Tree-based Grammar Genetic Programming to Evolve Particle Swarm Algorithms

Péricles B. C. Miranda and Ricardo B. C. Prudêncio
Universidade Federal de Pernambuco
Pernambuco, Recife

***Abstract*—Particle Swarm Optimization (PSO) is largely used to solve optimization problems effectively. Nonetheless, the PSO performance depends on the fine tuning of different parameters. To make the algorithm design process more independent from human intervention, some researchers have treated this task as an optimization problem. Grammar-guided Genetic Programming algorithms (GGGP), in special, have been widely studied and applied in the context of algorithm optimization. GGGP algorithms produce customized designs based on a set of production rules defined in the grammar; differently from methods that simply select designs in a pre-defined limited search space. In this work, we proposed a tree-based GGGP technique for the generation of PSO algorithms. This paper intends to investigate whether this approach can improve the production of PSO algorithms when compared to other GGGP techniques already used to solve the current problem. In the experiments, a comparison between the tree-based and the commonly used linearized GGGP approach for the generation of PSO algorithms was performed. The results showed that the tree-based GGGP produced better algorithms than the counterparts. We also compared the algorithms generated by the tree-based technique to state-of-art optimization algorithms, and the results showed that the algorithms produced by the tree-based GGGP achieved competitive results.**

***Index Terms*—Particle Swarm Optimization; Genetic Programming; Algorithm Generation;**

I. Introduction

Particle Swarm Optimization (PSO) has become a widely used and studied algorithm due to its flexibility and competitive results in different applications [1]. A variety number of studies were conducted aiming to improve the performance of the standard PSO algorithm [1]. Several of these and other developments have shown to be useful and improved the PSO performance significantly. Nonetheless, the PSO strongly depends on the fine tuning of its parameters to perform well [1]. This paper addresses the problem of designing PSO algorithms that is to determine a suitable configuration of parameters and components (i.e., a *design*) for the PSO algorithm when applied to a given optimization problem. Among the techniques already used for the current problem [2], [3], [4], [5], we emphasize the Grammar-guided Genetic Programming algorithm (GGGP) [6]. GGGP produces programs/algorithms based on production rules defined in a grammar. Differently from simply selecting designs from a pre-defined set of possible solutions, GGGP algorithms generate new and customized algorithm designs [7]. Different types of GGGP algorithms were proposed, such as Context-free Grammar Genetic Programming (CFG-GP) [6], Grammatical Evolution (GE) [7] and Logic grammar-based genetic programming (LOGENPRO) [8]. According to [6], these algorithms can be categorized by the way the individuals (solutions) are represented. Derivation tree and linear genome are considered the main forms to represent solutions [6]. Although the linearized GGGP approaches have become more popular than the tree-based ones for the algorithm design problem, some issues related to locality and generation of invalid individuals arose due to the linear genome representation [9]. These problems are critical and introduce a source of inconsistency into the search space, and become an obstacle for evolution and search [9]. In the context of the PSO algorithm design problem, we highlight that only linearized GGGP approaches, more specifically GE, have been adopted for the current problem [3], [4], [5]. Although the results presented in these works showed to be promising, the previously mentioned problems (e.g. locality and invalid individuals) interfered negatively the search. Thereby, we believe that an adequate individual representation would improve the construction of PSO algorithms.

In this work, we adopted a GGGP approach, CFG-GP, which uses a tree-based representation to define the individuals. The main reason we have adopted this representation is that it avoids the locality problem and the generation of invalid individuals. Nonetheless, as no tree-based GGGP technique was applied to the PSO algorithm design yet, our main goal is to investigate its performance on producing designs considering various optimization problems. In the experiments, we compared the designs generated by the CFG-GP with the designs produced by a GE approach, which uses a linear representation. The results showed that the designs generated by the CFG-GP were better than the counterpart. We also compared the algorithms generated by CFG-GP to state-of-art optimization algorithms. The results showed that the algorithms produced by the CFG-GP achieved competitive results. This work is organized as follows. Section II presents the basic concepts of GGGP approaches, focusing on the linear genome and tree-based representation. Section III contains related work to optimization of PSO algorithms. Section IV brings the experimental design. In Section V the results are presented and discussed. Finally, in Section VI, the conclusions and possible future works are presented.

