ISSN 2007-9737

Grammatical Evolution with Codons Selection Order as Intensification Process

Andrés Espinal, Marco Aurelio Sotelo-Figueroa*, Jorge Alberto Soria-Alcaraz

Universidad de Guanajuato,
Departamento de Estudios Organizacionales,
Mexico

masotelo@ugto.mx

Abstract. Grammar Evolution (GE) can be considered a form of Genetic Programming (GP) that has become very popular in the field of Automatic Programming (AP) over the last few years. There has been a lot of research on different aspects of GE, including its parts; the Search Engine, Mapping Process, and Grammar. However, it has been shown that it is possible to select the codons randomly to improve the GE, using a random permutation. This paper introduces a new approach to intensify a solution using permutation heuristics to guide the codon selection order. A non-parametric test was applied to discern between the results obtained by the proposal and those obtained by the canonical GE version and the GE with random permutations.

Keywords. Grammatical evolution, symbolic regression, intensification.

1 Introduction

Two important approaches to automatically generate computer programs are Genetic Programming (GP) [23, 26] and Automatic Programming (AP) [15, 3, 39]. These methods have proven invaluable in improving algorithms and resolving challenging issues.

They all have different restrictions and difficulties, and to solve some of these restrictions, a different evolution-inspired method appeared recently, namely Grammatical Evolution (GE) [37]. This method combines the strength of GP with a more methodical strategy based on formal grammar.

GP is an evolutionary algorithm that works based on a population of potential solutions, often represented as tree structures to develop programs [25, 31, 24]. By utilizing evolutionary operators including crossover, mutation, and selection, the goal of GP is to evolve programs that can carry out particular tasks.

Some of the issues GP can present in execution are program bloat, inefficiency, and a lack of control over the search space [32]. There are several proposals to improve GP [5, 10]; one of them [21] includes metaheuristics to improve GP performance.

By establishing a mapping mechanism that relates a given program genotype and phenotype, Grammatical Evolution (GE) offers a solution to these problems. It uses formal context-free grammars to do this [38]. Grammatical Evolution evolves strings of symbols as opposed to program trees directly.

A mapping process between these strings of symbols and computer programs is applied by this method, producing solutions with semantic and syntactic validity. The differentiation of genotype, the evolving string, from the phenotype, the completed program, is one of the major advances of Grammatical Evolution [34].

Because grammars may be created to contain specific structural and syntactical restrictions, this separation improves control over the search space by lowering the possibility that inappropriate or inefficient programs would be constructed.

Moreover, by altering the genotype, the search space may be explored more efficiently, producing

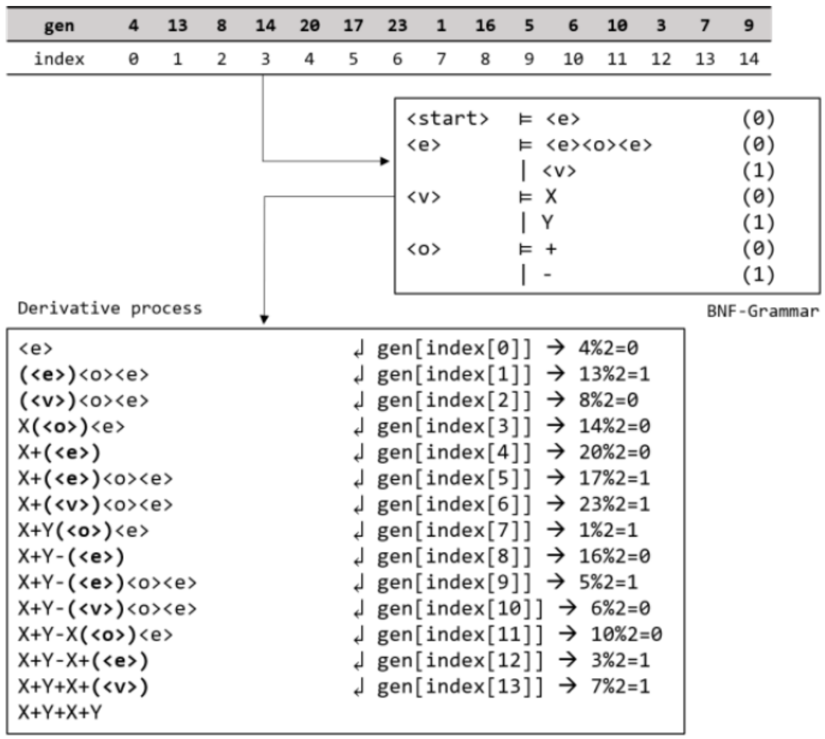

Fig. 1. Classical derivation example [45]

a more methodical program evolution. One of the research fields in GE is search engines.

