Dynamic Interconnection Approach With BLX-Based Search Applied to Multi-Swarm Optimizer: An Empirical Analysis to Real Constrained Optimization

DANIEL LEAL SOUZA1,2,5,7, (Member, IEEE), RODRIGO LISBÓA PEREIRA2,5, (Member, IEEE), MARIO T. R. SERRA NETO2,4, MARCO A. F. MOLLINETTI2,3, OTÁVIO NOURA TEIXEIRA2,5,6 and ROBERTO C. L. DE OLIVEIRA1,2,5

1Electrical Engineering Post Graduation Program of the Federal University of Pará (UFPA), Belém, PA, Brazil
2Computational Technologies Laboratory (LabTeC), Federal Rural University of Amazonia (UFRA), Paragominas, PA, Brazil
3Systems Optimization Laboratory, Tsukuba University Tsukuba, Ibaraki, Japan
4University of Porto (FCUP), Porto, Portugal
5Laboratory of Bionspired Computing (LCBIO), UFPA, Belém, Pará, Brazil
6Natural Computing Laboratory (NCL), UFPA, Tucuruí, PA, Brazil
7Cyberspatial Institute (ICIBE), Federal Rural University of Amazonia (UFRA), Belém, PA, Brazil

Corresponding author: Daniel Leal Souza (e-mail: daniel.leal.souza@gmail.com).

ABSTRACT

The Multi-Swarm approach allows the use of multiple configurations between two or more populations of particles, where each one can present different approaches (e.g. lbest, gbest, Unified, Guaranteed-Convergence) directed towards improving the optimization process. This article presents a proposal for local/global stochastic interconnection applied to the context of the Multi-Swarm algorithm, as well as for incrementing a local search method for refining previously obtained solutions. Two proposals are introduced for this new Multi-Swarm PSO (MSO). The first one is the inclusion of "counterpart particles", which establishes a sub-topology between inter-swarm particles, accessed by migration and evaluability rules. The other involves using customized crossover operators and is based on the BLX scheme (Blend Crossover) with direction information used as a reference for establish a subspace search around the particles. Performance and robustness of the new approaches were assessed by ten constrained engineering design optimization problems (COPs), as is compared to other solutions already published in the scientific literature. Results indicate significant performance improvements for all 10 COPs when compared to concurrent-based MSOs. By making available new references from other swarms, the counterpart particles approach tends to improve the optimization process in the search space, while an intermediate layer of local search based on a modified directed BLX crossover should provide an extra search around the particle, and thus, refining previously obtained solutions.

INDEX TERMS

PSO, Multi-Swarm, BLX, Swarm Intelligence, Topology, Constrained Optimization, CUDA.

I. INTRODUCTION

SINCe their first publication in 1995 [1], Particle Swarm Optimization (PSO) and its variants have been important in solving a wide variety of problems [2], [3], as well as being one of the major references for global optimization in the field of Swarm Intelligence (SI) [3]–[6]. Kennedy and Eberhart [1] developed the PSO mechanisms based on the research in behavioral dynamics by biologists Frank Heppner and Ulf Grenander [7], who observed flocks of birds seeking food and nesting sites in a given region [4], [7]. To simplify, one may say that the work of Heppner and Grenander provided the essential bioinspired references for the PSO algorithm and its variants: Each particle in the system (birds, fish) represents a localized solution in the
Six important characteristics make PSO a quite attractive metaheuristic in the search and optimization context [9], [10]: 1) Intrinsic memory associated with each solution (local); 2) Capacity for information exchanges; 3) Search based on references of good prior results; 4) High degree of parallelism; 5) Easily implemented; 6) Rapid convergence. Because of those attributes, PSO has been successfully applied to a diverse family of problems, with hybrid or derived versions being used in several fields of scientific knowledge (e.g., engineering, antenna design, robotics, machine learning) [11]. Such variations range from modifying control parameters and topologies, to introducing new learning techniques [2], [11]–[14].

Likewise, the use of multi-swarm structures is also noteworthy. They present algorithms with high diversification, flexibility and strategies directed towards the control and flow of information between different populations [9], [15]–[17]. Several proposals involving the use of cooperative PSOs organized into sub-swarms have been published in the scientific literature, which indicate superior performance compared to mono-swarm implementations [17]. Examples of multi-swarm PSOs (MSOs) include versions with topology configurations between populations and data transfers related to global bests results, as well as strategies involving communication hierarchies between swarms (e.g., master-slave) [9], [17]–[19].

Parallel to the multi-swarm methods, several hybrid versions of the PSO have been developed. Optimization strategies that include two or more search mechanisms working together with the original mechanism tend to increase efficiency in use of the search space, as well as providing additional resources for controlling diversity and local searches [20]. Considering the possibilities for improvements and customization capacity of Multi-swarm PSOs, as well as addition of techniques focused on sweeping the search space beyond the mechanisms found in the original PSO original, hybridizing particle swarms with crossover-based solutions for local search in subspaces presents a range of possibilities for allowing improvements in the optimization process.

A. PROBLEM DISCUSSION

Because it is a tool originally designed to have a strong tendency for refining previously obtained results, the original PSO [1] has some limitations in balancing between exploration and exploitation, because due to the rapid convergence, there is a risk of stagnation at the local bests [9], [22]–[24]. In contrast to that scenario, and with a history of five decades of development, GAs have emerged as one of the more successful evolutionary algorithms, mainly because of their capacity for adaptation regarding mechanisms for diversification [25]–[27].

Given the number of operators and adjustable parameters, as well as a great variety of crossover techniques, such algorithms present a better balance between exploration and exploitation. More specifically, the use of crossover methods between solutions and mutation are capable of generating new combinations with greater balancing between exploration of the search space and refinement of the best results already found [9], [28]–[30]. The combination of such mechanisms in a multi-swarm environment with strongly connected topology, as well as the exchange of data between the best sites and globally between all swarms through context and rules for migration should provide a sophisticated optimization market, with rapid convergence and greater use of space for solutions through the so-called "variant particles". A basic scheme for the proposed algorithms can be seen in Figure 1.

![FIGURE 1. Structural organization of the proposed algorithms](image-url)

Considering the mechanisms presented in Figure 1, this paper expands the research with hybrid multi-swarms begun in [31], [32], introducing a set of migration rules adapted for rapid access to memory data belonging to neighboring swarms (i.e., counterpart particles, better global value among all swarms), besides a control scheme based on stochastic rules for allowing access to swarm components (local/global stochastic access factor). At the particle level, each solution is equipped with mechanisms for generating variations based on its own values within a defined search subspace, established using the information on the direction (or trend) of the respective position data. New particles are then generated based on the BLX method modified inside that search space, providing a greater sweep capacity in an N-dimensional function.
II. A BRIEF RETROSPECTIVE AND RELATED WORKS

This paper may be considered an indirect outcome of the studies published in [31], [32], which developed a set of seven algorithms based on the classical and quantum particle swarm paradigms, both modified to include the Evolutionary Strategies scheme derived from the EPSO algorithm by Vladimiro Miranda and Nuno Fonseca [33]. Of the seven solutions proposed, five are of the multi-swarm type with master-slave topology on the model in Niu [18] and two of the monoswarm type [31], with all being implemented on a heterogeneous GPGPU computing paradigm powered by CUDA architecture.

Despite the good results achieved by including EPSO’s ES model in the context of a multi-swarm version of PSO [32] and QPSO [31], the experiments were conducted using a low number of restrictive functions, with no comparative analysis between the master and slave swarms, thus limiting the assertion of more precise conclusions.

A. RELATED WORKS ON HYBRID PSO WITH GAS

The use of mechanisms derived from Evolutionary Strategies or Genetic Algorithms in customized versions of PSO is a well-established approach with several papers available in the scientific literature, such as the previously highlighted EPSO algorithm by Miranda and Leite. Besides hybridization between PSO and GA, other variants of the particle swarm optimization algorithm with refinement of the search process based on other techniques (e.g. mathematical and numerical optimization, local search) have been applied in a wide variety of research fields [36]–[39].

Duong et al. [22] present a hybrid algorithm based on a particle swarm optimization and real-coded genetic algorithms, in which PSO is used to update the solution, while the GA recombination operators produced offsprings by crossover methods applied to a pair of particles.

In Ali et al. [40], two combinations between PSO and GA are applied in high-dimensional global optimization, as well as the development of electromagnetic structures and antenna modelling. In Grimaccia et al. [20], the population is divided at each iteration and then optimized based on both techniques (PSO and GA) simultaneously, one for each portion of the population, while in Robinson et al. [41] and Ali et al. [40], PSO and GA are performed sequentially.

Such implementations generally vary between (or include) the following options: 1) Split the population between both algorithms [40], [41]; 2) Scaling PSO for exploitation and GA for exploration [22]; 3) Modified equations for velocity and position of the particle swarm, which includes mechanisms for mutation and/or crossover (position and velocity) applied to a pair of particles from the same swarm [42], [43].

B. RELATED WORKS ON SINGLE OR MULTI PSO ON CONSTRAINED OPTIMIZATION (COP)

Several articles regarding the use of PSO for constrained optimization (COP) are briefly reviewed in this subsection, and the results show a good performance in solving problems associated with that class of problems [16], [44]–[46]. The implementations described cover a range of strategies for controlling and managing functions with constraints, including fitness normalization through basic [47], [48] and dynamic [49] penalty factors, feasibility rules [50], [51], optimization of the objective function and constraints separately [52], and others.

In Liu et al. [53], an adaptive approach for restrictive problems is implemented in order to maintain a fixed proportion of non-viable solutions, and is employed in cases where there are viable particles with fitness values higher than all the solutions with violations. Coath and Halgamuge [54] presented a comparison of two methods for manipulating restrictions in the context of the PSO algorithm (i.e. preservation of feasible solutions [55], and application of a dynamic penalty function [56]). It was demonstrated that the convergence rate observed in the PSO based on normalization by penalty factor tends to be faster when compared methods that use only feasible solutions in the search process [45], [54].

With regard to Multi-swarm environments, Zheng et al. [57] propose an aggregation of neighboring particles in several sub-swarms in order to explore a subspace of feasible solutions (without violations). In Yang et al. [44] proposed a master-slave PSO hierarchy in which the master swarm is responsible for optimizing the objective function while the slave swarm works by searching for viability in the restrictions. Zhao and Li [58] introduces a two-stage optimizer: Creation of sub-swarms so as to increase the chances of achieving better results (1): Merging the sub-swarms into a large swarm to refine the global best (2).
C. PROPOSED MECHANISMS AND SOLUTIONS

Despite the improvements proposed by the solutions described above, such algorithms have limitations. The inclusion of a GA applied to the PSO in a co-optimization context tends to increase the algorithm’s complexity, while the use of specific mechanisms found in the GA (modified crossover) and adapted to the PSO should provide an efficient, yet less complex, hybrid metaheuristic.

In the context of multi-swarm algorithms, the use of a master-slave topology entails the sequential execution of more than one layer of swarms, which causes data dependency. A concurrent approach using only a single set of interconnected swarms, where an environment based on information exchange ruled by a specific set of rules (synchronous execution of each population), would decrease data dependency as well as solution complexity when compared with master-slave topology.

By addressing these issues, the present paper introduces the following mechanisms and improvements:

1) Intercommunication between better locations for corresponding particles through counterpart particles (CP): We propose the inclusion of components associated to the best local value of all solutions connected in a trans-swarm neighborhood, with such equivalent particles identified by the same index. The "counterpart particles" provide additional information from their neighbors, located in other swarms.

2) A set of migration rules with tolerance threshold, adapted to constrained optimization (TMR): The values related to the local and global bests used by the particles are selected through comparison between the counterparts’ local best and the overall global result found by the swarm set, respectively. Seeking to avoid prevalence of exclusive access for the best components, a tolerance threshold is applied to the migration rules in order to increase the odds for a "more than two components scenario" in the velocity update. This approach is designed to include up to four components for adjusting particle trajectory;

3) Probabilistic Access Factor (PAF): Based on the stochastic star topology proposal from Miranda, Keko and Duque [59] the probabilistic access factor is extended to all components: Particle’s local best, best counterpart’s local best, swarm’s global best and best global value achieved by the multi-swarm environment. This scheme provides an option for alternating between a more explicit (selfish) local search and a more exploitative approach applied to the global best (selfless), with both using probability thresholds;

4) Variant particles generated in search subspaces by a direction-based crossover method (V1/V2): In order to expand exploration capabilities, each particle present in the swarm may generate up to two new alternative solutions. The variant particles are obtained through a modified crossover scheme based on the direction history (trajectory) of the original particle, where a set of positions inside the search subspaces is randomly selected as a reference for devising new solutions;

5) Dynamic limits applied to the inertia weight and acceleration coefficients: Based on the studies conducted by Harrison et al. [60], the inertia weight (w) is calculated through modified stochastic functions, and varies according to the iteration interval. For the acceleration coefficients (C1, Cc, Cg, Cs), also obtained through a customized stochastic function, the intervals are determined by combining the available (see TMR and PAF), which must be in conformity with the convergence parameters established by Poli’s theoretically defined region.

It is important to note that, besides the proposals mentioned above, based on data obtained through experiments, this article seeks to analyze the impact and viability of such modifications in the PSO, both in the isolated use of variant particles (4) and the probabilistic access factor (3) when implementing such approaches in the multi-swarm context presented herein (1 and 2).

