Improving the Generalisation of Genetic Programming Models with Evaluation Time and Asynchronous Parallel Computing

Aliyu Sani Sambo
School of Computing & Digital Tech.
Birmingham City University, UK
aliyu.sambo@mail.bcu.ac.uk

R. Muhammad Atif Azad
School of Computing & Digital Tech.
Birmingham City University, UK
atif.azad@bcu.ac.uk

Yevgeniya Kovalchuk
School of Computing & Digital Tech.
Birmingham City University, UK
Yevgeniya.Kovalchuk@bcu.ac.uk

Vivek Padmanaabhan Indramohan
Birmingham City University, UK
vivek.indramohan@bcu.ac.uk

Hanifa Shah
Computing Eng. & Built Environment
Birmingham City University, UK
hanifa.shah@bcu.ac.uk

ABSTRACT

In genetic programming (GP), controlling complexity often means reducing the size of evolved expressions. However, previous studies show that size reduction may not avoid model overfitting. Therefore, in this study, we use the evaluation time — the computational time required to evaluate a GP model on data — as the estimate of model complexity. The evaluation time depends not only on the size of evolved expressions but also their composition, thus acting as a more nuanced measure of model complexity than size alone. To constrain complexity using this measure of complexity, we employed an explicit control technique and a method that creates a race condition. We used a hybridisation of GP and multiple linear regression (MLRGP) that discovers useful features to boost training performance in our experiments. The improved training increases the chances of overfitting and facilitates a study of how managing complexity with evaluation time can address overfitting. Also, MLRGP allows us to observe the relationship between evaluation time and the number of features in a model. The results show that constraining evaluation time of MLRGP leads to better generalisation than both plain MLRGP and with an effective bloat-control.

CCS CONCEPTS

• **Computing methodologies → Classification and regression trees**; **Genetic programming**; **Artificial intelligence**.

KEYWORDS

genetic programming, generalisation, regression, complexity

ACM Reference Format:
Aliyu Sani Sambo, R. Muhammad Atif Azad, Yevgeniya Kovalchuk, Vivek Padmanaabhan Indramohan, and Hanifa Shah. 2021. Improving the Generalisation of Genetic Programming Models with Evaluation Time and Asynchronous Parallel Computing. In *2021 Genetic and Evolutionary Computation Conference Companion (GECCO '21 Companion), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3449726.3459583

1 INTRODUCTION

It has always been an important challenge in machine learning (ML) to avoid generating models that fit the training data very well but without generalising to the unseen data; this is termed overfitting. Often these overfitting models are overly complex [4]; however, determining how much complexity is just enough is a challenge. Another traditional concern in GP is complexity, which is often manifested by a tendency to grow model sizes to a point that renders the evolutionary search process ineffective. The most popular approach to controlling complexity in GP is bloat control, that is to limit the growth in size of the evolved expressions. However, previous studies have shown that bloat control alone does not always overcome the model overfitting problem [2]. This begs the question: is bloat control really complexity control?

To address the above limitation in bloat control, recent literature [6–8] has proposed alternative approaches to control the computational complexity of models in GP. Instead of using size as a measure of complexity, they use the evaluation time — the computational time it takes to evaluate a GP model on data. The use of evaluation time as a measure of complexity is built on the observation that a model that is made up of computationally expensive building blocks or that has large structures takes a long time to be evaluated, and hence it is computationally complex. The work in [6] empirically shows how the functional and structural complexity are different by plotting the evaluation times of identically sized but functionally diverse GP models, see a reproduction in Figure 1. Therefore, if the evaluation time of evolving models are constrained then the growth in the structural as well as the functional complexity will be discouraged. The same work also recommended various techniques to significantly minimise the noise in measuring evaluation times. This paper adopts the use of all these recommendations to measure complexity of the evolving models in MLRGP.

The next question is how to control the evaluation times. We used two approaches to control evaluation times. First, we use an effective bloat control method to discourage high evaluation times in the same way it discourages large size, named Time-control (TC). The second approach takes a simple view: induce a *race* among competing models, named the *Asynchronous Parallel Genetic Programming* (APGP) [6][8]. With APGP, the faster models can (if their fitness is competitive) join the breeding population before their slower counterparts and gain an evolutionary advantage.

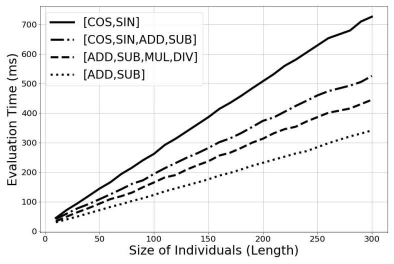

Figure 1: Size and composition affect evaluation times. Higher average evaluation times were returned by individuals made up of COS and SIN operators than same-sized individuals made up of simpler functions sets.

Method	Test Fitness	Size	Evaluation Times	No. of Features
Time-Control Success				
STD	9/10	10/10	10/10	10/10
BC	7/10	9/10	10/10	10/10
APGP Success				
STD	9/10	10/10	10/10	10/10
BC	7/10	0/10	0/10	0/10

Table 1: Summary of the test for significance in difference. The figures show the fraction of the tests where TC and APGP produced significantly better results, respectively.

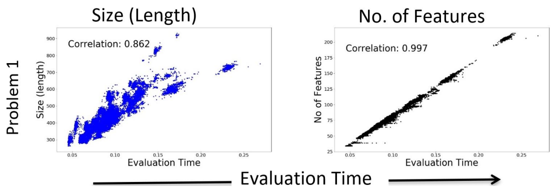

Figure 2: The correlation between evaluation time and the number of features is greater than it is with size.

