A Novel Context-Free Grammar to Guide the Construction of Particle Swarm Optimization Algorithms

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

Abstract—**Particle Swarm Optimization algorithm (PSO) has been largely studied over the years due to its flexibility and competitive results in different applications. Nevertheless, its performance depends on different aspects of design (e.g., inertia factor, velocity equation, topology). The task of deciding which is the best algorithm design to solve a particular problem is challenging due to the great number of possible variations and parameters to take into account. This work proposes a novel context-free grammar for Grammar-Guided Genetic Programming (GGGP) algorithms to guide the construction of Particle Swarm Optimizers. The proposed grammar addresses four aspects of the PSO algorithm that may strongly influence on its convergence: swarm initialization, neighborhood topology, velocity update equation and mutation operator. To evaluate this approach, a GGGP algorithm was set with the proposed grammar and applied to optimize the PSO algorithm in 32 unconstrained continuous optimization problems. In the experiments, we compared the designs generated considering the proposed grammar with the designs produced by other grammars proposed in the literature to automate PSO designs. The results obtained by the proposed grammar were better than the counterparts. Besides, we also compared the generated algorithms to state-of-art algorithms. The results have shown that the algorithms produced from the grammar achieved competitive results.**

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

I. Introduction

Particle Swarm Optimization (PSO) has been largely used and studied over the years due to its flexibility and competitive results in different applications [1]. Thus, various works have been conducted to improve further the performance of the algorithm [1]. Nonetheless, the PSO, as well as other algorithms, has a set of parameters values that need to be adjusted adequately. Otherwise, the algorithm performance can be harmed [1]. In this paper, the problem of algorithm design for the PSO is addressed. Our main goal is to determine an adequate configuration of parameters and components (i.e., a *design*) for the PSO algorithm regarding optimization performance when applied to a given optimization problem. The literature presents some previous works that performed empirical analysis on specific PSO parameters in isolation, which can be useful in practice [2], [3]. Among these works, we highlight the use of Grammar-Guided Genetic Programming algorithms (GGGP) [4], which generate programs by using production rules in a grammar definition. These algorithms are widely used and studied due to their capacity of generating new designs, instead of simply selecting designs from a pre-defined limited search space [5]. Two works addressed the task of designing PSO algorithms by using GGGP. The first deployed a GGGP algorithm to generate only velocity equations, and hence, it was not properly leveraged for automatic PSO design [2]. Recent work proposed a richer grammar considering more components of the PSO, such as topology, velocity equations, mutation operators and population size [3]. Although the results showed to be promising, the proposed grammar is limited.

In this work, we propose a novel context-free grammar to guide the construction of PSO algorithms. The proposed grammar addresses four aspects of the PSO algorithm that may strongly influence on its convergence: swarm initialization, neighborhood topology, velocity update equation and mutation operator. To evaluate this approach, a GGGP algorithm was set with the proposed grammar and applied to optimize the PSO design in 32 different continuous optimization functions associated with four categories of problems. Each category has different features and difficulty levels. In the experiments, we compared the designs generated by using the proposed grammar with the designs produced by other grammars defined in the literature to automate PSO designs. The results obtained by the proposed grammar were better than the counterparts. We also compared the generated algorithms to state-of-the-art algorithms. The results have shown that the algorithms produced from the grammar achieved competitive results. Finally, we investigated the components and parameter values of the PSO algorithms generated considering the proposed grammar for each category of problems.

This work is organized as follows. Section II presents the basic concepts of PSO, focusing on the parameters and components optimized in our proposal. Section III details related work to optimization of PSO algorithms. Section IV describes the proposed grammar in detail. Section V brings the experimental methodology and presents the obtained results. Finally, in Section VI, the conclusions are presented.