III. THEORETICAL BACKGROUND

So as to leave the PSO algorithm adapted for better use of the search space, the swarms of the proposed methods, besides the previously defined data on trajectory (i.e., position, velocity, fitness, normalized fitness), provide information associated with a direction trend based on the particle movement history, applied both for demarcating the search subspace around the original particle and to the modified crossover process for generating variant particles. Should the solutions produced by this directed-based crossover prove superior, they will replace the original one and be passed on to the next iteration.

In a matrix programming notation, Figure 2 presents the main variables for an individual in the context of PSO algorithms with elements related to the variant particles (subs-swarm limits) and information for guidance. Variables i, j, k and r represent the following indicators: Particle (i); Position or velocity (j); Swarm (k); Restrictions (r).
In terms of communication among particles, a fully-connected topology linking solutions from different swarms, together with stochastic flow control for position values provide quick access to new solutions at both local (counterpart particles) and global level (best result among all swarms). Figure 3 presents the memory components structure associated with one individual in the context of strongly connected PSO multi-swarm algorithms. Variables $i$, $j$, $k$ and $r$ represent the following indexes: Particle ($i$); Position or velocity ($j$); Swarm ($k$); Restrictions ($r$).

Through this new set of data and mechanisms, it is possible to expand the PSO particle search capacity beyond the classic velocity update, by adding procedures where all the information obtained by all the swarms can be available simultaneously during the optimization process (see Figure 3). One should note that the process for information exchanges between swarms also benefits from the generation of variant particles (V1/V2), which improves the search by generate new solutions in the subspace area around the original particles.

Every component associated to a particle (presented in Figure 2) is organized through the following data structures (one-dimensional, two-dimensional or three-dimensional):

- **Original Particles**: 3D position matrix ($X$); 3D velocity matrix ($V$); 2D matrix for fitness ($f$); 2D Matrix for normalized fitness ($nf$); 2D Matrix for number of violated constraints ($vf$); 3D Matrix for constraint values ($rf$);
- **Variant particles**: 3D position matrix ($\bar{X}$); 2D matrix for fitness ($\bar{f}$); 2D Matrix for normalized fitness ($\bar{nf}$); 2D Matrix for number of violated constraints ($\bar{vf}$); 3D Matrix for constraint values ($\bar{rf}$);

The memory components (presented in Figure 3) are described as follow:

- **Local best for the $i$-th particle and counterpart particle ($c$)**: 3D and 2D matrix for position ($Pl, Pc$); 2D and 1D matrix for fitness ($fPl, fPc$); 2D and 1D matrix for normalized fitness ($nfPl, nfPc$); 2D and 1D matrix for number of violated constraints ($vfPl, vfPc$); 3D and 2D matrix for constraint values ($rfPl, rfPc$);
- **Global best for the $k$-th swarm ($g$) and multi-swarm ($s$)**: 2D and 1D matrix for position ($Pg, Ps$); 1D and 1D matrix for fitness ($fPg, fPs$); 1D and 1D matrix for normalized fitness ($nfPg, nfPs$); 2D and 1D matrix for number of violated constraints ($vfPg, vfPs$); 2D and 1D matrix for constraint values ($rfPg, rfPs$);

Based on the aforementioned structures, the algorithms developed in this paper presents a intercommunication scheme of particles from different swarms during velocity update, which is organized into customized migration and access rules, thus allowing not just a new list of references for the particles, but a dynamic variation between different combinations of components applied to the velocity update equation. More details are presented in Subsection III-C1.
A. CLASSIC PARTICLE SWARM OPTIMIZATION (PSO)

The canonical version of the algorithm developed by Kennedy and Eberhart [1], [61] is a search method organized into clusters in a swarm structure [1], [58], [62]. It initializes (randomly or through preprocessed data) the values for position \((X_{i[j]}(t))\) and velocity \((V_{i[j]}(t))\) in the D-dimensional space for each \(i\)-th particle, where \(D - 1\) is the maximum limit allowed for the dimension index \(j\). \(P\) is the number of particles in a single swarm and \(T\) is the number of iterations. Later on, all solutions (particles) is submitted to fitness evaluation \(f[i]\).

The optimization process consists of adjusting the trajectory based on two references: Best individual position or local best \((P_l_{i[j]}(t))\) and best position obtained by all the particles in the swarm or global best \((P_g_{i[j]}(t))\). At each iteration, the particles are updated by means of Equations 1 and 2.

\[
V_{i[j]}(t+1) = wV_{i[j]}(t) + R_1C_1(P_l_{i[j]}(t) - X_{i[j]}(t)) + R_2C_2(P_g_{i[j]}(t) - X_{i[j]}(t)) \quad (1)
\]

\[
X_{i[j]}(t+1) = X_{i[j]}(t) + V_{i[j]}(t+1) \quad (2)
\]

Variables found in Equations 1 and 2 are defined as follows: Iteration index \((t)\); Inertia weight \((w)\); individual and social acceleration coefficients \((C_1, C_2)\); uniform random number generators distributed in the range between 0 and 1 \((R_1, R_2 = U(0,1))\).

By observing the composition of the velocity update in Equation 1, Figure 4 exemplifies the use of components in the particle movement, highlighting the individual memory (local best \((P_l_{i[j]}(t))\)), collective memory (global best \((P_g_{i[j]}(t))\)), and inertia applied to the velocity component.

\[\text{Algorithm 1: Classic PSO with inertia weight}\]

\[
\text{for } t \leftarrow 0 \text{ to } (P - 1) \text{ do for } i \leftarrow 0 \text{ to } (D - 1) \text{ do}
\]

Evaluate \(i\)-th particle \((\text{Fitness}: f_i, v_{f_i})\);

Obtain the \(i\)-th particle’s local best \((P_l_i, fP_l, v_{P_l})\);

Obtain the swarm’s global best \((P_g, fP_g, v_{P_g})\);

Update inertia weight \(w\):

Apply velocity update \((V_{i[j]}): \text{Eq. 1}\);

Apply velocity clamping \((V_{i[j]}): \text{Eqs. 3 a 6}\);

Apply position update \((X_{i[j]}): \text{Eq. 2}\);

Boundary conditions \((X_{i[j]}): \text{Eqs 7 e 8}\);

Evaluate \(i\)-th particle \((\text{Fitness}: f_i, v_{f_i})\);

Obtain the \(i\)-th particle’s local best \((P_l_i, fP_l, v_{P_l})\);

Obtain the swarm’s global best \((P_g, fP_g, v_{P_g})\);
B. MULTI-SWARM OPTIMIZATION

Multi-swarm models based on cooperative or competitive PSO schemes have been introduced over the years to more efficiently explore the search space [17]. These algorithms use swarms executing in parallel while sharing information or evolving concurrently. Based on the taxonomy developed by El-Abd and Kammel [17], it is possible to establish categories for classification of cooperative PSOs, as well as an extrapolation to competitive models where there is data exchange between swarms.

The application of cooperation and competition structures in PSOs tends to follow the same rationale when compared to multi-population solutions implemented in other search algorithms (e.g. Island GAs), with main differences in the communication methods between the groups of solutions. Cooperative search algorithms were extensively studied in the first decade of the 21st century to effectively solve many large optimization problems, where the basic approach involves the use of one or more search clusters (e.g. swarms, islands) exchanging information in order to explore the search space more efficiently and, consequently, obtain better solutions [5], [17].

In the context of the algorithms FC-CPMSO, FC-CPMSO V1 and FC-CPMSO-V1/V2, four swarm organization models were used as the basis for the proposed topological structure. Each of them has its own information exchange schemes, as well as different search space definition rules. Regarding the main changes of these type of algorithms when compared to the classic version of the PSO [1], [8], the inclusion of new components to the velocity update calculation stands out, by sharing the global best related information through all swarms.

All four approaches described below is classified by [17] as algorithms for use in static optimization\(^{11}\), and can be seen in Figure 5:

1) **Cooperative PSO - Standard e Hybrid** [16]: Two algorithms composed of two different strategies:
   - **Standard (CPSO_S)**: The search space is divided into smaller portions, where each subspace is distributed and optimized by separate swarms. It includes a solution vector which stores the global bests from each swarm;
   - **Hybrid (CPSO_H)**: It uses two optimization stages. It starts with the execution of CPSO_H, followed by a standard PSO, which receives the solution vector resulted from the previous operation;

2) **Multi-Population Cooperative PSO (MCPSO)** [18]: Swarms organized into two cluster layers, where the slave exams do not communicate with each other, by the same time they send their global best values to the master swarm, which will select the fittest among them;

3) **Concurrent PSO (CONPSO)** [64]: An algorithm based on simultaneous optimization, where swarms, executing different types of PSO, shares the global best among them, which increases the search capacity;

In comparison with the multi-swarm approaches previously described, the FC-CPMSO algorithm uses a concurrent structure to transfer data between the swarms, including global bests and counterpart particles, whereas in the FC-CPMSO-V1/V2 versions\(^{12}\), search subspaces are established based on the particle’s trajectory and direction.

---
\(^{11}\)Class of optimization problems with only one global optimum  
\(^{12}\)See Subsection III-D
C. A FULLY-CONNECTED INTER-SWARM APPROACH (FC-CPMSO)

For the purpose of improving performance in the multi-swarm PSO algorithm search, two additional components are included in the particle movement process: 1) Local best values for the set of interconnected particles (inter-swarm neighborhood), or counterpart particles \( (P_{Cij}(t)) \); 2) Global best result achieved by all the swarms \( (P_s(t)) \). The availability of that information during the search process seeks to expand capacity for optimization by accelerating the convergence process, with external references influencing velocity update.

Unlike the master-slave approach proposed by [18] and adapted to other metaheuristics in later works [31], the FC-CPMSO algorithms operates in a competitive multi-swarm environment. In that scenario, the extra components are associated with the information shared between particles from different populations operating competitively, thus excluding the use of master-swarms. Figure 6 illustrates four FC-CMPSO swarms, where each particle is connected with its counterparts located in other swarms.

According to the scheme exposed in Figure 6, all particles located in different swarms with equal label (numbers) share information with each other (i.e., position, velocity, fitness). They compose a sub-topology structure that connect particles from different swarms, providing additional components and references from other clusters. With a view to customization, the proposed algorithms are implemented so as to allow different topologies (e.g., rings, star), at both the swarm and counterpart levels.

Figure 7 illustrates the topological structure present in the proposed algorithms. Each particle is organized in two topologies: Fully-connected between A) all members of its respective swarm (solid black lines) and B) all the counterparts located in other swarms (dotted magenta lines). In both cases, the information is accessed through the members of a topology which is, by design, complete\(^{13}\).

Based on the proposed structure (see 6 and 7), each particle has access to the best results obtained by the other swarms, directly impacting the convergence and capacity for exploring the solution space. The next subsections will explore in details the equations used in FC-CPMSO algorithm for trajectory calculation, which also includes new access rules for each component and customized stochastic functions for inertia weight and acceleration coefficients.

\(^{13}\)By default, both the swarm’s particles and the counterparts are organized in a fully connected topology, similar to the structure found in Gbest PSO [1].
1) The Main Proposed Approach (FC-CPMSO)
The main idea of the FC-CPMSO algorithm consists of adding the aforementioned inter-swarm components (i.e., counterpart particles, global best for all swarms) in an access scheme based on 1) A comparison of Fitness values between components (migration rules, or TMR) and 2) Information (or component) availability during execution of the \( i \)-th particle’s velocity update (probabilistic access, or PAF). In order to establish an easy-to-implement solution, all modifications in FC-CPMSO algorithm are based on conditional rules, which will define both main parameters (i.e. inertia weight, acceleration coefficients) and components (TMR, PAF).

Regarding the use of migration with tolerance threshold, Equations 9 a 14 establish the rules for TMR. Figure 8 shows every possible combination of components using TMR and PAF inside the search space.

