The Problem Solving Benefits of Down-sampling Vary by Selection Scheme

Ryan Boldi
University of Massachusetts Amherst
rbahlousbold@umass.edu

Ashley Bao
Amherst College
abao26@amherst.edu

Martin Briesch
Johannes Gutenberg University
briesch@uni-mainz.de

Thomas Helmuth
Hamilton College
thelmuth@hamilton.edu

Dominik Sobania
Johannes Gutenberg University
dsobania@uni-mainz.de

Lee Spector
Amherst College
lspector@amherst.edu

Alexander Lalejini
Grand Valley State University
lalejina@gvsu.edu

ABSTRACT

Genetic programming systems often use large training sets to evaluate candidate solutions, which can be computationally expensive. Down-sampling training sets has long been used to decrease the computational cost of evaluation in a wide range of application domains. Indeed, recent studies have shown that both random and informed down-sampling can substantially improve problem-solving success for GP systems that use lexicase parent selection. We use the PushGP framework to experimentally test whether these down-sampling techniques can also improve problem-solving success in the context of two other commonly used selection methods, fitness-proportionate and tournament selection, across eight GP problems (four program synthesis and four symbolic regression). We verified that down-sampling can benefit the problem-solving success of both fitness-proportionate and tournament selection. However, the number of problems wherein down-sampling improved problem-solving success varied by selection scheme, suggesting that the impact of down-sampling depends both on the problem and choice of selection scheme. Surprisingly, we found that down-sampling was most consistently beneficial when combined with lexicase selection as compared to tournament and fitness-proportionate selection. Overall, our results suggest that down-sampling should be considered more often when solving test-based GP problems.

CCS CONCEPTS

• **Software and its engineering** → *Search-based software engineering*; • **Computing methodologies** → *Artificial intelligence.*

KEYWORDS

down-sampling, program synthesis, regression, genetic programming, selection

Extended version can be found at https://arxiv.org/abs/2304.07089

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

GECCO '23 Companion, July 15–19, 2023, Lisbon, Portugal

© 2023 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-0120-7/23/07.
https://doi.org/10.1145/3583133.3590713

ACM Reference Format:
Ryan Boldi, Ashley Bao, Martin Briesch, Thomas Helmuth, Dominik Sobania, Lee Spector, and Alexander Lalejini. 2023. The Problem Solving Benefits of Down-sampling Vary by Selection Scheme. In *Genetic and Evolutionary Computation Conference Companion (GECCO '23 Companion), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3583133.3590713

1 INTRODUCTION

Genetic programming (GP) applies evolutionary search algorithms to automatically synthesize programs instead of writing them by hand. GP systems use large training sets that 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. Down-sampling has been shown to be effective for reducing the per-generation cost of evaluating programs when using lexicase selection [13]. 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 increase success rates when the per-generation computational savings are reallocated to other aspects of evolutionary search, such as running for more generations [5, 10, 13, 18]. 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 [2, 10, 14]. To address this drawback, Boldi et al. [1] introduced informed down-sampling, which uses runtime population statistics to construct down-samples with distinct, more informative training cases. Informed down-sampling was found to improve success rates over random down-sampling for program synthesis runs using the

PushGP system. In each of these previous studies, down-sampling is applied in the context of standard lexicase selection. To our knowledge, these down-sampling techniques have yet to be evaluated in combination with other commonly used parent selection methods in GP, like tournament or fitness proportionate selection.

Here, we ask whether random or informed down-sampling can benefit GP systems beyond the context of lexicase selection. We analyze the problem-solving success of the PushGP system [20] on four program synthesis and four integer-based symbolic regression problems when using different combinations of selection scheme and down-sampling method. Specifically, for each of fitness-proportionate, tournament, and lexicase selection, we compare the impact of random, informed, and no down-sampling. We find that down-sampling either improved or did not significantly affect problem-solving success in all instances. Surprisingly, we find that lexicase selection benefits most consistently from the addition of down-sampling. Overall, our results highlight the potential problem-solving benefits of down-sampling in GP systems; though, some selection algorithms are likely to benefit more than others.

2 METHODS

We applied the PushGP system [20] to the following four program synthesis benchmark problems [6, 9]: Count Odds, Fizz Buzz, Small or Large, and Fuel Cost. These problems have been explored in previous work on informed down-sampling [1] and are therefore a good basis for our investigation. We included problems where informed down-sampling has been shown to improve problem-solving success (Count Odds and Fizz Buzz), reduce problem-solving success (Small or Large), and have no significant effect on problem-solving success (Fuel Cost).

In addition to the four program synthesis problems, we applied PushGP to four simple, integer-based symbolic regression problems to test whether the benefits of down-sampling extend beyond program synthesis. These symbolic regression problems included 3rd, 4th, and 5th-degree polynomials with randomly chosen coefficients (given in Table 2), and inputs and outputs were restricted to integers. For each problem, we limited the inputs of the training and testing sets $[-5, 5]$ and $[-10, -5) \cup (5, 10]$, respectively. A similar sized training set was used by Koza [17], and allowed our GP system to find solutions within our allotted computational budget.

