A Hybrid Approach of Grammar-based Genetic Programming and Differential Evolution for Symbolic Regression

Flávio A.A. Motta[1], João M. de Freitas[1], Felipe R. de Souza[1], Heder S. Bernardino[1], Itamar L. de Oliveira[1], and Helio J.C. Barbosa[1,2]

[1] Universidade Federal de Juiz de Fora, Juiz de Fora (UFJF), MG, Brazil
flavioaam@hotmail.com, joao@ice.ufjf.br, feliperafael@ice.ufjf.br, heder@ice.ufjf.br, itamar.leite@ufjf.edu.br
[2] Laboratório Nacional de Computação Científica (LNCC), Petrópolis, RJ, Brazil
hcbm@lncc.br

Abstract. Genetic Programming (GP) is used for solving many real world problems. From data classification to building phylogenetic trees, the technique can be applied to almost any problem. One way to improve GP performance is by using a formal grammar. We propose here the use of grammar-based genetic programming (GGP) with Differential Evolution (DE). DE is incorporated to GGP in order to improve the quality of solutions obtained by GGP by finding numerical coefficients when solving symbolic regression problems. In this proposal, the coefficients of the best individuals generated by GGP during the search are adjusted by DE. Also, this technique incorporates these values to the grammar; thus, the grammar is adapted during the search. The proposed method is applied to 24 symbolic regression problems and it is compared to a standard GGP. The results indicate that GGP hybridized with DE obtained better models, specially when one expects real-valued coefficients in the model.

Keywords: Grammar-based Genetic Programming, Differential Evolution, Hybridism, Symbolic Regression

1 Introduction

Genetic Programming (GP) is an evolutionary computation technique developed by Koza [1] which aims at automating the construction of a computer program. Supervised machine learning problems are one of the most common applications of this technique, where the programs represent models which map inputs into the outputs.

Grammars are widely used in computer science, and one of their main utility is to syntactically constrain symbolic expressions. This can be either to define valid expressions or to enforce type constraints [2]. Grammar-based Genetic Programming (GGP) is a GP technique in which a formal grammar is used to

constrain the expressions generated during the search. A survey of the grammar-based GP method can be found in [3].

When solving symbolic regression problems, the numerical coefficients of the obtained model are important. If the problem does not involve only integer coefficients the standard GGP may have difficulties to reach the real-valued ones [4]. One way to overcome this limitation is by performing some arithmetic operations. Another way to describe this type of values is to include them in the grammar used. Due the large number of symbolic regression problems in which the desired model is composed by not only integer numerical coefficients, another approach is desirable. Differential Evolution (DE) was first introduced by Storn and Price [5] and is often used for continuous parameter optimization problems. Considered a simple and efficient technique for many real-world optimization problems, such characteristics make DE a good choice to combine with other optimization techniques. Thus, DE is used here to obtain the numerical coefficients of some of the models found by GGP.

Using genetic programming together with another technique is very common [6]. Differential evolution can be used in some phases of GP, but just a few papers exploring this combination can be found in the literature. Shun and Teo [7] used DE to improve GP's mutation phase, and the results obtained showed improvement by using the hybridized method. The algorithm was capable of automatically designing and co-evolving both the controller and the morphology of modular robots.

Roy et al. [8] tried to generate DE's trial vector via GP, using a learning method which would choose the best strategy for the problem. The hybridization performance was compared with four numerical optimization methods and achieved a better success rate over all of them.

Howard and D'Angelo [9] explored a hybridism of genetic programming with a genetic algorithm. This method was named GA-P and it addressed the problem of symbolic regression; the GP part of the algorithm evolves the expression while the GA part concurrently evolves the coefficients used in the expression. The results show that GA-P has a slight advantage over the original GA and over GP.

Rayno et al. [10] also used a hybridism of GP with GA for 3-D Metamaterial Design. The technique uses genetic programming as the main evolver and a genetic algorithm in the evaluation phase. The GA tries to optimize the chromosomes and then insert them in the GP tree. Their work aims at reducing the number of generations needed to reach a good solution in metamaterial designs. They achieved fewer GP generations with their algorithm.

Here, the main objective is to generate better solutions with a GGP by using real-valued coefficients. A technique which hybridizes GGP with DE is proposed, where DE is used to improve the numerical coefficients of the models obtained by GGP. The values found by DE are also added to the grammar. Thus, the main contribution of this work is a procedure to generate better coefficients in a GGP than its standard variant, and which adapts its grammar during the search.

