ISSN 2007-9737

A Social Learning Based Particle Swarm Optimization Algorithm for Real-Parameter Single Objective Optimization Problems

Emmanuel Martínez-Guerrero*

Instituto Politécnico Nacional,
Centro de Investigación en Computación,
México

emartinezg2025@cic.ipn.mx

Abstract. The Particle Swarm Optimization (PSO) algorithm is a simple and effective method that has been widely used to solve complex optimization problems. However, it can easily get trapped in a local optima due to the loss of population diversity. This paper presents a new variant of the PSO algorithm based on social learning (SL-PSO) that aims to improve performance of traditional PSO. This is encouraged by the ability shown by diverse animal species to learn from the behavior of more experienced individuals. Specifically, the historical information of the best particle is utilized to modify the position and direction of the stagnant particles, and improve the exploration capability of the swarm. Experiments conducted on unimodal and multimodal test functions demonstrate the effectiveness of the SL-PSO algorithm compared to other variants of the PSO algorithm.

Keywords. Particle swarm optimization, social learning, bio-inspired algorithms, real-parameter single objective optimization.

1 Introduction

Nowadays, solving optimization problems is becoming increasingly challenging due to factors such as a larger number of variables, the use of constraints, and the handling of nonlinear functions. In the field of optimization, it has been shown that bio-inspired methods can effectively solve various optimization problems within a reasonable timeframe [5]. By replacing exhaustive search methods, these algorithms have significantly reduced the computational cost.

Among the bio-inspired algorithms, the Particle Swarm Optimization (PSO) algorithm [4] stands out for its simplicity and its ability to quickly reach a good solution, allowing it to obtain favorable results in the solution of various real-world optimization problems [13, 7].

However, the PSO encounters challenges in achieving a balance between exploring and exploiting the search space, which can lead to premature convergence. In addition, a major drawback of the PSO is that it is prone to getting stuck in a local optimal solution [2]. Therefore, two important problems in the research of the PSO algorithm are how to increase population diversity to improve solution accuracy and how to avoid stagnation in local optimal solutions. Strategies to address these issues can be classified as follows:

(1) Parameter Control: The PSO algorithm has two main parameters, the constraint factor, and the inertia factor. The search capability of the PSO algorithm has been improved by linearly decreasing the inertia weight, by using a random inertia weight, and by using a fuzzy adaptive inertia weight [1].On the other hand, the convergence of PSO has been improved by applying the constraint factor proposed by Clerc and Kennedy [2].

(2) Hybridization: To improve the search capabilities of the PSO algorithm, operators from other algorithms such as differential evolution and

ant swarm optimization have been introduced into the structure of the PSO algorithm [7].

(3) Particle swarm topology: Modifying the topology of the particle swarm can enhance the performance of the PSO algorithm. The global topology allows a faster convergence; however, it can get trapped in a local minimum. On the other hand, the local topology allows to obtain a better solution, but with a slower convergence [7, 8].

Although there are several variants of the PSO algorithm that achieve better performance than the standard version, certain problems remain, such as the difficulty of implementation, a greater number of parameters to adjust, or a higher computational cost, making it necessary to investigate how to avoid stagnation in local optimal solutions, improve the accuracy of the solutions found, while maintaining an easy to implement structure.

Motivated by the above, this article proposes a new variant of the PSO algorithm for solving real-parameter single objective optimization problems. Following concepts and ideas of social learning, the proposed approach identifies the stagnation of the particles in the swarm and uses the historical information of the best particle in the swarm to modify the position and direction of the stagnant particles. The proposed algorithm is called Social Learning based Particle Swarm Optimization (SL-PSO).

The remainder of the paper is organized as follows. In section 2, preliminary concepts related to this work are introduced. The proposed SL-PSO is described in section 3. Experimental studies and results are provided in section 4. Finally, the conclusions are presented in section 5.

2 Preliminaries

This section introduces the main concepts used throughout this paper.

2.1 Real-Parameter Single Objective Optimization Problems

A real-parameter single objective optimization problem can be stated as follows: Given a function $f(X) : \mathbb{R}^N \rightarrow \mathbb{R}$, an algorithm has to find the values of the variables of a vector X such that they minimize or maximize the function f (called the objective function). In this type of problems, it is assumed that the vector X can have N variables $(X = (x_1, x_2, ..., x_N))$ and the search space within which the search for such a vector of variables is performed is bounded by the lower and upper bounds corresponding to each variable, i.e., $a_i \leq x_i \leq b_i$, $i \in \{1, 2, ..N\}$, where a_i is the lower bound and b_i is the upper bound corresponding to the variable i.

