Harnessing Particle Swarm Optimization Through Relativistic Velocity

Mateus Roder, Gustavo Henrique de Rosa,
Leandro Aparecido Passos, João Paulo Papa
Department of Computing
UNESP - São Paulo State University
Bauru - SP, Brazil
{mateus.roder, gustavo.rosa,
leandro.passos, joao.papa}@unesp.br

André Luis Debiaso Rossi
UNESP - São Paulo State University
Itapeva - SP, Brazil
andre.rossi@unesp.br

***Abstract*—In the last century, Albert Einstein's perceptions of the world afforded a revolution in the understanding of the universe. In his theory of general relativity, he describes the space-time continuum, a concept capable of explaining several phenomena, ranging from gravity to black holes and supernovas. Further, it also provides a set of formulations to generalize classical physics concepts to accommodate the relativistic notions. Meanwhile, several mathematicians have been working on optimization tools aiming to solve complex problems associated with a large number of variables. Nowadays, despite the computational power, many daily tasks still pose a challenge and are becoming more prohibitives, mostly due to the massive amount of data to be processed. Therefore, efficient optimization techniques are more desirable than ever. In this context, meta-heuristic optimization has arisen, i.e., stochastic nature-inspired methods capable of finding sub-optimal solutions for complex problems with a reasonable computational effort. However, such approaches still suffer from some drawbacks related to low convergence and getting stuck on local optima, among others. Therefore, in this paper, we introduce relativistic concepts into the well-known meta-heuristic optimization technique Particle Swarm Optimization (PSO). The experimental results evince the robustness of the proposed approach compared to the standard PSO as well as three other variations for five benchmarking functions.**

***Index Terms*—Global Optimization, Meta-Heuristic Optimization, Particle Swarm Optimization, Theory of Relativity, Relativistic Particle Swarm Optimization**

I. Introduction

During humankind's evolution, several tools were developed in the most different areas of knowledge aiming to improve living quality and longevity. In the last decades, the advent of powerful computers has brought the development and use of such tools to a cumbersome baseline, being sophisticated technologies employed even in the most straightforward shores from daily life. However, such computer-aided approaches generally come with a high computational complexity cost, which demands either an optimization performed by experts or the development of "meta-tools", or the so-called optimization techniques [1].

Optimization techniques are algorithms developed to either maximize or minimize some target function. In this context, traditional methods generally employ a brute-force [2], i.e., testing all possible solutions, or gradient-based approaches [3], which gradually converges to some optimum. However, both methods may become prohibitive when the number of variables and possibilities are very high. Additionally, gradient-based approaches present some constraints concerning the objective function characteristics. Thus they are dependent on the problem definition.

On the other hand, meta-heuristic optimization methods [4] obtained notorious popularity in the last decades due to its simplicity and capability of finding sub-optimal solutions for complex problems with a reasonable low computational cost, such as hyperparameter fine-tuning of machine learning techniques [5], [6], feature selection [7], [8], and general-purpose benchmarking function optimization [9], [10], to cite a few. These methods are stochastic algorithms that mimic intelligent behavior observed on nature, like a swarm of birds or the evolution of species, among others, to tackle a stated problem. In short, these techniques employ a set of candidate solutions, denoted as individuals, who evolve according to some specific rules during several iterations towards the best solution.

Among a wide variety of meta-heuristic optimization techniques, the Particle Swarm Optimization (PSO) [11] has become one of the most popular due to its simplicity and consistent performance over a broad diversity of problems. In a nutshell, PSO particles share information about their position to perform an in-depth exploration and exploitation of the search space towards optima locations. Such information is employed to update the particle's velocity and, consequently, their position.

In this context, several works proposed variants of traditional PSO considering different methods for updating the velocity function. Nickabadi et al. [12], for instance, proposed the Adaptive Inertia Weight Particle Swarm Optimization (AIWPSO), which employs the dynamically change in inertia

The authors would like to thank São Paulo Research Foundation (FAPESP) grants #2013/07375-0, #2014/12236-1, #2017/25908-6, #2019/02205-5, #2019/07825-1, and #2019/07665-4, and National Council for Scientific and Technological Development (CNPq) grants #307066/2017-7 and #427968/2018-6.

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

weight to better global exploration or local exploitation in any dimension. Yang et al. [13] proposed the Vertical Particle Swarm Optimization (VPSO), claiming that the algorithm is capable of avoiding problems related to premature convergence and loss of population diversity. Recently, Ye et al. [14] proposed the PSO with Dynamic Learning Strategy (PSO-DLS) to improve the balance between global exploration and local exploitation.

Regarding velocity variation, in the last century, Albert Einstein [15] proposed the revolutionary Theory of General Relativity, which generalizes basics concepts of classical physics, such as the velocity, to accommodate the distortion exerted over the space-time continuum due to near to speed-of-light moving particles or extremely massive bodies in the universe. However, as far as we know, no work ever employed such concepts to improve PSO velocity formulation, and thus the model performance.

Therefore, this paper proposes the Relativistic Particle Swarm Optimization (RPSO) algorithm, a meta-heuristic approach that employs relativistic concepts to update PSO particles' velocities. The efficiency of RPSO is compared against the standard PSO algorithm and three PSO variants, namely AIWPSO, VPSO, and PSO-DLS, which were selected due to a similar working mechanism, i.e., to improving PSO by replacing the velocity formulation. Experiments were conducted to evaluate the proposed method over five benchmarking functions, which were selected considering distinct intrinsic characteristics.

