The Influence of Probabilistic Grammars on Evolution

Jessica Mégane
jessicac@dei.uc.pt
University of Coimbra
Coimbra, Portugal

Nuno Lourenço
naml@dei.uc.pt
University of Coimbra
Coimbra, Portugal

Penousal Machado
machado@dei.uc.pt
University of Coimbra
Coimbra, Portugal

Dirk Schweim
schweim@hs-mainz.de
University of Applied Sciences Mainz
Mainz, Germany

ABSTRACT

Context-Free Grammars (CFGs) are used in Genetic Programming (GP) to encode the structure and syntax of programs, enabling efficient exploration of potential solutions and generation of well-formed and syntactically correct programs. Probabilistic Context-Free Grammars (PCFG) can be used to model the distribution of solutions to help guide the search process. Structured Grammatical Evolution (SGE) is a grammar-based GP algorithm that uses a list of dynamic lists as its genotype, where each list represents the ordered indexes of production rules to expand for each non-terminal in the grammar. Two recent variants incorporate PCFG into the SGE framework, where the probabilities of the production rule change during the evolutionary process, resulting in improved performance.

This study examines the impact of these differences on the behavior of SGE and its variants, Probabilistic Structured Grammatical Evolution (PSGE) and Co-evolutionary Probabilistic Structured Grammatical Evolution (Co-PSGE), in terms of population tree depth, genotype size, new solutions generated, and runtime. The results indicate that the use of probabilistic alternatives can affect the growth of tree depth and size and increases the ability to generate new solutions.

CCS CONCEPTS

• **Mathematics of computing → Probabilistic algorithms**; • **Computing methodologies → Genetic programming**.

KEYWORDS

grammatical evolution, probabilistic

ACM Reference Format:
Jessica Mégane, Nuno Lourenço, Penousal Machado, and Dirk Schweim. 2023. The Influence of Probabilistic Grammars on Evolution. 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/10.1145/3583133.3590706

1 INTRODUCTION

Grammatical Evolution (GE) [11] is a Genetic Programming (GP) algorithm [2] that uses a grammar-based representation of the solution space. The grammar allows the search space to be restricted and to keep syntactically correct solutions, which has been an important addition to the evolution of computer programs [6]. Although there are other grammar-based GP approaches, GE has become one of the most popular.

One of the major differences to traditional GP, is that while in GP the individuals are represented as trees and the variation operators act directly on them, in GE there is a separation between genotype, a list of integers, and phenotype, the solution of the problem. The mapping from the genotype to the phenotype follows the rules of a Context-Free Grammar (CFG). The representation and variation operators used by GE present some known issues, such as low locality [15] and high redundancy [10, 14], which leads to a poor exchange between exploration and exploitation, sometimes having behaviour comparable to random search [16].

Structured Grammatical Evolution (SGE) [3] is a GE variant that tackles its main issues, showing better performance when compared to GE and other grammar-based approaches [4], but also improved locality and lower redundancy [5].

Recently, two variants of SGE have been proposed that resort to a Probabilistic Context-Free Grammar (PCFG) during the mapping [7, 8]. Probabilistic grammars contain probabilities associated with each production rule, which if changed during the evolutionary process can help constrain and bias the search space. In Probabilistic Structured Grammatical Evolution (PSGE) [8] at each generation, the probabilities of the grammar are updated based on the rules expanded by the best individual. In Co-evolutionary Probabilistic Structured Grammatical Evolution (Co-PSGE) [7] each individual contains a grammar, and each generation the grammar can mutate its probabilities.

In this work, we study the impact of the use of PCFG in the behavior of the algorithms and compare the results with the original SGE. Some metrics collected during the experiments from the population are tree depth, genotype size, new solutions generated, fitness and execution time of each generation.

The population tree depth and size play a crucial role in the evolution process. A shallow tree depth limits the diversity of the population, whereas a large tree size increases the computational cost and can lead to overfitting the training data. Therefore, it is crucial to find a balance between these two factors to ensure efficient exploration of the solution space and the generation of high-quality solutions. The results indicate that the use of probabilistic alternatives can affect the growth of tree depth and size, and generate new solutions. Our findings showed that Co-PSGE and PSGE tend to generate bigger trees, which results in a bigger percentage of unique solutions in the population. SGE has been shown to benefit from a larger population sample.