II. Particle Swarm Optimization

PSO is a well-succeeded optimization approach developed by Kennedy and Eberhart [1]. Considering the standard PSO, initially, a predefined number of particles are initialized randomly in the search space. While a stopping criterion is not reached, the particles update their velocity and position. Each particle explores the search space guided by its velocity, which is a combination between the best position found by itself ($pbest$) and by the best position found by its neighborhood ($gbest$) [1]. A new velocity is calculated by the following equation:

$$\begin{aligned} \vec{v}_i(t+1) = \vec{v}_i(t) + c_1 r_1(\vec{p}_i(t) - \vec{x}_i(t)) + \\ c_2 r_2(\vec{g}_i(t) - \vec{x}_i(t)). \end{aligned} \tag{1}$$

where $i = 1, ..., N$, N is the number of particles; $\vec{v}_i(t)$ is the current velocity; $\vec{x}_i(t)$ is the current position; $\vec{p}(t)$ and $\vec{g}(t)$ are the $pbest$ and $gbest$ respectively; c_1 and c_2 are the acceleration constants; 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:

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

Different developments were proposed to the standard PSO algorithm to improve its performance. Studies revealed that some PSO components such as swarm initialization, neighborhood topology, velocity update equation and mutation operators may interfere on its convergence [6]. Next, some developments are presented:

Initialization techniques: The swarm initialization is a phase of PSO algorithm that is not commonly studied. However, the initialization can influence on PSO performance in various problems [6]. In the case where particles are positioned close to a global optimum, the algorithm can quickly converge to a good solution in the search space. On the other hand, if they start close to a local solution, the convergence may be harmed. The most popular initialization technique is the uniform random initialization because it treats each part of the unrevealed search space equally [6]. However, different initialization strategies have been proposed to improve the evolution process. Among the initialization techniques, we highlight the Nonlinear Simplex Model (NSM) [6] and Opposition [7] which demonstrated to be useful.

Update velocity equations: A frequent phenomenon observed in PSO algorithms that use the standard velocity equation is the *explosion* of the particle's velocity. By using equation 1, the particles easily reach high-velocity values, making the convergence process more difficult. Shi and Eberhart proposed the addition of the inertia weight ω to the equation 1 of the standard PSO [1]. This weight can be a positive constant or a positive linear or nonlinear function of time [8]. As a consequence, the inertia weight controlled the *explosion* of the velocity and balanced the exploration and exploitation during the search. Another well-known variation of the velocity equation was proposed by Clerc et al. [1], called constriction factor. This proposal consists in the insertion of the parameter χ, which is derived from the acceleration constants, in the standard velocity equation. Besides the previous equations, other velocity equations were proposed aiming to improve the PSO performance [9], [10].

Neighborhood topology: It plays an important role in the PSO convergence because the update velocity equation depends on the knowledge acquired by the neighborhood. The first developed topologies were the Star and Ring [1]. The star is a fully-connected structure used to exploit the search space in promising regions. The ring connects each particle only with its direct neighbors, passing the information to the swarm indirectly. Both topologies gave rise to the creation of several other approaches that aimed to improve the search process. We can classify them as static or dynamic topologies. In static topologies, such as Star, Ring, Von-Neumann and Clan [1], the neighbors of a certain particle never change during the search process (i.e. the neighborhood structure is fixed) [11]. On the other hand, dynamic neighborhood allows the particles to be in different groups at different times balancing the particles' move between exploration and exploitation. Among the dynamic PSO variations already proposed, we highlight Niche and Species [12], Clubs [13] and Cluster [14].

Mutation operators: PSO can find solutions faster than other widespread search techniques like Genetic Algorithms (GA) [15]. However, it can suffer from premature convergence resulting in sub-optimal solutions, which can be a serious limitation in case of highly multimodal problems with several local optima [15]. Lack of population diversity in PSO can be a reason for convergence to local minima [15]. A strategy to improve diversity in a swarm is to include in the PSO a mutation operator, as commonly adopted in evolutionary algorithms. Several mutation operators can be seen in [15].

III. Related works

Previous work has dealt with the task of PSO design as another (meta)optimization problem [16], [17]. The search space, in this case, is the set of possible designs for the PSO on a problem. The objective function, in turn, is the same one adopted in the (base)optimization problem at hand. Different meta-heuristics, such as Differential Evolution (DE) and GA, were applied to optimize PSO parameters (e.g., inertia weight, acceleration constants) [16], [17]. Such optimization techniques, however, are not adequate to evolve more complex designs and also to exploit different combinations of PSO components [5].