II. Background

GP has used the potential of grammars almost from its creation. Since then, grammars have contributed to developments in the GP area. Initially, GGGP approaches represented a small

portion of the GP area. However, this scenario changed, and the number of works involving GGGP has increased. The most well known proposals are the tree-based and linearized GGGP algorithms [6]. Before going into detail about these GGGP algorithms, the concepts of genotype and phenotype need to be clarified. The term genotype is related to the data structure manipulated by genetic operators such as mutation and crossover. On the other hand, the phenotype represents an executable structure (program or algorithm) which represents the behavior of an individual. Next, both tree-based and linearized approaches are detailed.

In a tree-based GGGP approach, the genotypes are represented by a derivation tree produced from a grammar. This grammar describes a set of rules and expressions (language) as well as those used in standard GP [6]. The process of fitness evaluation works by reading the expression tree (phenotype) of each individual from the leaves of the grammar derivation tree. After that, it is executed and evaluated as in standard GP. Tree-based approaches as well as GP, use genetic operators such as selection mechanism, mutation, reproduction, and crossover. Although the selection and reproduction are the same used in standard GP, crossover and mutation operators are different. Regarding the mutation, an internal node is chosen randomly. After that, the subtree originated at that node is removed, and then, changed by a new one. It is worth mentioning that the novel subtree was randomly produced according to the grammar, beginning from the same non-terminal. In the case of crossover, two internal nodes of the derivation tree are selected at random, and their respective subtrees under them are swapped. It is important to mention that this crossover mechanism requires that both chosen nodes must be labeled with the same non-terminal symbol described in the grammar. The crossover operator in the tree-based GGGP is similar to standard crossover in GP, however an additional constraint defines that the crossover points have the same grammar label. We highlight that other variations of the same operators for tree-based techniques can be found in [6].

In the linearized GGGP algorithms, the genotype adopts a linear sequence structure (vector of genes) to be manipulated by genetic operators. Differently from the tree-based approach, the linearized algorithm does not convert the genotype to phenotype directly. There is an intermediate state which is built as a derivation tree according to the grammar, and then, translated to an expression tree (phenotype). The first conversion, from genotype to an intermediate state, is performed from the left of the vector to the right, and the intermediate state is generated respectively. The linear genome as genotype representation offers some advantages when compared to the tree-based one. The most obvious benefit is that this representation allows the use of theory and practice from fields of evolutionary computation which also use linear representations such as GA and Evolution Strategies (ES). Thus, commonly used genetic operators are available to operate genotype in the linear form. Because of that, linearized GGGP approaches have called the attention of the area, and it has been widely studied over the years. Along the years, different linearized approaches were proposed as presented in [6]. Nonetheless, Grammatical Evolution (GE) has become the most broadly used extension of the GGGP system with a linear genome representation described earlier in [7]. As well as in other linearized GGGP approaches, there are two critical issues associated with the linear representation adopted by the GE [9]. First, a supposedly valid genotype may result in a phenotype whose fitness can not be evaluated. This problem can be simply solved by assigning poor fitness values to these individuals, but this solution inserts inconsistent individuals into the search. It becomes an impediment for the linearized GGGP algorithms evolution whether the number of infeasible phenotypes is large. Different extensions have been developed to make linearized GGGP algorithms avoid the mentioned problem. Some of them are focused on correction of individuals and dynamic grammars. However, the extensions are considered too complex [9].