For each problem, we measured the problem-solving success of different down-sampling techniques in combination with each of the following well-established parent selection methods: fitness-proportionate (also known as roulette), tournament, and lexicase selection. We used standard implementations of these selection schemes, and for brevity, we refer readers to [3, 11] for detailed descriptions of each. For both fitness-proportionate and tournament selection, 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. We used a tournament size of $t = 10$.

For each selection scheme, we compared three down-sampling treatments: random down-sampling (Rnd), informed down-sampling (IDS), and no down-sampling (No). These down-sampling methods are detailed in [1]. Briefly, random-down sampling evaluates individuals on a random subset of training cases each generation. Informed down-sampling uses runtime population statistics to build down-samples that contain more distinct training cases. To estimate how distinct training cases are from one another, informed down-sampling evaluates a random subset of the population on the full training set every k generations (the generational interval), and then uses those error vectors to sample training cases solved by different subsets of the population. Both random and informed down-sampling afford per-generation computational savings by not evaluating the entire population on the entire training set. The no down-sampling treatment uses standard lexicase on the full training set each generation.

Table 1: System parameters used for experiments.

Parameter	Value (PS)	Value (SR)
GP system parameters		
runs per problem	50	200
population size	1000	1000
initial training set size	200	11
testing set size	1000	10
maximum program executions	60,000,000	3,300,000
variation operator	UMAD	UMAD
instruction set	Boldi et al. [1]	Boldi et al. [1]
Down-sampling parameters (when used)		
down-sample rate r	0.05	0.30
parent sample rate ρ	0.01	0.01
generational interval k	100	100

The configuration used for each problem is given in Table 1. For each treatment, we report the number of generalizing runs, which is the number of runs that produce a program that successfully passes all of the test cases in the held out testing set. For our experiments, we held the maximum allowed number of program evaluations constant (problem-specific, Table 1); as such, down-sampling treatments that require fewer per-generation evaluations were run for more generations than treatments that require greater per-generation evaluations.

3 RESULTS AND DISCUSSION

Table 2 shows problem-solving successes for the chosen program synthesis problems. A run is considered to be successful if a program that solves all training and unseen testing cases is found. Consistent with previous work with lexicase selection [1, 7, 10, 14], not all program synthesis problems benefited from down-sampling. However, we found no instances where selection configurations without down-sampling significantly outperformed configurations with down-sampling enabled. In fact, when using lexicase selection, problem-solving success was significantly improved for all problems by at least one of the down-sampling methods, and when using tournament selection, down-sampling significantly improved problem-solving success for all but one problem (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 four problems. We also found examples where fitness proportionate and tournament selection failed to find *any* solutions unless we used down-sampling.

Across all configurations of program synthesis problems, we detected a significant difference in problem-solving success between

Table 2: Number of generalizing solutions (successes) for program synthesis (Program Synth.) and symbolic regression (Sym. Reg.) problems (out of 50 and 200 runs, respectively). Bolded results indicate that performing down-sampling offers significant (p<0.05) benefits over the standard (no down-sampling) version of the same selection scheme. A dagger signifies where random down-sampling significantly outperforms informed down-sampling, and an asterisks indicates where informed down-sampling outperforms random down-sampling. All significance analysis was conducted with a two proportion z-test with a Bonferroni correction for multiple comparisons.

	Selection Scheme:	FPS			Tournament			Lexicase		
	Down-sample Type:	No	Rnd	IDS	No	Rnd	IDS	No	Rnd	IDS
Program synth.	Count Odds	0	1	0	0	**26**	**33**	10	10	**49***
	Fizz Buzz	0	0	0	0	0	1	5	**32**	**45***
	Fuel Cost	0	**19**	**14**	1	**28**	**25**	20	**40**	**41**
	Small or Large	0	5	5	13	18	**47***	16	**42**	**38**
Sym. Reg.	$2x^5 - x^4 + 2x^3 + 3x^2 + 2x + 6$	1	2	2	0	0	0	21	36	**43**
	$x^4 - 2x^3 + 3x^2 + 2x + 3$	7	8	11	19	16	11	120	**180**	**179**
	$x^4 - 2x^2 + 4x + 3$	2	0	0	131	**156**	**172**	187	**198**†	189
	$3x^3 - 4x^2 + 8x + 3$	0	0	0	0	0	0	34	**71**	**83**

informed and random down-sampling in three instances: the small or large problem with tournament selection and the count odds and fizz buzz problems with lexicase selection. In each of these instances, informed down-sampling outperformed random down-sampling.

Table 2 shows problem-solving success for four symbolic regression problems. Like our program synthesis results, not all symbolic regression problems benefited from down-sampling. In fact, we did not detect any significant problem-solving benefits from applying down-sampling to fitness-proportionate selection on any of the four symbolic regression problems. Though not statistically significant, we observed one problem where fitness proportionate found two solutions (out of 200 runs) without down-sampling and failed to find any solutions with either form of down-sampling enabled. As before, however, we detected no instances where selection configurations without down-sampling *significantly* outperformed configurations with down-sampling. When using tournament selection, one out of four problems significantly benefited from down-sampling, and when using lexicase selection, all four problems benefited from at least one type of down-sampling.