2 Grammar-based Genetic Programming

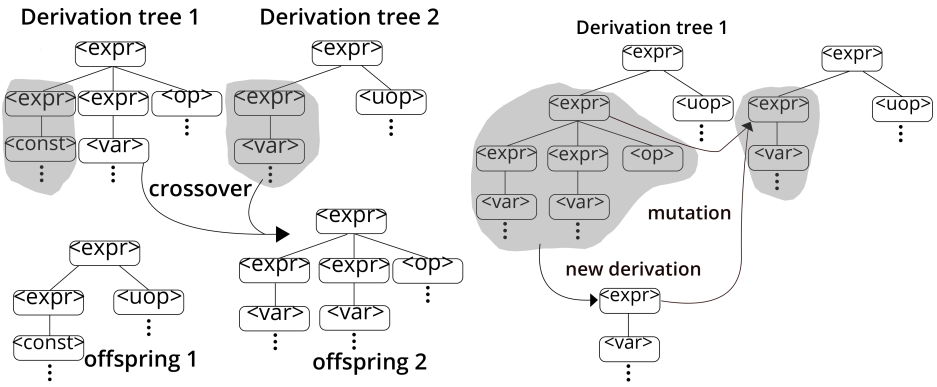
Nature has been a source of inspiration for various tools in computational systems. Two of these are genetic programming (GP) and grammatical evolution (GE). Genetic programming is a technique based on the Darwinian theory of natural selection in which, in short, the best individuals tend to survive, reproduce, and generate fitter offspring. The three main steps of this process are selection, crossover, and mutation. Differently from other algorithms, GP, as stated in [11, 12], has in its favor the production of symbolic models, allowing knowledge extraction from the model by an area expert. Although GP is able to generate interpretable models [13, 14], it demands high computational cost, given the number of objective function evaluations required to reach a good solution. When modeling dynamical phenomena, for example, it may require the numerical solution of ordinary differential equations during the evaluation phase.

In the grammar-based genetic programming (GGP) every individual is generated by means of a formal grammar. Therefore it has some advantages over GP, such as cleaner enforcement of semantic constraints, reduced search space, bias introduction, some possible extensions such as additional genetic operators, and the evolution of the grammar itself. On the other hand its concept is more complex and its implementation is more demanding.

Most of the GP elements are preserved in GGP. For instance, the selection procedure does not depend of the representation and can be the same used in the standard version of GP. During this step, some individuals are selected to be submitted to the genetic operations. Also, each individual is still represented by a tree, although in GGP they are represented by a derivation tree (sentence and nonterminal symbols). A grammar-based derivation tree is composed by sentences, nonterminal and terminal symbols. Every generated individual is syntactically correct with respect to the grammar rules adopted. A candidate expression is represented by the leafs of its tree in the post-order notation. Thus, one can notice that all leafs are terminal symbols of the grammar.

The crossover operates only over coincident nonterminal symbols. Two individuals (parents) generate two offspring. To perform this operation one sub-tree is chosen from each parent and the offspring is created with characteristics from both parents, as shown in Figure 1a The mutation is a perturbation that happens in an individual. First, a node is randomly chosen. Then a sub-tree is created respecting the restriction of the node, as shown in Figure 1b.

To evaluate an individual, GP uses known inputs mapped to outputs. Here, the objective function is the mean squared error between the real values and those inferred by the model. After the evaluation, the algorithm chooses which individuals will go into the next generation. The replacement can be performed using an elitism procedure, where a percentage of the best individuals is selected to compose the next generation. To fill the remaining individuals of the population, the newest generated individuals are chosen.



(a) Crossover (b) Mutation

Fig. 1: The two main genetic operators for GGP

Algorithm 1 Pseudo-code for the Grammar-based Genetic Programming

```
1: procedure GP
2:     Create initial population pop of size N
3:     evaluate (pop)
4:     while Termination criteria not met do
5:         selection(pop)
6:         crossover(pop)
7:         mutation(pop)
8:         evaluate(pop)
9:         replacement(pop)
```

3 Differential Evolution

Differential Evolution was proposed by Storn and Price [5] and, despite its simplicity, is a powerful search technique, specially for continuous variables optimization.

The basic operation performed by DE is the addition to a candidate vector of scaled difference(s) between other candidate solution vectors from the population. The basic mutation operator of the DE is used here, which picks randomly the individuals in the population, leading to $u_{i,j,G+1} = x_{r_1,j,G} + F.(x_{r_2,j,G} - x_{r_3,j,G})$, where r_1, r_2 and r_3 are randomly selected individuals with $r_1 \neq r_2 \neq r_3$. In addition, a crossover operation is performed, using the parameter CR, as explained in Algorithm 2.

