Studies in Computational Intelligence 862

Oscar Castillo
Patricia Melin
Janusz Kacprzyk *Editors*

Intuitionistic and Type-2 Fuzzy Logic Enhancements in Neural and Optimization Algorithms: Theory and Applications

Springer

Oscar Castillo · Patricia Melin ·
Janusz Kacprzyk
Editors

Intuitionistic and Type-2 Fuzzy Logic Enhancements in Neural and Optimization Algorithms: Theory and Applications

Springer

Editors
Oscar Castillo
Division of Graduate Studies and Research
Tijuana Institute of Technology
Tijuana, Baja California, Mexico

Patricia Melin
Division of Graduate Studies and Research
Tijuana Institute of Technology
Tijuana, Baja California, Mexico

Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
Warsaw, Poland

ISSN 1860-949X ISSN 1860-9503 (electronic)
Studies in Computational Intelligence
ISBN 978-3-030-35444-2 ISBN 978-3-030-35445-9 (eBook)
https://doi.org/10.1007/978-3-030-35445-9

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Contents

Intelligent Agents

Pattern Recognition

Studying Grammatical Evolution's Mapping Processes for Symbolic Regression Problems

B. V. Zuñiga-Nuñez, J. Martín Carpio, M. A. Sotelo-Figueroa, J. A. Soria-Alcaraz, O. J. Purata-Sifuentes, Manuel Ornelas and A. Rojas-Domínguez

Abstract Grammatical Evolution (GE) is a variant of Genetic Programming (GP) that uses a BNF-grammar to create syntactically correct solutions. GE is composed of the following components: the Problem Instance, the BNF-grammar (BNF), the Search Engine (SE) and the Mapping Process (MP). GE allows creating a distinction between the solution and search spaces using an MP and the BNF to translate from genotype to phenotype, that avoids invalid solutions that can be obtained with GP. One genotype can generate different phenotypes using a different MP. There exist at least three MPs widely used in the art-state: Depth-first (DF), Breadth-first (BF) and πGrammatical Evolution (piGE). In the present work DF, BF, and piGE have been studied in the Symbolic Regression Problem. The results were compared using a statistical test to determine which MP gives the best results.

Keywords Grammatical evolution · Symbolic problem · Mapping process

B. V. Zuñiga-Nuñez · J. M. Carpio · M. Ornelas · A. Rojas-Domínguez
Departamento de Estudios de Posgrado e Investigación, Tecnológico Nacional de México, Instituto Tecnológico de León, Avenida Tecnológico S/N, 37290 León, GTO, Mexico
e-mail: m18240006@itleon.edu.mx

J. M. Carpio
e-mail: jmcarpio61@hotmail.com

M. Ornelas
e-mail: mornelas67@yahoo.com.mx

A. Rojas-Domínguez
e-mail: alfonso.rojas@gmail.com

M. A. Sotelo-Figueroa (✉) · J. A. Soria-Alcaraz · O. J. Purata-Sifuentes
División de Ciencias Económico Administrativas, Departamento de Estudios Organizacionales, Universidad de Guanajuato, Fraccionamiento I El Establo, 36250 Guanajuato, GTO, Mexico
e-mail: masotelo@ugto.mx

J. A. Soria-Alcaraz
e-mail: jorge.soria@ugto.mx

O. J. Purata-Sifuentes
e-mail: opurata@ugto.mx

O. Castillo et al. (eds.), *Intuitionistic and Type-2 Fuzzy Logic Enhancements in Neural and Optimization Algorithms: Theory and Applications*, Studies in Computational Intelligence 862, https://doi.org/10.1007/978-3-030-35445-9_32

1 Introduction

Automatic Programming (AP) [1, 2] has been defined as a program that can construct other programs by itself. One of the AP's elements [2] is the application domain, the output is an efficient program that satisfies the requirements.

Genetic Programming (GP) [1] is an AP technique proposed by Koza in 1998, it takes inspiration from nature and genetics. GP uses a tree representation [3], it starts with an initial population of trees and each one represents candidate solutions to a specific problem, and each one of these individuals is evaluated according to its performance to solve the problem employing an objective function. Each individual is evolved using the Genetic Algorithm operators [4], like the selection, crossover, and mutation.