The second issue of the linear representation is related to the locality principle. This principle says that small modifications in genotype should result in small changes in phenotype [9]. To define operators for linearized GGGP algorithms which perform small modifications on the genotype is simple. However, to guarantee that the alterations in the phenotype will not be larger is challenging. This issue is critical because a simple modification at one position of the genotype may modify the expressiveness (coding or non-coding), or meaning of genes placed after that position. As a consequence, the phenotype may suffer too many modifications and produce an entirely different phenotype. In an extremely situation, the individual ceases to be feasible and becomes infeasible, or the search process works similarly to a random sampling [9]. Recognizing the problems related to the linear representation has started the development of a diverse number of works to understand and enhance the way linearized GGGP algorithms operate. Nonetheless, most of the solutions recurred to the use of operators from tree-based algorithms. It is worth mentioning that the two issues presented early can not happen with tree-based GGGP approaches [9]. The fact that the linearized GGGP has been adopted unanimously over the last decade does not mean that it is the most appropriate algorithm among all GGGP algorithms already proposed.

III. PSO ALGORITHM DESIGN PROBLEM

PSO is a successful optimization algorithm and it follows a set of simple steps [1]. Initially, a predetermined number of particles have their position and velocity randomly initialized. At each iteration of the algorithm, each particle moves through the search space by updating its own position and velocity until a stop criterion is reached. The particles' velocity is a combination between the best position found by itself $\vec{p}(t)$ ($pbest$) and by the best position found by its neighborhood $\vec{g}(t)$ ($gbest$) [1]. In the standard PSO, a new velocity is calculated by the following equation:

$$\vec{v_i}(t+1) = \vec{v_i}(t) + c_1 r_1(\vec{p_i}(t) - \vec{x_i}(t)) + c_2 r_2(\vec{g_i}(t) - \vec{x_i}(t)), \quad (1)$$

where $i = 1, ..., N$, N is the number of particles; c_1 and c_2 are the cognitive and social acceleration coefficients; r_1 and r_2 are two random numbers between [0,1]. After updating the particle's velocity, the following equation is used to update the particle's position:

$$\vec{x}_i(t+1) = \vec{x}_i + \vec{v}_i(t+1). \quad (2)$$

The flexibility and fast convergence of PSO in different applications called the attention of researchers. Among the various improvements to standard PSO, we highlight the development of new topologies, new velocity equations (e.g., by adopting the inertia weight or the constriction factor) and by adopting mutation operators [1], [10]. As a result of the great diversity of PSO components and hyper-parameters, to define the most suitable design for the PSO considering a given optimization problem becomes a hard task. Some early works handled the PSO algorithm design as another (meta)optimization problem to perform this process independently from human intervention and evaluate the candidate designs systematically [11], [2], [5]. In this case, each solution of the search space is a possible design for the PSO algorithm for a given problem. The objective function, in turn, is the same one used in the (base)optimization problem.

Initially, the optimization techniques adopted to tune the PSO algorithm were used simply to select parameters from a limited pool of candidates [11], [12], [2]. However, with the advent of GGGP approaches, some works [4], [5] adopted the GE approach, as said, a linearized GGGP approach to evolve/generate PSO designs. The results presented by the these works showed to be promising and that GE may be used to optimize PSO designs. Nonetheless, some problems (e.g. locality and invalid individuals) arose due to the linear genome representation adopted by the GE approach. These issues may introduce a source of inconsistency into the search space and consequently, harm the search process, as said in Section II.

The use of GGGP algorithms for the PSO algorithm design problem has not been investigated deeply. Few works adopted GGGP, and all of them used a linear genome representation to evolve PSO algorithms. Thus, we intend to evaluate a tree-based GGGP approach for the problem at hand. Our hypothesis is that the tree-based approach may improve the construction of PSO algorithms and avoid problems faced in previous works.

IV. Experimental design

In this section, we present the experiments performed to compare the tree-based and linearized GGGP approaches on the PSO algorithm design problem. To perform this comparison, we adopted two well-known algorithms: GE, using the linear genome representation, and the CFG-GP, using the tree-based representation. The GE implementation used here is that developed by [5]. This implementation, as previously said in section III, reached promising results in the generation of PSO algorithms. The CFG-GP implementation adopted here is based on the work developed in [13]. Complementary to the CFG-GP and GE, we also adopted a random search procedure in the experiments. This procedure was executed repeatedly producing solutions by using a method of random initialization of the GE. Our intention is to use the random search as a baseline measure against which the performance of the GE and CFG-GP on each optimization problem can be evaluated.