4 The Proposed Hybrid Approach

Numerical coefficients in GGP are generated only by means the values in the formal grammar. To reach values different of those ones in grammar, the algorithm

Algorithm 2 Pseudo-algorithm for the Differential Evolution

```
1: procedure DE(PS(populationsize), MUT(mutationscaling), CR(crossoverrate),
   GEN(generationsnumber))
2:     G ← 0
3:     CreateRandomInitialPopulation(PS)
4:     for i = 0 to PS do
5:         Evaluate f(x⃗_{i,G}) /*x⃗_{i,G} is an individual in the population*/
6:     for G ← 1 to GEN do
7:         for i ← 1 to NP do
8:             SelectRandomly(r1, r2, r3) /*r1 ≠ r2 ≠ r3 ≠ i*/
9:             jRand ← RandInt(1, N) /*N is the number of variables*/
10:            for j ← 1 to N do
11:                if Rand(0, 1) < CR or j = jRand then
12:                    u_{i,j,G+1} = x_{r3,j,G} + F.(x_{r1,j,G} − x_{r2,j,G})
13:                else
14:                    u_{i,j,G+1} = x_{i,j,G}
15:            if f(u⃗_{i,G+1}) ≤ f(x⃗_{i,G}) then
16:                x⃗_{i,G+1} = u⃗_{i,G+1}
17:            else
18:                x⃗_{i,G+1} = x⃗_{i,G}
```

must carry out arithmetic operations over them. This can be very costly as the tree has a limited depth and some numbers may require several operations to be calculated. A smaller tree is faster to execute and simpler to understand than the larges ones, so models represented by smaller trees are desirable.

One way to solve this problem is to use another technique to found the numerical coefficients. Here we propose the use of DE, a simple and efficient search technique. We termed our hybrid approach DE Mutation, as DE performs a perturbation in all numerical coefficients of an individual created by GGP.

DE mutation does not replace the original GGP mutation. The idea is to perform the original GGP operators first and, in the sequence, to improve the created individual using DE. All the numerical coefficients in an individual are then replaced by those found by DE.

The proposed grammar-based genetic programming executes the standard genetic operators in every generation. In equally spaced intervals, DE is used to carry out numerical coefficient adjustment. We use Ω to represent how many times DE is performed during the search.

Another feature that DE introduces to the proposed hybrid GGP concerns the grammar. All the constants generated by the best individual via DE Mutation are inserted into the grammar as terminals. Consequently, the evolutionary process can take advantage of these values in the remaining generations (without DE). A pseudo-code of the proposed approch is described in Algorithm 3.

The parameter *de_range* in Algorithm 3 indicates how many individuals of the *Population* will be improved by DE. Here, the options considered for *de_range* are: (i) *de_range* = 0 (the standard GGP), (ii) *de_range* = 1 (the

Algorithm 3 Pseudo-algorithm for Grammar-based Genetic Programming

```
1: procedure GGPDE
2:     Initialize Population of size p_size
3:     Initialize P_tmp empty
4:     Evaluate Population with DE optimization
5:     while evaluations to go do
6:         while |P_tmp| < p_size do
7:             Select p_1 and p_2 by tournament
8:             Copy p_1 to p'_1 and p_2 to p'_2
9:             if random < cross_factor then
10:                Apply crossover between p'_1 and p'_2
11:            if random < mut_factor then
12:                Apply mutation on p'_1 and p'_2
13:            Evaluate p'_1 and p'_2
14:            Insert p'_1 and p'_2 in P_tmp
15:        if Execute DE on this generation then
16:            for each individual in de_range best individuals on the Population do
17:                DE(individual)    //Algorithm 2
18:        Population ← elite from Population ∪ elite from P_tmp
19:        Empty P_tmp
```

hybrid technique with DE improving the best individual of GGP), and (iii) $de\_range = elite$ (similar to the previous case, but DE is applied to the *elite* best individuals of the population).

5 Computational Experiments

Some experiments were designed to compare the performance of classic GGP and the proposed hybrid. The goal is to show that on problems with real coefficients the hybrid algorithm performs better than GGP.

Some basic parameters required by GGP are the maximum tree depth, the population size, the crossover and mutation rates, the elite size, the grammar used, the selection used and the stop criterion. We used 8 as the maximum tree depth. The population size used was 500 individuals with an elite of 5%. For selection we used tournament with 2 randomly sampled individuals. Crossover and mutation occur in 90% of the cases.

The input data used is split into three groups. The first one is for training, which is used to evolve the population. The second group is named validation data, which is used to select the best individual. The last group is the test set which is used to evaluate the selected individual.