Hence, the contributions of the present study are twofold: (i) to propose the Relativistic Particle Swarm Optimization algorithm, and (ii) to promote the scientific community regarding meta-heuristic optimization approaches. The remainder of this paper is presented as follows. Section II introduces the theoretical background concerning PSO and its variations, while Section III briefly presents the Theory of General Relativity, as well as it describes the proposed method. Sections IV and V describes the methodology and present the experimental results, respectively. Finally, Section VI states conclusions and future works.

II. Particle Swarm Optimization

Particle Swarm Optimization is a meta-heuristic algorithm that models each candidate solution as a nature-inspired agent in search of a feasible objective. In other words, each agent represents a particle that belongs to a swarm and searches for optimal food sources (objective). More formally, each agent is composed of a tuple $(\boldsymbol{x}, \boldsymbol{v})$, where $\boldsymbol{x}$ stands for its position and $\boldsymbol{v}$ for its velocity. Its initial position $\boldsymbol{x}$ is represented by a n-dimensional randomly vector, while its velocity $\boldsymbol{v}$ is represented by an n-dimensional vector of zeros, where each dimension stands for a unique decision variable. Furthermore, the algorithm's objective is to search for the most feasible decision variables aiming to maximize or minimize a target function, i.e., fitness function.

Let $\boldsymbol{v}_i^t$ be the velocity of a particle i at iteration t, belonging to a swarm of size K, such that $i \in \{1, 2, \ldots, K\}$. One can update its velocity according to Equation 1, as follows:

$$\boldsymbol{v}_i^{t+1} = w\boldsymbol{v}_i^t + c_1 r_1(\boldsymbol{x}_i^* - \boldsymbol{x}_i^t) + c_2 r_2(\boldsymbol{g} - \boldsymbol{x}_i^t), \quad (1)$$

where $\boldsymbol{x}_i^*$ stands for the best position obtained by particle i so far, and $\boldsymbol{g}$ denotes the current best solution considering all the swarm. Additionally, w, c_1, and c_2 stand for the inertia weight, the cognitive and social parameters, respectively. Finally, r_1 and r_2 are uniformly distributed random numbers in the range $[0, 1]$.

After updating a particle's velocity, its position needs to be updated as well. Let $\boldsymbol{x}_i^t$ be the position of a particle i at iteration t. One can update its position according to Equation 2, as follows:

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

The following sections describe the PSO variants employed in this work, as well as its formulation and particularities.

A. Adaptive Inertia Weight Particle Swarm Optimization

In this variant [12], the inertia weight changes dynamically, providing a fast initial convergence when the particles are randomly located in the search space and a slow approximation to the minimum when the swarm becomes close to the optimal solution. To accommodate the variation, one can replace the standard inertia weight w from Equation 1 with the dynamic inertia weight ω, computed for each iteration t as follows:

$$\omega^t = (\omega_{max} - \omega_{min})\pi^t + \omega_{min}, \quad (3)$$

where ω_{min} and ω_{max} stand for the minimum and maximum inertia weight range, respectively, usually between $[0, 1]$. Further, $\pi^t \in [0, 1]$ denotes the swarm success, and represents the percentage of the particles improvement in the last iteration, computed as follows:

$$\pi^t = \frac{\sum_{i=1}^{K} S(i,t)}{K}, \quad (4)$$

where $S(\cdot)$ denotes the notion of individual particles' success, and is computed as follows:

$$S(i,t) = \begin{cases} 1, & \text{if } fitness(\boldsymbol{x}_i^{*(t)}) < fitness(\boldsymbol{x}_i^{*(t-1)}) \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

B. Particle Swarm Optimization with Dynamic Learning Strategy

The PSO-DLS [14] was proposed to improve the proper balance between global exploration and local exploitation from the original PSO algorithm, i.e., to achieve better solutions in a few iterations. Moreover, such a variant also tries to maintain the population diversity and to overcome some local optima or earlier convergence.

This version creates sub-swarms instead of a single population. Roughly speaking, this algorithm divides the original

swarm into several M sub-swarms, that in theory are forced to explore different areas in a multimodal optimization function, for instance. In this context, the author employed the $\boldsymbol{l}^{\star}$ term on the velocity equation, which may represent the global best solution (as in Equation 1) or the set of best solutions in each sub-swarm, with a random probability to decide each usage. Each particle's velocity can be updated as follows:

$$\boldsymbol{v}_i^{t+1} = w\boldsymbol{v}_i^t + c_1 r_1(\boldsymbol{x}_i^* - \boldsymbol{x}_i^t) + c_2 r_2 \left(\frac{1}{M} \sum_{m=1}^{M} \boldsymbol{l}_{\boldsymbol{m}}^{\star} - \boldsymbol{x}_i^t \right), \quad (6)$$

where $\boldsymbol{x}_i^*$ stands for the best position obtained by particle i, and $\boldsymbol{l}_{\boldsymbol{m}}^{\star}$ denotes best position achieved so far in the m^{th} sub-swarm. Finally, the position can be updated as in the Equation 2.

