Journal of Information & Optimization Sciences
ISSN 0252-2667 (Print), ISSN 2169-0103 (Online)
Vol. 47 (2026), No. 3, pp. 989–1010
DOI : 10.47974/JIOS-1787

Adaptively learning memory incorporating PSO

Dmytro Shchyrba
Faculty of Information and Communication Technology
Wroclaw University of Science and Technology
Wroclaw 51-627
Poland

Abstract

Swarm intelligence metaheuristics are a part of a large group of nature-inspired optimization algorithms. In this paper, we introduce a new PSO-inspired algorithm, that incorporates the positive experiences of the swarm to learn the geometry of the search space,thus obtaining the ability to consistently reach global optimum. During the comparative analysis, it has been found to consistently outperform the other swarm-intelligence based algorithms on the family of benchmarks.

Subject Classification: *65K10.*

Keywords: *Swarm intelligence, Optimization, Metaheuristics, Intelligent systems.*

1. Introduction

Optimization problems are fairly common in most complex human endeavors, such as engineering [9] [6], logistics [12] or artificial intelligence [1]. Currently, there are lots of methods available for finding either optimal or sub-optimal solutions for them, such as nature-based and gradient-based methods. Although gradient-based methods tend to have faster convergence to a solution, requirement of Jacobian and sometimes even Hessian matrix, becomes a strong disadvantage in real-life applications due to highly complicated objective functions, usually making computation of the derivatives unfeasible or even physically impossible. Thus, although perceived as slower comparatively to gradient-based, population-based algorithms are still sufficient for a wide variety of tasks. There is a thriving research field on building new metaheuristics, or improving existing

E-mail: 276186@student.pwr.edu.pl

© Taru Publications

algorithms via novel behavioural patterns[10] or efficient intialization techniques for the swarm population such as in the work of Khajeh and Arab [2]. Our work is mostly interested in nonsmooth semiconvex nondifferentiable optimization problems, where the number of cost evaluations is supposed to be as low as possible, but the cost function itself is nondeterministic and there is neither an analytical solution, nor possibility to apply gradient-based algorithms.

2. Backgrounds on PSO

Particle Swarm Optimization [4] is one of the most popular optimization algorithms. Being inspired by the social behaviour of birds, it has proven itself worthy in many cases and application. In PSO, search agents possess 2 cognitive factors for searching-social and individual.Each iteration, the coordinates are updated according to the equations

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{1}$$

$$v_i(t+1) = w \cdot r_i \cdot v_i(t) + c_1 \cdot r_i \cdot (g_i(t) - x_i(t)) + c_2 \cdot r_i \cdot (p_i(t) - x_i(t)) \tag{2}$$

Where x_i is the coordinates vector for i-th particle, v_i is respectively the velocity, w is the inertia hyperparameter, r_i is the random number generated from uniform distribution between, and c_1 and c_2 are the parameters for social and indiviual cognitive factors.

3. Proposed approach

The algorithm we are proposing consists of 3 main parts: incorporation of the preserved history into the swarm thinking abilities, reprojection across the best neighbors and natural selection component

1. Applications: The algorithm is especially focused on the multimodal large scale optimization tasks with a pattern towards the global optima. As it is shown in results section,it drastically outperforms it's opponents on the functions where optimization landscape has a tendency to bend towards the global optima. To obtain such abilities,three factors have been considered. Firstly, the direction choice. Inspired by the Unified PSO, after usual velocity update, particles are being projected across their best-performing neighbors. Secondly, the consistency of the learned tendencies of landscape being optimized. In order to make the algorithm less reliant on manually chosen parameters and free it from the demanded prior knowledge, it is learned on the run from preserved

history, concept of which will be explained in more detail in the next section.

2. Preserved history: To incorporate the information from the previous generations in a more effective way, we store the information about the best fitness values and their respective solutions in a set of size N_p, which is a custom parameter. Each of the objects which it contains possess 3 attributes-the coordinates, the fitness value, and the ,,age'' (the iteration when the solution was appended to the set).

3. Reprojection: The proposed approach involves two kinds or reprojections – involving individual and collective thinking components

3.1 *Individual:* Here is the update equation for an individual particle in

$$x_i(t+1) = x_i(t) + w_i(t+1) + 2 \cdot (l_i(t) - x_i(t)) \tag{3}$$

$$w_i(t+1) = -w_i(t) + 2 \cdot r_2 \cdot (x_i(t) - g_i(t)) + 2 \cdot r_2 \cdot (x_i(t) - p_i(t)) \tag{4}$$

Where l_i is the best-performing neighbor of the individual particle, thus following multidimensional ring topology [3] v_i is velocity, whose goal is to pull particles towards the global solution.