The termination criteria adopted was the maximum number of objective function evaluations. An evaluation occurs every time the program has to assign a fitness value to an individual. Therefore, the algorithm uses the generated function and applies it to the training set, calculating the root mean square

error of that individual. Here, the objective function is evaluated in the evaluation of both meta-heuristics, namely, GGP and DE. The maximum number of evaluations allowed was 100000.

The population is initialized based on a grammar developed for generating a feasible individual. Two methods to initialize the GGP population were implemented, each one having 50% of chance of being chosen. In the first one, a tree is built with a depth being an integer randomly chosen between one and the maximum depth defined by the user. The second option is a random approach, where any rule is allowed to be chosen from the grammar as long as the minimum tree depth plus the current level does not exceed the maximum tree depth.

The DE parameters are population size, and crossover and mutation rates. The values adopted here are 100, 90% and 80% respectively. The DE process ends when it performs 100 objective function evaluations, which here correspond to 10 generations.

Nicolau et al. [15] provided some guidelines for defining benchmark problems. Many functions were presented and discussed in their work; some of them are used here. Also, some new functions were generated replacing the integer values by real ones. It was done randomly including two randomly generated decimal places for each coefficient. In addition, eight polynomial functions were created. Four of them have only integer coefficient values while these integer coefficients are replaced by real values in the other four functions. To do that, 0.5 is added/subtracted to all integer coefficients. All functions from Nicolau et al. and the ones created here are in Table 1. As the proposed approaches are designed to efficiently found real-valued coefficients, the problems set also includes functions with this feature.

The adopted grammar is composed by the following rules:

$$
\begin{aligned}
< expr > &::= < expr >< expr >< op > \mid < var > \mid < expr >< uop > \mid < const > \\
< op > &::= + \mid - \mid * \mid / \mid pow \\
< uop > &::= exp \\
< var > &::= x_1 \mid x_2 \\
< const > &::= -5 \mid -4 \mid -3 \mid -2 \mid -1 \mid 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5
\end{aligned}
$$

Preliminary experiments were performed using DE in all generations and for every element of GGP. However, no improvement in the final result was observed. Thus the parameter Ω was defined, resulting in less objective function evaluations in the DE in the evolution process of GGP; Here $\Omega = 10$; so the algorithm will call DE 10 times in equally spaced intervals during the entire process. Also, the best individual generated by the DE replaces the original individual from GP only when an improvement is observed.

Two variants of the proposed approach are analyzed here: (i) DE is applied to the best individual of the population (GGPDE best), and (ii) DE is performed using the best individuals of GGP (GGPDE elite). Both methods are compared to a baseline GGP, where the numerical coefficients are created only by the elements in the formal grammar.

Table 1: Definition of the functions used in the computational experiments

	Nicolau et al. functions [15] and the created ones
F1	$f(x) = x^4 + 2x^3 - 13x^2 - 14x + 24$
F2	$f(x) = 2x^4 - 9x^3 - 4x^2 + 1$
F3	$f(x) = x^4 + 9x^3 - 3x^2 - x + 1$
F4	$f(x) = x^4 - x^2 + 25x + 2$
F5	$f(x_1, x_2) = 6sin(x_1)cos(x_2)$
F6	$f(x_1, x_2) = (x_1 - 3)(x_2 - 3) + 2sin((x_1 - 4)(x_2 - 4))$
F7	$f(x_1, x_2) = \frac{(x_1-3)^4+(x_2-3)^3-(x_2-3)}{(x_2-2)^4+10}$
F8	$f(x_1, x_2) = \frac{1}{1+{x_1}^{-4}} + \frac{1}{1+{x_2}^{-4}}$
F9	$f(x_1, x_2) = {x_1}^4 - {x_1}^3 + {x_2}^2/2 - x_2$
F10	$f(x_1, x_2) = \frac{8}{2+{x_1}^2+{x_2}^2}$
F11	$f(x_1, x_2) = {x_1}^3/5 + {x_2}^3/2 - x_2 - x_1$
F12	$f(x_1, x_2) = \frac{e^{-(x_1-1)^2}}{1.2+(x_2-2.5)^2}$
F13	$f(x_1, x_2) = e^{-x_1}{x_1}^3 cos(x_1)sin(x_1)(cos(x_1)sin^2 x_1 - 1)(x_2 - 5)$