C. Vertical Particle Swarm Optimization

The Vertical PSO [13] is a variant that claims to avoid some problems related to premature convergence to local optima and the loss of population diversity, among others. In short, the algorithm tries to prevent the swarm from evading the neighborhood of the global optimum after a few iterations with no improvements in the search. The procedure is implemented by introducing a vertical direction while updating each particle's position. Therefore, one can rewrite Equation 2 to fit the concept using Equation 7, as follows:

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + r\boldsymbol{v}_i^{t+1} + (1-r)\boldsymbol{s}_i^{t+1}, \quad (7)$$

where r represents a random number generated from a normal distribution in the range $[0, 1]$ at each iteration and $\boldsymbol{s}$ denotes the vertical velocity, described as follows:

$$\boldsymbol{s}_i^{t+1} = \boldsymbol{s}_i^t - \left(\frac{(\boldsymbol{v}_i^t \bullet \boldsymbol{s}_i^t)}{(\boldsymbol{v}_i^t \bullet \boldsymbol{v}_i^t)} \right) \boldsymbol{v}_i^t, \quad (8)$$

where $\bullet$ stands for the dot product between the vectors $\boldsymbol{v}$ and $\boldsymbol{s}$.

III. Relativistic Particle Swarm Optimization

In this section, we present the Relativistic Particle Swarm Optimization (RPSO), which improves the naïve PSO algorithm by considering the space-time distortion. First, we provide more in-depth concepts of the theory of relativity, which is the foundation for RPSO.

A. Theory of Relativity

Albert Einstein accomplished one of the most important physics works by proposing the Theory of Relativity, which comprises the special and general relativity theories [15]. The former is responsible for introducing new approaches to deal with celestial bodies' motion particularities, while the latter incorporated an acceleration concept. In other words, the theory combined space and time into a single notion, the well-known space-time continuum, which is distorted by massive bodies, such as planets and stars. Such distortion in the space-time continuum provides an elegant justification for several phenomena observed in nature, e.g., gravity laws and discrepancy of stars positions.

With that in mind, classical mechanics theory was extended to model the phenomena observed by Einstein. The particular one that we are interested in is the momentum, which is computed in a $three$-dimensional space as follows:

$$p_{(\boldsymbol{u})} = \gamma_{(\boldsymbol{u})} m \boldsymbol{u}, \quad (9)$$

where $\boldsymbol{u} = (u_x, u_y, u_z)$ denotes the velocity of an object with mass m in a $three$-dimensional space, and γ is the Lorentz factor, which is defined as follows:

$$\gamma_{(\boldsymbol{u})} = \frac{1}{\sqrt{1 - \left(\frac{|\boldsymbol{u}|}{c}\right)^2}}, \quad (10)$$

where $|\boldsymbol{u}|$ is the magnitude of vector $\boldsymbol{u}$ and c stands for the speed of light ($300,000$ km/s).

B. Algorithm Overview

The RPSO differs from the standard PSO by taking into account the particles' mass effect, the speed of light, and an enhanced social behavior feature, allowing all particles to know the space-time values of the swarm. With that in mind, RPSO maps $three$-dimensional momentum formula, depicted by Equation 9, into an n-dimensional approximation, which is capable of calculating the velocity of each particle in an n-dimensional search space. In order to accomplish such an approach, one can rewrite Equation 1 into Equation 11, as follows:

$$\boldsymbol{v}_i^{t+1} = p(\boldsymbol{v}_i^t) + c_1 r_1(\boldsymbol{x}_i^* - \boldsymbol{x}_i^t) + c_2 r_2(\boldsymbol{g} - \boldsymbol{x}_i^t), \quad (11)$$

where $p_{(\cdot)}$ denotes the standard $three$-dimensional relativistic momentum (9) expanded to an n-dimensional space. Additionally, as each particle needs a mass to calculate its relativistic momentum, we opted to draw the mass m from a uniform distribution in the range $[0, 1]$. The RPSO is show in Algorithm 1.

Algorithm 1: Relativistic Particle Swarm Optimization

```
Randomly initializes x;
Evaluate x;
x_i^* ← x; ∀i ∈ 1, 2, ..., K;
do
    for each candidate i do
        v_i^{t+1} = p(v_i^t) + c1 r1 (x_i^* − x_i^t) + c2 r2 (g − x_i^t);
        x_i^{t+1} = x_i^t + v_i^{t+1};
        Evaluate x_i and compute its fitness value;
        if (x_i is better than x_i^*) then
            x_i^* ← x_i;
        end
    end
    Update g as the best position found so far;
while (stop criterion is not met);
```

IV. Methodology

This section describes the experimental setup adopted to evaluate the RPSO and the other PSO variants for the optimization of five benchmark functions.

A. Benchmarking Functions

The benchmarking functions used to evaluate the approach proposed in this paper and their main characteristics are described as follows:

- Sphere (f_1): continuous, differentiable, separable, scalable, multimodal;
- Salomon (f_2): continuous, differentiable, non-separable, scalable, multimodal;
- Alpine #1 (f_3): continuous, non-differentiable, separable, non-scalable, multimodal;
- Rastrigin (f_4): continuous, differentiable, separable, scalable, multimodal; and
- Schwefel (f_5): continuous, differentiable, partially-separable, scalable, unimodal.

Moreover, Table I introduces each function's mathematical formulation, as well as their upper and lower boundaries and the global minimum.