2.2 Standard Particle Swarm Optimization

The PSO algorithm was proposed by Kennedy and Eberhart in 1995 [4]. This optimization algorithm attempts to mimic the complex socio-cooperative behavior exhibited by different animal species such as flocks of birds and schools of fish.

The standard version of PSO [14] utilizes a swarm of NP particles, where each particle represents a potential solution to an optimization problem. This swarm is randomly placed within the N-dimensional search space of the optimization problem. For the ith particle of the swarm, its position and velocity at iteration t are denoted by $X_i = (x_{i1}, x_{i2}, ..., x_{iN})$ and $V_i = (v_{i1}, v_{i2}, ..., v_{iN})$ respectively. Then the new velocity and position of this particle at iteration $t + 1$ are calculated as follows:

$$\begin{aligned} V_{i,t+1} &= \omega V_{i,t} + c_1 r_1 (X_{ipbest,t} - X_{i,t}) \\ &\quad + c_2 r_2 (X_{gbest,t} - X_{i,t}), \end{aligned} \tag{1}$$

where $i = 1, 2, ..., NP$, c_1 and c_2 are two non-negative acceleration factors, r_1, r_2 are two uniformly distributed random numbers, $X_{ipbest} = (p_{i1}, p_{i2}, ..., p_{iN})$ is the best solution found by the ith particle itself until iteration t, and X_{gbest} is the best solution found by the entire swarm until iteration t. ω is the inertia weigth to balance the

global and local search abilities of particles in the search space, which is given by:

$$\omega = \omega_{max} - (\omega_{max} - \omega_{min})\frac{t}{T}, \quad (2)$$

where t represents the current iteration, ω_{max} and ω_{min} are the maximum and minimum limits respectively of the inertia weigth ω, and T is the maximum number of iterations. Then, the position of the particles is updated as follows:

$$X_{i,t+1} = X_{i,t} + V_{i,t+1}, \quad (3)$$

and check that $X_{i,t+1}$ stays within the search space delimited by its boundary constraints.

In Equation (1), the first term represents the inertia of the previous velocity, the second term is known as the cognitive component and is related to the personal experience of each particle, the third term is called the social component and is related to the cooperation between particles.

2.3 Social Learning

Learning in various animal species consists of a process that allows them to acquire, store and subsequently use information from their environment. Some species, particularly those that live in groups, have developed the ability to learn from other members of their group [6]. Experienced individuals, such as parents, as well as long-lived and successful members, provide a reservoir of useful behaviors. Adopting behaviors from this reservoir allows new or inexperienced individuals to skip the many iterations of trial-and-error necessary for their individual learning and move directly to solutions previously tested by other individuals. Through social learning, behaviors can be transferred from one generation to the next, resulting in more complex behaviors that help to solve tasks such as foraging and predator avoidance in an efficient way [6, 16].

3 Proposed SL-PSO

In this section, we describe our proposed algorithm, the Social Learning Particle Swarm Optimization algorithm, in detail.

3.1 Motivation

The SL-PSO algorithm is a variant of the standard PSO algorithm that utilizes the historical information of the search process, specifically the information of the best individuals during the search process. The decision to use this information is made with the aim of mitigating the stagnation of particles within the search process, taking into account the fact that different species of animals living in groups use this information and learn from the behavior of other members of the group through social learning.

3.2 Algorithm Description

The SL-PSO algorithm incorporates an external archive called A with size NP, which stores the best part of the swarm $(X_{gbest,t})$ in each iteration. Unlike the standard PSO, which uses only information from the current generation, the proposed algorithm uses both current and historical information through A.

When the ith particle does not improve its result considering the fit value between two consecutive iterations, a stagnation counter $C_s \geq 0$ increments its value by one unit. If the stagnation counter exceeds a set threshold (δ_s), this particle is subjected to a social learning process in which the particle uses the historical knowledge of the best particle in the swarm to escape a possible stagnation and at the same time allows it to move towards regions with a higher improvement potential. This intervention of the social learning process implies that the position of the swarm is modified as follows:

$$X_{i,t} = X_{i,t} + \mathcal{N}\left(0, \exp^{-t/10}\right) + \gamma(X_{A_{r,t}} - X_{i,t}), \quad (4)$$

where $X_{i,t}$ is the position of the particle just when the stagnation counter C_s exceeds the set threshold, $\mathcal{N}\left(0, \exp^{-t/10}\right)$ is a vector of the same size as $X_{i,t}$ with randomly generated values from a normal distribution with zero mean and standard deviation given by $\exp^{-t/10}$, t represents the current iteration, γ is a uniformly generated random number, and $X_{A_{r,t}}$ is one of the best swarm positions randomly selected from A.