Secondly - custom coefficients are currently more of the structure element. As of now the main development and movement of the whole colony is more dependent on the other components of our algorithm, the purpose of the individual behaviour of each plant is ensuring of convergence, and allows for individual development of each plant genom. The choice of -2 as coefficients for social and cognitive elements in root factor is based on the fact that the easiest solution would be simply making the movement negative without any randomness. However, we insist on remaining some stochastic properties to ensure the certain level of diversity in colony behaviour.

3.2 *Collective*

3.3 *Theoretical Framework and Application:* In the realm of swarm intelligence algorithms, the utilization of weighted averages as a pivotal mechanism for search region exploitation is not a novel concept. This strategy has been effectively integrated into various algorithms, such as the Gravitational Search Algorithm (GSA)-PSO, Quantum Particle Swarm Optimization (QPSO), among others. Commonly, this approach operates under the hypothesis that, given substantial exploration capabilities or an efficacious initialization strategy, the global optimum tends to be proximal to the swarm's centroid, particularly nearer to the more proficient particles.

3.4 *Distinctive Aspects of weighted average application:* In our approach, the implementation of weighted averages diverges in two significant facets:

Historical Data Integration: Contrary to the traditional method of considering only the current positional and fitness data of the swarm, our approach incorporates a preserved historical perspective, as delineated in the section 'Preserved History'. This historical integration, while enriching the algorithm with past insights, also introduces potential complexities in heterogeneous search landscapes. Specifically, if the swarm initiates exploration in areas with diverse directional tendencies, discrepancies in historical memories may lead to a decoupling of the swarm's trajectory from its accumulated memory. Notwithstanding these challenges, our methodology exhibits robustness and effectiveness in multimodal functions with stable tendencies, effectively discerning promising search directions.

Dual-Function Utilization: The weighted average method serves a bifurcated purpose in our framework. Primarily, it contributes to generating a constellation of central points (as detailed later), thereby accelerating convergence. Its ancillary role lies in augmenting exploration capabilities by introducing an adaptive component into collective reprojection, which is steered by the backward average. This backward average, illustrated in sections A, B, C, is presumed to exhibit a positive divergence in environmental characteristics, as manifested through the integral of the objective function across their respective domains. Our algorithm exploits the tendency of optimized function to decrease, assuming that the complexity of the function being optimized is less then 0.5 (complexity measured in terms of the metric offered in section 5) Theoretical Contributions

$$\int_{F-\varepsilon}^{F+\varepsilon} g(x)dx < \int_{B-\varepsilon}^{B+\varepsilon} g(x)dx \tag{5}$$

Where epsilon is a distance metric,restricting the environments around the backward and forward weighted averages, B is the backward center, F is the forward weighted center, and g(x) is the function being minimized This assumption underpins the generation of subsequent points and collective reprojection in our model.

3.5 *Adaptive Reprojection Magnitude:* A crucial element in our algorithmic design is the incorporation of maximum deviation, which serves to tailor the magnitude of reprojection. This adjustment is attuned to the swarm's behavior and is contingent on the extent of exploitation within the current search region. By modulating reprojection in response

to these dynamics, our methodology achieves a nuanced balance between exploration and exploitation, enhancing the overall efficacy of the search process.

Firstly, here is the general equation of the weighted center.

$$\mu_w = \frac{\sum_{i=1}^{N_p}(w_i \cdot X_i)}{\sum_{i=1}^{N_p} w_i} \tag{6}$$

where w_i is the weight of i-th particle, X_i is a coordinates vector of the particle, and N_p is the number of particle in preserved memory. Here we see the equation of the weight formed by *j*-th function for an *i*-th preserved particle

$$w_{i,j} = \frac{f_j(t_i)}{a_i^{\alpha}} \tag{7}$$

$$t_i = \frac{c_i - \mu_c}{\sigma_c} \tag{8}$$

Let c_i represent the fitness of the i-th particle within the swarm. The parameters μ_c and σ_c denote the mean and standard deviation, respectively, of the fitness values as recorded in the preserved memory of the swarm. This historical data provides a contextual backdrop against which current fitness evaluations are assessed.

The term t_i refers to the preliminary weight assigned to the *i*-th particle. This initial weight is subject to further transformation, adhering to a specified mathematical relationship detailed in the equation.

An essential aspect of the weighting mechanism is the incorporation of particle age, represented as a_i^{α}. Here, a_i denotes the age of the *i*-th particle, and α is a tunable exponent with the constraint of being positive. The parameter α plays a pivotal role in modulating the swarm's sensitivity to temporal aspects of the search process. A higher value of α increases the swarm's responsiveness to solutions discovered in later generations, imbuing the search with a temporal depth. Conversely, reducing a_i^{α} shifts the focus predominantly towards immediate fitness values, deemphasizing the temporal dimension of the search.

In addition, $f_j(x)$ symbolizes one of the selected functions from a predefined set, each corresponding to a specific mean value within the algorithm's structure. The choice of these functions is fundamentally architectural, restricted to those with monotonically increasing derivatives. In scenarios where minimization is the objective, such as minimizing a

model's error, these functions are typically raised to the power of -1. This inversion aligns with the minimization goal, ensuring that the algorithm prioritizes lower function values, thereby steering the search towards optimal solutions.