Genetic Programming (GP) is an evolutionary approach which has potential advantages for algorithm design since it is flexible to represent and evolve more complex and flexible designs [5]. One of the most used GP approaches for the context of algorithm design is the GGGP algorithm, a GP approach guided by a grammar [5]. There are two reasons to combine GP with grammars [5]. First, the grammar incorporates prior knowledge about the problem domain helping to guide the GP search. Second, the production rules ensure syntactic correctness of the generated programs, differently from the traditional GP and GA. Different types of GGGP algorithms were proposed, such as Context-free Grammar Genetic Programming (CFG-GP) [5], Grammatical Evolution (GE) [5] and Logic grammar-based genetic programming (LOGENPRO) [18]. Nonetheless, GE has become the most broadly used extension of the GGGP system in the context of PSO [2]. Si et al. [2] adopted a GE algorithm to generate new velocity equations. However, the potential advantages of GE were not properly leveraged, since it was limited to evolve velocity updating equations. Other issues of PSO (e.g., the topology adopted) were not considered in this work. On the other hand, recent work, developed by Miranda and Prudêncio [3], used a GE algorithm with a richer grammar considering more components of the PSO. The neighborhood topology, mutation operators, velocity equations and population size were considered. Although the results showed to be promising, the proposed grammar is limited, since it considered standard static topologies (no dynamic topologies) and only two possible velocity equations.

IV. Proposal

As previously mentioned, few studies have applied GGGP in the PSO optimization and, moreover, the developed grammars considered a limited set of PSO parameters and components. In this work, we present a novel context-free grammar to guide the construction of PSO algorithms. Differently of previous works which used few and standard components to design the PSO, the proposed grammar was set with a complete set of modern and sophisticated components to build PSO algorithms with what is best already developed in the literature. We highlight that the proposed grammar follows the BNF format and can be used in any GGGP approach which adopts context-free grammar for algorithm generation.

To build a diverse and useful number of designs for the PSO, the grammar considers four aspects of the PSO algorithm that may strongly influence its performance. Algorithm 1 shows the parameters and components of the PSO algorithm to be optimized. The items in the format of tag <> correspond to the parameters or components that can be replaced by values defined in the grammar. Next, we present these aspects specified in the grammar (see Figure 1) with their optional values:

Initialization: the tag <INITIALIZATION> represents the strategy to initialize the swarm. We considered here three initialization methods: the traditional random uniform using Sobol sequences, NSM [6] and Opposition [7]. These initialization strategies were cited in Section II. We highlight that no previous work adopted the initialization in the optimization of PSO algorithms.

Velocity update: the tag <UPDATE-VELOCITY> represents the velocity equation to update particles' velocity. The proposed grammar can produce a variety of equations to combine the cognitive and social information (variables $gbest$ and $pbest$). The standard velocity equations presented in Section II are just special cases that can be derived from this grammar. The ω uses the linear decreasing strategy [8] and $\chi = 0.7$ (constriction factor) can be eventually adopted to control the particles' velocity.

Neighborhood topology: the tag <TOPOLOGY> corresponds to the topology mechanism that supplies the neighborhood positions for the velocity equation. Due to the rich PSO literature, choosing a set of relevant neighborhood topologies proved to be tough. Considering as criteria the inspiration source, the performance, and theoretical properties, the number of reported applications and the potential for further development and improvements, we selected seven neighborhood topologies from the literature: Star, Ring, Niche, Clubs, Cluster, Species and Dynamic Ring.

Mutation: Finally, possible mutation operators for the PSO are represented by the tag <MUTATION>. Five mutation operators, suggested by [15], can be assigned to this tag. Moreover, we also considered the case where the PSO does not use any mutation operator (the tag assumes the value λ). The tag <PROB-MUTATION>, in turn, defines the possible values of mutation probability.

It is worth mentioning that each PSO was run with a population size equals to 30 and the constant r, from tag <VAR>, assumes a value within 0 and 1. As it can be seen, the tags <INITIALIZATION>, <TOPOLOGY>, <MUTATION>, and <PROB-MUTATION> can be exchanged by terminals directly. In other words, it works as a selection of parameter values. However, the tag <UPDATE-VELOCITY> can generate different velocity equations with different formats and components.