- **Particle’s local best** \( fP_{l[i][k]} \), \( vP_{l[i][k]} \) and **counterpart’s local best** \( fP_{c[i]} \), \( vP_{c[i]} \):

  \[
  \lambda_l = \begin{cases} 
  (fP_{l[i][k]} - fP_{c[i]}) / TP_{c[i]} - 1 \times 100 & \text{if } fP_{c[i]} < fP_{l[i][k]} \\
  (fP_{l[i][k]} - fP_{c[i]}) / TP_{c[i]} - 1 \times 100 & \text{if } fP_{l[i][k]} < fP_{c[i]}
  \end{cases} \tag{9}
  \]

  \[
  \theta_l = \begin{cases} 
  1.0 & \text{if } (fP_{l[i][k]} < fP_{c[i]}) \text{ and } (\lambda_l \geq 2\%) \\
  0.5 & \text{if } (fP_{l[i][k]} < fP_{c[i]}) \text{ and } (\lambda_l < 2\%) \\
  0.5 & \text{if } fP_{c[i]} < fP_{l[i][k]} \text{ and } (\lambda_l = 0\%) \\
  0.0 & \text{if } fP_{c[i]} < fP_{l[i][k]} \text{ and } (\lambda_l \geq 2\%)
  \end{cases} \tag{10}
  \]

- **k-th swarm’s global best** \( fP_{g[k]} \), \( vP_{g[k]} \) and **all swarm’s global best** \( fP_{s[0]} \), \( vP_{s[0]} \):

  \[
  \lambda_g = \begin{cases} 
  (fP_{g[k]} - fP_{s[0]}) / TP_{s[0]} - 1 \times 100 & \text{if } fP_{s[0]} < fP_{g[k]} \\
  (fP_{s[0]} - fP_{g[k]}) / TP_{g[k]} - 1 \times 100 & \text{if } fP_{g[k]} < fP_{s[0]}
  \end{cases} \tag{12}
  \]

  \[
  \varphi_l = \begin{cases} 
  1.0 & \text{if } (vP_{g[k]} < vP_{s[0]}) \text{ and } (\varphi_l \leq 2\%) \\
  0.5 & \text{if } (vP_{g[k]} < vP_{s[0]}) \text{ and } (\varphi_l < 2\%) \\
  0.5 & \text{if } vP_{s[0]} < vP_{g[k]} \text{ and } (\varphi_l = 0\%) \\
  0.0 & \text{if } vP_{s[0]} < vP_{g[k]} \text{ and } (\varphi_l \geq 2\%)
  \end{cases} \tag{11}
  \]

Through the migration rules described in Equations 9 to 14, FC-CPMSO algorithm provides a new scheme for velocity update with dynamic behavior regarding access to information from other swarms. The component selection criteria uses the *fitness* values and the number of violations of each solution in order to select which one will be used in the optimization process. One should note that the evaluation methods applied by TMR rules (represented by the variables \( \theta_l \) and \( \varphi_l \)) are adapted for constrained functions, with the number of violations as the primary parameter, followed by the *fitness* values from the memory components.
With TMR, every particle can use up to four components at the same time for the velocity calculation with a competitive or semi-competitive approach\(^\text{14}\). In cases where three or more memory references are used, the values of associated components (i.e. local best and counterparts local best, global best and all swarms’ global best) are reduced by half, and that condition is reached when the difference (or tolerance) between the fitness values of associated components (\(\lambda_t\) or \(\lambda_g\)) is less than 2%.

A second mechanism found in FC-CPMSO is the probabilistic access factors for each component (PAF). Based on the stochastic star topology published by [59], the PAF method defines the probability which a given memory component is available for access in the context of the velocity update calculation. It works by including a binary variable (0 or 1) for each memory components associated with the local and global access factor (i.e. \(\rho_t\), \(\rho_c\), \(\rho_g\), \(\rho_s\)), expanding the number of possible combinations between them (see 8).

Based on the mechanisms described above, Equations 15 to 19 shows the FC-CPMSO’s new scheme for velocity update, which includes the migration rules TMR (\(\varphi_t\), \(\varphi_g\)) and the probabilistic access factor PAF (\(\rho_t\), \(\rho_c\), \(\rho_g\), \(\rho_s\)), both working in a single velocity equation (19).

- **Local best component** (\(cL_i\)):
  \[
  cL_i = \rho_t (\varphi_t (R_t C_t (P_{t[i][j][k]}(t) - X_{t[i][j][k]}(t))))
  \]  
  (15)

- **Counterpart local best component** (\(cL_c\)):
  \[
  cL_c = \rho_c (1 - \varphi_t) (R_c C_c (P_{c[i][j][k]}(t) - X_{t[i][j][k]}(t)))
  \]  
  (16)

- **Global best component** (\(cG_g\)):
  \[
  cG_g = \rho_g (\varphi_g (R_g C_g (P_{g[i][j][k]}(t) - X_{t[i][j][k]}(t))))
  \]  
  (17)

- **All swarms’ global best component** (\(cG_s\)):
  \[
  cG_s = \rho_s (1 - \varphi_g) (R_s C_s (P_{s[i][j][k]}(t) - X_{t[i][j][k]}(t)))
  \]  
  (18)

- **FC-CPMSO’s new velocity update equation**:
  \[
  V_{t[i][j][k]}(t + 1) = wV_{t[i][j][k]}(t) + cL_i + cL_c + cG_g + cG_s
  \]  
  (19)

Variables for Equations 15 to 19 are: Inertia weight (\(w\)); acceleration coefficients for local, counterparts, global and all swarms components (\(C_t, C_c, C_g, C_s\)); references for local and global migration rules (\(\rho_t, \rho_c, \rho_g, \rho_s\)); boolean flags for probabilistic access factors for local, counterpart, global and all swarms’ components (\(R_t, R_c, R_g, R_s\)); Uniform random values between 0 and 1 (\(R_t, R_c, R_g, R_s\)).

For the inertia weights (\(w\)) and acceleration coefficients (\(C_t, C_c, C_g, C_s\), a custom stochastic variation model in a closed interval were selected to be used in combination with FC-CPMSO. Setting the minimum and maximum limits for those variables based on different rules, as well as the procedures (functions) for obtaining the aforementioned factors are important for establish the relevance of each component during the optimization process [65].

Equations 20 to 24 presents the calculation for stochastic inertia weight (\(w\)) applied to velocity update\(^\text{15}\) of the FC-CPMSO algorithm, where the inertia limits \(w_{min} = 0\) and \(w_{max} = 1\), with constant \(T\) representing the maximum number of iterations.

\[
\begin{align*}
  w &= \begin{cases} 
    U(w_{r1}, w_{max}) & \text{if } (T \times 0.00) \leq t < (T \times 0.33) \\
    U(w_{r2}, w_{r3}) & \text{if } (T \times 0.33) \leq t < (T \times 0.66) \\
    U(w_{min}, w_{r0}) & \text{if } (T \times 0.66) \leq t < (T \times 1.00)
  \end{cases} \\
  \quad (20)
\end{align*}
\]

Onde:

\[
\begin{align*}
  w_{r0} &= \begin{cases} 
    w_{min} + (|w_{max} - w_{min}| \times 0.50) & \text{if } U(0, 1) \leq 0.7 \\
    w_{min} + (|w_{max} - w_{min}| \times 0.80) & \text{Otherwise}
  \end{cases} \\
  \quad (21)
\end{align*}
\]

\[
\begin{align*}
  w_{r1} &= \begin{cases} 
    w_{max} - (|w_{max} - w_{min}| \times 0.50) & \text{if } U(0, 1) \leq 0.7 \\
    w_{max} - (|w_{max} - w_{min}| \times 0.80) & \text{Otherwise}
  \end{cases} \\
  \quad (22)
\end{align*}
\]

\[
\begin{align*}
  w_{r2} &= \begin{cases} 
    w_{min} + (|w_{max} - w_{min}| \times 0.20) & \text{if } U(0, 1) \leq 0.7 \\
    w_{min} + (|w_{max} - w_{min}| \times 0.10) & \text{Otherwise}
  \end{cases} \\
  \quad (23)
\end{align*}
\]

\[
\begin{align*}
  w_{r3} &= \begin{cases} 
    w_{max} - (|w_{max} - w_{min}| \times 0.20) & \text{if } U(0, 1) \leq 0.7 \\
    w_{max} - (|w_{max} - w_{min}| \times 0.10) & \text{Otherwise}
  \end{cases} \\
  \quad (24)
\end{align*}
\]

As described by Equations 20 to 24, secondary variables for \(w\)’s minimum and maximum limits (i.e. \(w_{r1}, w_{r2}, w_{r3}\)) are defined based on the current iteration (see Equation 20), with the values being defined based on the condition criteria established by the Equations 21 to 24: Default (if \(U(0, 1) \leq 0.7\)) or expanded (if \(U(0, 1) > 0.7\)). These conditional rules defines the probability of \(w\) occurring outside the standard limit. In the context of this work, the selected threshold value was 0.7.

It’s important to emphasize that such limits has its parameters adjusted to smaller values after the execution of 1/3 of the iterations (see Equation 20), with the inertia weight has been generated stochastically within limits that tends to decrease. Such variation allows a more adequate velocity control considering the algorithm’s convergence behaviour, thus avoiding a deterioration of the algorithm’s exploration or exploitation capacity in the occurrence of successive values of \(w\) close to the minimum and maximum limits, respectively.

To illustrate the proposed stochastic inertia weight procedure, Figure 9 shows the variation of the limits in each iteration interval, as well as an example of \(w\) values obtained by a uniform random number generator. The areas shaded in red, blue and purple correspond to the standard (solid lines) and extended (dotted lines) intervals of iterations (\(t\)).

\(^{14}\)The use of 50% of the total value of a given component is called semi-cooperative. That classification is based on the cooperative approach developed by [18], where extra components are included to the velocity update

\(^{15}\)See Equations 15 to 19
After obtaining the inertia weight \(w\), the next procedure involves setting up the acceleration coefficients (i.e. \(C_l, C_c, C_g, C_s\)), which are obtained through stochastic functions intrinsically dependent on the control factors TMR and PAF. Equations 25 to 28 introduce the calculations of the probabilistic access factor (PAF) for the following components: Local \((\rho_l)\), counterpart \((\rho_c)\), global \((\rho_g)\) and all swarms \((\rho_s)\).

\[
\rho_l = \begin{cases} 
0 & \text{if } U(0, 1) < 0.3 \text{ or } \varphi_l = 0 \\
1 & \text{if } U(0, 1) \geq 0.3 
\end{cases} 
\] (25)

\[
\rho_c = \begin{cases} 
0 & \text{if } U(0, 1) < 0.3 \text{ or } \varphi_l = 1 \\
1 & \text{if } U(0, 1) \geq 0.3 
\end{cases} 
\] (26)

\[
\rho_g = \begin{cases} 
0 & \text{if } U(0, 1) < 0.3 \text{ or } \varphi_g = 0 \\
1 & \text{if } U(0, 1) \geq 0.3 
\end{cases} 
\] (27)

\[
\rho_s = \begin{cases} 
0 & \text{if } U(0, 1) < 0.3 \text{ or } \varphi_g = 1 \\
1 & \text{if } U(0, 1) \geq 0.3 
\end{cases} 
\] (28)

Based on the experiments conducted by Miranda, Keko and Duque [59], communication probability between 0.1 and 0.4 led to better performance than a fixed communication topology equal to 1. In this work, the limit value that defines the presence or absence of components in the velocity update equation is 0.3 (see Equations 25 to 28).

Another important variable associated with PAF is the number of available components by the time when the velocity update is executed. For the FC-CPMSO algorithm, the value of \(N_C\) verifies how many acceleration coefficients \((C_l, C_c, C_g, C_s)\) will be used in the particle’s movement. Equations 29 to 32 establishes the presence (1) or absence (0) of each isolated component \((S_l, S_c, S_g, S_s)\), while Equation 33 calculates the number of accessible components \((N_C)\).

\[
S_l = [\rho_l \varphi_l] 
\] (29)

\[
S_c = [\rho_c (1 - \varphi_l)] 
\] (30)

\[
S_g = [\rho_g \varphi_g] 
\] (31)

\[
S_s = [\rho_s (1 - \varphi_g)] 
\] (32)

\[
N_C = S_l + S_c + S_g + S_s 
\] (33)

The results from the ceiling\(^{16}\) functions in the Equations 29 to 32 ensures that only two results can be obtained from those math operations: 0 or 1. In the case where a TMR variable (i.e. \(\varphi_l, \varphi_g\)) has its value equals to 0.5, the present ceiling function will ensure the rounding of the product to the next integer value: 1.

\(^{16}\)Ceiling function maps a real number \(x\) to the least integer greater than or equal to the number \(x\)
After checking the component’s availability (Equations 29 to 32), the acceleration coefficients are calculated using a custom stochastic function (expressed by the Equations 34 to 37), which is based on uniform random number generators \((U(0, 1))\). In order to establish a suitable parameter combination to the convergence criteria observed by the Poli’s theoretical analysis \([60, 66, 67]\) (see Figure 10), the proposed calculations for the acceleration coefficients \((C_l, C_c, C_g, C_s)\) are designed to minimize the occurrence of values associated with the inertia weight \((w)\) that can generate a divergent behavior in the optimization process.

Considering the limits for the inertia weight (i.e. \(w_{\text{min}} = 0, w_{\text{max}} = 1\)), the sum of values related to acceleration coefficients must not be greater than 2 (see Figure 10). These specifications were adopted in order to reduce the occurrence of divergent combinations (red hatched area). In the context of the FC-CPMSO algorithm, Poli’s theoretical analysis was adapted to deal with four acceleration coefficients instead of two. As a direct result of these modifications, the values of \(C_l, C_c, C_g, C_s\) tend to oscillate between dynamic sub-intervals within the fixed limits of 0 to 1, which are defined according to the number of available components at the time of velocity update calculation.