This intricate weighting scheme, combining fitness evaluation, historical context, particle age, and function selection, exemplifies the algorithm's comprehensive approach. It encapsulates both the immediate and historical performance of particles, thereby balancing exploration and exploitation in the swarm's collective search dynamics.

In the subsequent stage of the algorithm, particle generation is meticulously orchestrated based on the number and nature of weight functions employed. This process is quantitatively defined as follows:

The algorithm initiates the creation of $2 \cdot f_n$ new particles, where f_n is a user-defined parameter that signifies the count of distinct weight functions being utilized within the algorithm's framework. This parameter, f_n, is central to determining the diversity and quantity of the new particles generated.

For each weight function, the algorithm generates a pair of particles. The first particle in this pair is directly derived from the weighted average calculated using the corresponding weight function. This weighted average particle embodies the collective characteristics influenced by the specific weight function, encapsulating the aggregate information processed through that function.

The second particle in each pair is generated through a process of reprojection. This reprojection is executed from a common reference point, denoted as μ_b, or backward weighted average. The reprojection involves recalculating the position of the particle by orienting it away from this less favorable average, thereby steering it towards potentially more promising regions of the search space.

This dual particle generation mechanism – one representing the weighted average and the other being reprojected from an unfavorable mean – is a strategic approach. It ensures that for each weight function, there is not only a representation of the collective trends dictated by that function but also an exploration away from less desirable regions. This balance between representation and exploration is crucial in enhancing the robustness and efficiency of the algorithm in navigating complex search spaces.

$$\mu_b = \frac{\sum_{i=1}^{N}(t_i \cdot X_i)}{\sum_{i=1}^{N} t_i} \tag{9}$$

Here is the equation for the bad mean, where N is a size of the swarm, X_i is a coordinate vector of *i*-th particle, and t_i is the inversed normalized fitness value of *i*-th particle(in case of maximization tasks it should the inversion is not required).

Then, the second particle in pair, is generated by reprojection of ,,bad mean'' across the corresponding weighted average, according to the equation

$$r = \mu_w - \mu_b \tag{10}$$

$$x_r = \mu_b + r \cdot \left(1 + \frac{\sigma}{\|r\|_n}\right) \tag{11}$$

Where μ_b is backward weighted average, r is the D-dimensional coordinates vector, containing the difference in position between the corresponding forward weighted average and the backward weighted center, $\| r \|_n$ is equivalent to Euclidean distance between them, x_r is a coordinates vector of a new particle, and σ is a span measure for the search space, considered at the moment.

$$\sigma = [\sigma_1, \sigma_2, \ldots, \sigma_{D-1}, \sigma_D]_{max} \tag{12}$$

As we can see here, sigma is an infinity order norm of the D-dimensional vector of standard deviations of coordinates in preserved history. The primary mechanism for directed movement within the swarm in our algorithm is the process of reprojection based on the optimal neighbors of each particle. This method may superficially resemble the numerical gradient estimation technique using finite differences, but fundamental differences exist between the two approaches.

In the finite difference method, a predetermined small value, commonly denoted as epsilon (ε), is used to approximate the gradient of a function. This method involves calculating the function values at points separated by ε and using these values to estimate the slope or gradient of the function (REF: [Standard reference on numerical methods]). Essentially, it employs a form of linear approximation where the choice of ε and the specific points of evaluation determine the accuracy of the gradient estimate.

Contrastingly, our algorithm adopts a distinct strategy where the 'epsilon' in each iteration, conceptualized as the distance between the backward and forward weighted averages, is not just a tool for estimation but serves as a significant metric in itself. This distance reflects the dynamic

scope of the search in the problem space and is adaptive to the swarm's behavior.

Moreover, in classical numerical gradient methods, irrespective of the chosen ε, there is always a fixed reference point. The method's nature (forward or backward difference) is defined by how this reference point is utilized. In our approach, however, the concept of a reference point is more fluid. The 'best neighbors' or the particle for which these neighbors are optimal can alternately function as reference points. This flexibility is crucial as it shifts the focus from absolute positions in the search space to the relative positioning and interactions within the swarm. We emphasize the relationships and dynamics between particles rather than their specific coordinates, aligning with the underlying principles of swarm intelligence where collective behavior and interaction patterns are paramount.

This fundamental difference in approach highlights the novelty of our method. By focusing on relative positions and adaptive distances within the swarm, our algorithm can effectively navigate complex search spaces, leveraging the collective intelligence and adaptability inherent in swarm-based techniques.

4. Natural selection: After each iteration, $2 \cdot f_n$ particles with the worst fitness value are removed from the swarm – according to the number of new particles generated by collective reprojection, thus leaving the size of the swarm the same.