Grammatical Evolution (GE) [5] is GP based form. GE uses a BNF-Grammar to generate syntactically correct solutions specifying the rules or restrictions that are necessary to create the solution. The individuals are a binary string that is evolved by a Search Engine (SE) [5], this allows making a distinction between the search space made by the individuals and the solution space created by the mapping process; thus this is possible to replace the SE used [6–8].

The main components of GE are the Problem Instance, the Search Engine, and the MP [7]. Each one of these represents a research area in itself and could be replaced. Different studies have been done in the field with the purpose of investigating the performance of using various SE like Differential Evolution (DE) [9], Estimation Distribution Algorithm (EDA) [8], Genetic Algorithm (GA) [10], Particle Swarm Optimization (PSO) [7] and Particle Evolutionary Swarm Optimization (PESO) [7]; as well as different types of grammars [11, 12].

The Mapping Process (MP) has the responsibility of making the genotype to phenotype mapping; it is possible to create different phenotypes from the same genotype just by changing the MP. There exist earlier research in the field of MPs, in [13] were studied 4 alternatives MPs: Depth-first (DF) [14], Breadth-first (BF) [15], πGrammatical Evolution (piGE) [16] and a Random control strategy applied to 4 benchmark problems; in [15] a two main MPs investigation between the DF and piGE processes is presented, but also another two MPs were studied. In both works it is concluded that piGE shows better performance as MP for GE.

GE has been used to solve diverse types of problems: The Bin Packing Problem (BPP) [7] , Even-5-parity [13, 15], Santa Fe Ant trail (SFAT) [13, 15], the design of Partially-Connected Artificial Neural Networks (DANN) [10], and the Symbolic Regression Problem (SRP) [8, 13, 15].

In this paper the DF, BF, and piGE are applied to ten well-known instances of the traditional SRP. The results are compared using a statistical test.

The paper is structured as follows: Sect. 2 presents a brief introduction to GE, BNF-Grammars and MPs, in Sect. 3 the Symbolic Regression Problem is described; the corresponding setup used in the experimental approach is explained in Sect. 4; and, finally the results and conclusions are exposed in Sects. 5 and 6 respectively.

2 Grammatical Evolution

Grammatical Evolution (GE) [5] is a variant of Genetic Programming (GP) [1], proposed by Ryan and O'Neill in 1998; GE differs from the traditional GP in 3 fundamental ways: it employs linear genotypes, it performs a genotype to phenotype mapping, and makes use of a grammar to create the solutions.

A Backus Naur Form Grammar (BNF-Grammar) is used to generate syntactically correct solutions (phenotypes) [5, 17], attending to certain restrictions and considering the context of the problem [18] taking the form of production rules. In order to select the corresponding production rule, a Genotype (binary or integer string) and a MP are employed [5, 7, 17, 19].

Derived from the MP a derivative tree is created, and a phenotype is obtained.

The generic algorithm of GE is described in Algorithm 1.

Algorithm 1 Grammatical Evolution Algorithm

```
Population = new_population(pop_size)
Generations = num_gen
Solution_found = false
Initialize (Population)
while Generations≥0 do
  PerformMappingProcess(Population)
  Evaluate(Population)
  if Solution_found then
    Return Best Solution
      end
  end if
  if Solution_found == false then
      PerformGeneticOperators(Population)
      Generations - -
  end if
end while
Return the best solution found
```

Due to the use of an MP, GE allows a distinction between the solution and search spaces, which means that depending on the use of one or another MP, a single genotype can be derived into distinct phenotypes. This characteristic allows GE to avoid the problem of getting stuck in local optima that the traditional GP has [15].

It is necessary to define the following components for the GE [7, 8]:

- Problem Instance.
- BNF-Grammar.
- Search Engine.

Figure 1 shows the methodology used in GE. The mapping process has been added as a fourth component to the original proposal taken from [7].

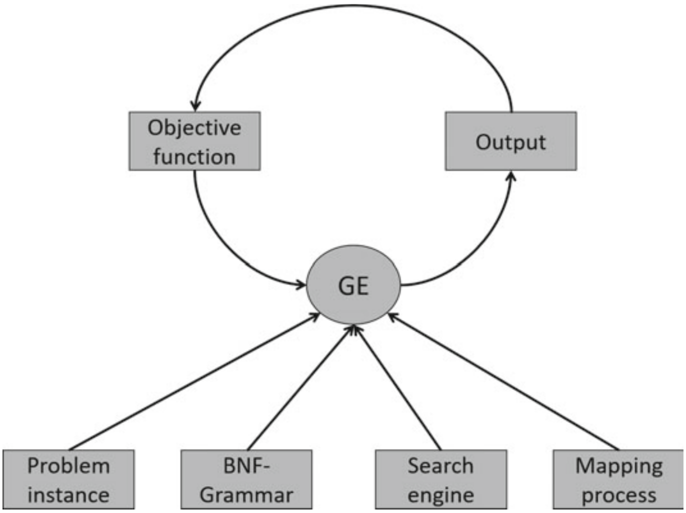

Fig. 1 Study approach GE's methodology based on [7]