In this work, both GGGP approaches and the random search evolved the PSO design (components and parameters) for a given optimization problem as input. The PSO algorithm with the parameters and procedures to be optimized is presented in Algorithm 1. The elements which are in the format of tag <> represent the parameters or procedures that can be replaced by values defined in the grammar.

Algorithm 1: PSO Algorithm

```
Require: size: size of swarm, nIt: number of iterations
  swarm = Random initialization of size particles
  Evaluate all p in swarm
  gbest = Choose best according to <GBEST>
  for i = 0 to maxIt do
    for all p in swarm do
      <UPDATE-VELOCITY> of p
      Update position of p
      if (rand(0-1) < <PROB-MUTATION>) then
        Mutate p according to <MUTATION>
      end if
      Evaluate p
      Update pbest
    end for
    Update gbest according to <GBEST>
  end for
```

Here, we used the grammar proposed by Miranda and Prudêncio [5] for the PSO algorithm generation. This grammar considers different components and parameters for PSO (see Figure 1), and it defines the possible options for a range of PSO designs. It is worth mentioning that the available values specified in the grammar are commonly used in the literature due to their importance in the generation of good solutions [7]. Next, all the PSO aspects considered by the grammar are presented:

Acceleration constants: the constants <C1> and <C2> denote the weight of the acceleration components which lead the particles' position toward to their best social or cognitive positions. The adjustment of these constants changes the balance of the search process. Lower values for <C1> and <C2> make the particles explore more the search space, whereas higher values make the particles focus on a target region [1]. The choice of these values is dependent on the problem. Nonetheless, these values are usually chosen between 0 and 3 [1].

Velocity update: we adopted two options of velocity equations. First, using the inertia weight with different possible values. Second, the constriction factor (χ) using a fixed value

equals to 0.7. These equations are the two most known velocity updating equations for PSO [1]. Other parameters compose the velocity equations such as the acceleration constants <C1>, <C2> and the inertia weight <INERTIA>;

Topology: the tag <GBEST> denotes the topology procedure that will be used to provide the neighborhood's positions for the velocity equation. As the topology defines how the particles interact with each other during the search, it performs a significant role in the optimization process. In this work, we chose seven well-known topologies in literature [1]. The selected topologies are heterogeneous, in other words, they present different strategies of exploitation and exploration aiming to improve the PSO performance;

Mutation: a strategy to improve diversity in a swarm is to include in the PSO a mutation operator, as regularly adopted in evolutionary algorithms. The tag <MUTATION> represents possible mutation operators for the PSO. This tag can assume five mutation operators suggested by [10]. This tag also considers the possibility of using no mutation operator (λ). The tag <PROB-MUTATION> defines the adopted values of mutation probability.

$\langle \text{C1} \rangle \models 1 \mid 1.25 \mid 1.5 \mid 1.75 \ldots \mid 3$

$\langle \text{C2} \rangle \models 1 \mid 1.25 \mid 1.5 \mid 1.75 \ldots \mid 3$

$\langle \text{INERTIA} \rangle \models 0.1 \mid 0.2 \mid \ldots \mid 1.0$

$\langle \text{PROB-MUTATION} \rangle \models 0.1 \mid 0.2 \mid \ldots \mid 1.0$

$\langle \text{GBEST} \rangle \models \text{Global}$
$\mid \text{Local}$
$\mid \text{Focal}$
$\mid \text{Von Neumann}$
$\mid \text{Hierarchical}$
$\mid \text{Four Clusters}$
$\mid \text{Clan}$

$\langle \text{UPDATE-VELOCITY} \rangle \models$
$(\langle \text{INERTIA} \rangle * v) + \langle \text{C1} \rangle * r_1 * (\langle \text{GBEST} \rangle - p) + \langle \text{C2} \rangle * r_2 * (pBest - p)$
$\mid \chi * (v + \langle \text{C1} \rangle * r_1 * (\langle \text{GBEST} \rangle - p) + \langle \text{C2} \rangle * r_2 * (pBest - p))$