Therefore, the lower bounds \((C_{l\min}, C_{c\min}, C_{g\min}, C_{s\min})\) and upper bounds \((C_{l\max}, C_{c\max}, C_{g\max}, C_{s\max})\) applied to the acceleration coefficients are defined by the following Equations:

\[
C_l = \begin{cases} 
U(C_{l\min}, 1) & \text{if } S_l = 1 \text{ and } N_C = 1 \\
C_{l\min} + U(0, 1) & \text{if } S_l = 1 \text{ and } N_C = 2 \\
C_{l\min} + U(0, 1) & \text{if } S_l = 1 \text{ and } N_C = 3 \ (34) \\
C_{l\min} & \text{if } S_l = 1 \text{ and } N_C = 4 \\
0 & \text{Otherwise}
\end{cases}
\]

\[
C_c = \begin{cases} 
U(C_{c\min}, 1) & \text{if } S_c = 1 \text{ and } N_C = 1 \\
C_{c\min} + U(0, 1) & \text{if } S_c = 1 \text{ and } N_C = 2 \\
C_{c\min} + U(0, 1) & \text{if } S_c = 1 \text{ and } N_C = 3 \ (35) \\
C_{c\min} & \text{if } S_c = 1 \text{ and } N_C = 4 \\
0 & \text{Otherwise}
\end{cases}
\]

\[
C_g = \begin{cases} 
U(C_{g\min}, 1) & \text{if } S_g = 1 \text{ and } N_C = 1 \\
C_{g\min} + U(0, 1) & \text{if } S_g = 1 \text{ and } N_C = 2 \\
C_{g\min} + U(0, 1) & \text{if } S_g = 1 \text{ and } N_C = 3 \ (36) \\
C_{g\min} & \text{if } S_g = 1 \text{ and } N_C = 4 \\
0 & \text{Otherwise}
\end{cases}
\]

\[
C_s = \begin{cases} 
U(C_{s\min}, 1) & \text{if } S_s = 1 \text{ and } N_C = 1 \\
C_{s\min} + U(0, 1) & \text{if } S_s = 1 \text{ and } N_C = 2 \\
C_{s\min} + U(0, 1) & \text{if } S_s = 1 \text{ and } N_C = 3 \ (37) \\
C_{s\min} & \text{if } S_s = 1 \text{ and } N_C = 4 \\
0 & \text{Otherwise}
\end{cases}
\]
Based on the aforementioned improvements, the particles submitted to the optimization process in the FC-CPMSO algorithm tend not only to perform the search based on variable control parameters at each iteration, but also the structure of the velocity update calculation can vary, according to the availability of the components for access a particle’s \( j \)-th variable (velocity). These features allows that a new combination of solutions can be generated by multiple approaches in a single particle.

Pseudocode 2 presents the FC-CPMSO algorithm. The red and blue lines highlight the changes that differentiate the proposed algorithm from the classic PSO.

**Algorithm 2: FC-CPMSO-V1/V2**

for \( k \leftarrow 0 \) to \((K-1)\) do

  for \( i \leftarrow 0 \) to \((P-1)\) do

    for \( j \leftarrow 0 \) to \((D-1)\) do

      Initialize velocity \( (V_{1/i}^{j/k}) \); Evaluate i-th particle's local best \( (P_{l,i}, f_{P_{l,i}}) \);
      Obtain the counterparts’ local best \( (P_{C_{i}}, f_P, v_P) \); Obtain the k-th swarm’s global best \( (P_{g}, f_{P_{g}}, v_{P_{g}}) \);

      Obtain the all swarm’s global best \( (P_{s}, f_{P_{s}}, v_{P_{s}}) \);

  for \( t \leftarrow 1 \) to \((T-1)\) do

    for \( k \leftarrow 0 \) to \((K-1)\) do

      for \( l \leftarrow 0 \) to \((D-1)\) do

        Tolerance-based Migration Rules TMR \( (\varphi_{2}, \varphi_{3}) \);

        Global migration TMR \( (\varphi_{4}, \varphi_{5}) \); Eq. 12 to 14);\n
        for \( j \leftarrow 0 \) to \((D-1)\) do

          Probabilistic Access Factor PAF \( (\rho_{1}, \rho_{2}, \rho_{3}, \rho_{4}, \rho_{5}) \);

          Access local PAF \( (\rho_{1}) \); Eq. 23);\n
          Access counterpart PAF \( (\rho_{2}) \); Eq. 24);\n
          Access swarm’s global PAF \( (\rho_{3}) \); Eq. 25);\n
          Access all swarm’s global PAF \( (\rho_{4}) \); Eq. 26);\n
          Apply custom stochastic inertia weight \( (w) \); Eq. 20 to 24);\n
          Define Component Availability \( (S_{l}, S_{c}, S_{g}, S_{s}) \);

          Local component \( (S_{l}) \); Eq. 29);\n
          Counterpart component \( (S_{c}) \); Eq. 30);\n
          Swarm’s global component \( (S_{g}) \); Eq. 31);\n
          All swarm’s global component \( (S_{s}) \); Eq. 32);\n
          Number of available components \( (N_{C}) \); Eq. 33);\n
          Get Acceleration Coefficients \( (C_{l}, C_{c}, C_{g}, C_{s}) \);

          Establish lower bounds \( (C_{l}) \); Eq. 34);\n
          Establish upper bounds \( (C_{c}) \); Eq. 35);\n
          Establish upper bounds \( (C_{g}) \); Eq. 36);\n
          Establish upper bounds \( (C_{s}) \); Eq. 37);\n
          Generate variants V1/V2 \( (V_{1/i}^{j/k}) \); Eq. 46 to 54);\n
        Evaluate i-th particle \( (Fitness: f_{i}, v_{f/j}) \);

        Compare V1/V2 with Originals \( (f, v_f, f_j) \); Eq. 17);\n
        Evaluate variants \( (Fitness: f_{j}^{k/1}, v_{f/j}^{k/1}) \);

        Compare \( (f_{j}^{k/1}, v_{f/j}^{k/1}) \); Eq. 18);\n
        if replace \( = \) (true) then \n
          for \( j \leftarrow 0 \) to \((D-1)\) do \n
            Update velocity \( (V_{1/i}^{j/k}) \); Eq. 55);\n
          Obtain the i-th particle’s local best \( (P_{l}, f_{PL}, v_{PL}) \);\n
          Obtain the counterparts’ local best \( (P_{C}, f_{PC}, v_{PC}) \);\n
        Obtain the k-th swarm’s global best \( (P_{g}, f_{P_{g}}, v_{P_{g}}) \);

        Obtain the all swarm’s global best \( (P_{s}, f_{P_{s}}, v_{P_{s}}) \);

The main reason why the stochastic approach was selected over other methods (e.g. linear, nonlinear, self-adaptive, chaotic) is found in the empirical analysis of the inertia weight \( (w) \) conducted by Harrison, Engelbrecht and Ombuki-Berman [60]. In this study, eighteen inertia weight \( (w) \) strategies were evaluated, with the local and global acceleration coefficients \( (C_{l}, C_{g}) \) assuming a constant value \( (C_{l} = C_{g} = 1.496180) \). The results indicates better performance for the stochastic (non-static) and constant (static) methods in the context of the inertia factor [60].

For the acceleration coefficients \( C_{l}, C_{c}, C_{g} \) and \( C_{s} \), the stochastic variations tends to alternate the component’s relevance every time the velocity values are updated. In a scenario with lower values of acceleration coefficient (variation between 0 and 1), a smoother particle movement is expected, and consequently, a lower convergence rate when compared to optimization schemes with high values of \( C_{l}, C_{c}, C_{g} \) and \( C_{s} \). These effects tends to attenuate with combined use of multi-swarm approach with counterpart particles. It is noteworthy that the movement of the particle over short distances allows a more detailed exploration of the search space.

As previously mentioned, the limits are determined in accordance with the theoretical analysis of Poly and applied to the acceleration coefficients \( (0.0 \leq (C_{l}+C_{c}+C_{g}+C_{s}) \leq 2.0) \) and inertia weight \( (0.0 \leq w \leq 1) \), which provides a combination of values that are more likely to converge (see Figure 10).
D. A DIRECTION-BASED CUSTOMIZED BLX SCHEME FOR LOCAL SEARCH (FC-CPMSO-V1/V2)

Seeking to minimize the incidence of stagnation of particles in local optimas, as well as to expand the search capacity in terms of exploration and exploitation, it is proposed the use of variant particles (VP), generated through a direction-based modified version of the BLX crossover. In this model\(^{17}\), called V1 and V2, the subspace search is defined through direction reference \((\nabla X_{[i][j][k]}[t])\) and the difference between current and previous position \((\Delta X_{[i][j][k]}[t])\), both expressed by Equations 46 and 47.

\[
\Delta X_{[i][j][k]} = X_{[i][j][k]}(t + 1) - X_{[i][j][k]}(t) \tag{46}
\]

\[
\nabla X_{[i][j][k]} = \begin{cases} +1 & \text{if } X_{[i][j][k]}(t + 1) > X_{[i][j][k]}(t) \\ 0 & \text{if } X_{[i][j][k]}(t + 1) = X_{[i][j][k]}(t) \\ -1 & \text{if } X_{[i][j][k]}(t + 1) < X_{[i][j][k]}(t) \end{cases} \tag{47}
\]

Each particle can generate up to two variants as offspring \((X'_{[i][j][k]})\), with both assigned for search and optimization in different subspaces. For FC-CPMSO-V1 and FC-CPMSO-V1/V2, the variants generated by modified directed-search crossover are configured as follows (see Figure 11):

- **First Variant (V1):** Establishes a subspace search with focus on exploration, where the limits located in the direction of the gradient \((\nabla X_{[i][j][k]}[t])\) may encompass the entire difference between the current and previous trajectory \((\Delta X_{[i][j][k]}[t])\) based on the current position \((X_{[i][j][k]}(t + 1))\), while in the opposite direction, a smaller percentage of \(\Delta X_{[i][j][k]}[t]\) completes the search region;

- **Second Variant (V2):** Focused on exploitation, the limits located in opposite direction of trajectory \((\nabla X_{[i][j][k]}[t])\) are distributed in a way to encompass positions located between the previous and current position, while in the main direction of the gradient, a reduced percentage of \(\Delta X_{[i][j][k]}[t]\) completes the search region;

The limits for V1 and V2 are calculated using Equations 48 and 49, where \(a_1, a_2, b_1, b_2\) define the ranges for the uniform random number generator \((U(a_1, a_2), U(b_1, b_2))\), which establish the actual location for the upper and lower bounds for the search subspace.

\[
A = X_{[i][j][k]}(t + 1) - (|\Delta X_{[i][j][k]}| \times U(a_1, a_2)) \tag{48}
\]

\[
B = X_{[i][j][k]}(t + 1) + (|\Delta X_{[i][j][k]}| \times U(b_1, b_2)) \tag{49}
\]

When analyzing Equations 48 \& 49, it is found that the main difference between V1 and V2 lies in \(a_1, a_2, b_1, b_2\). A combination of different values for those variables implies in different search behavioir. For the algorithms implemented in the context of this article, the configurations below (defined by the user) illustrate the scheme found in Figure 11, which shows a procedure for exploitation (V2) and exploration (V1).

\[
a_1 = \begin{cases} 0.1 & \text{if } V1 \text{ and } \nabla X_{[i][j][k]} = +1 \\ 0.8 & \text{if } V1 \text{ and } \nabla X_{[i][j][k]} = -1 \\ 0.6 & \text{if } V2 \text{ and } \nabla X_{[i][j][k]} = +1 \\ 0.1 & \text{if } V2 \text{ and } \nabla X_{[i][j][k]} = -1 \end{cases} \tag{50}
\]

\[
a_2 = \begin{cases} 0.3 & \text{if } V1 \text{ and } \nabla X_{[i][j][k]} = +1 \\ 1.0 & \text{if } V1 \text{ and } \nabla X_{[i][j][k]} = -1 \\ 0.9 & \text{if } V2 \text{ and } \nabla X_{[i][j][k]} = +1 \\ 0.3 & \text{if } V2 \text{ and } \nabla X_{[i][j][k]} = -1 \end{cases} \tag{51}
\]

\[
b_1 = \begin{cases} 0.8 & \text{if } V1 \text{ and } \nabla X_{[i][j][k]} = +1 \\ 0.1 & \text{if } V1 \text{ and } \nabla X_{[i][j][k]} = -1 \\ 0.1 & \text{if } V2 \text{ and } \nabla X_{[i][j][k]} = +1 \\ 0.6 & \text{if } V2 \text{ and } \nabla X_{[i][j][k]} = -1 \end{cases} \tag{52}
\]

\[
b_2 = \begin{cases} 1.0 & \text{if } V1 \text{ and } \nabla X_{[i][j][k]} = +1 \\ 0.3 & \text{if } V1 \text{ and } \nabla X_{[i][j][k]} = -1 \\ 0.3 & \text{if } V2 \text{ and } \nabla X_{[i][j][k]} = +1 \\ 0.9 & \text{if } V2 \text{ and } \nabla X_{[i][j][k]} = -1 \end{cases} \tag{53}
\]

After the calculation of search subspace limits \(A\) and \(B\), the new position values associated to the variant particles (V1 and/or V2) are obtained through the Equation 54.

\[
X_{[i][j][k]}(t + 1) = U(A, B) \tag{54}
\]
In evolutionary algorithms based on BLX-α crossover recombination, two individuals generate offspring based on their alleles, which also provides the search limits. In the case of FC-CPMSO-V1/V2 algorithms, the differences between the previous \( \Delta X[i][j][k](t) \) and current \( X[i][j][k](t+1) \) position, as well as the direction data \( \Delta X[i][j][k] \), will define the lower and upper boundaries. In short, the position values of the "parents" used to generate V1 and V2 are selected within the range established by Equation 54.

The fitness\(^{18}\) associated to the variant particles (VPs) are compared with those of their respective original versions. If V1 or V2 obtains better results than the original particle, this one will be discharged and replaced by the fittest variant. This new particle will then have its velocity values recalculated based on Equation 55, having its trajectory modified in relation to its previous location \( X[i][j][k](t) \).

\[
V[i][j][k](t+1) = X[i][j][k](t+1) - X[i][j][k](t)
\]

The use of local search based on directed BLX crossover, allows a refined search (exploitation) in a subspace closer to the current particle’s position (Second Variant (V2)). Concurrently, the information associated to particle’s displacement \( \Delta X[i][j][k] \) is used to establish a search region in the gradient direction (First Variant (V1)). In both cases, there’s a small probability in which V1 and V2 (Figure 11) get values opposite the specified trend, which allows for a wider range of possible combinations.