We used a GP system that is aided by Multiple Linear Regression (MLR) [5] in our experiments. Such *MLRGP* systems [3] have become increasingly popular lately because they improve the training performance significantly; this is because the traditional GP often underfits the data because it can not efficiently generate numeric constants [1]. As this improved training accuracy can result in serious overfitting [5], MLRGP is suitable for studying the effect of controlling complexity with evaluation time on overfitting. With this system, we compare the performance of the two methods that restrict evaluation time with plain MLRGP (STD) and with MLRGP combined with an effective bloat control technique (BC).

2 RESULTS

Ten widely used datasets were selected as test problems. We compared test fitness scores and our three indicators of complexity: size, evaluation times and the number of features. The Mann-Whitney U test was used to determine the significance of the difference in the final populations and the results are summarised in Table 1.

In terms of test-fitness accuracy (generalisation), the evaluation times methods (TC and APGP) prevailed over STD and BC. Also, they had matching results in terms of the number of tests they prevailed; they both produced significantly higher test scores in 9 out of 10 tests against STD and 7 out of 10 against BC. However, the two time control methods differed in how they handled complexity (size, evaluation times and the number of features). TC produced significantly simpler solutions against BC and STD with one exception out of 60 tests, this is despite TC and BC used the same techniques to control time and size, respectively. APGP complexity control was gentler; it produced significantly simpler solutions than STD in all tests but more complex solutions than BC.

In addition to the effective control of the number of features by TC and APGP, the final populations that MLRGP produced showed a stronger correlation between the evaluation times and the number of features (average of 0.996) than between evaluation times and the sizes (average of 0.843), see a representative example in Figure 2.

3 CONCLUSION

We showed that the evaluation time behaves differently from size. We demonstrated that it can discriminate between the size, the complexity of the components, and the number of features of the MLRGP individual. Also, the results asserts that using the evaluation time to mange complexity leads to better generalisation. Thus, this approach promises to be broadly applicable. Overall, this study highlights the feasibility and merits of using the evaluation time.

REFERENCES

[1] R. Muhammad Atif Azad and Conor Ryan. 2014. The Best Things Don't Always Come in Small Packages: Constant Creation in Grammatical Evolution. In *17th European Conference on Genetic Programming (LNCS, Vol. 8599)*, M. Nicolau et al. (Ed.). Springer, Spain, 186–197. https://doi.org/10.1007/978-3-662-44303-3_16
[2] Raja Muhammad Atif Azad and Conor Ryan. 2014. A Simple Approach to Lifetime Learning in Genetic Programming based Symbolic Regression. *Evolutionary Computation* 22, 2 (June 2014), 287–317. https://doi.org/10.1162/EVCO_a_00111
[3] Amir Hossein Gandomi and Amir Hossein Alavi. 2012. A new multi-gene genetic programming approach to nonlinear system modeling. Part I: materials and structural engineering problems. *Neural Comput. Appl* 21, 1 (2012), 171–187.
[4] Gregory Paris, Denis Robilliard, and Cyril Fonlupt. 2003. Exploring Overfitting in Genetic Programming. In *Evolution Artificielle, 6th International Conference (Lecture Notes in Computer Science, Vol. 2936)*, Pierre Liardet et al. (Ed.). Springer, Marseilles, France, 267–277. https://doi.org/10.1007/b96080
[5] Aliyu Sani Sambo, R. Muhammad Atif Azad, Yevgeniya Kovalchuk, Vivek Padmanaabhan Indramohan, and Hanifa Shah. 2020. Feature Engineering for Improving Robustness of Crossover in Symbolic Regression. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion* (Cancún, Mexico) *(GECCO '20)*. ACM, NY, USA, 249–250. https://doi.org/10.1145/3377929.3390078
[6] Aliyu Sani Sambo, R. Muhammad Atif Azad, Yevgeniya Kovalchuk, Vivek Padmanaabhan Indramohan, and Hanifa Shah. 2020. Leveraging Asynchronous Parallel Computing to Produce Simple Genetic Programming Computational Models. In *The 35th ACM/SIGAPP Symposium On Applied Computing*, Federico Divina and Miguel Garcia Torres (Eds.). ACM, Brno, Czech Republic, 521–528. https://doi.org/10.1145/3341105.3373921
[7] Aliyu Sani Sambo, R. Muhammad Atif Azad, Yevgeniya Kovalchuk, Vivek Padmanaabhan Indramohan, and Hanifa Shah. 2020. Time Control or Size Control? Reducing Complexity and Improving Accuracy of Genetic Programming Models. In *Genetic Programming - 23rd European Conference, EuroGP 2020, Held as Part of EvoStar 2020, Seville, Spain, April 15-17, 2020, Proceedings (Lecture Notes in Computer Science, Vol. 12101)*, Ting Hu et al. (Ed.). Springer, Berlin, Heidelberg, 195–210. https://doi.org/10.1007/978-3-030-44094-7_13
[8] Aliyu Sani Sambo, R. Muhammad Atif Azad, Yevgeniya Kovalchuk, Vivek Padmanaabhan Indramohan, and Hanifa Shah. 2021. Evolving simple and accurate symbolic regression models via asynchronous parallel computing. *Applied Soft Computing* 104 (2021), 107198. https://doi.org/10.1016/j.asoc.2021.107198