There are several metaheuristics used as search engines like Genetic Algorithm (GA) [16], Differential Evolution (DE) [33, 17], Particle Swarm Optimization (PSO) [36, 42], Estimation Distribution Algorithms (EDA) [40, 29], and Ant Colony Optimization (ACO) [13] among others.

In [44], it was proposed to change the codon selection using a random permutation; this process was used with GE as a search engine, obtaining better results than classic GE. GE has been used successfully in many problems; however, the Symbolic Regression Problem (SRP) has been proposed and utilized as a benchmark problem for GE [1, 4, 43, 30].

The result of applying GE to SRP is an expression or function that fits a given instance of the problem. It is necessary to use formal grammar to ensure that the generated expression or function has syntactic validity.

Finally, GE explores a wide range of possible expressions at execution time, allowing it to discover highly accurate expressions. This paper proposes a methodology to evolve the codon selection order using the 2-opt, 3-opt, and 5-opt, and inversion permute heuristics.

This proposal is tested with instances of the SRP problem. The results obtained by this proposal, the canonical GE version, and the GE with Random Permutations were compared statistically.

2 Grammatical Evolution

The Grammatical Evolution (GE) [37, 38, 34] is a grammar-based form of Genetic Programming (GP) [22, 23]. The concepts of genotype and phenotype are present in both GP and GE.

Originally, the genotype in GP is based on tree representations, which are evaluated directly to obtain the phenotype, whereas GE uses a

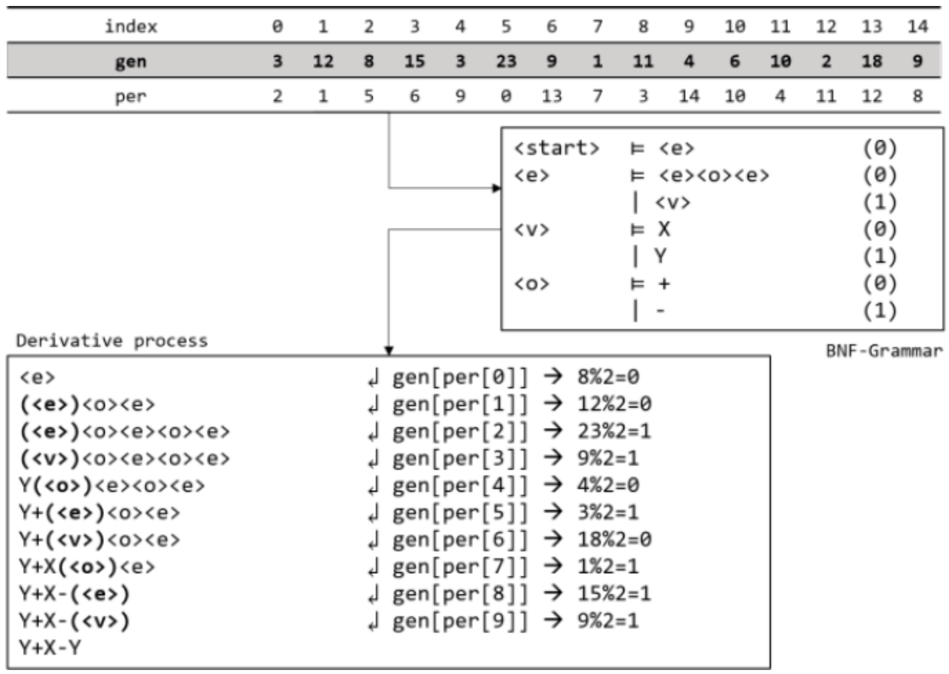

Fig. 2. Derivation example based on permutations [45]