B. Experimental Setup

The experiments were performed to optimize each benchmark function with $D \in \{10, 50, 100\}$ dimensions, i.e., decision variables, for $t = \{100, 1,000,$ and $10,000\}$ iterations employing 30 particles. Furthermore, the standard PSO as well as the three other variants were employed as baselines for comparison purposes: AIWPSO, PSO-DLS, and VPSO. The AIWPSO hyperparameter was set to $w_{min} = 0.1, w_{max} = 0.9$, $c_1 = c_2 = 1.7$ according to the authors' set up, while the following setting was employed for PSO, VPSO and PSO-DLS: $w = 0.7$, $c_1 = c_2 = 1.7$, in which the former algorithm employed $M = 10$ sub-swarms, as established by the authors [14]

For the statistical analysis, each experiment was executed 15 times and the best results according to the Wilcoxon signed-rank test [16] with 0.05 of significance are in bold. Concerning the implementation, we employed the Opytimizer[1] [17] library.

V. Experimental Results

This section presents and discusses the results obtained in the experiments. Tables II to VI present the mean fit value and standard deviation considering the proposed RPSO, standard PSO, and the PSO variants over the five benchmarking functions considering D dimensions and t iterations, as described in Section IV. Notice the best results, according to the Wilcoxon signed-rank test, obtained over each configuration considering the dimensionality and the number of iterations are presented in bold, while the best overall results considering each dimension are underlined.

[1]Available at http://github.com/gugarosa/opytimizer.

A. Overall Discussion

Considering general scenery, one can notice the RPSO was the most accurate technique overall since it obtained the minimum mean values over all benchmarking functions considering every dimensional configuration. Note that VPSO and PSO-DLS obtained similar statistical results considering the Sphere (Table II) and Rastrigin (Table V) functions, respectively, but only for $D = 10$. Such behavior is not surprising once both functions were the only ones whose results converge to the best values after less than $1,000$ iterations, denoting simpler problems to solve.

Another general fact observed in all scenarios concerns RPSO obtained alone the best results considering 50- and 100-dimensional configurations over $10,000$ iterations, while VPSO and PSO-DLS obtained similar results considering a 10-dimensional space over Sphere and Rastrigin functions, respectively. Further, RPSO also obtained the best results overall considering $1,000$ iterations over 10- and 50-dimensional problems. Regarding the configuration with 100 dimensions, it obtained the best results in three out of two benchmarking functions, i.e., Alpine $\#1$, Rastrigin, and Schwefel.

The main "drawback" of the proposed method relies on the necessity of a slightly higher number of iterations for convergence, as observed in all experiments with 50 and 100 dimensions using 100 iterations, except by Schwefel function, presented in Table VI. Two main points can explain such behavior: the first one is the mass dependence, which affects the momentum of the particles, and in this work employs a "scale factor" between $[0, 1]$. The second one relates directly to the implicit relativity behavior, in which slow or low massive particles tend to have less initial inertia (and consequently, less momentum) in the search space, requiring a few more steps to achieve good velocities that allow them to exploit and explore better the search space, and therefore, distorting the space-time continuum.

B. Convergence Analysis

Figures 1 to 5 depict the convergence from each benchmark function considering all baselines and dimensions over $10,000$ iterations. The behavior depicted in those images corroborates the results presented in previous tables, i.e., the proposed approach obtained the best convergence overall. However, in several cases, such as in Figures 1c, 2b, 2c, and 3c, one can observe the competitors converge faster than RPSO, but they usually get stuck at worse local optima. Thus, RPSO requires a few more iterations to converge to best positions and surpass such algorithms.

VI. Conclusion

In this work, a novel Particle Swarm Optimization variant was proposed changing the classical velocity interpretation by the one inspired in the relativistic momentum. The model intrinsically incorporates several relativistic phenomena, such as inertia, space-time distortion, and the mass effect in the swarm, while preserving the social intelligence behavior.

TABLE I
BENCHMARKING FUNCTIONS.

Function	Equation	Bounds	$f(\boldsymbol{x}^*)$
Sphere	$f_1(x) = \sum_{i=1}^{D} x_i^2$	$-10 \leq x_i \leq 10$	0
Salomon	$f_2(x) = 1 - \cos(2\pi\sqrt{\sum_{i=1}^{D} x_i^2}) + 0.1\sqrt{\sum_{i=1}^{D} x_i^2}$	$-100 \leq x_i \leq 100$	0
Alpine #1	$f_3(x) = \sum_{i=1}^{D} \lvert x_i sin(x_i) + 0.1x_i \rvert$	$-10 \leq x_i \leq 10$	0
Rastrigin	$f_4(x) = 10n + \sum_{i=1}^{D} \left[x_i^2 - 10cos(2\pi x_i)\right]$	$-5.12 \leq x_i \leq 5.12$	0
Schwefel	$f_5(x) = \left(\sum_{i=1}^{D} x_i^2\right)^{\sqrt{\pi}}$	$-100 \leq x_i \leq 100$	0

TABLE II
MEAN AND STANDARD DEVIATION BEST FITNESS VALUES ACHIEVED OVER SPHERE FUNCTION (f_1).