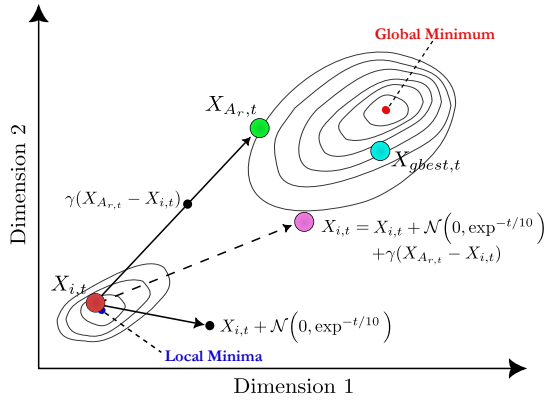


Fig. 1. Diagram showing the change in the position of a particle as a result of the process of social learning

Fig. 1 shows a diagram that illustrates the proposed modification. In this diagram, a particle $X_{i,t}$ (red circle) is close to a local minima, which can provoke the stagnation of the particle and thus finally activate the social learning process. In this process, the particle uses the information of one of the best particles of the swarm stored in the archive A (green circle) and a perturbation provided by the vector $\mathcal{N}\left(0, \exp^{-t/10}\right)$. This allows the particle to modify its position (magenta circle) to a new region of the search space and to improve its orientation with respect to the best particle of the swarm within the current generation (cyan circle).

Once the social learning process is applied to a particle, its stagnation counter is reset to zero. It is important to note that in the Equation (4), the vector $\mathcal{N}\left(0, \exp^{-t/10}\right)$ is intended to generate a random perturbation at the current position of the particle. Since this perturbation comes from a normal distribution with zero mean and standard deviation given by $exp^{-t/10}$, the values generated for this random perturbation vector tend to values closer to zero as the evolution process progresses. Thus, at the beginning of the algorithm execution, larger perturbations are obtained, which improve the exploration capacity of the algorithm, while at later stages of the evolution process, smaller perturbations are generated, which can benefit the local search of the algorithm. It is also worth noting that the speed of the particles is not modified within the social learning process and remains equal to the value obtained just when the social learning process is activated.

Initialize parameters:
$NP \leftarrow$ population size
$N \leftarrow$ the dimensionality of search space
$T \leftarrow$ the number of maximum iteration
$\delta_s \leftarrow$ stagnation threshold
$\omega_{max} \leftarrow$ maximum limit of inertia weigth
$\omega_{min} \leftarrow$ minimum limit of inertia weigth
$c_1, c_2 \leftarrow$ acceleration factors
$[X_{i_{min}}, X_{i_{max}}] \leftarrow$ the allowable position boundaries, $i = 1, 2, .., N$
Initialize Population:
Randomly initialize a swarm of NP particles
Randomly initialize particles velocities
Initialize pbest and gbest
Initialize archive and stagnation counter:
$A \leftarrow X_{gbest}$
$C_s = \emptyset$
for $t = 1, 2, ..., T$ **do**
  Update ω according to Equation (2)
  for *each particle* $X_i, i = 1, 2, .., NP$ **do**
    Update velocity according to Equation (1)
    Update position according to Equation (3) and check the boundaries
    if $f(X_{i,t+1}) < f(X_{ipbest})$ **then**
      $X_{ipbest} = X_{i,t+1}$
    else
      $X_{ipbest} = X_{i,t}$
      $C_{s_i} = C_{s_i} + 1$
    if $f(X_{i,t+1}) < f(X_{gbest})$ **then**
      $X_{gbest} = X_{i,t+1}$
    if $C_{s_i} > \delta_s$ **then**
      Social learning process:
      Modify position of X_i according to Equation (4)
      $C_{s_i} = 0$
  Add X_{gbest} to **A**
  Resize A if $|A| > NP$
  $t = t + 1$
Return the best solution

Fig. 2. SL-PSO algorithm

The pseudo-code in Fig. 2 provides a detailed description of the proposed SL-PSO algorithm.

4 Experimentation and Results

To evaluate the performance of SL-PSO, 10 benchmark functions are selected from the CEC2017 competition [10] as described in the section 4.1. The algorithms and parameter settings utilized in the tests are presented in section 4.2. The results of the comparison of SL-PSO with the standard PSO and other well-known variants of the PSO algorithm are presented in the section 4.3.