Here are the steps of ALMI-PSO

Algorithm 1 ALMI-PSO

```
for generation = 1,2,...,t do:
    Update the preserved history
    Generate new particles based on preserved history and evaluate them
    Natural selection
    Update current best solution
    for plant = 1,2,...,N do:
        Update personal and social information
        Update velocity factor
        Update the coordinates
        Update fitness values
    end for
end for
```

4. Computational Complexity Analysis of a Typical Iteration

There are 3 key variables defining the computational complexity of an iteration:

- Search space dimensionality D
- Number of particles in local history N
- Number of particles in the swarm n

A typical iteration (if not the first) involves 5 main elements:

1. Compute of the dissimilarity matrix for the current swarm with complexity of $O((n \cdot D)^2)$.
2. Standardization of points costs in preserved history with linear complexity of $O(N)$, as we only parse a single vector to find its mean and variance.
3. Update of the preserved history, which can be computed in two ways:

   3.1 Adding a number based on its value to a sorted vector, with computational complexity of $O(n \log N)$.

   3.2 Concatenating the memory and current swarm and sorting the cost vector, resulting in complexity of $O((n+N)\log(n+N))$.

   We will assume the pessimistic approach with sorting.
4. Building adjacency matrix for the particles for finding local best for each one, with quadratic complexity of $O(n^2)$.
5. Generation of weighted averages and reprojections according to the preserved history with linear complexity of $O(N)$, as it is a set of operations practically identical to standardization.

Thus, we estimate the complexity as:

$$O((n \cdot D)^2 + n^2 + (n+N)\log(n+N) + 2N)$$

As the quadratic complexity of dissimilarity matrix computation is assumed to dominate the logarithmic and linear scaling of other operations, the compute scaling with parameter number growth of a single iteration will be defined as $O((n \cdot D)^2)$.

5. Algorithm Comparison on functions

We are going to assess the performance of our algorithm on two sets of benchmark functions. Firstly, we evaluate it's performance in comparison to other algorithms on functions suiting it's main abilities, and then we compare the results to the evaluation on CEC2017 test functions. Data about the performance of the other algorithms[11][8][7][5] is taken from the article [5], involving a suitably diverse set of functions for test. We show the conditions in tables 1-2

Table 1

Expressions of tendency-possessing functions

Name of the function	Way to refer	Expression
Sphere	STF1	$\sum_{i=1}^{M} x_i^2$
Schwefel's P2.22	STF2	$\sum_{i=1}^{n}\sum_{j=1}^{i} x_j$
Quadric	STF3	$\sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$
Step	STF4	$\sum_{i=1}^{n} \lfloor x_i + 0.5 \rfloor^2$
Schwefel's	STF5	$\sum_{i=1}^{n} random\,[0,1] \cdot x_i^4$
Rastrigrin	STF6	$10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$
Ackley	STF7	$-20\exp\left(-0.2\sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}}\right) - \exp\left(\frac{\sum_{i=1}^{n}\cos(2\pi x_i)}{n}\right) + 20 + \exp(1)$
Griewank	STF8	$f_{10}(x) = 1 + \frac{\sum_{i=1}^{n} x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right)$

Table 2

Bounds and optimal solutions of manipulating variables

Function	Initialization variable bounds	Optimum
STF 1	[–100, 100] M	0
STF 2	[–10, 10] M	0
STF 3	[–100, 100] M	0
STF 4	[–100, 100] M	0
STF 5	[–500, 500] M	0

Contd...

STF 6	[–5.12, 5.12] M	0
STF 7	[–32, 32] M	0
STF 8	[–600, 600] M	0

In our case, we use two function groups-polynomial and exponential $(e^x)^{-1}$, $(x^3)^{-1}$ – as a basic example of the exponential function and the cubic polynomial, to emphasize the importance of the power being odd in case of polynomial space. These are the general functions, which were chosen to show the general performance, without tuning the architecture to suit the certain purpose better. Raising them to the power of -1 directly corresponds to the fact that we will be solving minimization problems.

The age is going to have the power of 1 to preserve generality, colony size to 10, so that one-fifth of particles would be the generated ones, memory length to 60 and maximum number of function evaluations to 10000*D, where D is the dimensionality, making 30 runs for more thorough analysis.