The remainder of this work is structured as follows: Section 2 presents the related work, Section 3 details the experimental setup and results, and Section 4 gathers the main conclusions and provides some insights regarding future work.

2 BACKGROUND

Probabilistic models in GP [1] have been widely explored in the literature and successful in solving complex problems. The models follow two main approaches, such as resampling the population with probabilistic distributions [9, 12] or use a grammar to define the search space and bias the search [7–9]. Some of these methods have been analyzed using metrics beyond performance, which have revealed various issues depending on the approach, including problems such as bloat or bias towards small trees.

Estimation Distribution Algorithm (EDA) is a class of optimization algorithms that search for solutions by modelling and then sampling from the probability distribution over candidate solutions. Estimation of Distribution Programming (EDP) combines EDA with GP, and replaces variation operators by sampling the population each generation from a Bayesian model based on observed relative frequencies of node values in the solutions of a population [17, 18].

Schweim et al. [13] studied the bias of the EDP model and found that the model overfits the training data. Adding noise to the model to prevent overfitting results in a bias towards small trees.

Ratle et al. [9] proposed Stochastic Grammar-based GP, a method that has a probability distribution associated with the grammar rules, and each generation the probabilities of the rules of the grammar are updated based on a set of best and worst solutions. This approach tackles one of the main limitations of GP which is the bloat problem.

In SGE [3] the genotype is a list of dynamic lists, where each list represents the ordered indexes of production rules to expand for each non-terminal in the grammar. This algorithm presented better performance when compared to GE, and other grammar-based approaches [4], but also improved locality and lower redundancy [5]. Recently, two variants of SGE have been proposed that resort to a PCFG during the mapping [7, 8]. These two methods maintain the same genotype structure as SGE, with the elements of each list corresponding to the probability of choosing a production rule. In PSGE [8] at each generation, the probabilities of the grammar are updated based on the rules expanded by the best individual. In Co-PSGE [7] each individual contains a grammar, and each generation the grammar can mutate its probabilities.

Table 1: Parameters used in the experimental analysis for SGE, PSGE and Co-PSGE.

	Params 1	Params 2
Population Size	1000	250
Generations	200	
Elitism Count	100	25
Mutation Rate	0.05	0.10
Crossover Rate	0.90	0.90
Tournament	3	
Max Depth	10	8

3 EXPERIMENTAL SETUP

In order to study the impact of the probabilistic grammars, SGE, PSGE and Co-PSGE were executed over 100 independent runs.

In each run we collect information about tree depth, number of nodes, percentage of unique individuals, runtime and fitness. The parameters used in the experiments are summarised in Table 1.

The mutation performed on PSGE and Co-PSGE adds to the value of the codon a value generated with a Gaussian distribution of N(0.0,0.5). The mutation of SGE replaces the codon with a different rule. The three methods have one-point crossover. Two experimental parameters were tested. *Params 1* was defined based on the analysis of Schweim et al. [13], however in this paper the evolution is fitness-driven and employ selection mechanisms such as elitism, while in [13] were conducted random walks. The second set, *Params 2*, was defined based on the previous experiments conducted with these methods [5, 7, 8].

The experiments were set for finding the function defined by Pagie polynomial, a symbolic regression problem. The fitness function used to evaluate the individuals is the Root Relative Squared Error (RRSE) between the individual's solution and the target on a data set. The function is sampled with x[0], x[1] ∈ [-5, 5] with a step of 0.4. The division and logarithm functions are protected, i.e., $1/0 = 1$ and $log(f(x)) = 0$

The solutions are generated using Grammar 1. PSGE and Co-PSGE initialize the probabilities of the rules of each non-terminal of the grammar with a uniform distribution.

```
<start> ::= <expr>
<expr> ::= <expr> <op> <expr> | <pre_op> (<expr>) | <var>
<op> ::= + | - | * | /
<pre_op> ::= sin | cos | exp | log
<var> ::= x[0] | x[1] | 1.0
```

Grammar 1: Symbolic Regression grammar.

3.1 Results analysis

The performance of the algorithms is analysed by averaging the fitness of the best individual of each generation of the 100 runs.