Table 1. Details of benchmark functions used in the experiments

Number	Function name	Type	Dimension (N)	Search space	Global optimum (F_{min})
F_1	Shifted and Rotated Bent Cigar	Unimodal	10/30	$[-100, 100]^N$	100
F_2	Shifted and Rotated Sum of Different Power	Unimodal	10/30	$[-100, 100]^N$	200
F_3	Shifted and Rotated Zakharov	Unimodal	10/30	$[-100, 100]^N$	300
F_4	Shifted and Rotated Rosenbrock	Multimodal	10/30	$[-100, 100]^N$	400
F_5	Shifted and Rotated Rastrigin	Multimodal	10/30	$[-100, 100]^N$	500
F_6	Shifted and Rotated Expanded Schaffer F6	Multimodal	10/30	$[-100, 100]^N$	600
F_7	Shifted and Rotated Lunacek Bi-Rastrigin	Multimodal	10/30	$[-100, 100]^N$	700
F_8	Shifted and Rotated Non-Continuous Rastrigin	Multimodal	10/30	$[-100, 100]^N$	800
F_9	Shifted and Rotated Levy	Multimodal	10/30	$[-100, 100]^N$	900
F_{10}	Shifted and Rotated Schwefel	Multimodal	10/30	$[-100, 100]^N$	1000

4.1 Benchmark Functions

To evaluate the performance of SL-PSO on different optimization problems, 10 benchmark functions were selected from the set of single-objective optimization problems with boundary constraints from the CEC2017 competition [10]. Three of the benchmark functions $(F_1 - F_3)$ are of the unimodal type, i.e., functions with a single global optimum, and the rest of them $(F_4 - F_{10})$ are functions with multiple local optima, also known as multimodal functions. The corresponding dimensions, search spaces, and values of their respective global optima are shown in Table 1.

4.2 Algorithms and Parameter Settings

The experiments were conducted using Python 3.11 programming language on a computer with 8 GB of RAM and a 3.6 GHz six-core processor. The performance of SL-PSO was compared with the standard PSO, the constraint factor PSO (CPSO) [2], the human behavior-Based PSO (HPSO) [11], and the PSO variant based on local stochastic search strategy (LSSPSO) [3] on the set of benchmark functions described in section 4.1.

In the simulations performed, all algorithms were run with the same number of iterations and particles: 1000 and 100, respectively. These values are commonly used in the PSO algorithm [9, 12]. For each algorithm, 51 independent runs were performed for each function with $N = 10$ and $N = 30$. The initial population was uniformly distributed over the search space. The inertia factor range was set to $[0.4, 0.9]$. Table 2 shows the configuration of the five algorithms. All the comparison algorithms adopt the authors' suggested parameter configurations.

4.3 Comparison of SL-PSO with PSO Algorithms

Tables 3 and 4 report the statistical results of the tests performed on the five algorithms for $N = 10$ and $N = 30$, respectively. The solution error measure $F_{min} - F(X^*)$ was used to obtain these results, where X^* represents the best solution found in each algorithm repetition, and F_{min} is the known solution to the problem. The mean and standard deviation of the solution error measure over 51 independent runs are shown for each function and algorithm. The best value for each function among all algorithms is shown in bold. To identify significant differences between the algorithms, the non-parametric Wilcoxon rank sum test with a confidence level of 0.05 was also implemented [15]. The values in the "W/T/L" row indicate the number of functions where the SL-PSO algorithm performed significantly better $(+)$, similar $(\approx)$, or worse $(-)$ than its counterpart. It is worth noting that the mean of the solutions is a measure of the solution quality of the algorithms, while the standard deviation is a measure of their stability.

According to the results presented in Table 3, SL-PSO achieves the best performance on the functions F_1,F_2,F_3,F_4,F_5,F_7,F_8 and F_{10}, while HPSO achieves the best performance on the

Table 2. Parameter settings

Parameter	Meaning	SL-PSO	PSO	CPSO	HPSO	LSSPSO
T	Number of iterations	1000	1000	1000	1000	1000
NP	Number of particles	100	100	100	100	100
C_1	Acceleration factor	2.0	2.0	2.0	N/A	2.0
C_2	Acceleration factor	2.0	2.0	2.0	N/A	2.0
ω	Inertia weight	Linear	Linear	Linear	Linear	Linear
δ_s	Stagnation threshold	15	N/A	N/A	N/A	N/A
χ	Constraint factor	N/A	N/A	0.7298	N/A	N/A
r	Constant	N/A	N/A	N/A	N/A	0.1

functions F_6 and F_9. PSO, CPSO, and LSSPSO do not perform well on any of the benckmark functions. As can be seen in this table, SL-PSO ranks first among all algorithms in terms of average mean performance, while the standard PSO algorithm ranks last. SL-PSO shows significant advantages except for three functions (F_6,F_7,F_9), but does not show significant disadvantages in any case. In addition, SL-PSO has the lowest standard deviation values for most functions, followed by HPSO.

