of the scheme.

Diversity maintenance is the capacity for a selection method to select a behaviorally diverse population of individuals. Estimating the behavior of an individual (so that you can maintain diversity) might be less accurate with the addition of down-sampling as a system has access to less information about the individual's behavior.

We investigate whether the diversity maintenance properties of a selection scheme affects its compatibility with down-sampling. To do this, we evaluate the problem-solving performance of selection schemes that use a variety of diversity preservation qualities. Specifically, we compare two selection schemes with no diversity preservation (FPS and Tournament) and two that do (Lexicase and IFS).

### Selection Pressure

We investigate how selection pressure interacts with down-sampling by comparing problem-solving success of tournament selection at a range of tournament sizes.

Selection pressure is the standard that is required to be met in order for an individual to be selected on average. With a high selection pressure, an individual needs to be better than more of its peers to have the same chance at selection. It is important to study whether the selection pressure affects the efficacy of down-sampling. This is because if the fitness evaluation is noisy due to an unbalanced sample, high selective pressure might result in worse selections being made. In other words, being highly selective with respect to an non-representative sample could result in selecting individuals that are globally suboptimal.

In order to vary the selection pressure, we vary the size of the tournament. We test tournament sizes of $t = 2, 5, 10, 30$ in combination with the various types of down-sampling.

## Results and Discussion

Table 2 shows problem-solving successes for the six program synthesis problems studied at a variety of selection scheme and down-sampling combinations. A run is considered to be successful if a perfect solution evolves (i.e., a program that solves all training and unseen testing cases). Consistent with previous work with lexicase selection (Helmuth and Spector, 2021; Helmuth and Kelly, 2022; Hernandez et al., 2022b; Boldi et al., 2024), not all program synthesis problems benefited from down-sampling. However, we found no instances where configurations without down-sampling significantly outperformed configurations with down-sampling enabled. In fact, when using lexicase selection and tournament selection, problem-solving success

Table 3: The effect that varying the selection pressure (tournament size) has on problem solving success when in conjunction with down-sampling. We report the number of generalizing solutions (successes) out of 50 program synthesis runs achieved by PushGP on the test set.

| | Tournament Size | $t = 2$ | | | $t = 5$ | | | $t = 10$ | | | $t = 30$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Down-sample Type | No | Rnd | IDS | No | Rnd | IDS | No | Rnd | IDS | No | Rnd | IDS |
| Problem | Count Odds | 1 | 0 | 0 | 0 | **7** | **10** | 0 | **26** | **33** | 1 | **31** | **34** |
| | Fizz Buzz | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 4 | **7** |
| | Fuel Cost | 0 | 0 | 0 | 0 | **14** | **11** | 1 | **28** | **25** | 0 | **28** | **37** |
| | Small or Large | 0 | 2 | 2 | 7 | 4 | **21*** | 13 | 18 | **47*** | 11 | **37** | **41** |
| | Middle Character | 0 | 1 | 1 | 0 | 0 | 3 | 1 | **13** | **10** | 4 | **17** | 10 |
| | Scrabble Score | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |

was significantly improved for all problems but one by at least one of the down-sampling method. When using implicit fitness sharing, down-sampling significantly improved problem-solving success for all but two problems (Scrabble Score and Fizz Buzz). Overall, fitness-proportionate selection benefited the least from the addition of down-sampling, as problem-solving success was significantly better for only one out of the six problems. We also found some examples where fitness-proportionate and tournament selection failed to find *any* solutions unless we used down-sampling.

We detected a significant difference in problem-solving success between informed and random down-sampling in five instances (across all problems and configurations). With implicit fitness sharing, we found significant differences between informed down-sampling and random down-sampling on the Count Odds, Fuel Cost and Small or Large problems. For lexicase selection, we found a significant difference on the Count Odds and Fizz Buzz problems. In each of these instances, informed down-sampling outperformed random down-sampling. We found no significant differences between the two down-sampling techniques on any problems when using fitness-proportionate or Tournament selection with $t = 30$.