2.1 BNF-Grammar

A Context-Free Grammar (CFG) [11] don't depend on the surroundings and is composed of a tuple defined as

$$G = \{N, T, R, S\}$$

where:

– N is the set of non-terminal symbols.
– R corresponds to the production rules.
– T is the set of terminal symbols.
– S corresponds to the start symbol, and $S \in$ N.

Backus Naur Form (BNF) [20] is a notation used to express a grammar in a language in the form of production rules.

BNF consists of two main components: the terminals which are objects that can appear in the final expression, and the non-terminals which can be expanded into one or more terminals or non-terminals. A grammar is defined by a set of rules that determine a complex structure from small blocks.

The Grammar 1 represents an example of a BNF-Grammar, the non-terminals symbols are surrounded by "<>", contrary to the terminals, which don't have any surrounding symbols, and each non-terminal has a set of production rules or options that are separated by the symbol "$\models$". The production rules are separated by the symbol "|".

$$
\begin{aligned}
\langle \text{start} \rangle &\models \langle \text{e} \rangle \\
\langle \text{e} \rangle &\models \langle \text{e} \rangle \langle \text{o} \rangle \langle \text{e} \rangle \;\mid\; \langle \text{v} \rangle \\
\langle \text{v} \rangle &\models \text{X} \;\mid\; \text{Y} \\
\langle \text{o} \rangle &\models + \;\mid\; \text{-}
\end{aligned}
$$

Grammar 1 Example of a BNF-Grammar

As the grammar determine the structure of the solutions, changing the complete behavior of the solution is as simple as changing the grammar [13].

2.2 Search Engine

The principal objective of the Search Engine (SE) is to evolve the genotypes using as an objective value the evaluation of the instances in their phenotypic representation. Originally GE uses the the Genetic Algorithm (GA) [1] as SE. GA takes inspiration from the natural selection proposed by Darwin in 1859 and uses genetic operators such as selection, crossover, and mutation to evolve the genotype. The GA is shown in Algorithm 2.

Algorithm 2 Genetic Algorithm

Population = new_population(pop_size)
Generations = num_gen
Solution_found = false
Initialize (Population)
while *termination condition not satisfied* ≥ 0 **do**
    Generations += 1
    PerformGeneticOperators(Population)
    Evaluate(Population)
    if *Solution_found* **then**
        Return Best Solution
        end
    end if
end while
Return the best solution found

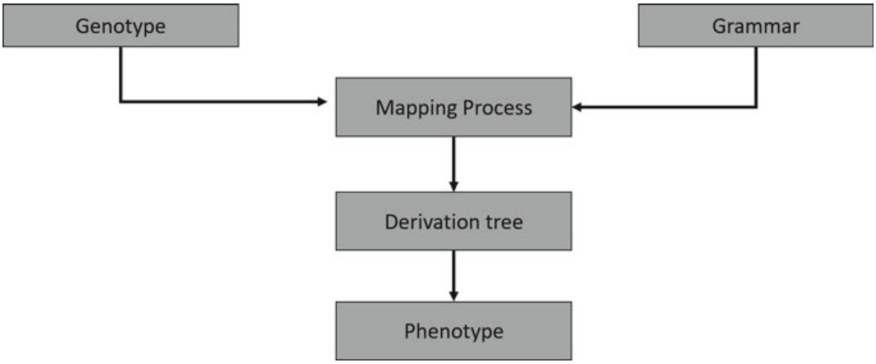

Fig. 2 Generic mapping process [15]

2.3 Mapping Process

The Mapping Process (MP) in GE is responsible for transforming a genotype to a phenotype, using a binary string as genotype and the productions derivation tree from the BNF grammar (Fig. 2 illustrates this process); the corresponding phenotype is obtained from this derivative tree. To do this, all the non-terminals (NT) are expanded using Eq. 1.

$$Prod\ rule = Codon\ value\ \%\ Number\ of\ production\ rules\ for\ the\ NT. \tag{1}$$

To exemplify the studied MPs, we will take the sample Grammar 1, and the following genotype to show the obtained derivative tree by each one of them.