Algorithm 1: PSO algorithm to be optimized.

```
   swarm_size = 30
   swarm = <INITIALIZATION>(swarm_size)
   evaluate_fitness(swarm)
   topology = <TOPOLOGY>
1: while !stop_criterion do
       <UPDATE-VELOCITY>(swarm, topology)
       update_position(swarm, topology)
       <MUTATION>(swarm)
       update_fitness(swarm)
       update_pbest(swarm)
       update_gbest(swarm, topology)
   end
```

V. Experiments and results

To evaluate the proposed approach, it was necessary to choose a GGGP algorithm to use the grammar. Thus, we chose the GE algorithm and incorporated the proposed grammar on it to optimize the PSO design (we call GE$_{new}$). The experiments were divided into three phases. First, we compared the designs produced by the GE$_{new}$ to the designs generated by GE using two other grammars already used for PSO algorithm design. Here, we considered the grammar developed by Miranda and Prudêncio [3] (we call GE$_1$) and the grammar created by Si et al. [2] (we call GE$_2$). Second, we compared the algorithms produced by the GE$_{new}$ to the algorithms adopted in the competition of the International Conference of Swarm Intelligence 2014 (ICSI 2014) [19], considering the same set of problems. Our goal is to verify whether the generated algorithms achieve competitive results in comparison to consolidated algorithms. Finally, we investigated the components and parameter values

```
⟨INITIALIZATION⟩ ⊨ Uniform Random | Opposition
                 | Non-linear Simplex Model
⟨TOPOLOGY⟩ ⊨ Star | Ring
                 | Niche | Clubs
                 | Cluster | Species
                 | Dynamic-Ring
⟨UPDATE-VELOCITY⟩ ⊨ (χ ∗ (v + ⟨EXP⟩)
                 | ω ∗ v + ⟨EXP⟩
⟨EXP⟩ ⊨ (⟨EXP⟩⟨OP⟩⟨EXP⟩)
                 | ⟨VAR⟩
⟨OP⟩ ⊨ + | - | / | *
⟨VAR⟩ ⊨ x | pBest | gBest | r
⟨MUTATION⟩ ⊨ Gaussian(⟨PROB-MUTATION⟩)
                 | Cauchy(⟨PROB-MUTATION⟩)
                 | Michalewicz(⟨PROB-MUTATION⟩)
                 | Levy(⟨PROB-MUTATION⟩)
                 | Random(⟨PROB-MUTATION⟩)
                 | λ
⟨PROB-MUTATION⟩ ⊨ 0.1 | 0.2 | ... | 1.0
```

Fig. 1. Proposed BNF grammar for PSO optimization.

of the PSO algorithms produced by the GE$_{new}$ for each category of problems. All approaches were evaluated considering a set of 32 unconstrained continuous optimization problems. All the functions are scalable and adopted 30 dimensions. The adopted functions can be classified in four categories: *Bowl-Shaped*, *Plate Shaped*, *Valley-Shaped* and *Many Local Minima*, allowing us to evaluate the approach proposed in different perspectives and levels of difficulty [20].

The *Bowl-Shaped* category presents functions with 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* category are composed by functions which have the global optimum located in an uniform plain. As the global minimum is located in the plain, the task of finding it can become difficult. 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}). Functions with multiple local minima are in the *Many Local Minima* category. These problems can trap optimization algorithms in a local minimum, turning the search more difficult. 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}).

TABLE I
GE PARAMETERS.

Parameter	**Value**
Population size	50
Number of generations	20
Crossover probability (LHS crossover)	0.8
Point mutation probability	0.01
Chromosome length	30
Selection mechanism	Roulette Wheel
Generation model	Steady state

It is noteworthy that the competition algorithms, GE$_1$ and GE$_2$ used the same setting values adopted by the PSO algorithms. In other words, the stop criterion is $5,000$ iterations per simulation and all algorithms executed 20 times to generate the mean of its fitness values.

A. Comparing grammars

This experiment compared the designs produced by GE$_{new}$ to the designs produced by the GE using previous grammars. The GE set with each grammar followed the same criterion of execution in this experiment: each PSO generated is executed (stop criterion is $5,000$ iterations) and the $pbest$'s fitness of the best particle is stored. To guarantee reliability, each PSO is executed 20 times, and the average fitness value is returned as the fitness of that design. Once the GE algorithm's stop criterion (20 generations) is reached, the best design found over all generations is returned as output. The parameter values of GE are listed in Table I. It is worth to mention that the experiments were performed on an Intel Core i7-5600U processor with 4M Cache, up to 3.20 GHz.