To better understand what might distinguish selection schemes for which downsampling helps from those for which it doesn't, we consider the role that the overall selection pressure of a selection scheme might play. One might expect, for example, that when using an informed downsample it would be beneficial to maintain high selection pressure with respect to those well-chosen cases, whereas high selection pressure could result in the loss of test coverage on the cases not in the down-sample when using random down-sampling. To cast some light on this and related possibilities, we conducted experiments using tournament selection, for which selection pressure is easily adjusted.

The results for a variety of different configurations of tournament selection can be found in Table 3. We find that varying the selection pressure imposed by tournament selection does have a significant effect on problem solving success. Over these configurations, we find multiple places where a down-sampling technique outperforms the non downsampled version, yet we only find one problem (Small or Large) where informed down-sampling outperforms random down-sampling significantly. For this reason, it seems as though selective pressure does not significantly affect the comparative performance between random and informed down-sampling. Given this, we can be reasonably certain that down-sampling will improve the performance of their systems, regardless of the selection pressure enacted by the selection scheme chosen.

Overall, our results indicate that down-sampling is often beneficial or neutral for problem-solving success. We did not find compelling evidence that down-sampling *impeded* success in any of our experiments. Though, we do note that others have found down-sampling to impede problem-solving success when there are strong trade-offs between training cases (e.g., low error on one excludes low error on another) or when a training set lacks some redundancy (Hernandez et al., 2022b).

We found that informed down-sampling was most consistently beneficial in the context of lexicase selection and implicit fitness sharing, as problem-solving success was improved by at least one down-sampling method across all problems for both of these selection schemes. We hypothesize that this is due to these schemes' ability to maintain diverse populations. Fitness-proportionate and tournament selection are known to be susceptible to premature convergence (Hornby, 2006; Hernandez et al., 2022c), while both lexicase selection and implicit fitness sharing are capable of maintaining both phenotypic and phylogenetic diversity

(Dolson et al., 2018; Helmuth et al., 2016b; Shahbandegan et al., 2022; Hernandez et al., 2022a). Given this, we hypothesize that lexicase selection and implicit fitness sharing benefit more from the increased number of generations afforded by down-sampling, and the presence of distinct and diverse training cases afforded by the "informed-ness", than tournament or fitness-proportionate selection. This also relates to the results of a different study, that showed that informed down-sampling maintains higher test coverage from successive selections than random down-sampling (Boldi et al., 2023b). That is, if a population evolving under fitness-proportionate and tournament selection has converged to a local fitness optimum, that population may not benefit from extra generations of evolution. In contrast, a more diverse population evolving under lexicase selection or IFS may benefit substantially from running for an increased number of generations when they are evaluated on a diverse set of training cases. Recent work also hints at the merit of increasing a population size instead of increasing the number of generations, potentially shrinking the gap between random and informed down-sampling (Briesch et al., 2023).

## Conclusion

In this work, we extended previous studies that evaluated the efficacy of random and informed down-sampling in the context of lexicase selection (Hernandez et al., 2019; Helmuth and Spector, 2021; Boldi et al., 2024).

Here, we show that the problem-solving benefits of both random and informed down-sampling generalize to other selection schemes, including fitness-proportionate selection, tournament selection and other diversity preserving selection schemes like implicit fitness sharing (IFS). This result suggests that evolutionary computing practitioners should experiment with different forms of down-sampling in combination with their preferred selection methods, as it can be used to improve problem-solving success by reallocating per-generation computational savings to running a deeper evolutionary search.

Previous studies have shown that the benefits of down-sampling stem from reallocating the computational savings to running an evolutionary search for more generations or evaluating more individuals (Helmuth and Spector, 2021; Hernandez et al., 2019; Ferguson et al., 2019). We hypothesize that this explanation holds across each of the selection schemes that we tested in this work. We did, however, find that different selection schemes benefited more or less from the addition of down-sampling: fitness-proportionate selection seemed to benefit the least, while lexicase, implicit fitness sharing, and even tournament selection with $t = 30$ benefited from down-sampling on five of the six problems.