Across all configurations of symbolic regression problems, we detected a significant difference in problem-solving success between informed and random down-sampling for only one problem in the context of lexicase selection. Unlike our program synthesis results, random down-sampling outperformed informed down-sampling on this problem.

Across both problem domains (program synthesis and symbolic regression) and all three selection methods, 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* problem-solving 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 [14].

Indeed, our findings are consistent with many previous studies that report substantial problem-solving gains in GP when applying down-sampling in order to reallocate computational resources to running deeper evolutionary searches [1, 10, 13]. These previous studies, however, focused on using down-sampling in the context of lexicase selection. Our results indicate that random and informed down-sampling are also potentially valuable additions to GP systems using other selection methods, such as fitness-proportionate selection or tournament selection. Further study is needed to verify that the value of using down-sampling with fitness-proportionate selection or tournament selection stems from the increase in generations that we can run evolution with fixed computational resources.

Surprisingly, we found that down-sampling was most consistently beneficial in the context of lexicase selection, as problem-solving success was improved by at least one down-sampling method across all problems. Further investigation is necessary to tease this apart. Fitness-proportionate and tournament selection are known to be susceptible to premature convergence [15, 16], while lexicase selection is more capable of maintaining both phenotypic and phylogenetic diversity [4, 8, 12, 19]. Given this, we hypothesize that lexicase selection benefits more from the increased number of generations afforded by down-sampling than tournament or fitness-proportionate selection. 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 may benefit substantially from running for an increased number of generations.

Overall, our results suggest that down-sampling, selection scheme, and search space topology interact to influence the likelihood of

problem-solving success. Even with our limited set of problems, neither down-sampling method completely dominated the other, and the benefits of down-sampling dramatically varied by problem and selection scheme.

4 CONCLUSION

We extended previous studies that evaluated the efficacy of random and informed down-sampling in the context of lexicase selection. We show that both random and informed down-sampling may also benefit GP systems that use fitness-proportionate or tournament selection. For fitness-proportionate, tournament, and lexicase selection, applying some form of down-sampling either helped or had no significant effect on problem-solving success (across four program synthesis and four symbolic regression problems). 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 [5, 10, 13]. 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 selection benefited on all eight problems. We hypothesize that populations evolving under lexicase selection are more diverse and therefore benefit the most from the extra generations afforded by down-sampling. In contrast, fitness-proportionate and tournament selection are known to be susceptible to premature convergence to local optima and might not always benefit from more generations of evolution; that is, if an entire population is stuck on a local optimum, tournament and fitness-proportionate selection have no built-in mechanisms to escape in subsequent generations.

We did not observe consistent differences between random and informed down-sampling. The best choice of down-sampling method depended on the selection strategy and the problem. Future work should investigate the interaction between down-sampling and selection methodology further to determine when informed down-sampling should be used over different down-sampling strategies.

ACKNOWLEDGMENTS

The authors would like to thank Charles Ofria and Franz Rothlauf for discussions that helped shape this work. This material is based upon work supported by the National Science Foundation under Grant No. 2117377. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation. This work was performed in part using high performance computing equipment obtained under a grant from the Collaborative R&D Fund managed by the Massachusetts Technology Collaborative.

REFERENCES

[1] Ryan Boldi, Martin Briesch, Dominik Sobania, Alexander Lalejini, Thomas Helmuth, Franz Rothlauf, Charles Ofria, and Lee Spector. 2023. Informed Down-Sampled Lexicase Selection: Identifying productive training cases for efficient problem solving. https://doi.org/10.48550/arXiv.2301.01488 arXiv:2301.01488.

[2] Ryan Boldi, Thomas Helmuth, and Lee Spector. 2022. The Environmental Discontinuity Hypothesis for Down-Sampled Lexicase Selection. In *2022 Conference on Artificial Life - Why it Didn't Work-Shop.* arXiv:2205.15931

[3] Thomas Bäck, D.B Fogel, and Z Michalewicz (Eds.). 1997. *Handbook of Evolutionary Computation* (0 ed.). CRC Press. https://doi.org/10.1201/9780367802486

[4] 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

[5] 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

[6] 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

[7] 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

[8] Thomas Helmuth, Nicholas Freitag McPhee, and Lee Spector. 2016. 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.

[9] 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

[10] Thomas Helmuth and Lee Spector. 2022. Problem-Solving Benefits of Down-Sampled Lexicase Selection. *Artificial Life* 27, 3–4 (March 2022), 183–203. https://doi.org/10.1162/artl_a_00341

[11] 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

[12] Jose Guadalupe Hernandez, Alexander Lalejini, and Emily Dolson. 2022. 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

[13] 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

[14] Jose Guadalupe Hernandez, Alexander Lalejini, and Charles Ofria. 2022. *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

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

[16] 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

[17] John R. Koza. 1992. . MIT Press, 138–139.

[18] 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

[19] 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

[20] 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