Statistical tests were performed in order to support the analysis. We first apply the non-parametric Kruskal-Wallis test, since the populations were initialized independently and the results do not meet the criteria for parametric tests, in order to ascertain significant differences between the methods. When differences are observed, the Mann-Whitney post-hoc test with Bonferroni correction is applied to verify in which pairs the difference exists, with a significance level of $\alpha = 0.05$. Additionally, the effect size r was calculated to determine how significant the differences are. The following notation was used: "$\sim$" was used when there are no significant differences between samples; "+" sign for small effect size ($r <= 0.3$); "++" for medium ($0.3 < r <= 0.5$), and "+++" for large

($r > 0.5$). Table 2 shows the statistical results between each method for each set of parameters. Statistical differences are only observed using the *Params 2*, between SGE and PSGE, and between SGE and Co-PSGE.

Table 2: Results of the Mann-Whitney post-hoc statistical tests applied to the performance data. The Bonferroni correction is used considering a significance level of $\alpha = 0.05$.

	p-value	effect size
Params 1		
SGE - PSGE	0.225	~
SGE - Co-PSGE	0.099	~
PSGE - Co-PSGE	1.355	~
Params 2		
SGE - PSGE	0.00	++
SGE - Co-PSGE	0.00	++
PSGE - Co-PSGE	0.371	~

In the case of Co-PSGE we tested the parameters 5% and 10% for grammar mutation probability. As 5% showed better performance, in this paper the results that were created using this value are presented. It should be noted that it was observed that, for this problem, a higher grammar mutation probability led to the creation of individuals with more nodes, but the same depth, on average. In the case of PSGE, 0.01 and 0.05 were tested for the learning factor and, as expected, with higher value there is premature convergence of performance, slightly smaller trees are created, and the percentage of unique solutions decreases.

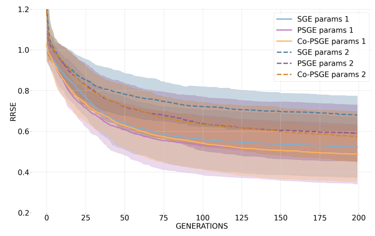

Figure 1: Performance results for the Pagie polynomial. Results are the mean best fitness of 100 runs.

Fig. 1 presents the performance results depicted as the mean of the fitness of the best individuals of each generation of the 100 runs. Looking at the figure, we can observe that the performance of PSGE and Co-PSGE is very similar, ending with better average fitness when compared with SGE in both set of parameters.

Using *Params 1* there were no statistical differences observed, however using the second set of parameters, both PSGE and Co-PSGE surpassed SGE with large statistical differences. These results indicate that SGE may need more individuals to improve results.

Looking at the average of the depths of the populations in Fig. 2 it is possible to see that SGE presents, in average, smaller individuals. Co-PSGE average increases rapidly, followed by PSGE. After 100 generations for *Params 2* and 50 generations for *Params 1*, PSGE stabilizes, ending up with mean and standard deviation values similar to the Co-PSGE. Despite initial differences in average tree depth, no difference in performance can be seen. It is also interesting to note that SGE tend to stabilize near the maximum depth defined (10 for

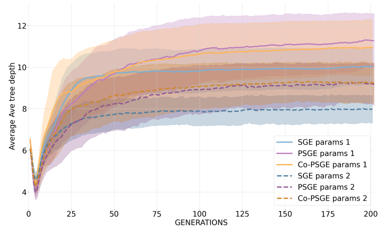

Figure 2: Average tree depth of the population over 100 runs.

Params 1 and 8 for *Params 2*), while the probabilistic alternatives surpass that value.

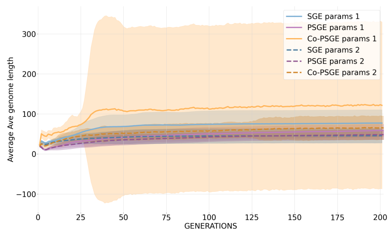

Figure 3: Average genome length of the population over 100 runs.

Fig. 3 shows the average genotype length. After initialization the length of the genotype increases in all methods. If we observe the results of *Params 1*, we are able to see that in average, the genotype length on Co-PSGE individuals is bigger, with a large standard deviation. All remaining methods have a smaller standard deviation in comparison. It is followed by SGE and, ending with smaller average, is PSGE. PSGE starts with even lower average than the results generated for *Params 2*. For the second set of parameters, PSGE starts with a smaller average of genotype size than SGE but grows faster, catching up and surpassing it after at least 125 generations. Co-PSGE has a similar curve as PSGE, but with an average of 20 codons higher.