We also detected that some selection schemes benefit more or less from the inclusion of informed down-sampling. Specifically, we found that for both lexicase selection and implicit fitness sharing multiple configurations using informed down-sampling significantly improved problem solving success over random down-sampling by including more unique and distinct cases in the down-samples. We hypothesize that populations evolving under lexicase selection or IFS are more diverse and therefore benefit the most from the extra generations of informative cases that are afforded by informed down-sampling.

To test the impact of selection pressure on down-sampling, we adjusted the tournament size for tournament selection and observed if it affected the performance of the two down-sampling techniques. Changing the tournament size did not consistently influence the relative performance between the two techniques. This strengthens the hypothesis that the improvement in problem-solving performance is due to diversity preservation rather than selective pressure.

Our study was limited to a relatively small set of problems, a single GP system (PushGP), and just two down-sampling techniques. Future work is needed to verify our findings beyond this context. Indeed, many down-sampling techniques have been developed for use in evolutionary computing and machine learning. Just as there has been recent progress in large-scale benchmarking for selection algorithms (Cava et al., 2021; Orzechowski et al., 2018), we argue that large-scale benchmarking efforts should be implemented for different down-sampling methods. Such efforts would help us to disentangle the circumstances where particular down-sampling methods are most appropriate. Furthermore, we only studied varied selection pressure and diversity maintenance when used in conjunction with tournament selection. Future work should explore this in conjunction with fitness proportionate selection in order to verify the generality of our conclusions. Additionally, a more unified theory on the effects of down-sampling on test-based problems could help to tie together disparate results from different application domains in evolutionary computing. Finally, future investigations should explore dynamic down-sampling; that is, can we use population statistics to automatically choose and parameterize down-sampling during an evolutionary search?

## Acknowledgements

We thank Charles Ofria, Franz Rothlauf, and the members of the PUSH Lab at Amherst College and UMass Amherst for their helpful discussions and comments.

This work was performed in part using high performance computing equipment obtained under National Science Foundation Grant No. 2117377. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation. This work was performed in part using high performance computing equipment obtained under a grant from the Collaborative R&D Fund managed by the Massachusetts Technology Collaborative.

## References


Sneha Aenugu and Lee Spector. 2019. Lexicase Selection in Learning Classifier Systems. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Prague, Czech Republic) *(GECCO '19)*. Association for Computing Machinery, New York, NY, USA, 356–364. https://doi.org/10.1145/3321707.3321828

Jarosław Arabas and Karol R. Opara. 2020. Population Diversity of Nonelitist Evolutionary Algorithms in the Exploration Phase. *IEEE Transactions on Evolutionary Computation* 24 (2020), 1050–1062.

Tobias Blickle and Lothar Thiele. 1996. A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation* 4, 4 (1996), 361–394.

Ryan Boldi, Ashley Bao, Martin Briesch, Thomas Helmuth, Dominik Sobania, Lee Spector, and Alexander Lalejini. 2023a. The Problem Solving Benefits of Down-sampling Vary by Selection Scheme. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*. 527–530.

Ryan Boldi, Martin Briesch, Dominik Sobania, Alexander Lalejini, Thomas Helmuth, Franz Rothlauf, Charles Ofria, and Lee Spector. 2024. Informed Down-Sampled Lexicase Selection: Identifying productive training cases for efficient problem solving. *Evolutionary Computation* (2024), 1–32.

Ryan Boldi, Thomas Helmuth, and Lee Spector. 2022. The Environmental Discontinuity Hypothesis for Down-Sampled Lexicase Selection. https://doi.org/10.48550/arxiv.2205.15931

Ryan Boldi, Alexander Lalejini, Thomas Helmuth, and Lee Spector. 2023b. A Static Analysis of Informed Down-Samples. In *Genetic and Evolutionary Computation Conference Companion (GECCO '23 Companion), July 15–19, 2023, Lisbon, Portugal*.

Martin Briesch, Dominik Sobania, and Franz Rothlauf. 2023. On the Trade-Off between Population Size and Number of Generations in GP for Program Synthesis. *Proceedings of the Companion Conference on Genetic and Evolutionary Computation* (2023). https://api.semanticscholar.org/CorpusID:260119504

Anne Brindle. 1980. Genetic algorithms for function optimization. (1980). https://doi.org/10.7939/R3FB4WS2W

Martin Volker Butz, Kumara Sastry, and David E. Goldberg. 2003. Tournament Selection: Stable Fitness Pressure in XCS. In *Annual Conference on Genetic and Evolutionary Computation*.