Table II shows an analysis performing a comparison between the GE$_{new}$ and the other GGGP approaches for each problem. As it can be seen on this table, the fitness values achieved by the algorithms generated by GE$_{new}$ overcame, statistically, the GE$_1$ results in 19 out of 32 problems. The remaining 13 problems, the results of both approaches were statistically equal. Regarding GE$_2$, the results obtained by the GE$_{new}$ were even more impressive. The GE$_2$ was overcome by the GE$_{new}$ in 27 out of the 32 problems, and the results of both approaches in the five remaining problems were considered statistically equal.

B. ICSI algorithms

Here, the goal is to investigate whether the algorithms generated by the GE$_{new}$ can reach competitive results. Thus, this experiment compares the algorithms generated by the GE$_{new}$ 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.

The experiment performed compares the average fitness values achieved by the algorithms for each problem. In order to compare all algorithms against each other, we performed an

TABLE II
MEAN OF FITNESS VALUES OF THE ALGORITHMS GE$_{new}$, GE$_1$ AND GE$_2$ IN EACH OPTIMIZATION PROBLEM.

Category	**Problems**	**Algorithms**		
		GE$_{new}$	GE$_1$	GE$_2$
Bowl	f_1	1.46e-148	1.41e-128 ▼	2.36e-118 ▼
	f_2	2.52e-73	2.34e-70	1.64e-64
	f_3	3.04e-127	3.14e-97 ▼	4.84e-64 ▼
	f_4	2.13e-138	2.13e-98 ▼	1.42e-73 ▼
	f_5	2.36e-144	1.16e-119 ▼	2.19e-119 ▼
	f_6	1.34e-31	2.31e-25	1.31e-18 ▼
	f_7	2.14e-135	4.47e-128	1.45e-123
	f_8	2.23e-29	2.86e-21	3.52e-19
Plate	f_9	1.02e-4	0.09 ▼	3.15 ▼
	f_{10}	1.34e-28	1.84e-15 ▼	0.34 ▼
	f_{11}	2.15	2.41	9.16 ▼
	f_{12}	2.13e-15	0.63 ▼	3.23 ▼
	f_{13}	1.2e-57	5.34e-22 ▼	5.34e-04 ▼
	f_{14}	3.21	5.92 ▼	10.72 ▼
	f_{15}	1.22e-21	3.32e-14 ▼	1.32e-04 ▼
	f_{16}	1.37e-19	4.31e-12	4.0e-05
Valley	f_{17}	2.79	5.41 ▼	15.21 ▼
	f_{18}	6.54	8.12 ▼	12.11 ▼
	f_{19}	8.25	11.55 ▼	20.75 ▼
	f_{20}	0.44	0.51	2.19
	f_{21}	2.42e-19	4.51e-14	0.023 ▼
	f_{22}	0.49	0.67	6.18 ▼
	f_{23}	0.61	2.73 ▼	6.17 ▼
	f_{24}	0.615	2.72 ▼	7.36 ▼
Many	f_{25}	2.14e-9	1.95e-4	11.15 ▼
	f_{26}	0.0	0.0	11.83 ▼
	f_{27}	2.96	6.21 ▼	14.97 ▼
	f_{28}	2.81e-23	1.57e-17	10.17 ▼
	f_{29}	0.31	2.42 ▼	16.18 ▼
	f_{30}	1.79e-12	0.475 ▼	15.42 ▼
	f_{31}	0.0	0.0	12.62 ▼
	f_{32}	3.52	6.21 ▼	17.27 ▼

experiment which ranks 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 GE$_{new}$ achieved the fourth best place in the rank from all algorithms. Aiming to verify which algorithms were overcome statistically by the GE$_{new}$, we applied the Wilcoxon signed-rank test. With this, we could see that the GE$_{new}$ is statistically equivalent to A_2, A_5 and A_6, and statistically better than A_1 and A_3.

In addition, we also carried out a deeper analysis performing a comparison between the GE$_{new}$ and the competition algorithms. For this, the average fitness value achieved by the best-competing algorithm from ICSI 2014 and by GE$_{new}$ were compared considering each problem. According to the collected results, the GE$_{new}$ generated algorithms which achieved fitness values close to the global optimum for all problems. To verify whether these results are competitive in comparison to the best algorithm, we applied the Wilcoxon statistical test. The achieved fitness values were considered statistically identical to the results of the best algorithm for each problem. 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 GE$_{new}$. In this sense, the comparison would be unfair. Our intention performing this experiment was to verify whether the GE$_{new}$ can generate adequate PSO algorithms for the considered optimization problems.