$\langle \text{MUTATION} \rangle \models Gaussian(\langle \text{PROB-MUTATION} \rangle)$
$\mid Cauchy(\langle \text{PROB-MUTATION} \rangle)$
$\mid Michalewicz(\langle \text{PROB-MUTATION} \rangle)$
$\mid Levy(\langle \text{PROB-MUTATION} \rangle)$
$\mid Random(\langle \text{PROB-MUTATION} \rangle)$
$\mid \lambda$

Fig. 1. Grammar for PSO algorithm optimization.

The experiments were divided into two phases. First, we compared the designs produced by the CFG-GP to the designs produced by the GE and the random search model considering a set of optimization problems. Second, we compared the algorithms produced by CFG-GP with algorithms adopted in the competition of the International Conference of Swarm Intelligence 2014 (ICSI 2014) [14], considering the same set of problems. Our goal is to verify whether the generated algorithms by the CFG-GP achieve competitive results in comparison to consolidated algorithms in the literature. The parameters adopted by the GGGP approaches were the same used in [9], except the crossover operator employed by the GE approach. We decided to choose the LHS crossover instead of a single point crossover, as suggested by [9], because the LHS was created for the linear genome representation and consequently performs better than previous proposed operators. All parameters are presented in Table I. Besides, it is noteworthy that the competition algorithms and the PSO designs generated by GE and CFG-GP used the same setting values: the stop criterion is $5,000$ iterations per simulation and all algorithms executed 20 times in order to generate the mean of its fitness values. The random search process was divided into 50 groups of 500 individuals for each run. This process is considered equal to sampling 50 initial swarms, each with 500 individuals, and keeping the best individual found during the run. These numbers were chosen to coincide with the values adopted in the experiments: a population size of 500 and a run of 50 generations.

TABLE I
THE PARAMETERS USED FOR ALL EXPERIMENTS

Parameter	Value	
	GE	CFG-GP
Population size	500	
Generations	50	
Selection method	Tournament	
Tournament size	3	
Elitism count	1	
Crossover rate	0.9	
Reproduction rate	0.1	
Mutation rate	0.5	-
Crossover	LHS crossover [15] (variable length)	Subtree
Mutation	Pointwise (per-codon)	Subtree
Crossover node selection	Uniform	Koza-Style
Initialization	Random	Ramped-Half & Half
Min. Initialization depth	-	2
Max. Initialization depth	-	6
Max. tree depth	-	17
Initial codon length	200	-
Wrapping	None	-

In this work, we adopted in the experiments 32 unconstrained continuous optimization problems as benchmarks from [16]. We highlight that all selected functions are scalable and adopted 30 dimensions. To evaluate the GGGP approaches in distinct aspects and levels of difficulty, we carefully chose problems that can be classified into four categories where each category is composed of eight different problems. The categories are named *Bowl-Shaped*, *Plate Shaped*, *Valley-Shaped* and *Many Local Minima*. Problems which belong to the *Bowl-Shaped* category presents a convex fitness landscape and only one global minimum solution. The eight problems in this category are: *Sphere* (f_1), *Sum of Different Powers* (f_2), *Sum Squares* (f_3), *Rotated Hyper-Ellipsoid* (f_4), *Axis parallel hyper-ellipsoid* (f_5), *Brown* (f_6), *Exponential* (f_7) and *Schwefel01* (f_8). The *Plate-Shaped* and *Valley-Shaped* categories are formed by problems where the global optimum is located in a uniform plain. This characteristic may become an obstacle to