William La Cava, Patryk Orzechowski, Bogdan Burlacu, Fabricio Olivetti de Franca, Marco Virgolin, Ying Jin, Michael Kommenda, and Jason H. Moore. 2021. Contemporary Symbolic Regression Methods and their Relative Performance. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*. https://openreview.net/forum?id=xVQMrDLyGst

Duc-Cuong Dang, Anton V. Eremeev, and P. Lehre. 2019. Runtime Analysis of Fitness-Proportionate Selection on Linear Functions. *ArXiv* abs/1908.08686 (2019).

Li Ding, Ryan Boldi, Thomas Helmuth, and Lee Spector. 2022. Lexicase Selection at Scale. In *Genetic and Evolutionary Computation Conference Companion (GECCO '22 Companion), July 9–13, 2022, Boston, MA, USA*.

Li Ding and Lee Spector. 2021. Optimizing neural networks with gradient lexicase selection. In *International Conference on Learning Representations*.

Emily Dolson and Charles Ofria. 2021. Digital Evolution for Ecology Research: A Review. *Frontiers in Ecology and Evolution* 9 (2021), 18. https://doi.org/10.3389/fevo.2021.750779

Emily L Dolson, Wolfgang Banzhaf, and Charles Ofria. 2018. *Ecological theory provides insights about evolutionary computation*. preprint. PeerJ Preprints. https://doi.org/10.7287/peerj.preprints.27315v1

Yongsheng Fang and Jun Li. 2010. A Review of Tournament Selection in Genetic Programming. In *Advances in Computation and Intelligence (Lecture Notes in Computer Science)*, Zhihua Cai, Chengyu Hu, Zhuo Kang, and Yong Liu (Eds.). Springer, Berlin, Heidelberg, 181–192. https://doi.org/10.1007/978-3-642-16493-4-19

Austin J. Ferguson, Jose Guadalupe Hernandez, Daniel Junghans, Alexander Lalejini, Emily Dolson, and Charles Ofria. 2019. Characterizing the effects of random subsampling and dilution on Lexicase selection. In *Genetic Programming Theory and Practice XVII*, Wolfgang Banzhaf, Erik Goodman, Leigh Sheneman, Leonardo Trujillo, and Bill Worzel (Eds.). Springer, East Lansing, MI, USA, 1–23. https://doi.org/doi:10.1007/978-3-030-39958-0-1

Chris Gathercole and Peter Ross. 1994. Dynamic Training Subset Selection for Supervised Learning in Genetic Programming. In *Parallel Problem Solving from*

*Nature III (LNCS, Vol. 866)*, Yuval Davidor, Hans-Paul Schwefel, and Reinhard Männer (Eds.). Springer-Verlag, Jerusalem, 312–321. https://doi.org/doi:10.1007/3-540-58484-6-275

Alina Geiger, Dominik Sobania, and Franz Rothlauf. 2023. Down-Sampled Epsilon-Lexicase Selection for Real-World Symbolic Regression Problems. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Lisbon, Portugal) *(GECCO '23)*. Association for Computing Machinery, New York, NY, USA, 1109–1117. https://doi.org/10.1145/3583131.3590400

David E. Goldberg and Kalyanmoy Deb. 1991. A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. In *Foundations of Genetic Algorithms*. Vol. 1. Elsevier, 69–93. https://doi.org/10.1016/B978-0-08-050684-5.50008-2

Ivo Goncalves and Sara Silva. 2013. Balancing Learning and Overfitting in Genetic Programming with Interleaved Sampling of Training data. In *Proceedings of the 16th European Conference on Genetic Programming, EuroGP 2013 (LNCS, Vol. 7831)*, Krzysztof Krawiec, Alberto Moraglio, Ting Hu, A. Sima Uyar, and Bin Hu (Eds.). Springer Verlag, Vienna, Austria, 73–84. https://doi.org/doi:10.1007/978-3-642-37207-0_7

Ivo Gonçalves, Sara Silva, Joana B. Melo, and Joao Carreiras. 2012. *Random Sampling Technique for Overfitting Control in Genetic Programming*. https://doi.org/10.1007/978-3-642-29139-5_19 Pages: 229.