TABLE III
RESULTING DESIGNS PER CATEGORY OF PROBLEMS.

Category	**Parameter values**			
	INITIAL.	MUTATION	PROB-MUT.	TOPOLOGY
Bowl	Random (68.5%)	*Random* (38%)	0.2 (41%)	Star (58%)
Plate	Opposit. (50%)	*Random* (32%)	0.2 (34.5%)	Niche (28%)
Valley	Opposit. (51%)	*Gauss.* (31%)	0.2 (31.5%)	Niche (32%)
Many	NSM (47%)	*Gauss.* (33%)	0.4 (39%)	Species (31%)

C. Analysis of PSO designs

In this section, we investigate the components and parameter values used by the generated PSO algorithms. To perform this battery of experiments, we stored, for each problem, the GE$_{new}$'s output designs and identified the most used components and parameter values across all runs and generations of the GE. Table III presents the resulting designs obtained for each category of problems. This table also shows the percentage of times that the resulting parameter values were used in PSO designs during the simulations.

As it can be seen, although the Uniform Random initialization is the most used strategy in literature, the Opposite and NSM showed to be useful for problems in the *Plate*, *Valley* and *Many Local Minima* categories. Considering the <MUTATION> parameter, the Gaussian and Random strategy were selected as the resulting mutation operators, and they were also the most used across the categories. The appearance of the mutation operator in the designs represents that they can be useful for the PSO results. Considering the <TOPOLOGY> parameter, we can see that the topology choice depends strongly on the problem difficulty. Differently from the other parameters, whose values were robust, the <TOPOLOGY> choices presented variations according to the category of the problem. It can be seen that the *Star* was the most adequate topology for problems in the *Bowl* category. Problems which belongs to this category are considered easier to be optimized, thus, topologies which favor *exploitation* would be more suitable. The problems in the *Plate* category, differently from the *Bowl* category, need more exploratory topologies to avoid stagnation in the plain. Thus, the *Niche*, *Clubs* and *Species* topologies were commonly used during the search. Similarly to the *Plate* category, problems which belong to the *Valley* category also used the *Niche* topology. Regarding the *Many Local Minima* category, more sophisticated topologies were massively applied in the PSO algorithms. The topologies *Niche* and *Species* were adopted in more than 60% of designs. This result can be explained due to the existence of mechanisms that provide a good balance between exploitation and exploration.

Next, examples of the *generated* velocity equations for the PSO algorithms are presented. Considering the *Bowl-shaped* category, the generated equation for the optimization problem f_1 is $\chi * (\vec{v}_i + [\vec{g}_i - r_1 * \vec{x}_i] - \vec{v}_i * r_2)$. This equation is suitable to operate on unimodal objective functions because it has a 100% social component and a random friction component. As the *Star* topology was adopted in the generated design for f_1, it makes the convergence to the global optimum faster. Regarding to the *Plate-shaped* category, the generated equation for f_9 is $\chi * (\vec{v}_i + [\vec{g}_i - r_1 * \vec{x}_i] + r_2 * \vec{g}_i * \vec{x}_i - r_3 * \vec{v}_i)$, and it has three components. The first component, $[\vec{g}_i - r_1 * \vec{x}_i]$, gives a greater weight to the social aspect. When the second component, $r_2 * \vec{g}_i * \vec{x}_i$, is near from zero it means that the current particle's position is close to the best social position. When this component is non-zero, it tends either to slow or to accelerate the motion of the particles, according to the positions of the current position and the position of the swarm best. Finally, $r_3 * \vec{v}_i$ is a random friction component. It is worth to mention that the *Niche* topology was used in the generated design for f_9. The combination of the generated equation with the selected topology made the resulting algorithm able to balance exploration and exploitation. Considering the *Valley-shaped* category, the generated equation for f_{17} is $\chi * (\vec{v}_i + r_1 * [\vec{g}_i - \vec{x}_i] + r_2 * [\vec{p}_i - \vec{x}_i] - r_3 * \vec{v}_i)$, and it has a similar structure when compared to the constricted equation. This equation considers the particle's social and cognitive information. As the resulting algorithm for the problem f_{17} also used the *Niche* topology, the combination of both equation and topology favored the balancing between exploration and exploitation. The generated equation for f_{25} (*Many* category) is $\chi * (\vec{v}_i + r_1 * [\vec{g}_i - \vec{x}_i] - r_2 * r_3 * \vec{x}_i * \vec{g}_i^2)$. As it can be seen, the particle's best information is not used in the equation, probably because, in a highly multimodal landscape, particles should not trust their observations too much. The second component tends to push the particles towards the origin unless the swarm best is near the origin. When this component is non-zero, it can slow or accelerate the motion of the particles, according to the positions of the current position and the position of the swarm best. Besides, the *Species* topology was adopted for f_{25}, and it helped the generated PSO algorithm to have a good performance on the problem at hand.