For $N = 30$, the results in Table 4 indicate that SL-PSO performs best on the functions F_1, F_2, F_3, F_4, F_5, F_8, and F_{10}, while HPSO performs best on functions F_6, F_7, and F_9. It is worth noting that the SL-PSO algorithm performs well on both unimodal and multimodal functions. However, none of the PSO, CPSO, and LSSPSO algorithms perform well on any of the benchmark functions. As with the results for $N = 10$, PSO ranks last in terms of performance, with SL-PSO achieving the best performance, followed by HPSO. The HPSO algorithm presents significant advantages in functions F_6, F_7, and F_9, while SL-PSO has significant advantages on functions F_1, F_2, F_3, F_4, and F_{10}. In addition, SL-PSO achieves the lowest standard deviation values for the majority of functions, showing its ability to solve problems with multimodal functions.

Fig. 3 shows the convergence curves of the evolution process of SL-PSO, PSO, CPSO, HPSO, and LSSPSO. The benchmark functions F_3, F_6, and F_7 are used to exemplify the convergence characteristics for cases where $N = 10$ and $N = 30$, with $T = 1000$ and $NP = 100$. Each curve represents the average of 51 runs over a function, obtained by a specific algorithm.

Figs. 3(a) and 3(b) show that for the unimodal function F_3, SL-PSO achieves the highest accuracy in both 10 and 30 dimensions, while the rest of the algorithms maintain higher average error. Figs. 3(c) and 3(d) show that for the multimodal function F_6, HPSO produces the lowest mean error, while SL-PSO remains in second place. The remaining algorithms achieve higher mean errors. This behavior is observed in both 10 and 30 dimensions. Finally, Figs. 3(e) and 3(f) show that for the multimodal function F_7, all algorithms keep relatively close error means in both 10 and 30 dimensions. However, SL-PSO achieves the lowest mean error, followed by HPSO.

Based on the accuracy of the solutions (Tables 3 and 4), as well as the convergence characteristics (Fig. 3), it can be clearly stated that SL-PSO significantly outperforms PSO, CPSO, and LSSPSO, in both unimodal and multimodal functions, even as the number of dimensions increases. On the other hand, SL-PSO performs competitively with respect to HPSO mainly as the number of dimensions increases.

5 Conclusion and Future Work

In this paper, a new variant of the PSO algorithm called Social Learning based Particle Swarm Optimization (SL-PSO) is proposed. It is inspired by the fact that several species of animals living in groups use social learning, which allows them to use information and learn from other individuals in the group. Stagnant swarm particles are redirected to regions with higher improvement potential using

Table 3. Comparison results of four PSO algorithms with SL-PSO in 10 functions and 10 dimensions

Function	Metrics	SL-PSO	PSO	CPSO	HPSO	LSSPSO
	Mean	**3.0750E+04**	8.7415E+09	9.7242E+06	3.2958E+06	2.1328E+09
F_1	Std	**2.1906E+05**	9.8876E+09	6.4430E+07	5.0008E+06	5.1754E+09
	Rank	**1**	5	3	2	4
	Significance		+	+	+	+
	Mean	**2.6103E-03**	6.6244E+08	4.6415E+04	2.6280E+03	1.2007E+08
F_2	Std	**1.0826E-02**	1.9478E+09	2.8757E+05	4.9877E+03	4.6080E+08
	Rank	**1**	5	3	2	4
	Significance		+	+	+	+
	Mean	**2.7864E-14**	1.2900E+02	9.2323E-08	4.9061E+01	7.0305E+02
F_3	Std	**2.8699E-14**	5.3368E+02	6.0167E-07	4.5129E+01	2.4047E+03
	Rank	**1**	4	2	3	5
	Significance		+	+	+	+
	Mean	**2.2166E+00**	4.5452E+01	7.7978E+00	6.7092E+00	2.0047E+01
F_4	Std	3.1996E+00	5.6387E+01	1.2717E+01	**2.0264E+00**	3.4095E+01
	Rank	**1**	5	3	2	4
	Significance		+	+	+	+
	Mean	**1.3667E+01**	3.2589E+01	2.0087E+01	2.9228E+01	2.7216E+01
F_5	Std	**5.1568E+00**	1.1341E+01	1.0310E+01	6.6641E+00	1.1248E+01
	Rank	**1**	5	2	4	3
	Significance		+	+	+	+
	Mean	3.4429E+00	1.3974E+01	9.8673E+00	**2.6507E+00**	1.0049E+01
F_6	Std	2.1599E+00	1.0043E+01	7.2500E+00	**1.4455E+00**	6.4726E+00
	Rank	2	5	3	**1**	4
	Significance		+	+	≈	+
	Mean	**2.8192E+01**	3.1687E+01	3.8718E+01	2.9148E+01	3.6623E+01
F_7	Std	8.5294E+00	1.0700E+01	1.1472E+01	**8.1587E+00**	1.0198E+01
	Rank	**1**	3	5	2	4
	Significance		≈	+	+	+
	Mean	**1.3210E+01**	2.4956E+01	2.1728E+01	1.9494E+01	2.4347E+01
F_8	Std	**5.6112E+00**	1.0126E+01	9.3034E+00	5.7626E+00	1.0231E+01
	Rank	**1**	5	3	2	4
	Significance		+	+	+	+
	Mean	1.3980E+00	2.3724E+01	3.2838E+01	**6.6562E-01**	5.7464E+00
F_9	Std	2.0021E+00	5.2860E+01	3.6176E+01	**7.2203E-01**	8.3653E+00
	Rank	2	4	5	**1**	3
	Significance		+	+	≈	+
	Mean	**4.5699E+02**	9.6344E+02	7.7064E+02	1.1971E+03	8.5483E+02
F_{10}	Std	**2.1481E+02**	3.1029E+02	2.3985E+02	2.1976E+02	3.3255E+02
	Rank	**1**	4	2	5	3
	Significance		+	+	+	+
Average rank		**1.2**	4.5	3.1	2.4	3.8
Final rank		**1**	5	3	2	4
W/T/L		-/-/-	9/1/0	10/0/0	8/2/0	10/0/0