	D = 10			**D = 50**			**D = 100**		
Algorithm	**t = 100**	**t = 1,000**	**t = 10,000**	**t = 100**	**t = 1,000**	**t = 10,000**	**t = 100**	**t = 1,000**	**t = 10,000**
AIWPSO	0.206 ± 0.194	0.197 ± 0.179	0.009 ± 0.014	24.905 ± 6.065	24.856 ± 6.071	7.854 ± 5.054	64.939 ± 10.959	64.663 ± 11.022	58.941 ± 13.68
PSO	0.027 ± 0.031	0.022 ± 0.028	0.022 ± 0.028	23.157 ± 10.971	23.025 ± 10.894	21.898 ± 11.469	63.871 ± 8.201	63.561 ± 8.144	61.751 ± 7.973
RPSO	0.008 ± 0.014	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	20.982 ± 8.379	$\mathbf{5.326 \pm 4.838}$	$\mathbf{2.291 \pm 2.830}$	92.446 ± 29.883	25.496 ± 6.683	$\mathbf{10.547 \pm 5.821}$
PSO-DLS	0.007 ± 0.006	0.009 ± 0.009	0.003 ± 0.006	$\mathbf{10.053 \pm 1.992}$	9.772 ± 1.916	4.693 ± 1.575	$\mathbf{23.615 \pm 4.309}$	$\mathbf{23.43 \pm 3.772}$	15.898 ± 2.989
VPSO	$\mathbf{0.001 \pm 0.001}$	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	13.016 ± 2.714	9.588 ± 2.500	7.653 ± 2.012	41.612 ± 10.438	34.207 ± 9.745	30.027 ± 8.769

TABLE III
MEAN AND STANDARD DEVIATION BEST FITNESS VALUES ACHIEVED OVER SALOMON FUNCTION (f_2).

	D = 10			**D = 50**			**D = 100**		
Algorithm	**t = 100**	**t = 1,000**	**t = 10,000**	**t = 100**	**t = 1,000**	**t = 10,000**	**t = 100**	**t = 1,000**	**t = 10,000**
AIWPSO	2.613 ± 0.897	2.613 ± 0.897	2.613 ± 0.897	11.393 ± 1.287	11.393 ± 1.287	11.393 ± 1.287	18.767 ± 1.336	18.767 ± 1.336	18.767 ± 1.336
PSO	1.640 ± 0.803	1.64 ± 0.803	1.64 ± 0.803	11.78 ± 2.137	11.780 ± 2.137	11.780 ± 2.137	18.213 ± 1.028	18.213 ± 1.028	18.213 ± 1.028
RPSO	$\mathbf{0.916 \pm 0.300}$	$\mathbf{0.393 \pm 0.129}$	$\mathbf{0.273 \pm 0.106}$	10.812 ± 1.384	$\mathbf{5.724 \pm 0.842}$	$\mathbf{3.441 \pm 0.901}$	20.441 ± 1.99	13.411 ± 1.911	$\mathbf{8.112 \pm 0.894}$
PSO-DLS	1.227 ± 0.449	1.180 ± 0.319	1.127 ± 0.353	$\mathbf{7.461 \pm 0.807}$	7.227 ± 0.695	7.44 ± 0.775	$\mathbf{11.040 \pm 0.700}$	$\mathbf{10.967 \pm 0.709}$	10.873 ± 0.751
VPSO	1.42 ± 0.564	1.413 ± 0.566	1.413 ± 0.566	9.9 ± 1.361	9.853 ± 1.338	9.853 ± 1.338	15.213 ± 1.115	15.18 ± 1.131	15.18 ± 1.131

TABLE IV
MEAN AND STANDARD DEVIATION BEST FITNESS VALUES ACHIEVED OVER ALPINE #1 FUNCTION (f_3).

	D = 10			**D = 50**			**D = 100**		
Algorithm	**t = 100**	**t = 1,000**	**t = 10,000**	**t = 100**	**t = 1,000**	**t = 10,000**	**t = 100**	**t = 1,000**	**t = 10,000**
AIWPSO	1.601 ± 0.886	1.513 ± 0.903	0.419 ± 0.685	35.873 ± 7.125	34.934 ± 7.156	30.999 ± 6.844	85.195 ± 9.976	83.786 ± 9.777	79.708 ± 13.121
PSO	1.596 ± 1.054	1.444 ± 0.973	0.822 ± 0.712	35.415 ± 5.763	34.738 ± 5.804	30.744 ± 5.040	83.194 ± 9.589	82.511 ± 9.434	73.965 ± 8.748
RPSO	$\mathbf{0.130 \pm 0.130}$	$\mathbf{0.005 \pm 0.005}$	$\mathbf{0.000 \pm 0.000}$	22.666 ± 3.427	$\mathbf{9.572 \pm 2.584}$	$\mathbf{4.935 \pm 2.303}$	76.568 ± 4.781	$\mathbf{47.523 \pm 7.616}$	$\mathbf{32.721 \pm 8.042}$
PSO-DLS	0.510 ± 0.543	0.360 ± 0.284	0.178 ± 0.183	$\mathbf{21.503 \pm 3.621}$	21.519 ± 3.086	17.232 ± 3.871	$\mathbf{52.927 \pm 4.322}$	52.652 ± 3.992	42.862 ± 5.834
VPSO	1.285 ± 1.03	0.516 ± 0.443	0.302 ± 0.299	33.414 ± 6.842	29.783 ± 6.624	27.928 ± 6.413	78.773 ± 15.036	74.66 ± 15.796	72.465 ± 15.492

The experiments exhibited great and impressive results considering five benchmark functions and four baseline approaches for comparison purposes, achieving better minima over them all. Furthermore, one can observe that growing the number of dimensions, i.e., increasing the optimization complexity, the proposed approach was able to escape from strong local minima, which traped the compared baselines.