By observing Algorithm 2 (in blue), the use of local search methods V1/V2 in the context of the PSO algorithm includes a direction-based crossover stage with up to three solutions per particle (original, V1, V2), followed by fitness comparison procedures, thus providing a local search engine with low impact on computational complexity.

**IV. IMPLEMENTATION, EXPERIMENTS AND RESULTS**

To perform benchmark tests and assess the impact of the proposed mechanisms, we evaluate the proposed algorithms on ten engineering constrained functions that are widely found in the scientific literature ( [68]–[74]).

For optimization problems, a common attribute present in the engineering field is the [75] constraints. They are associated with the limits that the value of a variable can have or the relationship between them. Thus, a global optimization problem with constraints is formalized based on the Equations 56 [76]:

\[
\begin{align*}
\text{minimize or maximize} & \quad f(x) \\
\text{subject to} & \quad l_i \leq x_i \leq u_i, \quad i = 1, \ldots, D, \\
& \quad g(x) \leq 0, \\
& \quad h(x) = 0,
\end{align*}
\]

\(^{18}\)In the case of constrained optimization problems, the evaluation is made based on the following priority order: 1) Number of constraint violations; 2) fitness value (minimization or maximization).

Where:

\[
f(x) : \mathbb{R}^n \rightarrow \mathbb{R}, \quad g(x) : \mathbb{R}^n \rightarrow \mathbb{R}^m, \quad h(x) : \mathbb{R}^n \rightarrow \mathbb{R}^p.
\]

Another important stage in particle’s evaluation is the fitness normalization \( n_f[i][k] \). This process is done according to Deb’s Rule [54], where violated solutions are converted into a single value, obtained through a penalty function. Equations 57 to 61 show the normalization process, where \( E \) represents the set of solutions without violations (feasible), and \( fP y[k] \) is the worst viable\(^{19}\) real fitness global value.

\[
rf[i][r][k] = g_r(x) \quad (57)
\]

\[
G_r = \begin{cases} 
rf[i][r][k] & \text{if } rf[i][r][k] \notin E \\
0 & \text{if } rf[i][r][k] \in E
\end{cases} \quad (58)
\]

\[
vG_r = \begin{cases} 
1 & \text{if } rf[i][r][k] \notin E \\
0 & \text{if } rf[i][r][k] \in E
\end{cases} \quad (59)
\]

\[
vf[i][k] = \sum_{r=0}^{R-1} vG_r 
\]

\[
nf[i][k] = \begin{cases} 
rf[i][k] & \text{if } vf[i][k] = 0 \\
fP y[k] + \sum_{r=0}^{R-1} G_r & \text{if } vf[i][k] > 0
\end{cases} \quad (61)
\]

Six algorithms were implemented in the context of this paper. Five of them are associated with the proposed mechanisms and improvements (FC-CPMSO, FC-CPMSO-V1, FC-CPMSO-V1/V2, MSO-ST-V1, MSO-ST-V1/V2), and one used only for comparative purposes (MSO-ST). The previously proposed stochastic procedures for inertia weight and acceleration coefficients are equally used in all tested algorithms. The following list describe the multi-swarm PSO approaches used for the experiments:

- **MSO-ST**: No data exchange during the optimization process. Only concurrent execution;
- **MSO-ST-V1**: No data exchange during the optimization process, with single directed-based crossover variant used for local search (V1);
- **MSO-ST-V1/V2**: Without exchanging data during optimization, with two directed-based crossover variants used for local search (V1 and V2);
- **FC-CPMSO**: Inter-swarm topology (counterparts and global). Only concurrent execution;
- **FC-CPMSO-V1**: Inter-swarm topology (counterparts and global) with single directed-based crossover variant used for local search (V1);
- **FC-CPMSO-V1/V2**: Inter-swarm topology (local and global) with two directed-based crossover variants used for local search (V1 and V2);

\(^{19}\)A feasible solution is defined as any one that does not have violations. If there’s only unfeasible particles in the swarm, the set of solutions with the lowest number of violations will accepted as the fittest. This definition is in line with the norms established by Deb’s Rule [51].
All six algorithms previously described were implemented using parallel computing in CUDA architecture, where each particle is executed by a Thread, while each swarm is organized as a Blocks of threads, which make up a set of swarms, represented by a Grid structure. Figure 12 illustrates a structure applied to a PSO algorithm into a single-dimensional CUDA grid.

In this parallel model used by FC-CPMSO and FC-CPMSO-V1/V2 algorithms, each particle is executed concurrently in a synchronous approach, using the data from other swarms that were obtained in the previous iteration (t). In terms of the data organization in threads, each instance has access to exclusive indexes in parallel, so as to avoid race conditions, overwriting and data loss. Figure 13 illustrates the organization of vectors for each thread.

Observing the structure of the PSO algorithms, it is possible to apply the complete parallelization model on GPUs described by Tan [5], thus, the swarm initialization and optimization occurs without using the CPU for tasks directly associated with the PSO. A second grid with only one block is subsequently executed to obtain the local counterpart particles and the best result achieved by the multiswarm approach.

The parameters used in the tests were selected for the purposes of boosting performance for all six algorithms. Those configurations may be verified in Table 1.

| Parameters                                | Value     |
|-------------------------------------------|-----------|
| Number of Experiments                     | 60        |
| Number of Iterations                      | T = 140   |
| Number of Particles per Swarms            | P = 100   |
| Number of Swarms                          | K = 4     |
| Number of Evaluations per Swarm           | 42000     |
| Weighted Constant (Minimum Velocity)      | β = 0.5   |
| Limits applied to the Inertia factor (w)  | Min (w_min = 0) | Max (w_max = 1) |

Tests were conducted in a machine with the following hardware configuration: Apple MacBook Pro mid-2012 with NVIDIA GeForce GT 650M GPU 1024 MB GDDR5 VRAM with 900 MHz clock and 384 CUDA Colors; Intel Core i7 "Ivy Bridge" 2.9 GHz CPU; 16 GB DDR3L RAM clocked at 1600 MHz. The operating system used was macOS Sierra 10.12.6, with CUDA driver version 9.1.

Regarding the ten constrained optimization problems (COPs) selected for evaluation and comparative analysis, each one of them is described as follows (see Figure 14):

- **Three Bar Truss (TBT):** Optimization of a two-dimensional function representing the volume subject to stress constraints on each side of the lattice [68];
- **Himmelblau Nonlinear Optimization (HNO1, HNO2):** Non-linear maximization problem widely used in the literature ([51], [68], [77], [78]);
- **Tubular Column Design (TCD):** Design a uniform tabular section column at a minimum cost. This is a two-dimensional problem [68];
- **Design of a Pressure Vessel (DPV1, DPV2):** Minimizing the total cost of the project, including cost of material, molding and welding of a cylindrical container limited at its extremities [79]–[83];
- **Speed Reducer Design (SRD11):** Design of a single gearbox associated with a velocity control system [84];
- **Welded Beam Design (WBD1, WBD2):** In order to find the minimum cost of fabricating a welded beam, this problem consists of four design variables [68];
- **Weight of Tension/Compression Spring (MWTCS):** Minimization of the weight of a tension/compression spring. The problem features four constraints describing the minimal deflection, shearing tension, surge frequency and external diameter limits [85], [86].

More details about the ten COPs used for benchmark, including their mathematical functions, can be found in the authors' previous work: Lisboa et al. [29].

21Regarding the minimum and maximum limits para for the acceleration coefficients (i.e. \( C_1, C_2, C_3, C_4 \)), Equations 38 to 45 describe achievement of those limits.

22Interval variables applied in the context of modified BLX crossover for local search in V1 and V2 (i.e. \((a_1, a_2, b_1, b_2)\) are defined arbitrarily by the user. The values used in the tests are found in Equations 50 to 53.
FIGURE 14. Technical schematics for six constrained engineering problems (WBD1, WBD2, DPV1, DPV2, MWTC, SRD, TCD, TBT).
## TABLE 2. Results - DPV1, DPV2, Himmelblau 1 (HNO1), Himmelblau 2 (HNO2), MWTCS

| Method      | DPV1          | Mean       | Median     | Std. Dev. | Best       | Worst      |
|-------------|---------------|------------|------------|-----------|------------|------------|
| HNO1 - Maximization |                |            |            |           |            |            |
| PSO         | MSO-ST-V1     | 30655.538204 | 30665.538384 | 5.29023e-04 | 30665.538204 | 30665.53709 |
|             | MSO-ST-V1/V2  | 30655.538635 | 30665.538661 | 6.90691e-05 | 30665.538681 | 30665.538340 |
|             | MSO-ST        | 30655.504982 | 30665.533647 | 1.77161e-01 | 30665.537456 | 30665.510194 |
|             | FC-CPSMO-V1   | 30655.538731 | 30665.538731 | 1.38865e-08 | 30665.538731 | 30665.538731 |
|             | FC-CPSMO-V1/V2| 30655.538731 | 30665.538731 | 3.34918e-09 | 30665.538731 | 30665.538731 |
|             | FC-CPSMO     | 30655.538730 | 30665.538730 | 7.46105e-08 | 30665.538731 | 30665.538730 |
| Others      | He et al. [77]| 30643.989   | NA         | 7.0043e+01 | 30655.539   | NA         |
|             | Dimopoulos [78]| NA        | NA         | 30665.54   | NA         | NA         |
|             | Gandomi et al. [84]| NA   | NA         | 30665.2327 | NA         | NA         |
|             | Garg [82]     | 30665.40461198 | 30665.493896 | 2.338366e-01 | 30665.56605465 | 30664.6269625 |
|             | Pereira et al. [29]| 30363.7   | 30348.1   | 8.0065e+01 | 30366.7   | NA         |
| HNO2 - Maximization |                |            |            |           |            |            |
| PSO         | MSO-ST-V1     | 31025.557005 | 31025.558038 | 4.04865e-03 | 31025.558322 | 31025.531303 |
|             | MSO-ST-V1/V2  | 31025.558853 | 31025.559234 | 1.24997e-03 | 31025.559687 | 31025.552046 |
|             | MSO-ST        | 31025.427667 | 31025.502972 | 2.56651e-01 | 31025.552101 | 31025.786965 |
|             | FC-CPSMO-V1   | 31025.559753 | 31025.559753 | 6.00263e-09 | 31025.559753 | 31025.559753 |
|             | FC-CPSMO-V1/V2| 31025.559753 | 31025.559753 | 1.79940e-09 | 31025.559753 | 31025.559753 |
|             | FC-CPSMO     | 31025.559753 | 31025.559753 | 3.71878e-08 | 31025.559753 | 31025.559753 |
| Others      | Pereira et al. [29]| 30901.5   | 30903.9   | 4.2102e+01 | 30107.3   | 30804.1    |
|             | Coello [93]   | NA         | NA         | 30810.359 | NA         | NA         |
|             | Coello [94]   | 30984.240703 | 31017.21369 | 7.36335e+01 | 31020.859 | 30792.407737 |
|             | Garg [82]     | 31025.55841263 | 31025.5621911 | 1.53528e+02 | 31025.57569195 | 31025.49205458 |
|             | Garg [95]     | 31025.557816 | 31025.561141 | 1.526e-02  | 31025.574717 | 31025.492054 |

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3119579, IEEE Access
### TABLE 3. Results - SRD11, TBT, TCD, WBD1, WBD2

| SRD11 | Mean      | Median    | Std. Dev. | Best      | Worst     |
|-------|-----------|-----------|-----------|-----------|-----------|
| PSO   | 2894.382128 | 2894.382128 | 2.31525e-07 | 2894.382128 | 2894.382129 |
|       | 2894.382127 | 2894.382127 | 7.35161e-08 | 2894.382127 | 2894.382128 |
|       | 2894.382130 | 2894.382130 | 9.11337e-07 | 2894.382129 | 2894.382133 |
|       | 2894.382127 | 2894.382127 | 1.20533e-09 | 2894.382127 | 2894.382127 |
|       | 2894.382127 | 2894.382127 | 3.28188e-10 | 2894.382127 | 2894.382127 |
|       | 2894.382127 | 2894.382127 | 5.72340e-09 | 2894.382127 | 2894.382127 |
| Others| 2894.382127 | 2894.382127 | 7.10543e-15 | 2894.382127 | 2894.382127 |

| TBT   | Mean      | Median    | Std. Dev. | Best      | Worst     |
|-------|-----------|-----------|-----------|-----------|-----------|
| PSO   | 263.895885 | 263.895880 | 2.93174e-05 | 263.895851 | 263.896005 |
|       | 263.895859 | 263.895856 | 1.06576e-05 | 263.895845 | 263.895892 |
|       | 263.895952 | 263.895920 | 1.00962e-04 | 263.895854 | 263.896449 |
|       | 263.895845 | 263.895844 | 2.29060e-06 | 263.895843 | 263.895853 |
|       | 263.895844 | 263.895844 | 1.00901e-06 | 263.895843 | 263.895848 |
|       | 263.895848 | 263.895845 | 6.42342e-06 | 263.895843 | 263.895879 |
|       | 264.067946 | NA        | 9.001e-05   | 263.971560 | NA        |
| Others| 263.895843 | 263.895843 | 9.001e-00   | 263.895843 | 263.895843 |