1. **Analysis with Functions Exhibiting Stable Tendencies:** The initial phase of our testing regime involves a set of functions characterized by stable tendencies. These functions are selected to assess the algorithm's ability to efficiently navigate optimization landscapes with consistent directional gradients. This evaluation will provide insights into the algorithm's core optimization proficiency in relatively straightforward scenarios. The results are presented in tables 3-10.
2. **Assessment Using Rotated Function Sets:** Subsequently, the algorithm will be subjected to a series of rotated functions. This phase is critical for demonstrating the algorithm's invariance to rotational transformations of the search space. By evaluating the algorithm's performance on these rotated functions, we aim to ascertain its independence from orientation biases, a vital attribute for effective multidimensional optimization. Subsequently, the results are presented in tables 11-19.
3. **Testing on Functions with Diverse Directional Tendencies:** The final set of benchmark tests comprises functions that deviate from the aforementioned stable tendencies. These functions are characterized by regions exhibiting diverse directional tendencies, potentially leading to discrepancies in the algorithm's historical memory. Such a testing environment is crucial for examining the resilience of the algorithm when confronted with complex and unpredictable search landscapes. The presence of varied directional

cues challenges the algorithm's ability to maintain a coherent search trajectory, particularly in relation to its accumulated historical data. Their another prominent trait is that, unlike in the first 2 groups, this group has the complexity of strictly over 0.5 by the metric introduced earlier. Note: As a reference convex function we used the highest order single term polynomial possible, in all of the cases except sphere that would be linear-as for genuinely convex or unimodal functions they can be perfectly approximate by a polynomial, thus we were able to use quadratic. None of the test functions possessed singular peaks on the landscape, requiring separation into more than 2 intervals-were the first region is the overall optimization space, and the second is the region where the function is locally unimodal around it's global optimum. The results are presented in tables 19-21 in a slightly different form that for previous test groups - due to us conducting cross-evaluation analysis

For each category of benchmark tests, we plan to conduct a thorough comparative analysis of the algorithm's performance. This will involve not only direct comparisons within each function set but also cross-comparisons between the different sets to gain a comprehensive understanding of the algorithm's versatility and adaptability across diverse optimization challenges.

Furthermore, to quantitatively assess the diversity and differentiation among the performance of various algorithms, including ours, we will employ the Friedman test. This non-parametric statistical test will be applied to each group of functions, provided they exhibit sufficient variability in performance. The Friedman test is particularly suitable for identifying instances where different algorithms may exhibit dominance under specific conditions or within certain function categories. This statistical analysis will add a layer of rigor to our evaluation, ensuring that our findings are grounded in statistically valid comparisons.

In summary, this multi-faceted testing approach is designed to provide a holistic assessment of the algorithm's capabilities, revealing strengths, potential limitations, and areas for further refinement. The outcome of these tests will offer valuable insights, guiding future developments and optimizations of the algorithm. Here, we are going to define the variable O- any number closer to the value of a function in a global optima than a certain, accepted distance will be simply set to O,and considered global optimum of a function. As accepted distance, we use $4.94 \cdot 10^{-324}$, as we operate in 64bit numbers, and any distance lower is not deemed perceivable.

Table 3

Test results on stable-tendency functions - STF1

Statistic	ALMI-PSO	FDR-PSO	UPSO	TSL-PSO	GGL-PSOD	DMSDL-PSO	MLDE-PSO
FV	0.000E + 00	2.182E-30	1.283E-149	1.283E-149	2.557E-203	1.959E-57	0.000E + 00
SP	3.63E + 01	1.231E + 05	1.148E + 05	2.333E + 04	3.313E + 04	1.493E + 05	9.012E + 04
SR	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Rank	1	3	4	2	6	5	2

Table 4

Test results on stable-tendency functions - STF2

Statistic	ALMI-PSO	FDR-PSO	UPSO	TSL-PSO	GGL-PSOD	DMSDL-PSO	MLDE-PSO
BF2	0.000E + 00	3.696E-63	3.696E-63	1.490E-97	5.398E-33	1.208E-34	0.000E + 00
SP	0.000E + 01	1.225E + 05	3.136E + 04	4.434E + 04	1.657E + 05	7.310E + 04	2.663E + 04
SR	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Rank	1	6	3	4	2	5	2

Table 5

Test results on stable-tendency functions - STF3

Statistic	ALMI-PSO	FDR-PSO	UPSO	TSL-PSO	GGL-PSOD	DMSDL-PSO	MLDE-PSO
BF3	0.000E + 00	4.603E-150	4.603E-150	2.000E-203	8.137E-56	5.657E-65	0.000E + 00
SP	4.4E + 01	1.185E + 05	2.588E + 04	3.329E + 04	1.543E + 05	5.510E + 04	2.009E + 04
SR	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Rank	1	3	4	2	6	5	2

Table 6

Test results on stable-tendency functions - STF4

Statistic	ALMI-PSO	FDR-PSO	UPSO	TSL-PSO	GGL-PSOD	DMSDL-PSO	MLDE-PSO
FV	0.000E + 00	1.667E-01	1.667E-01	0.000E + 00	0.000E + 00	0.000E + 00	0.000E + 00
SP	2.8E + 01	1.474E + 05	1.138E + 04	1.652E + 04	1.115E + 05	2.296E + 04	7.964E + 03
SR	100.00%	83.33%	100.00%	100.00%	100.00%	100.00%	100.00%
Rank	1	3	3	2	2	2	