Thomas Helmuth and Peter Kelly. 2021. PSB2: the second program synthesis benchmark suite. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, Lille France, 785–794. https://doi.org/10.1145/3449639.3459285

Thomas Helmuth and Peter Kelly. 2022. Applying genetic programming to PSB2: the next generation program synthesis benchmark suite. *Genetic Programming and Evolvable Machines* (June 2022). https://doi.org/10.1007/s10710-022-09434-y

Thomas Helmuth, Johannes Lengler, and William La Cava. 2022. Population Diversity Leads to Short Running Times of Lexicase Selection. In *Parallel Problem Solving from Nature – PPSN XVII*, Günter Rudolph, Anna V. Kononova, Hernán Aguirre, Pascal Kerschke, Gabriela Ochoa, and Tea Tušar (Eds.). Springer International Publishing, Cham, 485–498.

Thomas Helmuth, Nicholas Freitag McPhee, and Lee Spector. 2016a. Effects of Lexicase and Tournament Selection on Diversity Recovery and Maintenance. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion (GECCO '16 Companion)*. Association for Computing Machinery, New York, NY, USA, 983–990. https://doi.org/10.1145/2908961.2931657

Thomas Helmuth, Nicholas Freitag McPhee, and Lee Spector. 2016b. Lexicase Selection for Program Synthesis: A Diversity Analysis. In *Genetic Programming Theory and Practice XIII*, Rick Riolo, W.P. Worzel, Mark Kotanchek, and Arthur Kordon (Eds.). Springer International Publishing, Cham, 151–167. https://doi.org/10.1007/978-3-319-34223-8_9 Series Title: Genetic and Evolutionary Computation.

Thomas Helmuth, Edward Pantridge, and Lee Spector. 2020. On the importance of specialists for lexicase selection. *Genetic Programming and Evolvable Machines* 21, 3 (Sept. 2020), 349–373. https://doi.org/10.1007/s10710-020-09377-2

Thomas Helmuth and Lee Spector. 2015. General Program Synthesis Benchmark Suite. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, Madrid Spain, 1039–1046. https://doi.org/10.1145/2739480.2754769

Thomas Helmuth and Lee Spector. 2021. Problem-solving benefits of down-sampled lexicase selection. *Artificial Life* (jun 2021), 1–21. https://doi.org/10.1162/artl_a_00341 arXiv:2106.06085

Thomas Helmuth, Lee Spector, and James Matheson. 2015. Solving Uncompromising Problems With Lexicase Selection. *IEEE Transactions on Evolutionary Computation* 19, 5 (2015), 630–643. https://doi.org/10.1109/TEVC.2014.2362729

Jose Guadalupe Hernandez, Alexander Lalejini, and Emily Dolson. 2022a. What Can Phylogenetic Metrics Tell us About Useful Diversity in Evolutionary Algorithms? In *Genetic Programming Theory and Practice XVIII*, Wolfgang Banzhaf, Leonardo Trujillo, Stephan Winkler, and Bill Worzel (Eds.). Springer Nature Singapore, Singapore, 63–82. https://doi.org/10.1007/978-981-16-8113-4_4 Series Title: Genetic and Evolutionary Computation.

Jose Guadalupe Hernandez, Alexander Lalejini, Emily Dolson, and Charles Ofria. 2019. Random subsampling improves performance in lexicase selection. In *Proceedings of the Genetic and Evolutionary Computation*

*Conference Companion*. ACM, Prague Czech Republic, 2028–2031. https://doi.org/10.1145/3319619.3326900

Jose Guadalupe Hernandez, Alexander Lalejini, and Charles Ofria. 2022b. *An Exploration of Exploration: Measuring the Ability of Lexicase Selection to Find Obscure Pathways to Optimality*. Springer Nature Singapore, Singapore, 83–107. https://doi.org/10.1007/978-981-16-8113-4_5

Jose Guadalupe Hernandez, Alexander Lalejini, and Charles Ofria. 2022c. A suite of diagnostic metrics for characterizing selection schemes. https://doi.org/10.48550/ARXIV.2204.13839

Dorit S. Hochbaum and David B. Shmoys. 1985. A Best Possible Heuristic for the k-Center Problem. *Math. Oper. Res.* 10 (1985), 180–184.