Genotype = `2,12,7,9,3,15,23,1,11,4,6,13,2,7,8,3,35,19,2,6.`

Depth-First Mapping Process Depth-First (DF) [14] is considered the standard for GE, it begins from the start symbol and makes the expansion taking the left-most NT symbol available in the derivative tree. Figure 3 shows an example of the DF MP.

Equation 1 is performed using each codon value of the genotype. The number out of the parenthesis indicates the expansion order of the tree. In the example we take the start symbol <e>, the first codon value, which is 2, and the number of production rules that corresponds to the NT <e> which is 2; applying Eq. 1 we substitute 2%2 = 0, and this means that the corresponding production rule is the one in the position 0. Now we have three available NTs: <e>, <o> and <e>. We take the left-most NT, which corresponds to <e>.

We expand <e> into <e> <o> <e>, and so on, until no more NTs remain in the derivative tree. The corresponding algorithm for DF is shown in Algorithm 3.

Breadth-First Mapping Process The Breadth-First (BF) [15] uses an expansion process slightly different from DF. It uses the same equation to chose the

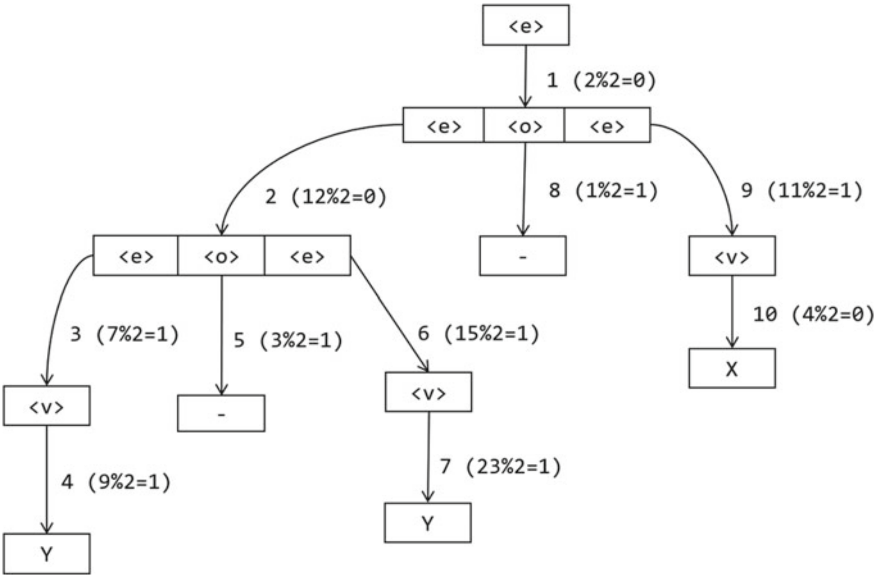

Fig. 3 Example of the Depth-First Mapping Process [14]

Algorithm 3 Depth-First Mapping Process Algorithm [14]

listNT {List to store NTs seen}
Add start symbol from grammar to *listNT*
wraps = 0
while *listNT* is not empty **do**
  if reached end of chromosome **then**
    wraps++
    if *wraps*> max wraps allowed **then**
      return false
    end if
    reset chromosome iterator
  end if
  CurrentNT = get head of *listNT*
  CurrentCodon = get next codon value
  newProduction = *currentCodon % number of productions for currentNT*
  set *currentNTs* children = *newProduction*
  {This is the key to depth first mapping}
  add *newProduction* to head of *listNT* {Only adds NTs}
end while
Generate phenotype by traversing the leaf nodes of the derivation tree
return true