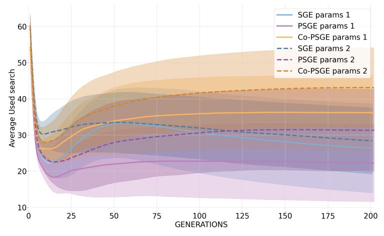

Figure 4: Average of the percentage of unique solutions in the population over 100 runs.

In both experiments, Co-PSGE has a sudden increase in the average and standard deviation, around 25 generations for *Params 1* and around 125 with *Params 2*. This may imply that the algorithm suffers from bloat, in which the individual's size increase, but there is no corresponding improvement in fitness. However Mégane et al. [7] tested this method in different problems, and showed that Co-PSGE was able to surpass SGE's performance in two complex problems, which may imply that this algorithm is better when finding more complex and difficult solutions.

Fig. 4 shows the percentage of unique solutions in the population. The algorithms show a higher percentage of unique solutions with *Params 2*, i.e. when less population is used. In the case of SGE the difference is lower than in the probabilistic versions, supporting the theory that it benefits more from a larger sample size.

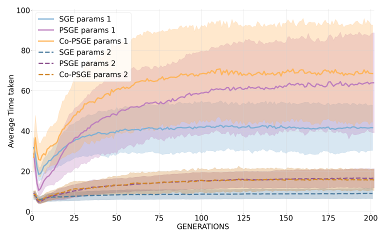

Figure 5: Average execution time of each generation over 100 runs.

Lastly, Fig. 5 represents the average time of each generation for each algorithm, and we can see that the probabilistic versions take longer, and for a large sample population Co-PSGE has a slightly higher average.

4 CONCLUSIONS

In this paper, the performance and behaviour of three algorithms - SGE, PSGE, and Co-PSGE - was analyzed in two sets of parameters.

The results showed that two of the algorithms, PSGE and Co-PSGE, performed very similar and outperformed SGE in the set of parameters with a smaller sample of individuals. The average of the depths of the populations indicated that SGE presented, on average, smaller individuals. The results of SGE tended to stabilize near the maximum depth defined, while the probabilistic alternatives exceeded this value. Co-PSGE starts with higher average, but after a certain number of generations, PSGE stabilized with mean and standard deviation values similar to Co-PSGE.

The analysis of the average genotype length showed that after initialization, the length of the genotype increased in all methods. The results indicated that Co-PSGE had a bigger average genotype size with a large standard deviation, followed by SGE and PSGE. For the second set of parameters, PSGE grew faster and surpassed SGE. Co-PSGE presented a higher average of codons.

In conclusion, the analysis of the percentage of unique solutions in the population showed that the probabilistic algorithms had a higher percentage of unique solutions when a smaller population was used. The probabilistic versions had a higher difference than SGE, which may imply that SGE benefits more from a larger sample size. Previous work [7] tested the Co-PSGE method on different problems and showed that it was able to surpass SGE's performance in two complex problems, implying that this algorithm is better at finding more complex solutions.

This study can be extended by analyzing the impact of other factors like grammar-design, problem complexity and variation operators. The analysis can help to understand the strengths and weaknesses of the algorithms, relevant for their future development.

ACKNOWLEDGMENTS

This work was funded by FEDER funds through the Operational Programme Competitiveness Factors - COMPETE and national funds by FCT - Foundation for Science and Technology (POCI-01-0145-FEDER-029297, CISUC - UID/CEC/00326/2020) and within the scope of the project A4A: Audiology for All (CENTRO-01-0247-FEDER-047083) financed by the Operational Program for Competitiveness and Internationalisation of PORTUGAL 2020 through the European Regional Development Fund. The first author is funded by FCT, Portugal, under the grant 2022.10174.BD.