John H. Holland. 1992. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence* (1st mit press ed ed.). MIT Press, Cambridge, Mass.

Gregory S. Hornby. 2006. ALPS: the age-layered population structure for reducing the problem of premature convergence. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation - GECCO '06*. ACM Press, Seattle, Washington, USA, 815. https://doi.org/10.1145/1143997.1144142

Krzysztof Krawiec and Mateusz Nawrocki. 2013. Implicit Fitness Sharing for Evolutionary Synthesis of License Plate Detectors. In *Applications of Evolutionary Computation*, Anna I. Esparcia-Alcázar (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 376–386.

Alexander Lalejini, Emily Dolson, Anya E Vostinar, and Luis Zaman. 2022. Artificial selection methods from evolutionary computing show promise for directed evolution of microbes. *eLife* 11 (Aug. 2022), e79665. https://doi.org/10.7554/eLife.79665

Alexander Lalejini, Austin J. Ferguson, Nkrumah A. Grant, and Charles Ofria. 2021. Adaptive Phenotypic Plasticity Stabilizes Evolution in Fluctuating Environments. *Frontiers in Ecology and Evolution* 9 (Aug. 2021), 715381. https://doi.org/10.3389/fevo.2021.715381

Alexander Lalejini, Matthew Andres Moreno, Jose Guadalupe Hernandez, and Emily Dolson. 2023. Phylogeny-informed fitness estimation. *ArXiv* abs/2306.03970 (2023). https://api.semanticscholar.org/CorpusID:259095608

W. Langdon. 2011. Minimising testing in genetic programming. *RN* 11, 10 (2011).

Christian W.G. Lasarczyk, Peter Dittrich, and Wolfgang Banzhaf. 2004. Dynamic Subset Selection Based on a Fitness Case Topology. *Evolutionary Computation* 12, 2 (June 2004), 223–242. https://doi.org/10.1162/106365604773955157

Richard E. Lenski, Charles Ofria, Robert T. Pennock, and Christoph Adami. 2003. The evolutionary origin of complex features. *Nature* 423, 6936 (May 2003), 139–144. https://doi.org/10.1038/nature01568

Yi Liu and Taghi Khoshgoftaar. 2004. Reducing overfitting in genetic programming models for software quality classification. In *Proceedings of the Eighth IEEE international conference on High assurance systems engineering (HASE'04)*. IEEE Computer Society, USA, 56–65.

R I (Bob) McKay. 2000. Fitness Sharing in Genetic Programming. In *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation* (Las Vegas, Nevada) *(GECCO'00)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 435–442.

Blossom Metevier, Anil Kumar Saini, and Lee Spector. 2019. Lexicase Selection Beyond Genetic Programming. In *Genetic Programming Theory and Practice XVI*, Wolfgang Banzhaf, Lee Spector, and Leigh Sheneman (Eds.). Springer International Publishing, Cham, 123–136. https://doi.org/10.1007/978-3-030-04735-1_7

Jared M. Moore and Adam Stanton. 2017. Lexicase selection outperforms previous strategies for incremental evolution of virtual creature controllers. In *Proceedings of the Fourteenth European Conference Artificial Life, ECAL 2017, Lyon, France, September 4-8, 2017*, Carole Knibbe, Guillaume Beslon, David P. Parsons, Dusan Misevic, Jonathan Rouzaud-Cornabas, Nicolas Bredèche, Salima Hassas, Olivier Simonin 0001, and Hédi Soula (Eds.). MIT Press, 290–297. http://cognet.mit.edu/journal/ecal2017