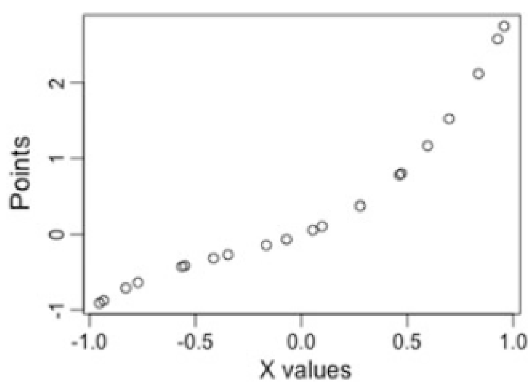

Fig. 3. Available data

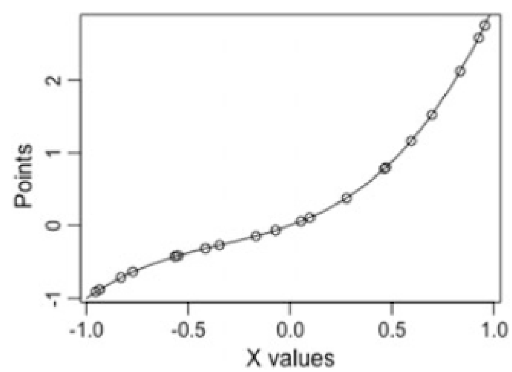

Fig. 4. Representative expression

linear representation, which is employed with a grammar to obtain the phenotype. The four main components, shown in Figure 5.

- The **Problem Instance** defines the problem domain and its conditions. It is used as a measure to guide the search engine and make the optimization process.
- The **Grammar** establishes elements and rules that fit the problem instance's specifics. In GE, Backus-Naur Form is commonly employed [2], although alternatives such as Attribute Grammar [20, 11] or Christiansen Grammar [7, 35] can also be applied.
- The **Search Engine** oversees optimization and adjusts the genotype based on the quality of the phenotype applied to the problem instance.

ISSN 2007-9737

Algorithm 1 Grammatical evolution algorithm

```
Require: pop_size, dimtext, search_engine, cont_apply_LS, heuristic
Population ← new_population(pop_size, dim)
Per ← generate_permutation(dim)
Fitness ← Evaluate(Population, Per)
Cont ← 0
GBest ← get_Best(Population, Fitness)
while termination condition not met do
    Population ← search_engine(Population)
    Fitness ← Evaluate(Population, Per)
    Best ← get_Best(Population, Fitness)
    if GBest=Best then
        Cont ← Cont + 1
    else
        GBest ← Best
        Cont ← 0
    end if
    if Cont = cont_apply_LS then
        Per ← heuristic(Per)
        Cont ← 0
    end if
end while
return Best Solution
```

While the Genetic Algorithm stands as the canonical search engine [16], numerous others have been implemented.

– The **Mapping Process** facilitates the conversion between genotype and phenotype, employing specific strategies such as Depth-First [34], Breadth-First [12], π Grammatical Evolution [36], etc.

Figure 1 shows an example of a classic GE, it uses a sequential codon selection order, and the search engine is applied to the codon values.

Figure 2 shows the proposal [44] example, it uses a permute codon selection order. Figure 1 and 2 uses the same codon values and the results show that is possible to obtain different phenotypes using a selection order.

3 Symbolic Regression Problem

Symbolic Regression Problem (SRP) [1, 4, 18] is the process of obtaining a representative expression from available data that we use when we want to know what was the equation behind our instance.

Table 1. Symbolic regression functions used as instances set

Function	Polynomial
F_1	$f\left(x\right) = X^3 + X^2 + X$
F_2	$f\left(x\right) = X^4 + X^3 + X^2 + X$
F_3	$f\left(x\right) = X^5 + X^4 + X^3 + X^2 + X$
F_4	$f\left(x\right) = X^6 + X^5 + X^4 + X^3 + X^2$ $+X$
F_5	$f\left(x\right) = sin\left(x^2\right) cos\left(x\right) - 1$
F_6	$f\left(x\right) = sin\left(x\right) + sin\left(x + x^2\right)$
F_7	$f\left(x\right) = log\left(x + 1\right) + log\left(x^2 + 1\right)$
F_8	$f\left(x\right) = \sqrt{x}$
F_9	$f\left(x, y\right) = sin\left(x\right) + sin\left(y^2\right)$
F_10	$f\left(x, y\right) = 2sin\left(x\right) cos\left(y\right)$
$Keijzer_1$	$f\left(x\right) = 0.3xsin(2\pi x)$
$Keijzer_2$	$f\left(x\right) = 1 + 3x + 3x^2 + x^3$
$Keijzer_3$	$f\left(x, y\right) = 8/(2 + x^2 + y^2)$
$Keijzer_4$	$f\left(x, y\right) = x^4 - x^3 + y^2/2 - y$
$Keijzer_5$	$f\left(x, y\right) = x^3/5 + y^3/2 - y - x$
$Keijzer_6$	$f\left(x_1, x_2, \ldots, x_10\right) = 10.59x_1x_2 +$ $100.5967x_3x_4 - 50.59x_5x_6 +$ $20x_1x_7x_9 + 5x_3x_6x_10$

In SRP, the goal is to seek a model (an equation or a mathematical formula) that describes the relationship between the input variables and the target output variable, without prior knowledge of the functional form.