Table 7

Test results on stable-tendency functions - STF5

Statistic	ALMI-PSO	FDR-PSO	UPSO	TSL-PSO	GGL-PSOD	DMSDL-PSO	MLDE-PSO
FV	0.000E + 00	2.532E-03	2.532E-03	3.517E-03	2.471E-03	2.094E-03	1.953E-04
SP	4.0E + 01	Inf	Inf	Inf	Inf	Inf	Inf
SR	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Rank	1	4	5	6	3	2	7

Table 8

Test results on stable-tendency functions - STF6

Statistic	ALMI-PSO	FDR-PSO	UPSO	TSL-PSO	GGL-PSOD	DMSDL-PSO	MLDE-PSO
FV	0.000E + 00	2.802E + 01	2.802E + 01	0.000E + 00	5.329E-16	0.000E + 00	0.000E + 00
SP	4.0E + 01	Inf	Inf	3.349E + 04	2.178E + 05	6.946E + 04	2.322E + 04
SR	100.00%	0.00%	0.00%	100.00%	100.00%	100.00%	100.00%
Rank	1	3	4	1	2	1	1

Table 9

Test results on stable-tendency functions - STF7

Statistic	ALMI-PSO	FDR-PSO	UPSO	TSL-PSO	GGL-PSOD	DMSDL-PSO	MLDE-PSO
FV	0.000E + 00	2.315E-14	2.315E-14	3.417E-14	8.112E-15	6.454E-15	8.882E-16
SP	2.002E + 02	1.300E + 05	4.545E + 04	4.354E + 04	1.688E + 05	7.549E + 04	2.658E + 04
SR	100.00%	96.67%	100.00%	100.00%	100.00%	100.00%	100.00%
Rank	1	5	6	7	4	3	2

Table 10

Test results on stable-tendency functions - STF8

Statistic	ALMI-PSO	FDR-PSO	UPSO	TSL-PSO	GGL-PSOD	DMSDL-PSO	MLDE-PSO
FV	0.000E + 00	1.532E-02	1.532E-02	6.049E-13	3.281E-03	0.000E + 00	0.000E + 00
SP	4.0E + 01	6.346E + 05	1.160E + 05	3.443E + 04	2.423E + 05	6.193E + 04	1.901E + 04
SR	100.00%	36.67%	80.00%	100.00%	76.67%	100.00%	100.00%
Rank	1	5	6	3	4	2	2

From these results we can see the superiority of the ALMI-PSO on tendency-possessing functions both in term of success rate and convergency speed

Table 11

Test results on rotated functions - RF1

Statistic	ALMI-PSO	FDR-PSO	UPSO	TSL-PSO	GGL-PSOD	DMSDL-PSO	MLDE-PSO
FV	0.000E + 00	2.156E-114	1.252E-98	2.340E-121	1.161E-52	1.631E-43	5.373E-28
SP	3.63E + 01	1.198E + 05	3.040E + 04	4.859E + 04	1.535E + 05	7.736E + 03	1.009E + 05
SR	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Rank	1	3	4	2	5	6	7

Table 12

Test results on rotated functions - RF2

Statistic	ALMI-PSO	FDR-PSO	UPSO	TSL-PSO	GGL-PSOD	DMSDL-PSO	MLDE-PSO
BF2	0.000E + 00	3.911E-01	1.094E + 01	1.123E + 01	4.965E-04	1.991E-05	3.290E-11
SP	4.2E + 01	Inf	Inf	Inf	2.003E + 05	1.458E + 06	2.017E + 05
SR	100.00%	0.00%	0.00%	0.00%	93.33%	20.00%	100.00%
Rank	1	5	6	7	4	3	2

Table 13

Test results on rotated functions - RF3

Statistic	ALMI-PSO	FDR-PSO	UPSO	TSL-PSO	GGL-PSOD	DMSDL-PSO	MLDE-PSO
FV	0.000E + 00	6.727E-19	1.288E-10	1.872E-31	3.761E-12	8.727E-12	5.154E-16
SP	4.4E + 01	1.919E + 05	1.944E + 05	8.210E + 04	2.307E + 05	2.497E + 05	1.568E + 05
SR	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Rank	1	3	7	2	5	6	4

Table 14

Test results on rotated functions - RF4

Statistic	ALMI-PSO	FDR-PSO	UPSO	TSL-PSO	GGL-PSOD	DMSDL-PSO	MLDE-PSO
FV	0.000E + 00	4.100E + 00	0.000E + 00	9.667E-01	3.333E-02	0.000E + 00	0.000E + 00
SP	4.4E + 01	Inf	3.567E + 04	2.204E + 05	1.269E + 05	3.783E + 04	4.726E + 04
SR	100.00%	0.00%	100.00%	96.67%	96.67%	100.00%	100.00%
Rank	1	5	2	4	3	2	2