| TCD   | Mean      | Median    | Std. Dev. | Best      | Worst     |
|-------|-----------|-----------|-----------|-----------|-----------|
| PSO   | 26.531327  | 26.531327 | 6.08374e-12 | 26.531327 | 26.531327 |
|       | 26.531327  | 26.531327 | 4.99822e-13 | 26.531327 | 26.531327 |
|       | 26.531327  | 26.531327 | 7.14114e-11 | 26.531327 | 26.531327 |
|       | 26.531327  | 26.531327 | 7.08935e-15 | 26.531327 | 26.531327 |
|       | 26.531327  | 26.531327 | 1.80211e-12 | 26.531327 | 26.531327 |
| Others| 26.535084  | NA        | 1.930e-03   | 26.53217   | 26.53972 |
|       | NA        | NA        | NA        | 26.5323    | NA        |
|       | NA        | NA        | NA        | 26.5316    | NA        |
|       | 26.531332  | 26.531330 | 3.94e-04    | 26.53128   | 26.53515 |
|       | 26.5448    | 26.5434   | 1.05607e-02 | 26.5315    | 26.5849 |

| WBD1  | Mean      | Median    | Std. Dev. | Best      | Worst     |
|-------|-----------|-----------|-----------|-----------|-----------|
| PSO   | 2.382944  | 2.381080  | 8.49064e-04 | 2.3809567 | 2.446189 |
|       | 2.381317  | 2.380985  | 6.81497e-04 | 2.3809567 | 2.384135 |
|       | 2.395980  | 2.383267  | 2.58112e-02 | 2.3809568 | 2.514657 |
|       | 2.380957  | 2.380957  | 1.91808e-09 | 2.3809567 | 2.380957 |
|       | 2.380957  | 2.380957  | 4.28943e-10 | 2.3809567 | 2.380957 |
|       | 2.380957  | 2.380957  | 3.80576e-09 | 2.3809567 | 2.380957 |
| Others| 2.38552   | 2.38538   | 1.90285e-03 | 2.3821692  | 2.39076 |
|       | 2.38119489 | NA        | 3.85388e-04 | 2.3809573 | NA        |
|       | 2.38108932 | NA        | 1.01227e-04 | 2.3809961 | NA        |
|       | 2.38117861 | NA        | 2.381134   | NA        | NA        |
|       | 2.3794755  | 2.3786824 | 1.18226e-04 | 2.3809597 | 2.380147 |

| WBD2  | Mean      | Median    | Std. Dev. | Best      | Worst     |
|-------|-----------|-----------|-----------|-----------|-----------|
| PSO   | 1.695257  | 1.695248  | 6.94261e-05 | 1.695247 | 1.695790 |
|       | 1.695856  | 1.695247  | 3.24592e-03 | 1.695247 | 1.715355 |
|       | 1.702840  | 1.695247  | 2.09371e-02 | 1.695290 | 1.824886 |
|       | 1.695247  | 1.695247  | 1.70439e-10 | 1.695247 | 1.695247 |
|       | 1.695247  | 1.695247  | 4.06796e-11 | 1.695247 | 1.695247 |
|       | 1.695247  | 1.695247  | 7.56532e-10 | 1.695247 | 1.695247 |
| Others| 1.724852  | 1.724852  | 3.730e-12    | 1.724852  | 1.724852 |
|       | 1.9648    | NA        | NA        | 1.7315     | NA        |
|       | 1.695247  | NA        | 2.190e-09   | 1.695247  | 1.695247 |
|       | 1.69526436 | NA        | 5.264e-05   | 1.695247  | 1.695247 |
|       | 1.69530842 | NA        | 2.840e-05   | 1.6952638 | NA        |
TABLE 4. Ranking Table – Mean and Best Result

| Algorithms        | DPV1 | DPV2 | HNO1 | HNO2 | MWTC1 | SRD11 | TBT | TCD | WBD1 | WBD2 | Best Rank |
|-------------------|------|------|------|------|-------|-------|-----|-----|------|------|-----------|
| MSO-ST-V1         | 4    | 4    | 5    | 5    | 5     | 5     | 5   | 5   | 4    | 4    | 4.7       |
| MSO-ST-V1/V2      | 5    | 5    | 4    | 4    | 4     | 4     | 4   | 4   | 4    | 5    | 4.3       |
| MSO-ST            | 6    | 6    | 6    | 6    | 6     | 6     | 6   | 6   | 6    | 6    | 6         |
| FC-CPMSO-V1       | 2    | 1    | 2    | 2    | 2     | 2     | 2   | 2   | 2    | 2    | 1.9       |
| FC-CPMSO-V1/V2    | 1    | 2    | 1    | 1    | 1     | 1     | 1   | 1   | 1    | 1    | 1.1       |
| FC-CPMSO          | 3    | 3    | 3    | 3    | 3     | 3     | 3   | 3   | 3    | 3    | 3         |

RANKING: MEAN

| Algorithms        | DPV1 | DPV2 | HNO1 | HNO2 | MWTC1 | SRD11 | TBT | TCD | WBD1 | WBD2 | Best Rank |
|-------------------|------|------|------|------|-------|-------|-----|-----|------|------|-----------|
| MSO-ST-V1         | 4    | 4    | 5    | 5    | 5     | 5     | 5   | 5   | 5    | 5    | 4.8       |
| MSO-ST-V1/V2      | 5    | 5    | 4    | 4    | 4     | 4     | 4   | 4   | 4    | 4    | 4.2       |
| MSO-ST            | 6    | 6    | 6    | 6    | 6     | 6     | 6   | 6   | 6    | 6    | 6         |
| FC-CPMSO-V1       | 3    | 2    | 2    | 2    | 2     | 2     | 2   | 2   | 2    | 2    | 2.1       |
| FC-CPMSO-V1/V2    | 1    | 1    | 1    | 1    | 1     | 1     | 1   | 1   | 1    | 1    | 1         |
| FC-CPMSO          | 2    | 3    | 3    | 3    | 3     | 3     | 3   | 3   | 3    | 3    | 2.9       |

RANKING: BEST

Ranking of the best observed results for mean and best fitness statistic for the proposed mechanisms (i.e. counterpart particles, migration rules with degree of restrictive tolerance, probabilistic access factor, generation of variant particles by crossover BLX based on direction, tolerance thresholds for inertia weight and acceleration coefficient limits) is displayed in Table 4 (mean rank and best results). In it, we can observe that the combination of all resources (FC-CPMSO-V1/V2) provided the best results in nine functions on mean (i.e., DPV1, MWTCS, WBD1, WBD2, HNO1, HNO2, SRD11), and the best fitness values in all constrained problems. In the case of DPV2, the FC-CPMSO-V1 algorithm has archived the best results in terms of mean and median.

Regarding the algorithms that did not use counterpart particles to exchange information between swarms (i.e., MSO-ST-V1, MSO-ST-V1/V2, MSO-ST), the results point to an improvement in performance detected in all evaluation criteria, specially those which uses the directed-based BLX crossover (MSO-ST-V1, MSO-ST-V1/V2r. Despite the good results for the MSO-ST-V1/V2 algorithm, which uses two variant particles, a less significant performance in three benchmark functions were observed, with the MSO-ST-V1 algorithm obtaining the best results in the following evaluation criteria: Mean (WBD2), median (DPV2), and worst (DPV1, DPV2, WBD2) fitness value obtained in 60 executions.

Finally, it is important to emphasize that, with the exception of WBD2, HNO1 and HNO2, the proposed algorithms outperform the methods published in the scientific literature and used for comparison. That statement is made based on the best results found during the experiments.

1) Statistical Analysis (p-Test)
Non-parametric tests were performed using the data collected from the experiments. For the p-test calculation, the Mann-Whitney U method was selected, where the statistical significance (non-equivalence) of one algorithm in relation to the other is verified whether \( p < 0.05 \), otherwise, both solutions are equivalent with no statistical difference; therefore, the null hypothesis (in bold, see Table 6) must be considered.

The data displayed Tables 3 was obtained from the mean of the best observed fitness of each run, with all algorithms being executed with the same set of initial populations. Based on the results collected, statistical relevance for FC-CPMSO-V1/V2 algorithms can be observed when compared with FC-CPMSO-V1 (single variant particle) in 7 of 10 engineering functions. For the rest, p-values less than 0.05 is detected in all evaluations.

P-values for MSO-ST-V1/V2 algorithm indicates statistical relevance in 8 optimization functions (i.e., HNO1, HNO2, SRD11, WBD1, WBD2, MWTCS, TBT, TCD), and in the constrained functions DPV1 and DPV2, no statistical significance were achieved when comparing MSO-ST-V1/V2 MSO-ST-V1/V2 (two variant particles) and MSO-ST-V1 (one variant particle). In most cases, based on the tests performed, it is possible to conclude that the use of counterpart particles together with rule-based access procedures (i.e., TMR, PAF) significantly improves the performance, with the use of modified crossover BLX methods providing an additional optimization layer.

---

23In Result Tables 2 and 3, these algorithms are identified as "Others"
### TABLE 5. Mann-Whitney U test- DPV1, DPV2, Himmelblau 1 (HNO1), Himmelblau 2 (HNO2), MWTCS

|        | FC-CPMSO-V1/V2 | FC-CPMSO | MSO-ST-V1 | MSO-ST-V1/V2 | MSO-ST |
|--------|----------------|----------|-----------|--------------|--------|
| HNO1   | 1.05948e-17    | 1.22986e-20 | 1.77282e-21 | 1.77282e-21 | 1.77282e-21 |
|        | -              | 1.77801e-21 | 1.77801e-21 | 1.77801e-21 | 1.77801e-21 |
|        | -              | -         | 1.77829e-21 | 1.77829e-21 | 1.77829e-21 |
|        | -              | -         | -          | 2.49436e-16 | 4.81886e-21 |
| HNO2   | 1.91207e-18    | 5.32e-21  | 1.77801e-21 | 1.77801e-21 | 1.77801e-21 |
|        | -              | 1.77801e-21 | 1.77801e-21 | 1.77801e-21 | 1.77801e-21 |
|        | -              | -         | 1.77829e-21 | 1.77829e-21 | 1.77829e-21 |
|        | -              | -         | -          | 1.81106e-10 | 4.36378e-21 |
| WBD1   | 8.08578e-19    | 1.41334e-19 | 1.77773e-21 | 1.77773e-21 | 1.77773e-21 |
|        | -              | 1.77467e-21 | 1.77467e-21 | 1.77467e-21 | 1.77467e-21 |
|        | -              | -         | 1.77745e-21 | 1.77745e-21 | 1.77745e-21 |
|        | -              | -         | -          | 4.17091e-10 | 3.54953e-13 |
| WBD2   | 1.18292e-01    | 1.44783e-03 | 1.38121e-19 | 4.73617e-17 | 5.04340e-20 |
|        | -              | 2.51753e-12 | 2.51753e-12 | 2.51753e-12 | 2.51753e-12 |
|        | -              | -         | 5.57891e-03 | 5.57891e-03 | 5.57891e-03 |
|        | -              | -         | -          | 4.63478e-08 |           |
| TBY    | 1.44783e-03    | 1.38955e-02 | 2.52389e-21 | 3.81107e-18 | 1.38955e-02 |
|        | -              | 2.56728e-05 | 2.56728e-05 | 2.56728e-05 | 2.56728e-05 |
|        | -              | 2.51753e-12 | 2.51753e-12 | 2.51753e-12 | 2.51753e-12 |
|        | -              | -         | 9.7056e-12  | 9.7056e-12  | 9.7056e-12  |
|        | -              | -         | -          | 6.28765e-07 | 1.92382e-16 |
| TCD    | 1.84060e-01    | 2.17857e-24 | 2.77948e-24 | 9.02178e-10 | 2.84906e-24 |
|        | -              | 2.17857e-24 | 2.17857e-24 | 2.17857e-24 | 2.17857e-24 |
|        | -              | 1.05149e-20 | 1.05149e-20 | 1.05149e-20 | 1.05149e-20 |
|        | -              | -         | 3.99367e-22 | 3.99367e-22 | 3.99367e-22 |
|        | -              | -         | -          | 4.07519e-22 |           |
| DPV1   | 6.37855e-03    | 8.40688e-03 | 4.87625e-12 | 1.7371e-12  | 5.43591e-15 |
|        | -              | 2.0556e-06 | 2.0556e-06 | 2.0556e-06 | 2.0556e-06 |
|        | -              | 3.07035e-08 | 3.07035e-08 | 3.07035e-08 | 3.07035e-08 |
|        | -              | -         | 2.81439e-03 | 2.81439e-03 | 2.81439e-03 |
| DPV2   | 2.88088e-01    | 4.17971e-05 | 1.06684e-15 | 7.92999e-16 | 2.19643e-18 |
|        | -              | 2.3884e-04 | 2.3884e-04 | 2.3884e-04 | 2.3884e-04 |
|        | -              | 7.33695e-07 | 7.33695e-07 | 7.33695e-07 | 7.33695e-07 |
|        | -              | -         | 3.24912e-01 | 3.24912e-01 | 3.24912e-01 |
|        | -              | -         | -          | 2.84695e-06 |           |
2) Graphs (Convergence)

The following graph illustrate the behavior of the mean of all runs for assessment of convergence throughout the iterations for each algorithm, exposed in Table 4. The data collected by the experiments were normalized using Equations 57 to 61 in order to adjust results with violations to the plots. Equation 62 is used to generate the convergence lines displayed in Figure 15.