Jean-Baptiste Mouret and Jeff Clune. 2015. Illuminating search spaces by mapping elites. *ArXiv* abs/1504.04909 (2015). https://api.semanticscholar.org/CorpusID:14759751

Peter Nordin and Wolfgang Banzhaf. 1997. An On-Line Method to Evolve Behavior and to Control a Miniature Robot in Real Time with Genetic Programming. *Adaptive Behavior* 5, 2 (Jan. 1997), 107–140. https://doi.org/10.1177/105971239700500201

Patryk Orzechowski, William La Cava, and Jason H. Moore. 2018. Where are we now? a large benchmark study of recent symbolic regression methods. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, Kyoto Japan, 1183–1190. https://doi.org/10.1145/3205455.3205539

Brian J Ross. 1999. *The Effects of Randomly Sampled Training Data on Program Evolution*. Technical Report CS-99-03. Dept. of Computer Science, Brock University, Canada.

Dirk Schweim, Dominik Sobania, and Franz Rothlauf. 2022. 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)*. 1–8. https://doi.org/10.1109/CEC55065.2022.9870337

Shakiba Shahbandegan, Jose Guadalupe Hernandez, Alexander Lalejini, and Emily Dolson. 2022. 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*. ACM, Boston Massachusetts, 2322–2325. https://doi.org/10.1145/3520304.3534028

Robert E. Smith, Stephanie Forrest, and Alan S. Perelson. 1993. Population Diversity in an Immune System Model: Implications for Genetic Search. In *Foundations of Genetic Algorithms*, L. DARRELL WHITLEY (Ed.). Foundations of Genetic Algorithms, Vol. 2. Elsevier, 153–165. https://doi.org/10.1016/B978-0-08-094832-4.50016-7

Dominik Sobania, Dirk Schweim, and Franz Rothlauf. 2022. A comprehensive survey on program synthesis with evolutionary algorithms. *IEEE Transactions on Evolutionary Computation* (2022).

Lee Spector. 2001. Autoconstructive Evolution: Push, PushGP, and Pushpop. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, Lee Spector, Erik D. Goodman, Annie Wu, W. B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke (Eds.). Morgan Kaufmann, San Francisco, California, USA, 137–146. http://hampshire.edu/lspector/pubs/ace.pdf

Lee Spector. 2012. Assessment of Problem Modality by Differential Performance of Lexicase Selection in Genetic Programming: A Preliminary Report. In *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation* (Philadelphia, Pennsylvania, USA) *(GECCO '12)*. Association for Computing Machinery, New York, NY, USA, 401–408. https://doi.org/10.1145/2330784.2330846

Lee Spector, Jon Klein, and Maarten Keijzer. 2005. The Push3 Execution Stack and the Evolution of Control. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation* (Washington DC, USA) *(GECCO '05)*. Association for Computing Machinery, New York, NY, USA, 1689–1696. https://doi.org/10.1145/1068009.1068292

Lee Spector and Alan Robinson. 2002. Genetic Programming and Autoconstructive Evolution with the Push Programming Language. *Genetic Programming and Evolvable Machines* 3, 1 (March 2002), 7–40. https://doi.org/10.1023/A:1014538503543

Adam Stanton and Jared M. Moore. 2022. Lexicase Selection for Multi-Task Evolutionary Robotics. *Artificial Life* 28, 4 (Nov. 2022), 479–498. https://doi.org/10.1162/artl_a_00374 _eprint: https://direct.mit.edu/artl/article-pdf/28/4/479/2043352/artl_a_00374.pdf.

Xuyang Yan, Mohammad Razeghi-Jahromi, Abdollah Homaifar, Berat A. Erol, Abenezer Girma, and Edward Tunstel. 2019. A Novel Streaming Data Clustering Algorithm Based on Fitness Proportionate Sharing. *IEEE Access* 7 (2019), 184985–185000. https://doi.org/10.1109/ACCESS.2019.2922162

Jinghui Zhong, Xiaomin Hu, Jun Zhang, and Min Gu. 2005. Comparison of Performance between Different Selection Strategies on Simple Genetic Algorithms., Vol. 2. 1115–1121. https://doi.org/10.1109/CIMCA.2005.1631619