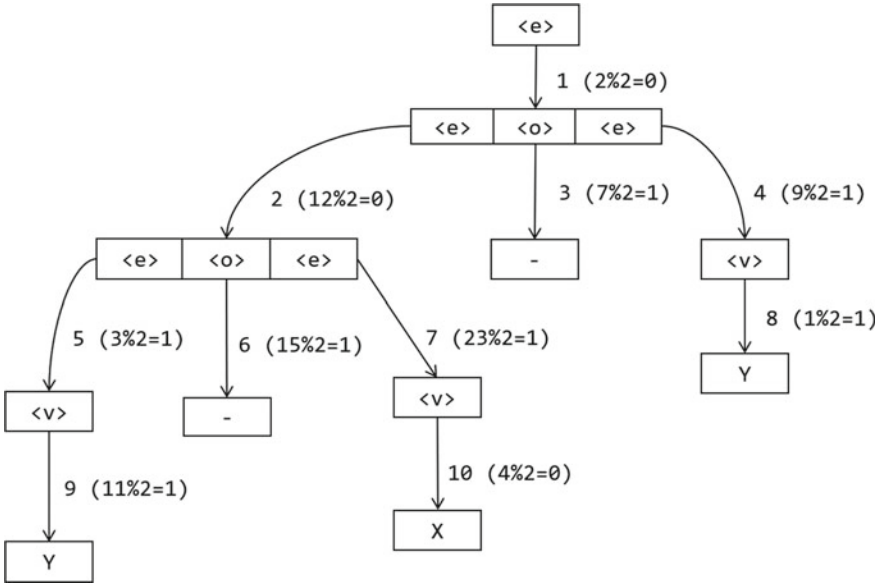

Fig. 4 Example of the Breadth-First Mapping Process [15]

corresponding production rule to expand, but the expansion is done by expanding all the NTs at the same level (taking the NTs from left to right) before moving to the next level in the expansion tree.

The process begins with the start symbol <e>, applying the module rule, the corresponding production rule to expand is <e><o> <e>, we continue the expansion by taking these last three obtained NTs, and so on level by level, until no more NTs remain.

An example of this process can be seen in Fig. 4, where the expansion order is indicated by the numbers out the parenthesis. The corresponding algorithm for BF is shown in Algorithm 4.

π Grammatical Evolution Mapping Process The π Grammatical Evolution (piGE) [16] differs from the previous ones because it uses a pair of codons, the first one (the order codon) dictates the order of the expansion (it says which one of the available NTs in the derivative tree will be expanded); and the second one, called the content codon, works in the traditional way as in the other MPs.

piGE removes the linear dependency by evolving the expansion order in the evolutionary search [21] by making use of the order codon.

Equation 2 is used to choose the expansion order.

$$NT\ to\ expand = Order\ Codon\ value\ \%\ Number\ of\ available\ NTs. \quad (2)$$

Figure 6 represents the order choice list of the expansions for the NTs, and Fig. 5 shows the corresponding derivative tree.

Algorithm 4 Breadth-First Mapping Process Algorithm [15]

```
listNT {List to store NTs seen}
Add start symbol from grammar to listNT
wraps = 0
while listNT is not empty do
  if reached end of chromosome then
    wraps++
    if wraps> max wraps allowed then
      return false
    end if
    reset chromosome iterator
  end if
  CurrentNT = get head of listNT
  CurrentCodon = get next codon value
  newProduction = currentCodon % number of productions for currentNT
  set currentNTs children = newProduction
  {This is the key to breadth first mapping}
  add newProduction to tail of listNT {Only adds NTs}
end while
Generate phenotype by traversing the leaf nodes of the derivation tree
return true
```

The example shown in Fig. 5 begins with the start symbol, as in the first step the only available NT is <e>, Eq. 2 chose this NT that corresponds to the option in position 0. Using the second codon we select the corresponding production rule, and now we have three different options of NTs to expand. Taking the next pair of codons, we select and expand the available NT <o>. We repeat this process until there is no more available NTs to expand in the order choice list.

The corresponding algorithm for piGE is shown in Algorithm 5.

3 Symbolic Regression Problem

The Symbolic Regression Problem (SRP) [1, 8, 15] is the process of obtaining a representative expression for a given set of finite points, it is used to know what was the expression that generated this data.

SRP represents an important task studied in the GP community [22].

The main objective of the SRP is finding the best combination of variables, coefficients, and symbols.

For example, the expression $\cos 2x$ can be represented with another equation, such as $1 - 2\sin^2 x$. Both expressions give as a result the same values in a specific range of points.