REFERENCES

[1] K. Kim, Y. Shan, N. X. Hoai, and R. I. McKay. 2013. Probabilistic model building in genetic programming: a critical review. *Genetic Programming and Evolvable Machines* 15, 2 (Sept. 2013), 115–167. https://doi.org/10.1007/s10710-013-9205-x

[2] J. R. Koza. 1994. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing* 4, 2 (June 1994). https://doi.org/10.1007/bf00175355

[3] N. Lourenço, F. Assunção, F. B. Pereira, E. Costa, and P. Machado. 2018. Structured Grammatical Evolution: A Dynamic Approach. In *Handbook of Grammatical Evolution*. Springer International Publishing, 137–161. https://doi.org/10.1007/978-3-319-78717-6_6

[4] N. Lourenço, J. Ferrer, F. B. Pereira, and E. Costa. 2017. A Comparative Study of Different Grammar-Based Genetic Programming Approaches. In *Lecture Notes in Computer Science*. Springer International Publishing, 311–325. https://doi.org/10.1007/978-3-319-55696-3_20

[5] N. Lourenço, F. B. Pereira, and E. Costa. 2016. Unveiling the properties of structured grammatical evolution. *Genetic Programming and Evolvable Machines* 17, 3 (Feb. 2016), 251–289. https://doi.org/10.1007/s10710-015-9262-4

[6] R. I. McKay, N. X. Hoai, P. A. Whigham, Y. Shan, and M. O'Neill. 2010. Grammar-based Genetic Programming: a survey. *Genetic Programming and Evolvable Machines* 11, 3-4 (May 2010), 365–396. https://doi.org/10.1007/s10710-010-9109-y

[7] J. Mégane, N. Lourenço, and P. Machado. 2022. Co-evolutionary probabilistic structured grammatical evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM. https://doi.org/10.1145/3512290.3528833

[8] J. Mégane, N. Lourenço, and P. Machado. 2022. Probabilistic Structured Grammatical Evolution. In *2022 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. https://doi.org/10.1109/cec55065.2022.9870397

[9] A. Ratle and M. Sebag. 2002. Avoiding the Bloat with Stochastic Grammar-Based Genetic Programming. In *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 255–266. https://doi.org/10.1007/3-540-46033-0_21

[10] F. Rothlauf and D. E. Goldberg. 2003. Redundant Representations in Evolutionary Computation. *Evolutionary Computation* 11, 4 (Dec. 2003), 381–415. https://doi.org/10.1162/106365603322519288

[11] C. Ryan, M. O'Neill, and J.J. Collins (Eds.). 2018. *Handbook of Grammatical Evolution*. Springer International Publishing. https://doi.org/10.1007/978-3-319-78717-6

[12] R. Sałustowicz and J. Schmidhuber. 1997. Probabilistic Incremental Program Evolution: Stochastic search through program space. In *Machine Learning: ECML-97*. Springer Berlin Heidelberg, 213–220. https://doi.org/10.1007/3-540-62858-4_86

[13] D. Schweim and F. Rothlauf. 2018. An Analysis of the Bias of Variation Operators of Estimation of Distribution Programming. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Kyoto, Japan) *(GECCO '18)*. Association for Computing Machinery, New York, NY, USA, 1191–1198. https://doi.org/10.1145/3205455.3205582

[14] D. Schweim, A. Thorhauer, and F. Rothlauf. 2018. *On the Non-uniform Redundancy of Representations for Grammatical Evolution: The Influence of Grammars*. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-319-78717-6_3

[15] A. Thorhauer and F. Rothlauf. 2014. On the Locality of Standard Search Operators in Grammatical Evolution. In *Parallel Problem Solving from Nature – PPSN XIII*. Springer International Publishing, 465–475. https://doi.org/10.1007/978-3-319-10762-2_46

[16] P. A. Whigham, G. Dick, J. Maclaurin, and C. A. Owen. 2015. Examining the "Best of Both Worlds" of Grammatical Evolution. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM. https://doi.org/10.1145/2739480.2754784

[17] P. Wong, L. Lo, M. Wong, and K. Leung. 2014. Grammar-Based Genetic Programming with Bayesian network. In *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. https://doi.org/10.1109/cec.2014.6900423

[18] Kohsuke Y. and Hitoshi I. 2004. Program Evolution by Integrating EDP and GP. In *Genetic and Evolutionary Computation – GECCO 2004*, Kalyanmoy Deb (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 774–785.