SRP represents an important problem studied for the GP community [18, 43, 30].

Figure 3 shows the available data from the instance without knowing the function and Figure 4 shows the proposed expression to generate the data.

Table 2. Median of the results obtained for each instance and GE variant

Function	GA	Rnd	Inv	2opt	3opt	5opt
F_1	0.000	0.000	0.000	0.000	0.000	0.000
F_2	0.033	0.012	0.034	0.033	0.031	0.035
F_3	0.082	0.082	0.105	0.100	0.065	0.081
F_4	0.143	0.112	0.112	0.123	0.130	0.154
F_5	0.049	0.044	0.046	0.056	0.046	0.046
F_6	0.036	0.043	0.044	0.040	0.023	0.040
F_7	0.254	0.256	0.254	0.221	0.223	0.253
F_8	0.095	0.103	0.094	0.100	0.094	0.098
F_9	0.036	0.046	0.047	0.066	0.036	0.043
F_10	0.044	0.044	0.046	0.044	0.044	0.045
$Keijzer_1$	0.061	0.055	0.061	0.059	0.061	0.059
$Keijzer_2$	0.000	0.000	0.000	0.000	0.000	0.000
$Keijzer_3$	0.408	0.408	0.408	0.408	0.404	0.408
$Keijzer_4$	0.677	0.677	0.677	0.677	0.677	0.677
$Keijzer_5$	0.427	0.426	0.426	0.426	0.425	0.428
$Keijzer_6$	2.448	2.233	3.826	3.137	2.943	4.283

4 Proposed Approach

Figure 2 shows the proposal from [44] using a codon position random permutation. The proposal is shown in Algorithm 1, the permutation heuristics is applied if the best proposal solution is not improved for n generations.

In this study, the n value to apply the perturbation heuristic was determined empirically. The permutation heuristics were taken from the state-of-art based on those that can be applied to the Traveler Salesman Problem (TSP) [28, 8] and widely studied permutation problem.

4.1 k-opt

The k-opt heuristic [9, 27, 6] is classified as a local search method because it only slightly modifies the current solution to try to make it better.

It does not ensure the identification of the globally optimal solution, but it remains effective in perturbing an initial solution.

A k-opt heuristic is an attempt to enhance the quality of a solution using iterative swapping of pairs of items. Figure 6 shows an example of 2-opt.

Table 3. GA parameters

Parameter	Value
Population Size	300
Dimensions	100
Iterations to apply the heuristic	5
Function Evaluations	250,000
Selection	Binary Tournament
Crossover	2-points
Mutation	Bit-flip
Mapping Process	Depth-First

4.2 Inversion

The inversion heuristic [16, 8] involves selecting a subset of the proposal solution and investing the order of its elements. This heuristic neither ensures the identification of the globally optimal but is effective in perturbing an initial solution.

This operation introduces diversity in the population and can potentially explore different regions of the solution space. Figure 7 shows an example of Inversion.

5 Experimental Setup

Table 1 shows the Symbolic Regression functions used, the functions F_1 to F_{10} were taken from [18, 45] and the Keijzer from [19, 43, 14]. Mean Root Squared Error (MRSE), Equation 1, was used as a fitness function to discern the quality of each expression proposed by GE:

$$\mathrm{MRSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(t_i - y_i)^2}, \qquad (1)$$

where:

– n is the number of data points.

– y_i is the real value.

– t_i corresponds to the value obtained.

Grammars used by each function [45] are the followings:

– Grammar 1 for functions F_1 to F_8.

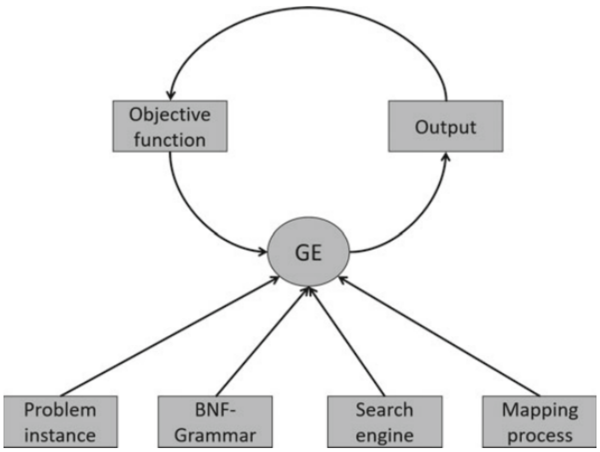

Fig. 5. GE methodology [41]