historical information from the best particle in the swarm.

This increases its exploration capability while maintaining a simple implementation that does not require the configuration of a large number of additional parameters.

The SL-PSO algorithm was tested on 10 single-objective optimization problems with boundary constraints from the CEC2017 competition.

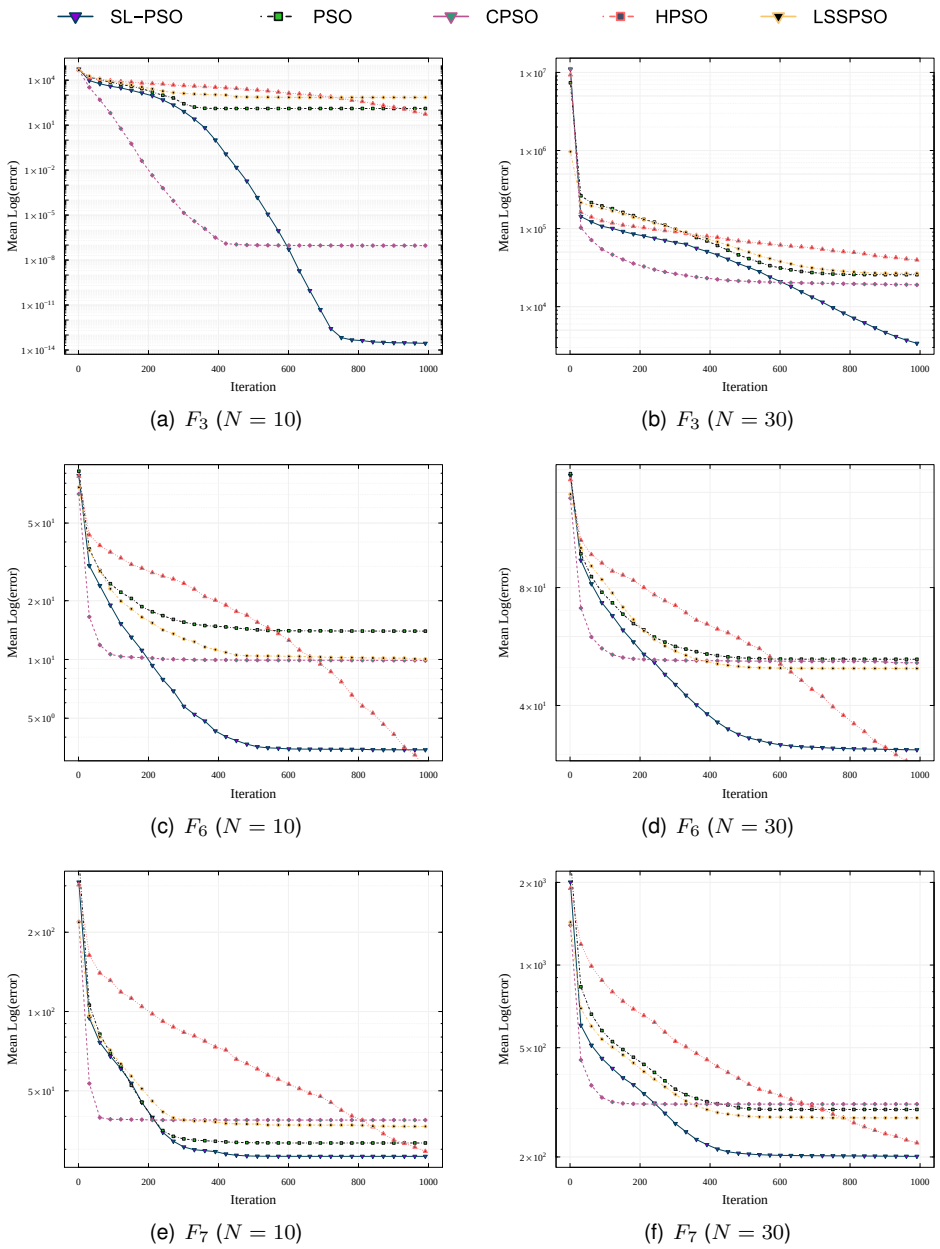
These problems require function minimization, which, due to their mathematical structure,