Algorithm 5 π Grammatical Evolution Mapping Process Algorithm [16]

```
listNT {List to store NTs seen}
Add start symbol from grammar to listNT
wraps = 0
while listNT is not empty do
  if reached end of chromosome then
    wraps++
    if wraps> max wraps allowed then
      return false
    end if
    reset chromosome iterator
  end if
  {This is where the piGE order comes in}
  currentOrderCodon = get next codon value
  nextProductionIndex = currentOrderCodon % size of listNT
  currentNT = get listNT[nextProductionIndex]
  currentContentCodon = get next codon value
  newProduction = currentCodon % number of productions for currentNT
  set currentNTs children = newProduction
  {The new NTs are added where the parent NT was removed from}
  insert newProduction at listNT[nextProductionIndex]{Only adds NTs}
end while
Generate phenotype by traversing the leaf nodes of the derivation tree
return true
```

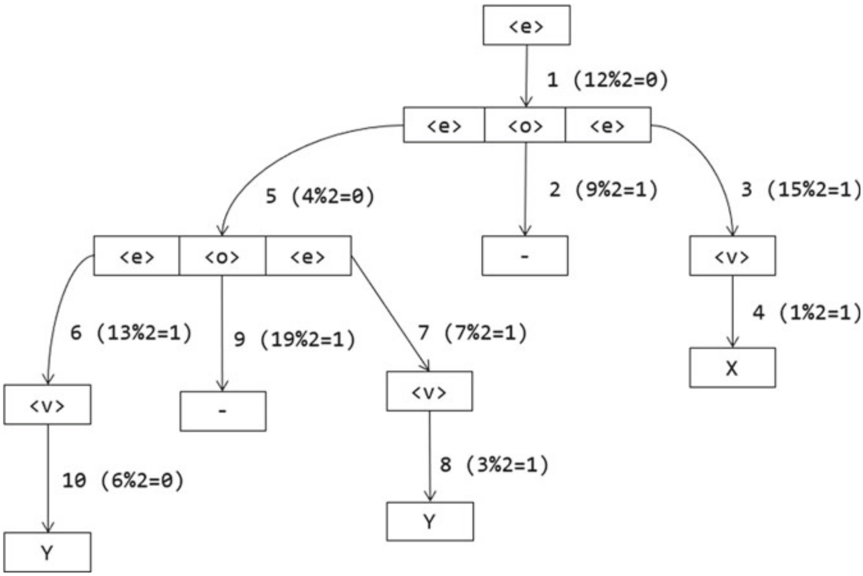

Fig. 5 Example of the π Grammatical Evolution Mapping Process [16]

Fig. 6 Order choice list for piGE

```
1. [(e)]        →  2%1=0
2. [e, (o), e]  →  7%3=1
3. [e, (e)]     →  3%2=1
4. [e, (v)]     →  23%2=1
5. [(e)]        →  11%1=0
6. [(e), o, e]  →  6%3=0
7. [v, o, (e)]  →  2%3=2
8. [v, o, (v)]  →  8%3=2
9. [v, (o)]     →  35%2=1
10.[(v)]        →  2%1=0
```

4 Experimental Setup

GA was used as SE, Table 1 shows the GA parameters used.

The SRP functions used as instances set are shown in Table 2, and it were taken from [8].

The Mean Root Squared Error (MRSE), Eq. 3, was used as objective function to evaluate the performance on the candidate solutions.

The grammar used for the SRP is shown in Grammar 2.

It was performed 33 independently runs for each instance with each MP. It was taken the median result to make the statistical comparison.

A statistical test was applied to discern about the correct test to be applied.

$$MRSE = \sqrt{\frac{\sum_{i=1}^{N}(x_i - y_i)^2}{N}} \tag{3}$$

where:

- N is the number of data points.
- x corresponds to the obtained value.
- y is the real value.

⟨start⟩ ⊨ ⟨expr⟩
⟨expr⟩ ⊨ (⟨expr⟩⟨op⟩⟨expr⟩) | ⟨pre-op⟩(⟨expr⟩) | ⟨var⟩
⟨var⟩ ⊨ X | Y | 1.0
⟨pre-op⟩ ⊨ sin | cos | exp | log
⟨op⟩ ⊨ + | - | / | *

Grammar 2 Grammar used for the Symbolic Regression Problem [8]

Table 1 Parameter setup used

Parameter	Value
Population size	100
Dimensions	100
Function calls	250,000
Selection method	Random
Crossover	2 points
Mutation rate	1.0
Elitism rate	0.30