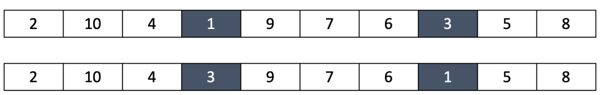

Fig. 6. 2-opt example, the first array contains a permutation from 1 to 10, it was chosen two items to interchange it into the second array

$$
\begin{array}{rcl}
\langle\text{start}\rangle & \models & \langle\text{expr}\rangle \\
\langle\text{expr}\rangle & \models & (\langle\text{expr}\rangle\langle\text{op}\rangle\langle\text{expr}\rangle) \mid \langle\text{pre}\rangle(\langle\text{expr}\rangle) \mid \langle\text{var}\rangle \\
\langle\text{var}\rangle & \models & \texttt{1} \mid \texttt{x} \\
\langle\text{pre}\rangle & \models & \texttt{exp} \mid \texttt{log} \mid \texttt{sin} \mid \texttt{cos} \\
\langle\text{op}\rangle & \models & \texttt{*} \mid \texttt{/} \mid \texttt{+} \mid \texttt{-}
\end{array}
$$

Grammar 1. Grammar for the functions F_1 to F_8

- Grammar 2 for functions F_9 to F_{10}.
- Grammar 3 for functions $Keijzer_1$ to $Keijzer_5$.
- Grammar 4 for function $Keijzer_6$.

The parameters used in the classic GE, Random Permutation GE, and current proposal are shown in Table 3. Those parameters were taken from [44], and the new parameter used to apply the permutation heuristic was chosen empirically. To conduct the comparison, 33 individual runs were executed for each function, using the proposed approach, classical GE, and Random Permutation GE.

Table 4. Ranking based on medians

Algorithm	Ranking
3opt	2.28125
Rnd	3.03125
GA	3.53125
2opt	4.03125
5opt	4.03125
Inv	4.09375

$$
\begin{array}{rcl}
\langle\text{start}\rangle & \models & \langle\text{expr}\rangle \\
\langle\text{expr}\rangle & \models & (\langle\text{expr}\rangle\langle\text{op}\rangle\langle\text{expr}\rangle) \mid \langle\text{pre}\rangle(\langle\text{expr}\rangle) \mid \langle\text{var}\rangle \\
\langle\text{var}\rangle & \models & \texttt{1} \mid \texttt{x} \mid \texttt{y} \\
\langle\text{pre}\rangle & \models & \texttt{exp} \mid \texttt{log} \mid \texttt{sin} \mid \texttt{cos} \\
\langle\text{op}\rangle & \models & \texttt{*} \mid \texttt{/} \mid \texttt{+} \mid \texttt{-}
\end{array}
$$

Grammar 2. Grammar for the functions F_9 to F_{10}

$$
\begin{array}{rcl}
\langle\text{start}\rangle & \models & \langle\text{expr}\rangle \\
\langle\text{expr}\rangle & \models & (\langle\text{expr}\rangle\langle\text{op}\rangle\langle\text{expr}\rangle) \mid \langle\text{var}\rangle \\
\langle\text{var}\rangle & \models & \texttt{1} \mid \texttt{x} \mid \texttt{y} \\
\langle\text{op}\rangle & \models & \texttt{*} \mid \texttt{/} \mid \texttt{+} \mid \texttt{-}
\end{array}
$$

Grammar 3. Grammar for the functions Keijzer from 1 to 5

$$
\begin{array}{rcl}
\langle\text{start}\rangle & \models & \langle\text{expr}\rangle \\
\langle\text{expr}\rangle & \models & (\langle\text{expr}\rangle\langle\text{op}\rangle\langle\text{expr}\rangle) \mid \langle\text{pre}\rangle(\langle\text{expr}\rangle) \mid \langle\text{var}\rangle \\
\langle\text{pre}\rangle & \models & \texttt{exp} \mid \texttt{log} \mid \texttt{sin} \mid \texttt{cos} \\
\langle\text{var}\rangle & \models & \texttt{1} \mid \texttt{x}\langle\text{c}\rangle \\
\langle\text{op}\rangle & \models & \texttt{+} \mid \texttt{-} \mid \texttt{*} \mid \texttt{/} \\
\langle\text{c}\rangle & \models & \texttt{0} \mid \texttt{1} \mid \texttt{2} \mid \texttt{3} \mid \texttt{4} \mid \texttt{5} \mid \texttt{6} \mid \texttt{7} \mid \texttt{8} \mid \texttt{9}
\end{array}
$$

Grammar 4. Grammar for the function Keijze 6

The median of the results was used for statistical comparison among the proposed approach, GE, and Random Permutation GE.