	Functions adapted to real valued coefficients
F1r	$f(x) = 0.5x^4 + 1.5x^3 - 13.5x^2 - 13.5x + 24.5$
F2r	$f(x) = 2.5x^4 - 9.5x^3 - 3.5x^2 + 1.5$
F3r	$f(x) = 1.5x^4 + 9.5x^3 - 2.5x^2 - 0.5x + 1.5$
F4r	$f(x) = 1.5x^4 + 0.5x^3 - 0.5x^2 + 25.5x + 1.5$
F5r	$f(x_1, x_2) = \frac{(x_1-3.96)^4+(x_2-3.41)^3-(x_2-3.11)}{(x_2-2.94)^4+10.57}$
F6r	$f(x_1, x_2) = \frac{1.91}{1.93+{x_1}^{-4}} + \frac{1.5}{1.64+{x_2}^{-4}}$
F7r	$f(x_1, x_2) = {x_1}^4 - {x_1}^3 + {x_2}^2/2.71 - x_2$
F8r	$f(x_1, x_2) = \frac{8.47}{2.59+{x_1}^2+{x_2}^2}$
F9r	$f(x_1, x_2) = {x_1}^3/5.92 + {x_2}^3/2.98 - x_2 - x_1$
F10r	$f(x_1, x_2) = 6.7sin(x_1)cos(x_2)$
F11r	$f(x_1, x_2) = (x_1 - 3.91)(x_2 - 3.33) + 2.26sin((x_1 - 4.74)(x_2 - 4.53))$

Thirty independent runs were performed for each technique in each function. The results obtained in the functions from [15] and the generated ones with integer coefficients can be seen in Table 2. Table 3 presents the results found for the modified functions. These tables present the following values for the errors of the models found: minimum, median, mean, standard deviation, and maximum. Also, the best results are highlighted in boldface.

As F12 already has real-valued coefficients, no modification was made. F13 has trigonometric functions and these functions are not in the grammar used here. Thus, F13 is not adapted either.

"Inf" values can be seen in some lines of both tables. In F13, the worst model obtained by GGP contains indefiniteness (like division by zero). The standard deviations calculated for the solutions found by the variant GGPDE best in functions F13 and F5r are very large.

Table 4 shows the number of times in which the best results are obtained by each method considered here. For the minimum value, GGPDE best overcomes

Table 2: Results for the models from [15] and those with integer coefficients.

F	Approach	Min	Median	Mean	Std	Max
F1	GGP	**1.27267e-20**	3.20295e+02	3.83958e+02	3.38433e+02	2.06861e+03
	GGPDE best	1.24314e+02	3.19164e+02	**3.23125e+02**	**9.66903e+01**	**5.25894e+02**
	GGPDE elite	9.39024e+01	**2.84030e+02**	3.91188e+05	2.10417e+06	1.17225e+07
F2	GGP	**1.18479e-18**	**3.69810e-01**	**1.41985e+02**	3.98085e+02	1.68803e+03
	GGPDE best	**1.18489e-18**	1.02453e+00	1.87379e+02	4.55823e+02	2.31287e+03
	GGPDE elite	**1.18489e-18**	1.00000e+00	1.48102e+02	**3.11546e+02**	**1.26865e+03**
F3	GGP	3.41348e-19	3.22930e+01	4.05211e+02	6.64576e+02	3.21866e+03