- Mean of the global best (Figure 15): Calculates the mean of the best results of each experiment in iteration \( t \). Variables from the 62: Global best’s fitness \( f_g \) at iteration \( t \) associated to experiment \( e \) \( \left( e fPg[k][t] \right) \); Number of experiments \( (E) \); experiment index \( (e) \).

\[
Mean[e] = \frac{1}{E} \sum_{e=0}^{E-1} e fPg[k][t] \quad (62)
\]

Based on the graphs found in Figure 13, it is possible to establish the following affirmations:

1) The algorithms that used the counterparts’s inter-swarm topology (i.e. FC-CPMSO, FC-CPMSO-V1, FC-CPMSO-V1/V2) were the ones that obtained the best fitness values in all benchmark functions. It was also verified that the use of counterpart particles \( (Pc) \) and multi-swarm’s global best \( (Ps) \) increased the convergence speed in all constrained optimization.

2) Modest improvements were observed for the algorithms FC-CPMSO-V1 and FC-CPMSO-V1/V2 when comparing them to FC-CPMSO (without variant particles) in six optimization functions (i.e., WBD1, WBD2, SRD11, TCD, HNO1, HNO2), while the difference in the remaining functions has been more significant (i.e., MWTCS, DPV1, DPV2, TBT). Based on the collected data, it is possible to infer that the information exchange among swarms through the TMR and PAF methods affects the performance of the modified BLX crossover \( (V1, V2) \), given the fact that both mechanisms tend to increase exploration in space search, which may affect the exploitation aspect of the VPs;

3) A comparative analysis regarding FC-CPMSO-V1 and FC-CPMSO-V1/V2 convergence behaviour shows few differences in performance with one or two variant particles. With exception of DPV1 and TBT, the optimization pattern is almost similar (i.e., SRD11, HNO2) for both algorithms. In other cases (i.e., DPV2), the use of a single variant particle \( (V1) \) outperformed the results obtained with two variant particles \( (V1/V2) \).

V. FINAL REMARKS AND CONCLUSION

In this article, a set of new mechanisms were introduced in order to establish a sub-topology for information exchange between swarms (i.e., PAF, TMR), also including a gradient-based crossover for use as local search. In the first case, the objective was the inclusion of components from other swarms during velocity updates, in order to amplify the search capacity by incorporate new references for particle’s trajectory, whether from local information (counterpart particles) or global information (best particle among all swarms). For the second case, alternative particles (variants) generated by directed-base modified crossover is restricted only to a local search, in which particles \( V1 \) and \( V2 \) are generated based on exploration and exploitation schemes.

By considering the other implementations of additional components in the velocity update stage \([17, 44, 57, 58]\), the counterpart particles approach is a novel and improved option for data exchange between swarms, where different particle (or combinations) connected by an inter-swarm sub-topology, is shared among swarms by a set of rules (PAF, TMR), thus enhancing the algorithm’s exploration capabilities in the search space. Given the fact that information from other swarms (i.e. counterpart particles, all swarms’ global best) were obtained in previous iteration, a copy of those references are distributed for all swarms, which limits the impact on the computational complexity by the proposed mechanisms.

The main reason why the BLX crossover was selected as the foundation for the proposed method \( (V1/V2) \) is the fact that BLX is a simple – yet efficient – approach, which...
FIGURE 15. Convergence graphs associated with engineering problems.
presented good results in previous experiments when applied to constrained optimization problems (COPs) [29]. It is important to emphasize that, as in FC-CPMSO, the use of V1/V2 had a low impact on computational complexity.

Experiment data suggests viability and significant improvements in the MSO algorithms enhanced with inter-swarm topology structure based on the TMR and PAF methods, with best fitness values obtained in 8 out of 10 instances of constrained problems. With respect to the directed-based BLX crossover (V1, V2), despite the expressive impact when used with MSOs that don’t use counterpart particles and all swarms global best (i.e., MSO-ST-V1, MSO-ST-V1/V2), an unsatisfactory convergence behaviour was noticed when comparing FC-CPMSO-V1 and FC-CPMSO-V1/V2. Two possible solutions includes the use of additional trajectory information (e.g. particle’s direction tendency in a interval of iterations) and a new combination of boundary parameters (see Equations 50 to 53) in order to establish new search strategies that increases search capacities for both exploration and exploitation.

Even through the aforementioned results, additional mechanisms for improving optimization performance in FC-CPMSO and MSO-ST algorithms are in development. Some of these mechanisms includes a hyperheuristic-based parameter tuning (e.g. inertia weight, acceleration coefficients, boundaries for search subspaces) which uses information related to the swarm’s state, such as diversity and convergence counter. Future works includes the improvement of the mechanisms proposed in this article. We can highlight some upgrades regarding the multi-swarm structure, such as the inclusion of a master-slave hierarchy with modified rules for TMR and PAF. For the variant particles and its directed-based crossover, the addition of a game-based crossover scheme (i.e. GBX2\(^{27}\))

Finally, a multi-swarm adaptation of Quantum-Bevalued Particle Swarm Optimization (QPSO) enhanced with PAF, TMR and directed-based crossover is in development, and the results will be released in a future publication.

REFERENCES

[1] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in Proceedings of the IEEE Int. Conf. on Neural Networks, IEEE. IEEE Press, 1995, pp. 1942–1948.

[2] K. Parsopoulos and M. N. Vrahatis, Particle Swarm Optimization and Intelligence: Advances and Applications, 1st ed. IGI Global, 2010.

[3] J. Sun, C. H. Lai, and X. J. Wu, Particle Swarm Optimisation - Classical and Quantum Perspectives, ser. Numerical Analysis and Scientific Computing. CRC Press, 2012.

[4] M. Clerc, Particle Swarm Optimization. Wiley-ISTE, 2006.

[5] Y. Tan, GPU-based Parallel Implementation of Swarm Intelligence Algorithms. Morgan Kaufmann, 2016.

[6] X. S. Yang, Z. Cui, R. Xiao, and A. Hossein, Swarm Intelligence and Bio-inspired Computation: Theory and Applications, M. Karamanouglu, Ed. Elsevier, 2013.

[7] F. Heppner and U. Grenander, A Stochastic Nonlinear Model for Coordinate Bird Flocks, ser. The Ubiquity of Chaos. Washington: AAS Publications, 1990.

[8] R. Eberhart and Y. Shi, “A modified particle swarm optimizer,” IEEE. IEEE Press, 1998, pp. 69–73.

[9] H. Sho, “Particle multi-swarm optimization: A proposal of multiple particle swarm optimizers with information sharing,” in 2017 IEEE 10th International Workshop on Computational Intelligence and Applications (IWCIA), 2017, pp. 109–114.

[10] Y. del Valle, G. K. Venayagamoorthy, S. Mohagheghi, J. C. Hernandez, and R. G. Harley, “Particle swarm optimization: Basic concepts, variants and applications in power systems,” IEEE Transactions on Evolutionary Computation, vol. 12, no. 2, pp. 171–195, 2008.

[11] R. Poli, “Analysis of the publications on the applications of particle swarm optimization,” Journal of Artificial Evolution and Applications, vol. 2008, pp. 1–10, February 2008.

[12] K. Deb, Multi-objective Optimization. Boston, MA: Springer US, 2014, pp. 403–449. [Online]. Available: https://doi.org/10.1007/978-1-4614-9404-1_15

[13] M. R. Alrashidi and M. E. El-Hawary, “A survey of particle swarm optimization applications in power system operations,” Electric Power Components and Systems, vol. 34, no. 12, pp. 1349–1357, 2006. [Online]. Available: https://doi.org/10.1080/15325000600748871

[14] H.-R. Liu, J.-C. Cui, Z.-D. Lu, D.-Y. Liu, and Y.-J. Deng, “A hierarchical simple particle swarm optimization with mean dimensional information,” Applied Soft Computing, vol. 76, pp. 712–725, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494619300080

[15] Z. Zhang, S. Ding, and W. Jia, “A hybrid optimization algorithm based on cuckoo search and differential evolution for solving constrained engineering problems,” Engineering Applications of Artificial Intelligence, vol. 85, pp. 254–268, October 2019.

[16] H. V. den Bergh and A. P. Engelbrecht, “A cooperative approach to particle swarm optimization,” IEEE Transactions on Evolutionary Computation, pp. 225–239, 2004.

[17] M. El-Abd and M. Kamel, “A taxonomy of cooperative particle swarm optimizers,” International Journal of Computational Intelligence Research, pp. 137–144, 2008.

[18] B. Niu, Y. Zhu, X. He, and H. Wu, “MCPSO: A multi-swarm cooperative particle swarm optimizer,” Applied Mathematics and Computation, pp. 1050–1062, 2007.

[19] Q. He and L. Wang, “An effective co-evolutionary particle swarm optimization for constrained engineering design problems,” Engineering Applications of Artificial Intelligience, vol. 20, no. 1, pp. 89–99, February 2007.

[20] F. Grimaccia, M. Mussetta, and R. E. Zich, “Genetical swarm optimization: Self-adaptive hybrid evolutionary algorithm for electromagnetics,” IEEE Transactions on Antennas and Propagation, vol. 55, no. 3, pp. 781–785, March 2007.

[21] X. Yu and M. Gen, Introduction to Evolutionary Algorithms, 1st ed., ser. Decision Engineering. Springer-Verlag London, 2010.

[22] S. C. Duong, H. Kinjo, E. Uezato, and T. Yamamoto, “Particle swarm optimization with genetic recombination: A hybrid evolutionary algorithm,” Artificial Life and Robotics, vol. 15, no. 4, pp. 444–449, 2010. [Online]. Available: https://doi.org/10.1007/s10015-010-0846-z

[23] H. Zhang, “An analysis of multiple particle swarm optimizers with inertia weight for multi-objective optimization,” International Journal of Computer Science, vol. 9, no. 2, 2012.

[24] C. F. Juang, “A hybrid of genetic algorithm and particle swarm optimization for recurrent network design,” IEEE Transactions on Systems, Man, and Cybernetics, vol. 34, no. 2, pp. 977–985, March 2004.

[25] J. Rechenberg, “Evolutionstrategien: Optimierung technischer systeme nach prinzipien der biologischen evolution,” Frommann-Holzboog Verlag, 1973.

[26] H. P. Schwefel, “Kybernetische evolution als strategie der experimentellen forschung in der strömungstechnik,” Ph.D. dissertation, Universität des Saarlandes, 1975.

[27] M. A. F. Mollinetti, “Analyzing genetic algorithm with game theory and Multi-objective Optimization,” Hybrid Intelligent Systems - Advances in Intelligent Systems and Computing, vol. 420, no. 1, pp. 177–188, November 2016.
[29] R. L. Pereira, D. L. Souza, M. A. F. Mollinetti, M. T. R. S. Neto, E. K. K. Yasojima, O. N. Teixeira, and R. C. L. de Oliveira, “Game theory and social interaction for selection and crossover pressure control in genetic algorithms: An empirical analysis to real-valued constrained optimization,” IEEE Access, vol. 8, pp. 144 839–144 865, August 2020.

[30] M. Tomassini, “Introduction to evolutionary game theory,” in Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation. ACM, 2014, pp. 877–890.

[31] D. L. Souza, O. N. Teixeira, D. C. Monteiro, R. C. L. de Oliveira, and M. A. F. Mollinetti, “Novel Competitive Quantum-Behaviour Evolutionary Multi-Swarm Optimizer Algorithm Based on CUDA Architecture Applied to Constrained Engineering Design,” Springer-Verleg, 2014.

[32] D. L. Souza, O. N. Teixeira, D. C. Monteiro, and R. C. L. de Oliveira, “A new cooperative evolutionary multi-swarm optimizer algorithm based on CUDA architecture applied to engineering optimization,” in Combinations of Intelligent Methods and Applications, I. Hatzilygeroudis and V. Palade, Eds., vol. 23. Springer-Verlag, 2013, pp. 95–115.

[33] V. Miranda and N. Fonseca, “EPSO - evolutionary particle swarm optimization, a new algorithm with applications in power systems;,” in Transmission and Distribution Conference and Exhibition 2002: Asia Pacific. IEEE/PES, vol. 2. IEEE Press, 2002, pp. 745–750.

[34] D. B. Kirk and W. W. Hwu, Programming Massively Parallel Processors: A Hands-on Approach, 1st ed. Elsevier and Morgan Kaufman, 2010.

[35] NVIDIA, “NVIDIA CUDA C programming guide,” https://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf, NVIDIA Corporation, Tech. Rep., November 2019.

[36] K. M. Anga, W. H. Lima, N. A. M. Isab, S. S. Tianga, and C. H. Wonga, “A constrained multi-swarm particle swarm optimization without velocity for constrained optimization problems,” Expert Systems With Applications, vol. 140, pp. 1–23, February 2018.

[37] R. Lan, L. Zhang, Z. Tang, Z. Liu, and X. Liao, “A hierarchical sorted swarm optimizer for large-scale optimization,” IEEE Access, vol. 7, pp. 40 625–40 635, 2019.

[38] T. P. Q. Nguyen and R. J. Kuo, “Automatic fuzzy clustering using non-dominated sorting particle swarm optimization algorithm for categorical data,” IEEE Access, vol. 7, pp. 99 721–99 734, 2019.

[39] G. Coath and S. K. Halgamuge, “An improved particle swarm algorithm for solving constrained optimization problems,” in The 2003 Congress on Evolutionary Computation, 2003. CEC ‘03., vol. 4, Dec 2003, pp. 2419–2425.