The statistical test was performed using the non-parametric Friedman test, which aimed to establish if any implementation was capable of outperforming the others.

2	10	4	1	9	7	6	3	5	8

2	10	4	6	7	9	1	3	5	8

Fig. 7. Inversion example, the first array contains a permutation from 1 to 10, it was chosen a range to invert it into the second array

6 Results

Table 2 shows the median of the results of the 33 experiments for each instance. To discern between the results, a non-parametric Friedman test was performed. The value obtained was 11.99107143 with a p-value of 0.034910328.

With this p-value less than 0.1, it was possible to make a post-hoc procedure to determine which GE variant obtained the best results. The results of the post-hoc test are shown in Table 4.

7 Conclusions

This paper introduced a methodology to intensify the Grammatical Evolution solutions. The intensification was based on the codon position random permutation used previously, where it was shown that a random permutation improves the Grammatical Evolution results.

The permutation was guided by permutation heuristics from the state-of-the-art, using a k-opt and inversion heuristics that have been applied to permutation problems. Symbolic Regression Problems were used because they are widely used in Grammatical Evolution and Genetic Programming to analyze improvement and performance.

The results obtained using the proposal with a 3-opt heuristic are better than the random permutation and the classic Grammatical Evolution. Not all variations of the opt heuristic gave good results, nor did the inversion heuristic. The improvement of n generations was used as a parameter to apply the permutation heuristics; however, it is possible to identify a way to apply it without being an extra parameter.

References

1. **Augusto, D. A., Barbosa, H. J. (2000).** Symbolic regression via genetic programming. Proceedings of the VI Brazilian Symposium on Neural Networks, IEEE Computer Society, pp. 173–178. DOI: 10.1109/SBRN.2000.889734.

2. **Backus, J. W., Bauer, F. L., Green, J., Katz, C., McCarthy, J., Naur, P., Perlis, A. J., Rutishauser, H., Samelson, K., Vauquois, B., Wegstein, J. H., van-Wijngaarden, A., Woodger, M. (1963).** Revised report on the algorithmic language ALGOL 60. Communications of the ACM, Vol. 5, No. 4, pp. 349–367. DOI: 10.1093/comjnl/5.4.349.

3. **Balzer, R. (1985).** A 15 year perspective on automatic programming. IEEE Transactions on Software Engineering, Vol. SE-11, No. 11, pp. 1257–1268. DOI: 10.1109/TSE.1985.231877.

4. **Barmpalexis, P., Kachrimanis, K., Tsakonas, A., Georgarakis, E. (2011).** Symbolic regression via genetic programming in the optimization of a controlled release pharmaceutical formulation. Chemometrics and Intelligent Laboratory Systems, Vol. 107, No. 1, pp. 75–82. DOI: 10.1016/j.chemolab.2011.01.012.

5. **Brameier, M. F., Banzhaf, W. (2007).** Linear genetic programming. Springer US. DOI: 10. 1007/978-0-387-31030-5.

6. **Brodowsky, U. A., Hougardy, S., Zhong, X. (2023).** The approximation ratio of the k-opt heuristic for the euclidean traveling salesman problem. SIAM Journal on Computing, Vol. 52, No. 4, pp. 841–864. DOI: 10.1137/21M146199X.

7. **Christiansen, H. (1990).** A survey of adaptable grammars. ACM SIGPLAN Notices, Vol. 25, No. 11, pp. 35–44. DOI: 10.1145/101356.101357.

8. **Cook, W. J., Applegate, D. L., Bixby, R. E., Chvátal, V. (2011).** The traveling salesman problem: A computational study. Princeton

University Press, Vol. 17. DOI: 10.1515/9781400841103.

9. **Croes, G. A. (1958).** A method for solving traveling-salesman problems. Operations Research, Vol. 6, No. 6, pp. 791–812. DOI: 10.1287/opre.6.6.791.

10. **Cárdenas-Florido, L., Trujillo, L., Hernandez, D. E., Muñoz-Contreras, J. M. (2024).** M5GP: Parallel multidimensional genetic programming with multidimensional populations for symbolic regression. Mathematical and Computational Applications, Vol. 29, No. 2, pp. 25. DOI: 10.3390/mca29020025.

11. **de-la-Cruz-Echeandía, M., de-la-Puente, A. O., Alfonseca, M. (2005).** Attribute grammar evolution. Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach, Springer Berlin Heidelberg, pp. 182–191. DOI: 10.1007/11499305_19.