Table 2 Symbolic regression functions [8] used as instances set

Function	Fit cases
$F_1 = X^3 + X^2 + X$	20 random points $\in [-1, 1]$
$F_2 = X^4 + X^3 + X^2 + X$	
$F_3 = X^5 + X^4 + X^3 + X^2 + X$	
$F_4 = X^6 + X^5 + X^4 + X^3 + X^2 + X$	
$F_5 = \sin(X^2)\cos(X) - 1$	
$F_6 = \sin(X) + \sin(X + X^2)$	
$F_7 = \log(X + 1) + \log(X^2 + 1)$	20 random points $\in [0, 2]$
$F_8 = \sqrt{X}$	20 random points $\in [0, 4]$
$F_9 = \sin(X) + \sin(Y^2)$	200 random points $\in [-1, 1]$, X $\in [-1, 1]$, Y $\in [-1, 1]$
$F_{10} = 2\sin(X)\cos(Y)$	

5 Results

Table 3 shown the median obtained by each instance with an MP.

A Shapiro-Wilk test [23] was performed with the results obtained to know if the data belong to a Normal distribution [24]. The Shapiro-Wilk test gives a *p*-value of 1.201e-05 that indicates that the data don't come from a Normal distribution.

The Shapiro-Wilk test shows that the data doesn't belong to a Normal distribution. Considering this, a Friedman non-parametric test [25] was applied to know if there is a significant difference between the results obtained by the MPs.

The *p*-value obtained by the Friedman non-parametric test is 0.067.

Table 3 Obtained results for each MP at each function

	Depth-First	Breadth-First	piGE
F_1	0.0476	0.0653	0.0532
F_2	0.0443	0.1170	0.0572
F_3	0.0986	0.1062	0.0709
F_4	0.0851	0.1948	0.0747
F_5	0.0374	0.0379	0.0291
F_6	0.0262	0.0355	0.0402
F_7	0.0092	0.0096	0.0120
F_8	0.0354	0.0591	0.0249
F_9	0.0291	0.0301	0.0147
F_{10}	0.0256	0.0221	0.0221

6 Conclusions and Future Work

In the present paper, three Mapping Processes were applied to ten instances of the Symbolic Regression Problem. The obtained results were compared with a Friedman non-parametric test with the purpose of know if there is a statistical difference between the studied Mapping Processes.

Derived from the statistical test, we could conclude that there is no statistical evidence to discern about the performance of the three studied Mapping Processes for Grammatical Evolution applied to the Symbolic Regression Problem. This implies that is possible to use any Mapping Process.

As future work, it is proposed to determine the algorithmic complexity for each MP to know if one of them can be easily applied. There exist other MP, like the Univariate Model-Based Grammatical Evolution (UMBGE), that can be applied to the Symbolic Regression Problem and can compare it with the results obtained.

Acknowledgements The authors want to thank National Council for Science and Technology of Mexico (CONACyT) through the scholarship for postgraduate studies: 703582 (B. Zuñiga) and the Research Grant CÁTEDRAS-2598 (A. Rojas), the León Institute of Technology (ITL), and the Guanajuato University for the support provided for this research.

References

1. Koza, J.R.: Genetic Programming On the Programming of Computers by Means of Natural Selection. Massachusetts Institute of Technology, Cambridge (1998)
2. Rich, C., Waters, R.C.: Automatic programming: myths and prospects. Computer **21**, 40–51 (1988)