the search process making the problems in the latest two categories harder than the *Bowl-Shaped* problems. The functions which belong to these categories are, respectively: *Zakharov* (f_9), *Bent Cigar* (f_{10}), *Elliptic* (f_{11}), *Discus* (f_{12}), *AMGM* (f_{13}), *Rotated High Conditioned Elliptic* (f_{14}), *Rotated Bent Cigar* (f_{15}) and *Rotated Discus* (f_{16}); and *Rosenbrock* (f_{17}), *Shifted Rosenbrock* (f_{18}), *Shifted and rotated Rosenbrock* (f_{19}), *Dixon-Price* (f_{20}), *Schwefel04* (f_{21}), *Rotated Dixon-Price* (f_{22}), *Shifted Dixon-Price* (f_{23}) and *Shifted and rotated Dixon-Price* (f_{24}). Finally, problems which present several local minima belongs to the *Many Local Minima* category. The functions in this category are *Ackley* (f_{25}), *Griewank* (f_{26}), *Rastrigin* (f_{27}), *Alpine* (f_{28}), *Salomon* (f_{29}), *Shifted Ackley* (f_{30}), *Shifted and rotated Griewank* (f_{31}) and *Shifted rastrigin* (f_{32}). Each category is composed of minimization problems and their global minimum is zero. All benchmark optimization problems adopted as search space's bounds $(-100, 100)$. We followed the same instructions defined in the competition of ICSI 2014.

V. Results

This section presents in details the results obtained from the comparison among the GGGP approaches when applied to the PSO algorithm design problem. As previously mentioned in the section IV, we performed two investigations. First, we compared the PSO designs generated by the CFG-GP with the designs produced by the GE and random search model considering all 32 optimization problems. Second, we compared between the resulting algorithms generated by CFG-GP with the algorithms adopted in the ICSI 2014, for each optimization problem.

A. GGGP approaches

Here in this section, we present the results related to the first investigation that attempts to determine which GGGP approach generates better PSO designs for an input problem. The table II shows the mean of the fitness values of the algorithms produced by the CFG-GP, GE and random search model for each optimization problem. To evaluate the results adequately, we applied the Wilcoxon test to certify whether the results of the CFG-GP are statistically better than the results of the counterparts. The symbol ▼indicates that the results achieved by an approach are statistically worse than the CFG-GP's ones. As it can be seen in the table II, the algorithms generated randomly were not able to reach satisfactory results, and consequently were overcome statistically by the CFG-GP's generated algorithms in 100% of the problems. Differently of the random approach, the algorithms produced by the GE attained promising results. Nonetheless, the CFG-GP surpassed the GE statistically in 23 out of the 32 optimization problems, and in the other 9 problems, both approaches reached results statistically equal.

This massive superiority of CFG-GP over GE is explained due to the way each approach represents their individuals. As previously mentioned in section II, the mutation and crossover operators developed for linear genome representation cause the locality problem and the generation of invalid individuals. These issues harm the performance of linearized approaches in the task of optimizing the PSO algorithm design. The same problems do not happen in the tree-based representation. Although a tree-based approach had not been investigated for PSO algorithm design, the results presented here showed that this method, specifically the CFG-GP algorithm, performed better than the GE, and it may be useful in other contexts.

TABLE II
Mean of fitness values of the algorithms CFG-GP, GE and random search model in each optimization problem.