Also, the relativistic momentum provided opportunities for better exploitation and exploration by the swarm, taking into account the inertia that particles accumulate over the iterations, simultaneously to the natural acceleration or deceleration inherent by the Lorentz factor.

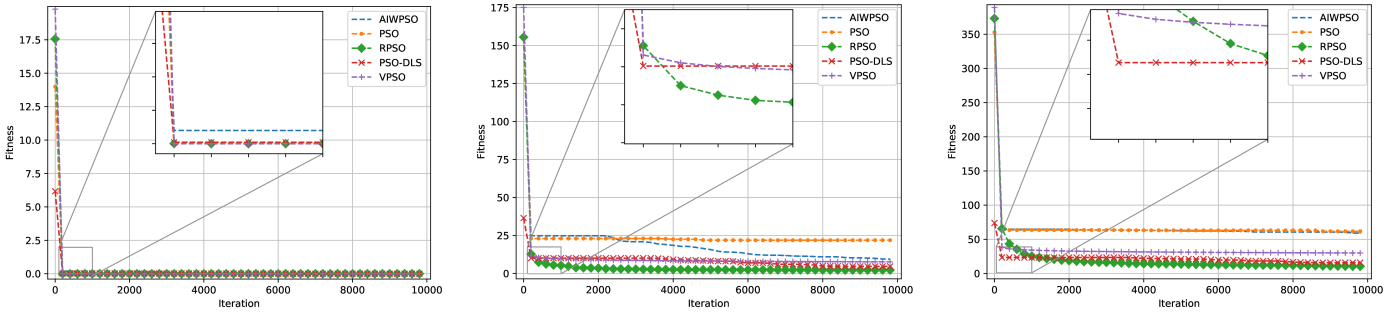
Regarding future works, we aim to deeply explore the mass' effect on RPSO performance, analyzing different distributions, or even achieving a straightforward correlation with other variables. Moreover, we consider applying the proposed

TABLE V
MEAN AND STANDARD DEVIATION BEST FITNESS VALUES ACHIEVED OVER RASTRIGIN FUNCTION (f_4).

	D = 10			D = 50			D = 100		
Algorithm	**t = 100**	**t = 1,000**	**t = 10,000**	**t = 100**	**t = 1,000**	**t = 10,000**	**t = 100**	**t = 1,000**	**t = 10,000**
AIWPSO	**22.922 ± 10.404**	22.859 ± 10.42	22.473 ± 10.279	346.246 ± 31.865	345.62 ± 32.005	292.589 ± 53.445	847.832 ± 79.37	845.891 ± 79.093	726.530 ± 103.608
PSO	**23.190 ± 12.263**	23.001 ± 12.326	22.992 ± 12.326	311.214 ± 28.381	307.965 ± 27.112	307.936 ± 27.129	838.335 ± 82.915	832.918 ± 80.977	830.185 ± 79.347
RPSO	**19.599 ± 8.852**	**6.067 ± 6.067**	**4.179 ± 4.179**	306.723 ± 34.497	**184.839 ± 34.462**	**122.796 ± 20.329**	759.052 ± 90.736	**571.774 ± 50.584**	**441.719 ± 52.780**
PSO-DLS	**19.296 ± 9.255**	**9.974 ± 5.343**	**8.260 ± 4.691**	320.255 ± 45.510	279.524 ± 50.155	249.476 ± 49.801	**724.119 ± 87.179**	659.626 ± 61.733	579.543 ± 55.305
VPSO	**21.868 ± 8.752**	21.756 ± 8.794	21.756 ± 8.794	**288.390 ± 28.800**	244.257 ± 30.140	216.787 ± 28.546	765.363 ± 70.529	696.636 ± 73.564	657.815 ± 78.257

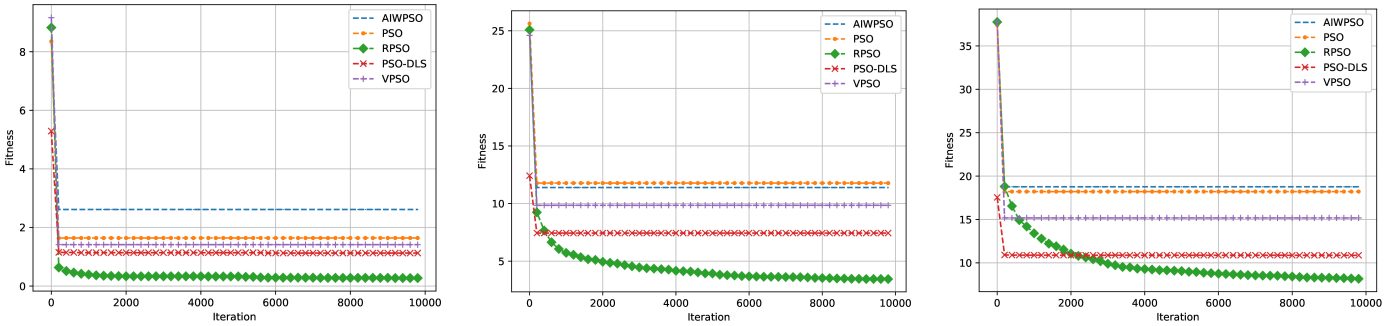
TABLE VI
MEAN AND STANDARD DEVIATION BEST FITNESS VALUES ACHIEVED OVER SCHWEFEL FUNCTION (f_5).