3. Igwe, K., Pillay, N.: A study of genetic programming and grammatical evolution for automatic object-oriented programming: a focus on the list data structure. In: Pillay, N., Engelbrecht, A.P., Abraham, A., du Plessis, M.C., Snášel, V., Muda, A.K. (eds.) Advances in Nature and Biologically Inspired Computing, pp. 151–163. Springer, Cham (2016)
4. Holland, J.H.: Adaptation in Natural and Artificial Systems. In: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT Press (1992)
5. Ryan, C., Collins, J., Neill, M.O.: Grammatical evolution: evolving programs for an arbitrary language. In: Banzhaf, W., Poli, R., Schoenauer, M., Fogarty, T.C. (eds.) Genetic Programming, vol. 1391, pp. 83–96. Springer, Berlin, Heidelberg (1998)
6. Brabazon, A., O'Neill, M., McGarraghy, S.: Natural Computing Algorithms. Natural Computing Series, 1st edn. Springer, Berlin, Heidelberg (2015)
7. Sotelo-Figueroa, M.A., Soberanes, H.J.P., Carpio, J.M., Huacuja, H.J.F., Reyes, L.C., Soria-Alcaraz, J.A.: Improving the bin packing heuristic through grammatical evolution based on swarm intelligence. Math. Probl. Eng. (2014)
8. Sotelo-Figueroa, M.A., Hernández-Aguirre, A., Espinal, A., Soria-Alcaraz, J.A., Ortiz-López, J.: Symbolic regression by means of grammatical evolution with estimation distribution algorithms as search engine. In: Fuzzy Logic Augmentation of Neural and Optimization Algorithms: Theoretical Aspects and Real Applications, vol. 749, pp. 169–177. Springer (2018)
9. O'Neill, M., Brabazon, A.: Grammatical differential evolution. In: IC-AI, pp. 231–236 (2006)
10. Quiroz-Ramírez, O., Espinal, A., Ornelas-Rodríguez, M., Rojas-Domínguez, A., Sánchez, D., Puga-Soberanes, H., Carpio, M., Espinoza, L.E.M., Ortíz-López, J.: Partially-connected artificial neural networks developed by grammatical evolution for pattern recognition problems. In: Fuzzy Logic Augmentation of Neural and Optimization Algorithms: Theoretical Aspects and Real Application, pp. 99–112. Springer, Cham (2018)
11. Hemberg, E.A.P.: An exploration of grammars in grammatical evolution. Ph.D. thesis, University College Dublin (2010)
12. Nicolau, M., Agapitos, A.: Understanding grammatical evolution: Grammar design. In: Handbook of Grammatical Evolution, pp. 23–53. Springer (2018)
13. Fagan, D., O'Neill, M., Galván-López, E., Brabazon, A., McGarraghy, S.: An analysis of genotype-phenotype maps in grammatical evolution. In: European Conference on Genetic Programming, pp. 62–73. Springer, Berlin, Heidelberg (2010)
14. O'Neill, M., Ryan, C.: Grammatical evolution. IEEE Trans. Evo. Comput. **5**, 349–358 (2001)
15. Fagan, D., O'Neill, M.: Analysing the Genotype-Phenotype Map in Grammatical Evolution. Ph.D. thesis, University College Dublin (2013)
16. O'Neill, M., Brabazon, A., Nicolau, M., Garraghy, S.M., Keenan, P.: π grammatical evolution. In: Deb, K. (ed.) Genetic and Evolutionary Computation–GECCO 2004, pp. 617–629. Springer, Berlin, Heidelberg (2004)
17. Dempsey, I., O'Neill, M., Brabazon, A.: Foundations in Grammatical Evolution for Dynamic Environments. Vol. 194 of Studies in Computational Intelligence, 1st edn. Springer, Berlin, Heidelberg (2009)
18. Hugosson, J., Hemberg, E., Brabazon, A., O'Neill, M.: Genotype representations in grammatical evolution. Appl. Soft Comput. **10**(1), 36–43 (2010)
19. McKay, R.I., Hoai, N.X., Whigham, P.A., Shan, Y., O'Neill, M.: Grammar-based genetic programming: a survey. Gen. Program. Evol. Mach. **11**, 365–396 (2010)
20. Backus, J.W., Bauer, F.L., Green, J., Katz, C., McCarthy, J., Perlis, A.J., Rutishauser, H., Samelson, K., Vauquois, B., Wegstein, J.H., van Wijngaarden, A., Woodger, M.: Revised report on the algorithm language algol 60. Commun. ACM **6**, 1–17 (1963)
21. Fagan, D., Murphy, E.: Mapping in grammatical evolution. In: Handbook of Grammatical Evolution, pp. 79–108. Springer (2018)
22. White, D.R., McDermott, J., Castelli, M., Manzoni, L., Goldman, B.W., Kronberger, G., Jaskowski, W., O'Reilly, U.-M., Luke, S.: Better GP benchmarks: community survey results and proposals. Gen. Program. Evol. Mach. **14**, 3–29 (2012)
23. Shapiro, S.S., Wilk, M.B.: An analysis of variance test for normality (complete samples). Biometrika **52**, 591 (1965)

24. Soria-Alcaraz, J.A., Sotelo-Figueroa, M.A., Espinal, A.: Statistical comparative between selection rules for adaptive operator selection in vehicle routing and multi-knapsack problems. In: Fuzzy Logic Augmentation of Neural and Optimization Algorithms: Theoretical Aspects and Real Applications, pp. 389–400. Springer (2018)
25. Derrac, J., García, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol. Comput. **1**, 3–18 (2011)