Category	Problems	Algorithms		
		CFG-GP	GE	Random
Bowl	f_1	1.22e-168	1.32e-156	2.19 ▼
	f_2	1.42e-103	3.37e-90	3.74 ▼
	f_3	1.67e-157	2.16e-127 ▼	5.94 ▼
	f_4	1.33e-158	3.28e-118 ▼	7.48 ▼
	f_5	2.63e-164	1.61e-129 ▼	12.19 ▼
	f_6	1.43e-61	3.13e-35 ▼	8.31 ▼
	f_7	2.41e-155	4.74e-145	7.45 ▼
	f_8	2.32e-59	2.68e-41	8.52 ▼
Plate	f_9	1.20e-24	1.32e-04 ▼	13.15 ▼
	f_{10}	1.43e-58	1.48e-25 ▼	10.34 ▼
	f_{11}	2.51e-15	2.14e-02 ▼	19.16 ▼
	f_{12}	1.31e-35	0.36 ▼	23.23 ▼
	f_{13}	1.14e-77	5.43e-32 ▼	15.34 ▼
	f_{14}	3.12e-10	4.29e-01 ▼	20.27 ▼
	f_{15}	1.18e-51	2.23e-24 ▼	10.32 ▼
	f_{16}	1.73e-39	3.13e-22	14.2 ▼
Valley	f_{17}	2.97e-04	2.41 ▼	15.21 ▼
	f_{18}	6.54e-05	4.21 ▼	12.11 ▼
	f_{19}	8.25e-08	8.55 ▼	20.75 ▼
	f_{20}	1.12e-08	1.51e-03	12.91 ▼
	f_{21}	1.24e-49	3.15e-20 ▼	9.76 ▼
	f_{22}	1.07e-23	2.67e-15	16.18 ▼
	f_{23}	1.16e-12	0.37 ▼	21.17 ▼
	f_{24}	1.56e-11	0.72 ▼	27.36 ▼
Many	f_{25}	1.41e-29	1.59e-12 ▼	30.19 ▼
	f_{26}	0.0	0.0	21.83 ▼
	f_{27}	2.69e-08	3.12 ▼	24.97 ▼
	f_{28}	2.18e-53	1.75e-27 ▼	25.17 ▼
	f_{29}	1.13e-12	2.24e-02 ▼	26.18 ▼
	f_{30}	1.23e-32	1.57e-07 ▼	25.42 ▼
	f_{31}	0.0	0.0	22.62 ▼
	f_{32}	1.25e-07	3.12 ▼	27.27 ▼

B. ICSI algorithms

In the last section, we could see the potential of the CFG-GP approach for the generation of PSO algorithm designs when compared to the largely used GE approach. Aiming to investigate whether these generated algorithms can reach competitive results, this experiment compares the algorithms produced by the CFG-GP with the algorithms from the ICSI 2014 competition: HSDB (A_1), MPCPSO (A_2), MBO (A_3), dynFWA (A_4), DESP (A_5) and EFWA (A_6). We highlight that all algorithms from A_1 to A_6 adopted their default parameters used in the competition. We highlight that all algorithms from A_1 to A_6 used their default parameters used in the competition.

Here, we performed an experiment which compares the average fitness values achieved by the algorithms generated

TABLE III
RANKING OF ALGORITHMS CONSIDERING ALL PROBLEMS.

Rank	1	2	3	4	5	6	7
Algs.	A_2	CFG-GP	A_5	A_6	A_4	A_1	A_3
Avg. rank	1.62	1.92	3.46	3.84	4.83	6.07	6.36

by the CFG-GP for each problem and the average fitness values obtained by the ICSI 2014 algorithms adopted here. Table III presents the average ranks of all algorithms across all problems (for each problem, rank = 1 is assigned to the best algorithm, rank = 2 is assigned to the second best, and so on). The CFG-GP achieved the second best place in the rank from all algorithms. Aiming to verify which algorithms were overcome by the CFG-GP statistically, we applied the Wilcoxon signed-rank test. With this, we could see that the CFG-GP is statistically equivalent to A_2, and statistically better than A_5, A_6, A_4, A_1 and A_3.

We also carried out a deeper analysis performing a paired comparison between the CFG-GP and the competition algorithms. We compared the average fitness value achieved by the best competing algorithm from ICSI 2014 and by the CFG-GP for each problem. The collected results shown that the CFG-GP's generated algorithms achieved fitness values close to the global optimum for all problems. To verify whether these results are competitive in comparison to the best competition algorithm, we applied the Wilcoxon statistical test. The results achieved by the CFG-GP's generated algorithms are promising. In 7 out of 32 optimization problems the CFG-GP algorithms overcame the results of the competition algorithms and in the other 25 problems, both approaches reached results statistically equal.

The comparative results considering the competition algorithms should not be considered in an absolute sense because the competition algorithms were executed using their default parameters, whereas the PSO algorithms were optimized by the CFG-GP. In this sense, the comparison would be unfair. Our intention performing this experiment was to verify whether the CFG-GP can generate adequate PSO algorithms for the considered optimization problems.