Table 15

Test results on rotated functions - RF5

Statistic	ALMI-PSO	FDR-PSO	UPSO	TSL-PSO	GGL-PSOD	DMSDL-PSO	MLDE-PSO
FV	0.000E + 00	2.637E-03	1.347E-02	5.054E-03	2.128E-03	3.495E-03	5.338E-03
SP	4.0E + 01	Inf	Inf	Inf	Inf	Inf	Inf
SR	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	90.00%
Rank	1	3	7	5	2	4	6

Table 16

Test results on rotated functions - RF6

Statistic	ALMI-PSO	FDR-PSO	UPSO	TSL-PSO	GGL-PSOD	DMSDL-PSO	MLDE-PSO
FV	0.000E + 00	4.985E + 01	9.170E + 01	8.626E + 01	3.323E + 01	3.293E + 01	1.810E + 02
SP	4.0E + 01	Inf	Inf	Inf	Inf	Inf	3.442E + 04
SR	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	100.00%
Rank	1	4	6	5	3	2	8

Table 17

Test results on rotated functions - RF7

Statistic	ALMI-PSO	FDR-PSO	UPSO	TSL-PSO	GGL-PSOD	DMSDL-PSO	MLDE-PSO
FV	0.000E + 00	1.866E + 00	2.281E + 00	2.519E + 00	1.036E-14	7.832E-09	8.882E-16
SP	0.000E + 00	4.340E + 06	Inf	7.666E + 05	1.746E + 05	1.833E + 05	2.795E + 04
SR	0.00%	6.67%	0.00%	30.00%	100.00%	100.00%	100.00%
Rank	1	5	7	8	2	4	1

Table 18

Test results on rotated functions - RF8

Statistic	ALMI-PSO	FDR-PSO	UPSO	TSL-PSO	GGL-PSOD	DMSDL-PSO	MLDE-PSO
FV	0.000E + 00	8.865E-03	2.712E-03	6.271E-04	3.122E-03	4.138E-12	0.000E + 00
SP	0.000E + 00	6.526E + 05	1.835E + 05	5.821E + 04	3.157E + 05	1.358E + 05	2.145E + 04
SR	100%	36.67%	70.00%	100.00%	66.67%	100.00%	100.00%
Rank	1	8	6	5	7	3	1

Here we can accept the rotation-invariance abilities of ALMI-PSO.

Table 19

Test results on CEC2017 functions

Function	ALMI-PSO	FDR-PSO	UPSO	TSL-PSO	GGL-PSOD	DMSDL-PSO	MLDE-PSO
F1	5.71e + 08	1.677E + 03	3.377E + 03	1.677E + 03	2.876E + 03	7.447E + 01	0.000E + 00
F2	3.92e + 03	4.914E + 10	1.486E + 16	4.914E + 10	1.509E + 06	1.948E + 10	1.667E + 00
F3	3.72e + 02	4.145E-08	4.754E + 01	4.145E-08	1.103E-07	1.199E-05	2.519E-09
F4	8.62e + 02	2.465E + 01	1.240E + 02	2.465E + 01	6.404E + 01	1.728E + 00	4.913E + 01
F5	5.00e + 02	5.877E + 01	8.236E + 01	5.877E + 01	3.042E + 01	3.841E + 01	2.584E + 01
F6	6.31e + 03	1.860E-01	1.294E + 00	1.860E-01	5.917E-08	1.137E-13	1.091E-07
F7	7.49e + 02	1.009E + 02	1.334E + 02	1.009E + 02	6.282E + 01	7.200E + 01	6.198E + 01
F8	8.01e + 02	5.860E + 01	8.430E + 01	5.860E + 01	3.279E + 01	4.059E + 01	2.790E + 01

And here,the group with negative results is presented,where there is a drastic performance difference seen.

6. Statistical Analysis

Analysis of Diversity Within Groups

In our research, we employed the Friedman test to statistically evaluate the performance variations of seven distinct implementations of Particle Swarm Optimization (PSO) across eight functions. This test examines the null hypothesis that repeated measurements on the same entities follow an identical distribution. It's noteworthy that the test's reliability, predicated on a chi-squared distribution assumption, is most robust when the sample size (n) exceeds 10 and includes more than six repeated measurements.