	GGPDE best	**3.41343e-19**	**2.02782e+00**	2.04296e+02	3.27345e+02	1.09942e+03
	GGPDE elite	3.41353e-19	1.89276e+01	**1.25639e+02**	**1.91379e+02**	**8.71560e+02**
F4	GGP	2.67329e-20	**2.67330e-20**	**1.25102e+01**	**5.56011e+01**	3.05324e+02
	GGPDE best	2.67327e-20	**2.67330e-20**	1.87492e-01	9.44870e-01	**5.26439e+00**
	GGPDE elite	**2.67325e-20**	**2.67330e-20**	1.47353e+01	7.84320e+01	4.37096e+02
F5	GGP	7.15415e+00	**8.23153e+00**	**2.43162e+01**	**8.16072e+01**	**4.62883e+02**
	GGPDE best	6.85063e+00	8.29127e+00	1.15226e+08	6.20513e+08	3.45679e+09
	GGPDE elite	**6.46682e+00**	8.45763e+00	3.31625e+28	1.78586e+29	9.94876e+29
F6	GGP	1.83507e+00	4.25916e+00	9.97359e+00	1.07128e+01	4.32296e+01
	GGPDE best	1.88241e+00	**1.93989e+00**	**8.03398e+00**	**8.29932e+00**	**2.77297e+01**
	GGPDE elite	**1.80789e+00**	4.27954e+00	9.19945e+00	9.40705e+00	3.42718e+01
F7	GGP	9.48119e+01	4.09538e+02	**3.98684e+02**	2.07941e+02	1.02015e+03
	GGPDE best	**5.56259e+01**	**3.73582e+02**	7.24423e+132	3.90114e+133	2.17327e+134
	GGPDE elite	9.81467e+01	4.04810e+02	4.24675e+02	**1.94837e+02**	**9.01534e+02**
F8	GGP	5.17117e-03	6.18720e-02	8.34767e-02	5.51492e-02	2.18350e-01
	GGPDE best	4.54639e-03	**6.12495e-02**	**7.54036e-02**	**4.68644e-02**	**1.61780e-01**
	GGPDE elite	**2.57640e-03**	9.20400e-02	9.63285e-02	5.57474e-02	1.98522e-01
F9	GGP	**2.91398e-20**	6.76676e+01	8.15942e+01	7.44445e+01	2.52412e+02
	GGPDE best	5.41265e-01	**3.07666e+01**	**6.67409e+01**	**6.25851e+01**	**1.81362e+02**
	GGPDE elite	1.05793e+01	1.27820e+02	1.24008e+02	9.92508e+01	4.27681e+02
F10	GGP	3.96578e-02	1.84812e-01	**1.84424e-01**	**9.35003e-02**	**3.49632e-01**
	GGPDE best	**3.94160e-02**	**1.15905e-01**	1.14604e+10	6.17160e+10	3.43811e+11
	GGPDE elite	4.34469e-02	1.32551e-01	3.67080e+07	1.97679e+08	1.10124e+09
F11	GGP	2.64535e+00	**1.92827e+01**	**2.31053e+01**	**1.46673e+01**	**5.52548e+01**
	GGPDE best	**2.08399e+00**	2.07451e+01	2.43688e+01	1.54383e+01	7.97426e+01
	GGPDE elite	1.24761e+01	2.64740e+01	3.04023e+01	1.77095e+01	7.05079e+01
F12	GGP	1.15570e-03	**2.80619e-03**	1.61054e+29	8.67302e+29	4.83162e+30
	GGPDE best	**5.90153e-04**	3.02171e-03	**3.58328e-03**	2.32394e-03	1.28618e-02
	GGPDE elite	1.23993e-03	4.13272e-03	4.26648e-03	**2.18519e-03**	**1.02590e-02**
F13	GGP	3.21105e+140	1.43894e+145	inf	inf	inf
	GGPDE best	8.53042e+139	3.29360e+145	7.07463e+210	inf	2.12239e+212
	GGPDE elite	**8.47539e+139**	**5.86418e+144**	**9.91778e+145**	**1.95611e+146**	**9.20379e+146**