12. **Fagan, D., O'Neill, M., Galván-López, E., Brabazon, A., McGarraghy, S. (2014).** An analysis of genotype-phenotype maps in grammatical evolution. Esparcia-Alcázar, A.I., Ekárt, A., Silva, S., Dignum, S., Uyar, A.Ş. (eds) Genetic Programming. EuroGP 2010. EuroGP 2010. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, Vol. 60216021. DOI: 10.1007/978-3-642-12148-7_6.

13. **Gaddam, J., Barca, J. C., Nguyen, T. T., Angelova, M. (2023).** Grammatical evolution with adaptive building blocks for traffic light control. 2023 IEEE Congress on Evolutionary Computation (CEC), pp. 1–10. DOI: 10.1109/CEC53210.2023.10254190.

14. **Gupt, K. K., Raja, M. A., Murphy, A., Youssef, A., Ryan, C. (2022).** GELAB – The cutting edge of grammatical evolution. IEEE Access, pp. 1–1. DOI: 10.1109/ACCESS.2022.3166115.

15. **Hintze, G. (1966).** Automatic programming. Fundamentals of Digital Machine Computing, Springer Berlin Heidelberg, pp. 177–211. DOI: 10.1007/978-3-662-40151-4_7.

16. **Holland, J. H. (1992).** Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT Press.

17. **Indu, M. T., Shunmuga, V. C. (2024).** Differential evolution ensemble designer. Expert Systems with Applications: An International Journal, Vol. 238, No. C. DOI: 10.1016/j.eswa.2023.121674.

18. **Karaboga, D., Ozturk, C., Karaboga, N., Gorkemli, B. (2012).** Artificial bee colony programming for symbolic regression. Information Sciences, Vol. 209, pp. 1–15. DOI: 10.1016/j.ins.2012.05.002.

19. **Keijzer, M. (2003).** Improving symbolic regression with interval arithmetic and linear scaling. Genetic Programming, Springer, Berlin, Heidelberg, pp. 70–82. DOI: 10.1007/3-540-36599-0_7.

20. **Knuth, D. E. (1968).** Semantics of context-free languages. Mathematical systems theory, Vol. 2, No. 2, pp. 127–145. DOI: 10.1007/BF01692511.

21. **Korns, M. F. (2011).** Abstract expression grammar symbolic regression. Chapter 7, Springer New York, pp. 109–128. DOI: 10.1007/978-1-4419-7747-2_7.

22. **Koza, J. R. (1989).** Hierarchical genetic algorithms operating on populations of computer programs. IJCAI'89: Proceedings of the 11th international joint conference on Artificial intelligence, Vol. 1, pp. 768–774.

23. **Koza, J. R. (1992).** Genetic programming. Massachusetts Institute of Technology.

24. **Koza, J. R. (1994).** Genetic programming II: Automatic discovery of reusable programs, Vol. 1, The MIT Press. DOI: 10.5555/183460.

25. **Koza, J. R. (2010).** Human-competitive results produced by genetic programming. Genetic Programming and Evolvable Machines, Vol. 11, pp. 251–284. DOI: 10.1007/s10710-010-9112-3.

26. **Langdon, W. B. (1998).** Genetic programming and data structures: genetic programming+ data structures= automatic programming! Springer Science & Business Media. DOI: 10.1007/978-1-4615-5731-9.

27. **Laporte, G., Gendreau, M., Potvin, J. Y., Semet, F. (2000).** Classical and modern heuristics for the vehicle routing problem. International Transactions in Operational Research, Vol. 7, No. 4, pp. 285–300. DOI: 10.1016/S0969-6016(00)00003-4.

28. **Larrañaga, P., Kuijpers, C., Murga, R. H., Inza, I., Dizdarevic, S. (1999).** Genetic algorithms for the travelling salesman problem: A review of representations and operators. Artificial Intelligence Review, Vol. 13, No. 2, pp. 129–170. DOI: 10.1023/A:1006529012972.

29. **Mégane, J., Lourenço, N., Machado, P., Schweim, D. (2023).** The influence of probabilistic grammars on evolution. Proceedings of the Companion Conference on Genetic and Evolutionary Computation, Association for Computing Machinery, pp. 611–614. DOI: 10.1145/3583133.3590706.

30. **Nicolau, M., Agapitos, A. (2021).** Choosing function sets with better generalisation performance for symbolic regression models. Genetic Programming and Evolvable Machines, Vol. 22, No. 1, pp. 73–100. DOI: 10.1007/s10710-020-09391-4.