VI. CONCLUSION

This paper discusses the problem of designing PSO algorithms. The intention is to generate automatically PSO designs that are adequate for a given problem regarding optimization performance.

In this work, we adopted a tree-based GGGP technique for the generation of PSO designs. Besides never having been used in the problem at hand, another reason for choosing this approach is that it avoids certain issues such as locality and infeasible solutions presented by other GGGP approaches for the same problem. Thus, the goal of this paper is to contrast the performance of a tree-base GGGP with a commonly used linearized GGGP in the production of PSO designs.

In the experiments, it was performed a comparison among the tree-based and linearized approaches, and a random search model as a baseline considering 32 unconstrained optimization problems with different levels of difficulty. The results showed that the tree-based GGGP produced better algorithms than the counterparts in most of the selected problems. We also compared the algorithms generated by the tree-based technique to state-of-art algorithms, and the results showed that the produced algorithms achieved competitive results.

ACKNOWLEDGMENT

The authors would like to thank CNPq, CAPES and FACEPE (Brazilian Agencies) for their financial support.

REFERENCES

[1] A. P. Engelbrecht, *Computational intelligence: an introduction*. John Wiley & Sons, 2007.

[2] J.-L. Liu *et al.*, "Evolving particle swarm optimization implemented by a genetic algorithm," *Source: Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 12, 2008.

[3] R. Poli, C. Di Chio, and W. B. Langdon, "Exploring extended particle swarms: a genetic programming approach," in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*. ACM, 2005, pp. 169–176.

[4] T. Si, A. De, and A. K. Bhattacharjee, "Grammatical swarm based-adaptable velocity update equations in particle swarm optimizer," in *Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2013*. Springer, 2014, pp. 197–206.

[5] P. B. Miranda and R. B. Prudêncio, "Gefpso: A framework for pso optimization based on grammatical evolution," in *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*. ACM, 2015, pp. 1087–1094.

[6] R. I. Mckay, N. X. Hoai, P. A. Whigham, Y. Shan, and M. O'Neill, "Grammar-based genetic programming: a survey," *Genetic Programming and Evolvable Machines*, vol. 11, no. 3-4, pp. 365–396, 2010.

[7] M. O'Neil and C. Ryan, "Grammatical evolution," in *Grammatical Evolution*. Springer, 2003, pp. 33–47.

[8] M. L. Wong and K. S. Leung, "Evolutionary program induction directed by logic grammars," *Evolutionary Computation*, vol. 5, no. 2, pp. 143–180, 1997.

[9] P. A. Whigham, G. Dick, J. Maclaurin, and C. A. Owen, "Examining the best of both worlds of grammatical evolution," in *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*. ACM, 2015, pp. 1111–1118.

[10] C. Li, S. Yang, and I. Korejo, "An adaptive mutation operator for particle swarm optimization," in *Proceedings of the 2008 UK Workshop on Computational Intelligence*. IEEE, 2008, pp. 165–170.

[11] T. Hendtlass, "A combined swarm differential evolution algorithm for optimization problems," in *Engineering of intelligent systems*. Springer, 2001, pp. 11–18.

[12] W.-J. Zhang, X.-F. Xie *et al.*, "Depso: hybrid particle swarm with differential evolution operator," in *IEEE International Conference on Systems Man and Cybernetics*, vol. 4, 2003, pp. 3816–3821.

[13] P. A. Whigham, "Inductive bias and genetic programming," in *Genetic Algorithms in Engineering Systems: Innovations and Applications, 1995. GALESIA. First International Conference on (Conf. Publ. No. 414)*. IET, 1995, pp. 461–466.

[14] Y. Tan, J. Li, and Z. Zheng, "Introduction and ranking results of the icsi 2014 competition on single objective optimization," *arXiv preprint arXiv:1501.02128*, 2015.

[15] R. Harper and A. Blair, "A structure preserving crossover in grammatical evolution," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 3. IEEE, 2005, pp. 2537–2544.

[16] A. Gavana, "Global Optimization Benchmarks and AMPGO," http://infinity77.net/global_optimization/, 2005, [Online; accessed 19-July-2015].