the other approaches. The same minimum value was obtained by all approaches in F2, so it is not considered in this analysis.

Analyzing the median values, the results obtained shows GGP with better values in seven of all 23 problems. The same median value was found by all approaches in F4 and it is not considered in this analysis. Comparing to the other approaches, GGPDE best achieved most of the best medians: 11 of the 23 problems.

Looking for the mean value, GGP and GGPDE best had better results. When comparing only GGP with GGPDE best, the second one goes to 15 best values while standard GGP stays with 9.

For both standard deviation and the maximum values, GGPDE variants obtained values better than those found by GGP.

Table 3: Results obtained for the models with real-valued coefficients.

F	Approach	Min	Median	Mean	Std	Max
F1r	GGP	5.48189e+01	1.76317e+02	2.38351e+130	1.28356e+131	7.15053e+131
	GGPDE best	**8.51386e+00**	**1.23058e+02**	1.50917e+02	1.01282e+02	4.14614e+02
	GGPDE elite	2.49015e+01	1.35062e+02	**1.38455e+02**	**5.61589e+01**	**2.33040e+02**
F2r	GGP	**2.50000e-01**	1.27940e+02	5.39328e+02	7.85030e+02	2.91256e+03
	GGPDE best	7.98032e-01	9.79430e+01	3.85574e+02	7.04658e+02	2.90090e+03
	GGPDE elite	2.59192e-01	**1.22137e+01**	**2.26970e+02**	**5.07131e+02**	**2.12380e+03**
F3r	GGP	3.74884e+01	5.10014e+02	1.33653e+03	2.39767e+03	1.24998e+04
	GGPDE best	**5.81255e-01**	4.41056e+02	8.12076e+02	1.21800e+03	5.71448e+03
	GGPDE elite	8.34346e+00	**4.18485e+02**	**4.83822e+02**	**3.86485e+02**	**1.36211e+03**
F4r	GGP	1.24521e+01	**1.32684e+02**	**1.74190e+02**	**1.31735e+02**	**5.17977e+02**
	GGPDE best	**2.50000e-01**	1.42351e+02	1.84998e+02	1.75329e+02	8.49106e+02
	GGPDE elite	4.24787e+00	2.00104e+02	9.78321e+02	2.70527e+03	1.22537e+04
F5r	GGP	7.69905e+00	**1.03494e+01**	**1.02325e+01**	**9.72613e-01**	**1.15175e+01**
	GGPDE best	**7.49249e+00**	1.04060e+01	6.18707e+178	inf	1.85612e+180
	GGPDE elite	8.71417e+00	1.06273e+01	1.35408e+03	7.23534e+03	4.03176e+04
F6r	GGP	2.54229e+00	**1.26288e+01**	1.99437e+01	1.68676e+01	5.73162e+01
	GGPDE best	**2.48153e+00**	1.62218e+01	**1.88118e+01**	**1.67733e+01**	**5.67298e+01**
	GGPDE elite	2.50581e+00	1.88640e+01	2.35538e+01	1.99124e+01	7.69267e+01
F7r	GGP	1.96530e+02	7.14748e+02	8.50404e+02	5.03237e+02	**2.46552e+03**
	GGPDE best	**1.74813e+02**	**7.11527e+02**	**7.67132e+02**	**4.85504e+02**	2.54786e+03
	GGPDE elite	2.50323e+02	1.02802e+03	1.03900e+03	5.45372e+02	3.07262e+03
F8r	GGP	1.63017e-02	8.62828e-02	8.27299e-02	3.87543e-02	1.75797e-01
	GGPDE best	**6.21844e-03**	**6.91660e-02**	**6.70838e-02**	**3.80853e-02**	**1.52002e-01**
	GGPDE elite	1.15992e-02	9.12135e-02	8.32915e-02	3.95185e-02	1.65612e-01
F9r	GGP	**1.00405e-01**	1.32341e+02	1.49218e+02	2.45887e+02	1.42234e+03
	GGPDE best	6.63786e-01	**8.77002e+01**	1.45180e+02	3.37647e+02	1.93108e+03
	GGPDE elite	2.84583e+00	1.41733e+02	**1.16240e+02**	**7.19148e+01**	**2.48074e+02**
F10r	GGP	1.62983e-02	1.20113e-01	**1.31228e-01**	9.36401e-02	4.21901e-01
	GGPDE best	**1.16336e-02**	1.29129e-01	2.41550e+07	1.30079e+08	7.24650e+08
	GGPDE elite	2.34820e-03	**1.18365e-01**	1.35053e-01	**8.51397e-02**	**2.62876e-01**
F11r	GGP	**3.58123e+00**	1.46003e+01	1.55218e+01	6.35614e+00	3.31064e+01
	GGPDE best	5.52777e+00	**1.22693e+01**	**1.37339e+01**	**5.88497e+00**	**2.75655e+01**
	GGPDE elite	6.61749e+00	1.32096e+01	1.57459e+01	7.43996e+00	3.55630e+01

When all cases are considered (total), GGPDE best can be considered the best performing method between those considered here. Also, one can highlight the results obtained GGPDE variants when the functions contains real-valued coefficients. GGPDE best obtained the best results in 8 of the 11 problems, the best medians in 5, and the best mean in 4.

Finally, Table 5 presents the mean run time of the techniques for each function. These values indicate that the processing time required by the baseline GGP is similar to those observed in both GGPDE approaches. Also, one can notice that GGPDE elite is faster than the other methods considered here.

6 Concluding Remarks and Future Works

The hybridization of GGP with DE is proposed here, where DE is used to improve the numerical coefficients of the models obtained by GGP. Also, the grammar is adapted during the search when the proposed method is used.

Table 4: Total of best results for each category

Approach	Min	Median	Mean	Std	Max	Total
GGP	5	7	9	6	6	33
GGPDE best	13	11	9	8	8	49
GGPDE elite	5	5	6	10	10	36

Results show that using DE into GGP is useful. The proposed technique obtained the best results in most of the problems when DE is applied to the best individual of GGP. It indicates that the fine tuning of the coefficient values performed by the proposed hybrid approach improves the original search technique. In special, the best results were found by the variant GGPDE best in 13 of the 24 functions used in the computational experiments. When the functions contains real-valued coefficients, the best results are achieved in 8 of the 11 situations.

Other applications can be considered in future works. Also, it is desired perform a sensibility analysis of the parameter values; for instance, with respect to the frequency of DE execution and the number of fitness evaluations.

Acknowledgements

The authors thank the financial support provided by CAPES, CNPq (grant 310778/2013-1), FAPEMIG (grant APQ-03414-15), and PPGCC/UFJF.