31. **Nordin, P. (1999).** Genetic programming III - Darwinian invention and problem solving. Evolutionary Computation, Vol. 7, No. 4, pp. 451–453. DOI: 10.1162/evco.1999.7.4.451.

32. **O'Neil, M., Ryan, C. (2003).** Grammatical evolution. Springer US. DOI: 10.1007/978-1-4615-0447-4_4.

33. **O'Neill, M., Brabazon, A. (2006).** Grammatical differential evolution. IC-AI, pp. 231–236.

34. **O'Neill, M., Ryan, C. (2001).** Grammatical evolution. IEEE Transactions on Evolutionary Computation, Vol. 5, No. 4, pp. 349–358. DOI: 10.1109/4235.942529.

35. **Ortega, A., de-la-Cruz, M., Alfonseca, M. (2007).** Christiansen grammar evolution: Grammatical evolution with semantics. IEEE Transactions on Evolutionary Computation, Vol. 11, No. 1, pp. 77–90. DOI: 10.1109/TEVC.2006.880327.

36. **O'Neill, M., Brabazon, A. (2004).** Grammatical swarm. Genetic and Evolutionary Computation–GECCO 2004: Genetic and Evolutionary Computation Conference, pp. 163–174.

37. **Ryan, C., Collins, J. J., O'Neill, M. O. (1998).** Grammatical evolution: Evolving programs for an arbitrary language. Genetic Programming: First European Workshop, pp. 89–96. DOI: 10.1007/BFb0055930.

38. **Ryan, C., O'Neill, M., Collins, J. J. (2018).** Handbook of grammatical evolution. Springer International Publishing. DOI: 10.1007/978-3-319-78717-6.

39. **Schmid, U. (2003).** Automatic programming. Inductive Synthesis of Functional Programs: Universal Planning, Folding of Finite Programs, and Schema Abstraction by Analogical Reasoning, pp. 99–166. DOI: 10.1007/978-3-540-44846-4_6.

40. **Sotelo-Figueroa, M. A., Hernández-Aguirre, A., Espinal, A., Soria-Alcaraz, J. A., Ortiz-López, J. (2018).** Symbolic regression by means of grammatical evolution with estimation distribution algorithms as search engine. Fuzzy Logic Augmentation of Neural and Optimization Algorithms: Theoretical Aspects and Real Applications, Springer International Publishing, pp. 169–177. DOI: 10.1007/978-3-319-71008-2_14.

41. **Sotelo-Figueroa, M. A., Puga-Soberanes, H. J., Carpio-Valadez, J. M., Fraire-Huacuja, H. J., Cruz-Reyes, L., Soria-Alcaraz, J. A. (2014).** Improving the bin packing heuristic through grammatical evolution based on swarm intelligence. Mathematical Problems in

Engineering, Vol. 2014, No. 1, pp. 1–12. DOI: 10.1155/2014/545191.

42. **Tsoulos, I. G., Tzallas, A. (2023).** A feature construction method that combines particle swarm optimization and grammatical evolution. Applied Sciences, Vol. 13, No. 14, pp. 8124. DOI: 10.3390/app13148124.

43. **White, D. R., McDermott, J., Castelli, M., Manzoni, L., Goldman, B. W., Kronberger, G., Jaśkowski, W., O'Reilly, U. M., Luke, S. (2013).** Better GP benchmarks: Community survey results and proposals. Genetic Programming and Evolvable Machines, Vol. 14, No. 1, pp. 3–29. DOI: 10.1007/s10710-012-9177-2.

44. **Zúñiga, B. V., Carpio, J. M., Sotelo-Figueroa, M. A., Espinal, A., Purata-Sifuentes, O. J., Ornelas, M., Soria-Alcaraz, J. A., Rojas, A. (2020).** Exploring random permutations effects on the mapping process for grammatical evolution. Journal of Automation, Mobile Robotics and Intelligent Systems, pp. 65–72. DOI: 10.14313/JAMRIS/1-2020/8.

45. **Zuñiga-Nuñez, B. V., Carpio, J. M., Sotelo-Figueroa, M. A., Soria-Alcaraz, J. A., Purata-Sifuentes, O. J., Ornelas, M., Rojas-Domínguez, A. (2020).** Studying grammatical evolution's mapping processes for symbolic regression problems. Intuitionistic and Type-2 Fuzzy Logic Enhancements in Neural and Optimization Algorithms: Theory and Applications, Springer, pp. 445–459. DOI: 10.1007/978-3-030-35445-9_32.

Article received on 30/01/2024; accepted on 06/05/2024.
Corresponding author is Marco Aurelio Sotelo-Figueroa.