Table 4. Comparison results of four PSO algorithms with SL-PSO in 10 functions and 30 dimensions

Function	Metrics	SL-PSO	PSO	CPSO	HPSO	LSSPSO
	Mean	**1.2385E+07**	6.3888E+10	2.9003E+10	1.9072E+09	1.6355E+10
F_1	Std	**2.3163E+07**	6.3034E+10	1.7661E+10	1.1510E+09	3.9199E+10
	Rank	**1**	5	4	2	3
	Significance		+	+	+	+
	Mean	**6.3208E+22**	7.0401E+39	4.0791E+31	8.2597E+23	1.1473E+43
F_2	Std	**3.5340E+23**	2.8986E+40	2.8381E+32	5.7518E+24	8.1915E+43
	Rank	**1**	4	3	2	5
	Significance		+	+	+	+
	Mean	**5.1230E+02**	2.0306E+04	1.8521E+04	9.8818E+03	1.8536E+04
F_3	Std	**3.0889E+02**	2.1862E+04	6.9042E+03	3.6645E+03	2.5974E+04
	Rank	**1**	5	3	2	4
	Significance		+	+	+	+
	Mean	**1.8055E+02**	1.5944E+03	8.4919E+02	2.0912E+02	6.2117E+02
F_4	Std	6.4990E+01	1.6563E+03	4.2227E+02	**4.3289E+01**	7.4243E+02
	Rank	**1**	5	4	2	3
	Significance		+	+	+	+
	Mean	**1.0057E+02**	1.9181E+02	1.7107E+02	1.0556E+02	1.8041E+02
F_5	Std	2.2009E+01	4.8255E+01	3.6120E+01	**1.6339E+01**	4.1253E+01
	Rank	**1**	5	3	2	4
	Significance		+	+	≈	+
	Mean	2.9756E+01	5.1896E+01	5.1283E+01	**1.7762E+01**	4.9453E+01
F_6	Std	5.4414E+00	1.1102E+01	1.1844E+01	**4.0451E+00**	1.4327E+01
	Rank	2	5	4	**1**	3
	Significance		+	+	−	+
	Mean	1.9693E+02	2.9447E+02	3.1142E+02	**1.7496E+02**	2.7421E+02
F_7	Std	3.7166E+01	1.0133E+02	6.5002E+01	**2.3012E+01**	6.5471E+01
	Rank	2	4	5	**1**	3
	Significance		+	+	−	+
	Mean	**9.0583E+01**	1.5041E+02	1.3941E+02	9.6197E+01	1.4292E+02
F_8	Std	**2.0908E+01**	4.0311E+01	3.1154E+01	2.1218E+01	3.4358E+01
	Rank	**1**	5	3	2	4
	Significance		+	+	≈	+
	Mean	9.4548E+02	3.3655E+03	2.6033E+03	**3.9235E+02**	2.6611E+03
	Rank	2	5	3	**1**	4
F_9	Std	3.9333E+02	1.4856E+03	1.0561E+03	**2.0978E+02**	1.2553E+03
	Significance		+	+	−	+
	Mean	**3.2987E+03**	5.1759E+03	4.4595E+03	5.2560E+03	4.8679E+03
F_{10}	Std	**5.7575E+02**	8.4169E+02	7.4558E+02	9.8490E+02	9.6935E+02
	Rank	**1**	4	2	5	3
	Significance		+	+	+	+
Average rank		**1.3**	4.7	3.4	2	3.6
Final rank		**1**	5	3	2	4
W/T/L		-/-/-	10/0/0	10/0/0	5/2/3	10/0/0

provides insight into the performance of the proposal under complex scenarios.