[40] J. Liang and P. N. Suganthan, “Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism,” in Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics, vol. 5, 2002, pp. 14–18.

[41] K. E. Parsopoulos and M. N. Vrahatis, “Parameter selection and adaptation in unified particle swarm optimization,” Mathematical and Computer Modelling, vol. 46, no. 1–2, pp. 198–213, July 2007.

[42] X. Hu and R. Eberhart, “Solving constrained nonlinear optimization problems with particle swarm optimization,” in Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics, vol. 5, 2002, pp. 14–18.

[43] A. F. U. Ali and M. A. Tawhid, “A hybrid particle swarm optimization and genetic algorithm with population partitioning for large scale optimization problems,” Ain Shams Engineering Journal, vol. 8, no. 2, pp. 191–206, 2017. [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3119579, IEEE Access

[44] Q. Zhao and C. Li, “Two-stage multi-swarm particle swarm optimizer for unconstrained and constrained global optimization,” IEEE Access, vol. 8, pp. 124 905–124 927, 2020.

[45] R. Poli, “Mean and variance of the sampling distribution of particle swarm optimization revisited,” in Proceedings of the 2014 IEEE Congress on Evolutionary Computation, vol. 8, no. 3, pp. 204–210, June 2004.

[46] A. F. Kuri-Morales and J. Gutierrez-Garcia, “Penalty function methods for constrained optimization with genetic algorithms: A statistical analysis,” in MICAI 2002: Advances in Artificial Intelligence, C. A. Coello Coello, A. de Alvarez, L. E. Sucarrat, and O. C. Montes, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 108–117.

[47] J. A. Joines and C. R. Houck, “On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with ga’s,” in Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence, June 2004, pp. 579–584.

[48] Y. Wang and Z. Cai, “A hybrid multi-swarm particle swarm optimization to solve constrained optimization problems,” Frontiers of Computer Science in China, vol. 3, no. 1, pp. 38–52, 2009. [Online]. Available: https://doi.org/10.1007/s11704-009-0010-x

[49] K. Deb, “An efficient constraint handling method for genetic algorithms,” Computer Methods in Applied Mechanics and Engineering, vol. 186, no. 2-4, pp. 313–338, June 2000.

[50] Y. Wang and Z. Cai, “Two-stage multi-swarm particle swarm optimization for constrained optimization,” in The 2003 Congress on Evolutionary Computation, 2003. CEC ’03., vol. 4, Dec 2003, pp. 2419–2425.

[51] R. Eberhart and J. Kennedy, “A new approach to particle swarm optimization,” in Proceedings of the 6th International Symposium on Micro Machine and Human Science, Oct 1995, pp. 39–43.

[52] R. Mendes, J. Kennedy, and J. Neves, “The fully informed particle swarm: behaviour and its tuning,” IEEE Transactions on Evolutionary Computation, vol. 8, no. 3, pp. 204–210, June 2004.

[53] K. E. Parsopoulos and M. N. Vrahatis, “Parameter selection and adaptation in unified particle swarm optimization,” Mathematical and Computer Modelling, vol. 46, no. 1–2, pp. 198–213, July 2007.

[54] X. Hu and R. Eberhart, “Solving constrained nonlinear optimization problems with particle swarm optimization,” in Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics, vol. 5, 2002, pp. 14–18.

[55] K. R. Harrison, A. P. Engelbrecht, and B. M. Ombuki-Berman, “Inertia weight control strategies for particle swarm optimization: Too much momentum, not enough analysis,” Swarm Intelligence, vol. 10, no. 4, pp. 267–305, December 2016.

[56] R. Eberhart and J. Kennedy, “New optimizer using particle swarm theory,” in MHS ’95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Oct 1995, pp. 39–43.

[57] R. Mendes, J. Kennedy, and J. Neves, “The fully informed particle swarm: Simpler, maybe better,” IEEE Transactions on Evolutionary Computation, vol. 8, no. 3, pp. 204–210, June 2004.

[58] S. Xu and Y. Rahmat-Samii, “Boundary conditions in particle swarm optimization,” in Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation, ser. GECCO’01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 469–476.

[59] B. Yang, Y. Chen, Z. Zhao, and Q. Han, “A master-slave particle swarm optimization algorithm for solving constrained optimization problems,” in 2006 6th World Congress on Intelligent Control and Automation, vol. 1, June 2006, pp. 3208–3212.

[60] M. Daneshyari and G. G. Yen, “Constrained multiple-swarm particle swarm optimization within a cultural framework,” IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, vol. 42, no. 2, pp. 475–490, March 2012.

[61] R. N. Samal, A. Konar, S. Das, and A. Abraham, “A closed loop stability analysis and parameter selection of the particle swarm optimization dynamics for faster convergence,” in 2007 IEEE Congress on Evolutionary Computation, Sep. 2007, pp. 1769–1776.

[62] G. C. Coelho, M. A. de Macedo, and C. M. Coello, “Mathematical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art,” Computer Methods in Applied Mechanics and Engineering, vol. 191(11-12), pp. 1245–1287, 2002.

This work has been licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
[68] H. Garg, “A hybrid gsa-ga algorithm for constrained optimization problems,” Information Sciences, vol. 478, pp. 499–523, April 2019.

[69] I. Brajevic, M. Tuba, and M. Subotic, “Improved artificial bee colony algorithm for constrained problems,” Proceedings of the 11th WSEAS International Conference on Neural Networks, Fuzzy Systems and Evolutionary Computing, pp. 185–190, 2010.

[70] R. L. Pereira, M. A. F. Mollinetti, M. T. R. S. Neto, A. de Almeida Neto, D. L. Souza, E. K. Yasojima, O. N. Teixeira, and R. C. L. de Oliveira, “Parametric analysis of iterated game environments as social interaction model for genetic algorithm to solve constrained engineering problems,” in IEEE Congress on Evolutionary Computation (CEC), July 2018, pp. 1–8.

[71] S. G. de-los Cobos-Silva, M. A. Gutiérrez-Andrade, R. A. Morán-Gutiérrez, P. Lara-Velázquez, E. A. Rincón-García, and A. Ponsich, “An efficient algorithm for unconstrained optimization,” Mathematical Problems in Engineering, vol. 2015, pp. 1–17, 2015.

[72] G. Wu, R. Mallipuddi, and P. N. Suganthan, “Problem definitions and evaluation criteria for CEC 2017 competition on constrained real-parameter optimization,” IEEE Tech Report, September 2017.

[73] R. Lisboa, E. K. Yasojima, R. M. S. de Oliveira, M. A. F. Mollinetti, O. N. Teixeira, and R. C. L. de Oliveira, “Parallel genetic algorithm with social interaction for solving constrained global optimization problems,” in 2015 7th International Conference of Soft Computing and Pattern Recognition (SoCPAR), Nov 2015, pp. 351–356.

[74] E. K. Yasojima, R. C. L. de Oliveira, O. N. Teixeira, and R. L. Pereira, “CAM-ADX: A new genetic algorithm with intensified intensification and diversification for design optimization problems with real variables,” Robotica, vol. 37, no. 9, pp. 1595–1640, September 2019.

[75] A. Ravindran, K. M. Ragsdell, and G. V. Reklaitis, Handbook of Research on Artificial Intelligence Techniques and Applications, IGI Global, 2015, ch. A Hybrid GA - GSA Algorithm Using an Artificial Bee Colony Algorithm, pp. 223–229, June 1990.

[76] H. Garg, “A hybrid gsa-ga algorithm for constrained optimization problems,” Applied Mathematics and Computation, vol. 274, no. 1, pp. 292–305, 2016.

[77] E. M. Montes and C. A. C. Coello, “An empirical study about the usefulness of evolution strategies to solve constrained optimization problems,” International Journal of General Systems, vol. 37, no. 4, pp. 443–473, July 2008.

[78] S. Mirjalili, “Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm,” Knowledge-Based Systems, vol. 89, pp. 228–249, November 2015.

[79] L. S. Coelho, “Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems,” Expert Systems with Applications, pp. 1676–1683, 2010.

[80] A. W. Mohamed, “A novel differential evolution algorithm for solving constrained engineering optimization problems,” Journal of Intelligent Manufacturing, vol. 29, no. 3, pp. 659–692, January 2017.

[81] W. Gong, Z. Cai, and D. Liang, “Engineering optimization by means of an improved constrained differential evolution,” Computer Methods in Applied Mechanics and Engineering, vol. 268, no. 1, pp. 884–904, January 2014.

[82] G. Diham and V. Kumar, “Emperor penguin optimizer: A bio-inspired algorithm for engineering problems,” Knowledge-Based Systems, vol. 159, no. 1, pp. 20–50, November 2018.

[83] H. Eskandar, A. Sadollah, A. Bahreininejad, and M. Hamdi, “Water cycle algorithm – a novel metaheuristic optimization method for solving constrained engineering optimization problems,” Computers and Structures, vol. 110-111, pp. 151–166, November 2012.

[84] A. Sadollah, A. Bahreininejad, H. Eskandar, and M. Hamdi, “Mine blast algorithm: A new population-based algorithm for solving constrained engineering optimization problems,” Applied Soft Computing, vol. 13, no. 5, pp. 2592–2612, May 2013.

[85] S. S. Rao, Engineering Optimization: Theory and Practice. John Wiley & Sons, 1996, no. 3rd.

[86] Y. L. Hsu and T. C. Liu, “Developing a fuzzy proportional-derivative controller optimization engine for engineering design optimization problems,” Engineering Optimization, pp. 679–700, 2007.

[87] V. K. Mehta and D. Dasgupta, “A constrained optimization algorithm based on the simplex search method,” Engineering Optimization, vol. 44, no. 3, pp. 537–550, September 2012.

[88] M. Shahrour and A. Salehi, “An efficient competitive learner-based optimization: A hybrid method to solve engineering problems,” International Journal of Optimization in Civil Engineering, vol. 10, no. 1, pp. 155–180, December 2020.

[89] H. Garg, A Hybrid GA - GSA Algorithm for Optimizing the Performance of an Industrial System by Utilizing Uncertain Data, ser. Handbook of Research on Artificial Intelligence Techniques and Algorithms. IGI Global, 2015.

[90] S. K. Haldar, “Chapter 9 - statistical and geostatistical applications in geology,” in Mineral Exploration, S. Haldar, Ed. Boston: Elsevier, 2013, pp. 157–182. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B978012416005700009X

[91] J. D. Hunter and M. Droettboom, “Matplotlib: Python plotting,” 2020. [Online]. Available: https://matplotlib.org
Author et al.: Preparation of Papers for IEEE TRANSACTIONS and JOURNALS

DANIEL LEAL SOUZA was born in Belém, Pará, Brazil, in 1989. He holds a Bachelor’s Degree in Computer Science from the University Centre of the State of Pará (CESUPA), in 2010. Master’s Degree in Computer Science from the Federal University of Pará (UFPA) in 2014. He is currently a Ph.D. candidate in Electrical Engineering by the UFPA. Since 2011, develops research at Laboratory of Bionspired Computing (LCBIO/UFPA). He’s currently a professor and researcher in Cyberspatial Institute at the Federal Rural University of the Amazonia (UFRA). His research interests includes Computational Intelligence, Artificial Intelligence Techniques applied to Engineering, Evolutionary Computing, Swarm Intelligence, High Performance Computing and Machine Learning.

RODRIGO LISBÔA PEREIRA (Member, IEEE) was born in Belém, Pará, Brazil, in 1985. He holds a Bachelor’s Degree in Computer Science from the University Centre of the State of Pará (CESUPA), in 2006. Postgraduate in Software Project Management, Master’s Degree and Ph.D Degree in Electrical Engineering in Area of Applied Computing from the Federal University of Pará (UFPA), respectively, in 2008, 2011 and 2020, and is member of the Institute of Electrical and Electronics Engineers (IEEE), since 2016. He is currently Professor and Researcher, leader of research group Computational Technologies Laboratory (LabTeC) at the Federal Rural University of the Amazonia (UFRA). Has experience in the areas of Computer Science and Computer Engineering, with an emphasis on Software Engineering, Computational Intelligence, Natural Computing, Artificial Intelligence Techniques applied to Engineering and High Performance Computing.

OTÁVIO NOURA TEIXEIRA was born in Belém, Pará, Brazil, in 1977. He holds a degree in Data Processing Technology from the University Centre of the State of Pará (CESUPA), in 1998, and a Bachelor’s Degree in Computer Science from the Federal University of Pará (UFPA), in 2003. Master and Ph.D. in Electrical Engineering, with emphasis in Applied Computing, research line in Computational Intelligence, from the UFPA in 2005 and 2012. He is currently an Adjunct Professor at the UFPA, Professor and Researcher of the Postgraduate Program in Applied Computing (PPCA/UFPA). Has experience in the areas of Computer Science and Computer Engineering, with emphasis on Computational Intelligence and Computer Theory.

ROBERTO C. L. DE OLIVEIRA was born in Belém, Brazil, in 1965. He received the Ph.D. degree in electrical engineering from the Federal University of Santa Catarina, Florianópolis, Brazil, in 1999. He is currently a Full Professor of Electrical Engineering Faculty, Institute of Technology, Federal University of Pará (UFPA), Brazil. He was Visiting Professor at Texas A&M University, College Station, TX, USA, from 2017 to 2018. He published over 200 refereed papers in journals and conferences and published a book. His research interests include Natural Computing, Computational Intelligence, Artificial Neural Networks and Evolutionary Computing.

VOLUME 4, 2016 27