	D = 10			D = 50			D = 100		
Algorithm	**t = 100**	**t = 1,000**	**t = 10,000**	**t = 100**	**t = 1,000**	**t = 10,000**	**t = 100**	**t = 1,000**	**t = 10,000**
AIWPSO	1,629 ± 397	1,628 ± 397	1,623 ± 399	13,453 ± 780	13,433 ± 779	11,856 ± 1,143	30,861 ± 1,726	30,838 ± 1,723	28,015 ± 2,477
PSO	1,516 ± 381	1,511 ± 383	1,511 ± 383	12,220 ± 1,233	12,163 ± 1,249	12,153 ± 1,243	30,016 ± 1,893	29,898 ± 1,949	29,727 ± 2,075
RPSO	**701 ± 218**	**519 ± 259**	**448 ± 270**	**9,167 ± 1,033**	**5,145 ± 747**	**4,282 ± 654**	**25,102 ± 2,192**	**17,190 ± 1,838**	**12,799 ± 2,104**
PSO-DLS	1,656 ± 319	1,044 ± 278	664 ± 369	14,995 ± 735	12,635 ± 977	11,286 ± 837	32,949 ± 1,154	30,114 ± 1,281	27,804 ± 1,566
VPSO	1,478 ± 488	1,465 ± 484	1,465 ± 484	11,956 ± 1,398	10,979 ± 1,526	10,468 ± 1,487	30,084 ± 1658	28,252 ± 1,736	27,186 ± 1,910



(a) (b) (c)

Fig. 1. Convergence considering Sphere function over 10,000 iterations with 10 (a), 50 (b), and 100 (d) dimensions.



(a) (b) (c)

Fig. 2. Convergence considering Salomon function over 10,000 iterations with 10 (a), 50 (b), and 100 (d) dimensions.

approach and methodology in other benchmark functions. Besides, we also aim to employ the proposed approach to some applications, such as the fine-tuning of hyperparameters of artificial neural networks, Restricted Boltzmann Machines and Deep Belief Networks.

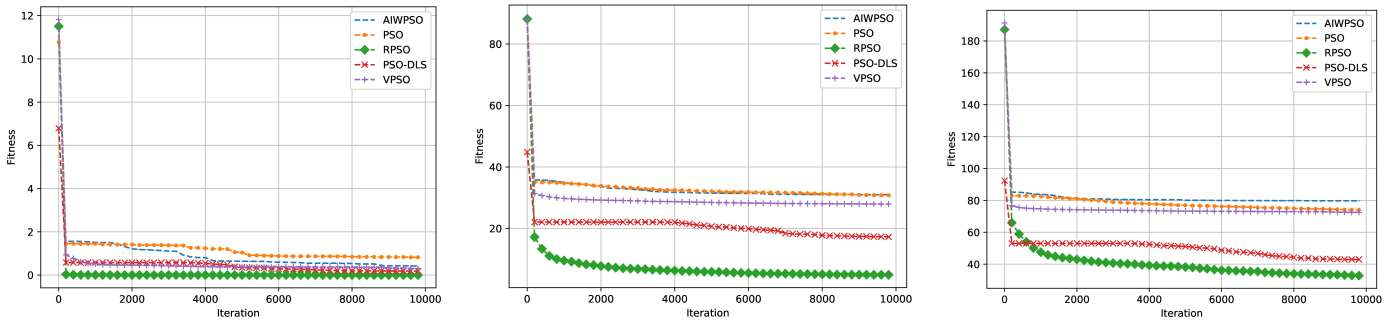
REFERENCES

[1] A. Törn and A. Žilinskas, *Global optimization*. Springer, 1989, vol. 350.

[2] B. Korte, J. Vygen, B. Korte, and J. Vygen, *Combinatorial optimization*. Springer, 2012, vol. 2.

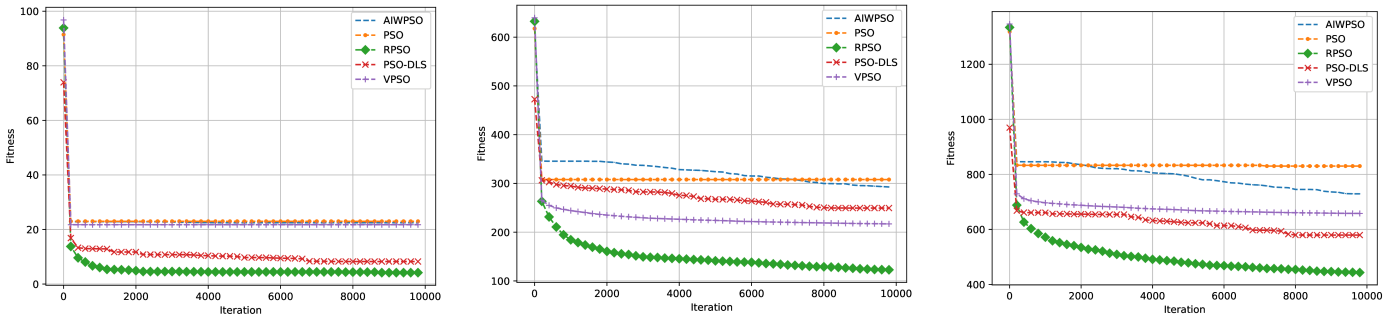
[3] Y. Bengio, "Gradient-based optimization of hyperparameters," *Neural computation*, vol. 12, no. 8, pp. 1889–1900, 2000.