The statistical results as well as the convergence curves confirm that SL-PSO is able to achieve a high level of performance concerning other PSO variants in both unimodal and multimodal functions, even when the number of dimensions are increased. In future works, we will focus on improving the performance of the SL-PSO algorithm and comparing it to other bio-inspired algorithms. Furthermore, the application of the proposed algorithm to solving real-world



(a) F_3 $(N = 10)$ (b) F_3 $(N = 30)$

(c) F_6 $(N = 10)$ (d) F_6 $(N = 30)$

(e) F_7 $(N = 10)$ (f) F_7 $(N = 30)$

Fig. 3. Convergence curves of SL-PSO, PSO, CPSO, HPSO, and LSSPSO for the selected functions. The cases of $N = 10$ and $N = 30$ are considered, with $T = 1000$ and $NP = 1000$. A logarithmic scale has been used for visualization purposes

optimization problems is another area to be investigated.

Acknowledgments

The author acknowledges support from the Mexican National Council of Humanities, Sciences and

Technologies (CONAHCyT) through a scholarship to pursue graduate studies.

References

1. **Bansal, J. C., Singh, P. K., Saraswat, M., Verma, A., Jadon, S. S., Abraham, A. (2011).** Inertia weight strategies in particle swarm optimization. 2011 Third World Congress on Nature and Biologically Inspired Computing, IEEE. DOI: 10.1109/nabic.2011.6089659.

2. **Clerc, M., Kennedy, J. (2002).** The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation, Vol. 6, No. 1, pp. 58–73. DOI: 10.1109/4235.985692.

3. **Ding, J., Liu, J., Chowdhury, K. R., Zhang, W., Hu, Q., Lei, J. (2014).** A particle swarm optimization using local stochastic search and enhancing diversity for continuous optimization. Neurocomputing, Vol. 137, pp. 261–267. DOI: 10.1016/j.neucom.2013.03.075.

4. **Eberhart, R. C., Kennedy, J. (1995).** A new optimizer using particle swarm theory. MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, pp. 39–43.

5. **Fan, X., Sayers, W., Zhang, S., Han, Z., Ren, L., Chizari, H. (2020).** Review and classification of bio-inspired algorithms and their applications. Journal of Bionic Engineering, Vol. 17, No. 3. DOI: 10.1007/s42235-020-0049-9.

6. **Galef, B. (2006).** Social Learning in Animals, Vol. 64. pp. . DOI: 10.1002/0470018860.s00721.

7. **Houssein, E. H., Gad, A. G., Hussain, K., Suganthan, P. N. (2021).** Major advances in particle swarm optimization: Theory, analysis, and application. Swarm and Evolutionary Computation, Vol. 63, pp. 100868. DOI: 10.1016/j.swevo.2021.100868.

8. **Kennedy, J., .** Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), IEEE. DOI: 10.1109/cec.1999.785509.

9. **Kennedy, J., Mendes, R., .** Population structure and particle swarm performance. Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600), IEEE. DOI: 10.1109/cec.2002.1004493.

10. **Kumar, A., Price, K. V., Mohamed, A. W., Hadi, A. A., Suganthan, P. N. (2016).** Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization. pp. .

11. **Liu, H., Xu, G., Ding, G.-y., Sun, Y.-b. (2014).** Human behavior-based particle swarm optimization. The Scientific World Journal, Vol. 2014, pp. 1–14. DOI: 10.1155/2014/194706.

12. **Piotrowski, A. P., Napiorkowski, J. J., Piotrowska, A. E. (2020).** Population size in particle swarm optimization. Swarm and Evolutionary Computation, Vol. 58, pp. 100718. DOI: 10.1016/j.swevo.2020.100718.

13. **Shami, T. M., El-Saleh, A. A., Alswaitti, M., Al-Tashi, Q., Summakieh, M. A., Mirjalili, S. (2022).** Particle swarm optimization: A comprehensive survey. IEEE Access, Vol. 10, pp. 10031–10061.

14. **Shi, Y., Eberhart, R. (1998).** A modified particle swarm optimizer. 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360), pp. 69–73. DOI: 10.1109/ICEC.1998.699146.

15. **Smalheiser, N. R. (2017).** Nonparametric Tests. Elsevier. DOI: 10.1016/b978-0-12-811306-6.00012-9.

16. **Whiten, A. (2019).** Cultural evolution in animals. Annual Review of Ecology,

Evolution, and Systematics, Vol. 50, No. 1, pp. 27–48. DOI: 10.1146/annurev-ecolsys-110218-025040.

Article received on 23/04/2024; accepted on 19/03/2025.
**Corresponding author is Emmanuel Martınez-Guerrero.*