References

1. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press (1992)
2. Hopcroft, J.E.; Motwani, R.U.J.: Automata Theory, Languages, and Computation, 3rd ed. Pearson (2014)
3. McKay, R.I., Hoai, N.X., Whigham, P.A., Shan, Y., O'Neill, M.: Grammar-based genetic programming: a survey. Genetic Programming and Evolvable Machines **11**(3-4) (2010) 365–396
4. Evett, M., Fern, T.: Numeric mutation: Improved search in genetic programming. In: Proc. of the 11th Intl. Florida Artificial Intelligence Research Society Conference. (1998)
5. Storn, R., Price, K.: Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces. international computer science institute, berkeley. Technical report, CA, 1995, Tech. Rep. TR-95–012 (1995)
6. El-Mihoub, T.A., Hopgood, A.A., Nolle, L., Battersby, A.: Hybrid genetic algorithms: A review. Engineering Letters **13** (2006) 124–137
7. Shun, C.W., Teo, J.: Hybridized gp and self-adaptive de for morphology-controller co-evolution of heterogeneous modular robots. In: 2014 International Conference on Computational Science and Technology (ICCST). (Aug 2014) 1–6
8. Roy, P., Islam, M.J., Islam, M.M.: Self-adaptive genetically programmed differential evolution. In: 7th Intl. Conf. on Electrical and Computer Engineering. (Dec 2012) 639–642

Table 5: Means of the run times (in seconds) for each function.

F	GGP	GGPDE best	GGPDE elite
F1	3.06758e+01	3.03063e+01	**2.83879e+01**
F2	**2.93674e+01**	3.10546e+01	2.97596e+01
F3	3.01554e+01	**2.77856e+01**	2.79418e+01
F4	2.66668e+01	2.73552e+01	**2.64481e+01**
F5	2.95411e+01	2.88897e+01	**2.81170e+01**
F6	2.26458e+01	**2.20589e+01**	2.26317e+01
F7	2.34498e+01	2.31985e+01	**2.31423e+01**
F8	2.55344e+01	2.50382e+01	**2.44331e+01**
F9	2.26389e+01	2.28062e+01	**2.02327e+01**
F10	2.53310e+01	2.59047e+01	**2.35279e+01**
F11	2.23458e+01	2.26310e+01	**2.01952e+01**
F12	2.65347e+01	2.48998e+01	**2.35781e+01**
F13	2.56674e+01	2.52329e+01	**2.39746e+01**
F1r	2.85950e+01	2.95963e+01	**2.73044e+01**
F2r	3.47155e+01	3.39089e+01	**3.15222e+01**
F3r	3.22507e+01	3.22618e+01	**3.07822e+01**
F4r	3.12391e+01	3.05945e+01	**2.88004e+01**
F5r	2.86495e+01	2.90972e+01	**2.75503e+01**
F6r	2.35143e+01	2.31774e+01	**2.26686e+01**
F7r	2.41495e+01	2.36710e+01	**2.24067e+01**
F8r	2.48463e+01	2.56763e+01	**2.30100e+01**
F9r	2.26670e+01	2.14072e+01	**1.94738e+01**
F10r	2.50913e+01	2.45826e+01	**2.41729e+01**
F11r	2.30184e+01	2.02559e+01	**1.98415e+01**

9. Howard, L.M., D'Angelo, D.J.: The GA-P: a genetic algorithm and genetic programming hybrid. IEEE Expert **10** (Jun 1995) 11–15
10. Rayno, J., Iskander, M.F., Kobayashi, M.H.: Hybrid genetic programming with accelerating genetic algorithm optimizer for 3-d metamaterial design. IEEE Antennas and Wireless Propagation Letters **15** (2016) 1743–1746
11. Pappa, G.L., Talbot, H., Menotti, D., Meignan, M.: Towards automated lymphoma prognosis based on pet images. In: 2008 IEEE Workshop on Machine Learning for Signal Processing. (Oct 2008) 279–284
12. Devaney, J.E., Hagedorn, J.G.: The role of genetic programming in describing the microscopic structure of hydrating plaster. In: Late Breaking papers at the Genetic and Evolutionary Computation Conference (GECCO-2002). (Jul 2002) 9–13
13. Meier, A., Gonter, M., Kruse, R.: Accelerating convergence in cartesian genetic programming by using a new genetic operator. In: Proc. of the 15th Annual Conf. on Genetic and Evolutionary Computation, ACM (2013) 981–988
14. Koza, J.R.: Darwinian invention and problem solving by means of genetic programming. In: Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings. 1999 IEEE Intl. Conf. on. Volume 3. (1999) 604–609 vol.3
15. Nicolau, M., Agapitos, A., O'Neill, M., Brabazon, A.: Guidelines for defining benchmark problems in genetic programming. In: 2015 IEEE Congress on Evolutionary Computation (CEC). (May 2015) 1152–1159