VI. Conclusion

This paper addresses the problem of algorithm design for the PSO. Our goal is to produce a PSO algorithm that is adequate for a given problem regarding optimization performance. Thus, we proposed a novel context-free grammar for Grammar-Guided Genetic Programming algorithms to guide the construction of PSO algorithms. The proposed grammar considers four aspects of the PSO algorithm that may strongly influence on its convergence: swarm initialization, neighborhood topology, velocity update equation and mutation operator.

To evaluate this approach, a GGGP algorithm was set with the proposed grammar and applied to optimize the PSO algorithm in 32 unconstrained continuous optimization problems. In the experiments, we compared the designs generated by using the proposed grammar with the designs produced by other grammars defined in the literature to automate PSO designs. The results obtained by the proposed grammar were better than the counterparts. Besides, we also compared the algorithms generated by state-of-art algorithms. The results have shown that the algorithms produced from the grammar 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] T. Si, A. De, and A. K. Bhattacharjee, "Grammatical swarm based-adaptable velocity update equations in particle swarm optimizer," in *International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2013*. Springer, 2014, pp. 197–206.

[3] 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.

[4] 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.

[5] P. A. Whigham *et al.*, "Grammatically-based genetic programming," in *Workshop on genetic programming: from theory to real-world applications*, vol. 16. Citeseer, 1995, pp. 33–41.

[6] K. E. Parsopoulos, *Particle Swarm Optimization and Intelligence: Advances and Applications: Advances and Applications*. IGI Global, 2010.

[7] H. Jabeen, Z. Jalil, and A. R. Baig, "Opposition based initialization in particle swarm optimization (o-pso)," in *11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*. ACM, 2009, pp. 2047–2052.

[8] J. Xin, G. Chen, and Y. Hai, "A particle swarm optimizer with multi-stage linearly-decreasing inertia weight," in *Computational Sciences and Optimization, 2009. CSO 2009. International Joint Conference on*, vol. 1. IEEE, 2009, pp. 505–508.

[9] Z. Xinchao, "A perturbed particle swarm algorithm for numerical optimization," *Applied Soft Computing*, vol. 10, no. 1, pp. 119–124, 2010.

[10] M. M. El-Sherbiny, "Particle swarm inspired optimization algorithm without velocity equation," *Egyptian Informatics Journal*, vol. 12, no. 1, pp. 1–8, 2011.

[11] Y.-X. Wang and Q.-L. Xiang, "Particle swarms with dynamic ring topology," in *IEEE Congress on Evolutionary Computation, 2008*. IEEE, 2008, pp. 419–423.

[12] R. Brits, A. P. Engelbrecht, and F. Van den Bergh, "A niching particle swarm optimizer," in *4th Asia-Pacific conference on simulated evolution and learning*, vol. 2. Singapore: Orchid Country Club, 2002, pp. 692–696.

[13] W. Elshamy, H. M. Emara, and A. Bahgat, "Clubs-based particle swarm optimization," in *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*. IEEE, 2007, pp. 289–296.

[14] A. Passaro and A. Starita, "Particle swarm optimization for multimodal functions: a clustering approach," *Journal of Artificial Evolution and Applications*, vol. 2008, p. 8, 2008.

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

[16] K. E. Parsopoulos and M. N. Vrahatis, "Recent approaches to global optimization problems through particle swarm optimization," *Natural computing*, vol. 1, no. 2-3, pp. 235–306, 2002.

[17] 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.

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

[19] 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.

[20] S. Surjanovic and D. Bingham, "Virtual library of simulation experiments: test functions and datasets," *Retrieved December*, vol. 4, p. 2014, 2014.