For the number of function evaluations (SP), a significant diversity in outcomes was evident at the 0.01 significance level, contingent on the algorithm variant utilized, keeping our requirement of diverse algorithm behaviours intact, in order to keep different efficiency levels and provide more unbiased analysis. Analogously, the function values (FV) demonstrated a similar pattern of significant variation at the same statistical threshold. These findings are particularly salient in the context of the ALMI-PSO algorithm, which consistently achieved the top rank (rank 1) across evaluations. This prominence was especially pronounced for rotated functions and stable tendency functions. However, in the case of the cec2017 functions, although a significant difference at the 1Cross-Group Comparative Analysis Wilcoxon Signed-Rank Test for Comparative Assessment

Further analysis was conducted using the Wilcoxon signed-rank test to compare paired samples' distributions. This test was applied to evaluate the consistency of rankings between different types of functions. It was observed that for rotated functions and stable tendency functions, the ALMI-PSO algorithm maintained consistent rankings. Conversely, for cec2017 functions, the rankings differed significantly at the 1Statistical Results

The results for RBF and stable tendency functions (both for FV and SP) indicated significant differences, as evidenced by the obtained p-values:

Table 20

Friedman Test Results for Function Values (FV)

Function set	Statistic	p-value
RF – FV	18.857	0.0044
STF – FV	29.017	6.04e-05
CEC2017 – FV	31.636	1.92e-05

Table 21

Statistical Test Results for Comparisons

Comparison	Statistic	p-value
RF vs. CEC2017	0.0	0.0078
STF vs. CEC2017	0.0	0.0078

There was no difference in ranks between RF and STF set, as they were identical

Summary of Findings

The statistical analysis indicates that the ALMI-PSO algorithm demonstrates superior performance for rotated and stable tendency functions. However, its efficacy varies notably with cec2017 functions. These results suggest that ALMI-PSO is particularly effective for specific types of functions, underscoring the importance of algorithm selection based on the function architecture in optimization tasks.

As we can see, in case of functions fulfilling it's assumptions ALMI-PSO consistently outperforms the other versions by a wide margin, not only being more efficient in finding the global minima, but also doing it considerably faster than the other versions, although it's performance drastically falls back on functions falling out of it's range.

To compare the performance results of our algorithm between the functions that contain the tendency trait, and these that do not, we apply the Friedman's test, that shows the test statistic of 2.714

7. Conclusions

In this paper, a new version of PSO for determining the parameters of low pass filters, such as exponential smoothing is proposed. By incorporating the learning of the search space via preserving positive

experiences, it also speeds up the convergence and offers high capabilities in finding global optima on multimodal functions with stable tendencies towards global optima,although losing in performance on function without that trait. Tests on it's primary application, searching for the perfect smoothing parameter show it's effectiveness.

8. Declarations

Ethical Approval: Not applicable

Funding: Not applicable

Data avialability statement: The authors declare that all the data supporting the findings of this study are available within the paper.

Competing Interest declaration: The authors declare that no funding was received to assist with the preparation of this manuscript.

Acknowledgments: Izabela Aleksandra Paniczek for valuable help in performance analysis.

References

[1] R. Abdulkadirov, P. Lyakhov, and N. Nagornov, "Survey of optimization algorithms in modern neural networks," *Mathematics,* vol. 11, no. 11, pp. 2466 (2023).

[2] M. R. Ghasemi Atieh Khajeh and H. Ghohani Arab, "Modified particle swarm optimization with novel population initialization," *Journal of Information and Optimization Sciences*, vol. 40, no. 6, pp. 1167–1179 (2019).

[3] 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).

[4] J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proceedings of ICNN'95-International Conference on Neural Networks, vol. 4, pp. 1942–1948, *IEEE* (1995).

[5] A. Lin, S. Li, and R. Liu, "Mutual learning differential particle swarm optimization," *Egyptian Informatics Journal*, vol. 23, no. 3, pp. 469–481 (2022).

[6] M. Nutakki and S. Mandava, "Review on optimization techniques and role of artificial intelligence in home energy management

systems," *Engineering Applications of Artificial Intelligence*, vol. 119, p. 105721 (2023).

[7] K. E. Parsopoulos and M. N. Vrahatis, "UPSO: A unified particle swarm optimization scheme," in International Conference of Computational Methods in Sciences and Engineering (ICCMSE 2004), pp. 868–873, CRC Press (2019).

[8] T. Peram, K. Veeramachaneni, and C. K. Mohan, "Fitness-distance-ratio based particle swarm optimization," in Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706), pp. 174–181 (2003).

[9] S. P. S. Rajput and S. Datta, "A review on optimization techniques used in civil engineering material and structure design," *Materials Today: Proceedings,* vol. 26, pp. 1482–1491 (2020).

[10] S. Sengupta, S. Basak, and R. A. Peters, "Particle swarm optimization: A survey of historical and recent developments with hybridization perspectives," *Machine Learning and Knowledge Extraction*, vol. 1, no. 1, pp. 157–191 (2019).

[11] G. Xu, Q. Cui, X. Shi, H. Ge, Z.-H. Zhan, H. P. Lee, Y. Liang, R. Tai, and C. Wu, "Particle swarm optimization based on dimensional learning strategy," *Swarm and Evolutionary Computation,* vol. 45, pp. 33–51 (2019).

[12] Y. Zhou, W. Xia, and J. Dai, "The application of nature-inspired optimization algorithms on the modern management: A systematic literature review and bibliometric analysis," *Journal of Management & Organization,* vol. 29, no. 4, pp. 655–678 (2023).

Received December, 2023