[4] X.-S. Yang, "Metaheuristic optimization: algorithm analysis and open



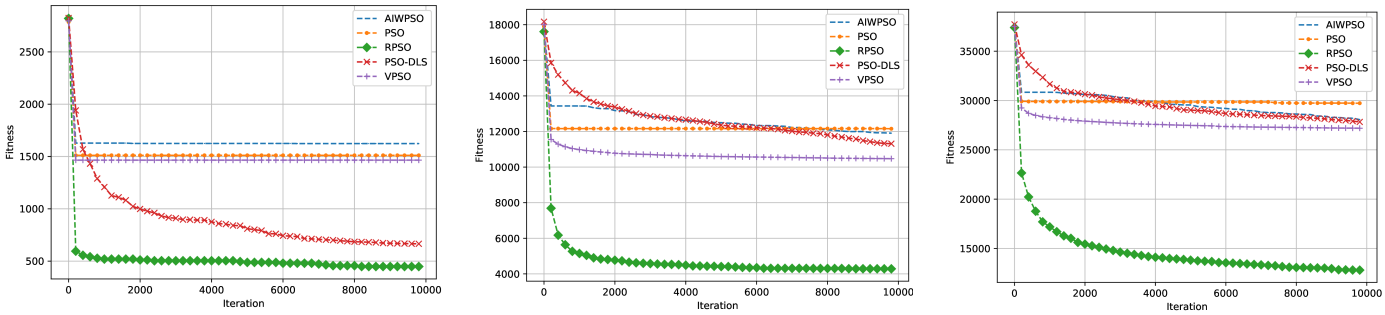
(a) (b) (c)

Fig. 3. Convergence considering Alpine #1 function over 10,000 iterations with 10 (a), 50 (b), and 100 (d) dimensions.



(a) (b) (c)

Fig. 4. Convergence considering Rastrigin function over 10,000 iterations with 10 (a), 50 (b), and 100 (d) dimensions.



(a) (b) (c)

Fig. 5. Convergence considering Schwefel function over 10,000 iterations with 10 (a), 50 (b), and 100 (d) dimensions.

problems," in *International Symposium on Experimental Algorithms*. Springer, 2011, pp. 21–32.

[5] J. P. Papa, G. H. Rosa, A. N. Marana, W. Scheirer, and D. D. Cox, "Model selection for discriminative restricted boltzmann machines through meta-heuristic techniques," *Journal of Computational Science*, vol. 9, pp. 14 – 18, 2015, computational Science at the Gates of Nature.

[6] L. A. Passos and J. P. Papa, "A metaheuristic-driven approach to fine-tune deep boltzmann machines," *Applied Soft Computing*, p. 105717, 2019.

[7] S. C. Yusta, "Different metaheuristic strategies to solve the feature selection problem," *Pattern Recognition Letters*, vol. 30, no. 5, pp. 525–534, 2009.

[8] Y. Chen, D. Miao, and R. Wang, "A rough set approach to feature selection based on ant colony optimization," *Pattern Recognition Letters*, vol. 31, no. 3, pp. 226–233, 2010.

[9] G.-G. Wang, A. Hossein Gandomi, X.-S. Yang, and A. Hossein Alavi, "A novel improved accelerated particle swarm optimization algorithm for global numerical optimization," *Engineering Computations*, vol. 31, no. 7, pp. 1198–1220, 2014.

[10] D. Rodrigues, G. H. de Rosa, L. A. Passos, and J. P. Papa, "Adaptive improved flower pollination algorithm for global optimization," in *Nature-Inspired Computation in Data Mining and Machine Learning*. Springer, 2020, pp. 1–21.

[11] R. Eberhart and J. Kennedy, "Particle swarm optimization," in *Proceedings of the IEEE international conference on neural networks*, vol. 4. Citeseer, 1995, pp. 1942–1948.

[12] A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh, "A novel particle swarm optimization algorithm with adaptive inertia weight," *Applied Soft Computing*, vol. 11, no. 4, pp. 3658–3670, 2011.

[13] W.-P. Yang, "Vertical particle swarm optimization algorithm and its application in soft-sensor modeling," in *2007 International Conference on Machine Learning and Cybernetics*, vol. 4. IEEE, 2007, pp. 1985–1988.

[14] W. Ye, W. Feng, and S. Fan, "A novel multi-swarm particle swarm optimization with dynamic learning strategy," *Applied Soft Computing*, vol. 61, pp. 832–843, 2017.

[15] A. Einstein, "Relativity: The special and general theory," *Holt and Company*, 1916.

[16] F. Wilcoxon, “Individual comparisons by ranking methods,” *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.
[17] G. H. de Rosa and J. P. Papa, “Opytimizer: A nature-inspired python optimizer,” 2019.