


RESEARCH

Open Access



A novel improvement of particle swarm optimization using an improved velocity update function based on local best murmuration particle

Elvis Twumasi^{1*} , Emmanuel Asuming Frimpong¹, Nicholas Kwesi Prah II¹ and David Boah Gyasi¹

*Correspondence:
etwumasi.coe@knust.edu.gh

¹Department of Electrical
and Electronic Engineering,
Kwame Nkrumah University
of Science and Technology
Kumasi, Kumasi, Ghana

Abstract

Improvement of particle swarm optimization (PSO) is relevant to solving the inherent local optima and premature convergence problem of the PSO. In this paper, a novel improvement of the particle swarm optimization is provided to curb the problem of the classical PSO. The proposed improvement modifies the updating velocity function of the PSO, and it uses a local best murmuration particle which is found using the k-means clustering technique. In this contribution, each particle moves towards the global best position by not only using the personal best and global best, but particles are modelled to move in murmuration towards the global best using the personal best, global best and a local best particle known as the local best murmuration particle. The improved model was tested against the traditional PSO and two other variants of the PSO and genetic algorithm (GA) using 18 benchmark test functions. The proposed improvement demonstrated superior exploration abilities by achieving the best optimum values in 15 out of 18 functions, particularly in the multimodal functions, where it achieved the best optimum value in all 6 cases. It also achieved the best worst-case values in 12 out of 18 functions, especially in the variable-dimension functions, where other algorithms showed significant escalation, indicating the proposed improvement's reliability and robustness. In terms of convergence, the proposed improvement exhibited the best convergence rate in all 18 functions. These findings highlight the impressive ability of the proposed improvement to converge swiftly without compromising accuracy.

Keywords: Particle swarm optimization, Swarm intelligence, Murmuration, K-means

Introduction

The use of the collective behaviour of animals as evolutionary optimization algorithms in finding optimal solution to real-life problems has increased over the past decades [1, 2]. These heuristic algorithms provide simple step-by-step approach in solving problems that traditional linear and nonlinear programming techniques would find difficulty in solving. Over the years, new metaheuristic techniques have been developed using the behaviour of several animals as additions to the popular

particle swarm optimization and genetic algorithm [2–7]. Though new algorithms have come in over the years, the efficiency of the particle swarm optimization cannot be overemphasized. PSO is well known and is a widely used algorithm because of its simplicity, high efficiency and accuracy [8]. The superiority of PSO over other algorithms has been shown in [8–10]. Notwithstanding, the efficacy and simplicity of the PSO algorithm as compared to other techniques, the PSO has a major drawback of sometimes converging at local optima and premature convergence. Many researchers have over the years provided various variants of the PSO to address these challenges. The work in [11] made the acceleration constants of PSO adaptive to optimize the two acceleration constants in the updating velocity function. Xu et al. [12] also used an adaptive weight to optimize the inertia weight and other variants to simultaneously remodel the inertia weight and acceleration constants [13, 14]. An intensive study has shown that the focus of researchers has mainly been on modifying either the inertia weight or the acceleration constants of the velocity function of the PSO in order to solve the local optima and premature convergence problem of the PSO.

The velocity function for the updating of particles positions of the PSO was modelled based on the social behaviour of a flock of birds when searching for food, where each particle (bird) moves towards the best position where there is food (global best) by looking at the position of the global best and its personal knowledge of where there is food (personal best). However, recent study and models have shown that birds also move to their global best position in murmuration. Murmuration of birds refers to the pattern-like movement of birds towards an area in order to keep warmth, stay away from predators and also create beauty [15]. According to [16] there is a ‘sound of low murmur’ birds at an area make from several wingbeats and soft flight calls in order to move in murmuration towards their proposed place of more food. The work in [17] who modelled the behaviour of birds in murmuration discovered that flock of birds in murmuration towards a new position or place adapts their direction of flight and speed or velocity to about seven of the birds that fly closest to it. Richardson and Chemo [18] also revealed that birds in a flock can influence one another to account for their speed variability within groups inside of the flock. Smaldino [19] also discovered that when one bird changes its motion, its neighbours imitate it and change spreads within the subgroup of the flock in order to change speed and lead to a collective beautiful display called murmuration. Evidence from biological data reiterates that there is a group-to-group speed variability among birds in search for food [20]. In a flock with many birds, it is clear that not every bird will be able to keep track of the other birds but rather move in sub groups [21]. Moreover, [22, 23] has shown that a ‘cheerleader’ bird always serves as the cornerstone for the self-organized speed variation of each bird in each subgroup inside of a flock. Cavagna et al. [24] recommended that we need to consider the velocity of the murmuration as a whole, in modelling the velocities of individual birds.

The above shows that each bird in an area does not only move towards the global best position by just looking at its personal best position but also imitate the movement of a leading bird in its locality. Inspired by these phenomenal behaviours of birds in choosing a local leader to murmurate towards the global best position, this contribution remodels the velocity updating function of the particle swarm optimization to include a local best

murmuration particle in other to solve the local optima problem and also ensure fast convergence of the PSO.

The remaining sections of this paper are organized as follows: In section “[Introduction](#)”, the concept of particle swarm optimization and k-means clustering that was used to cluster particles into groups is discussed. The novel approach in modifying the velocity function of the PSO using a local best murmuration particle is detailed in section “[Methodology](#)”. Section “[Methodology](#)” presents the testing functions used to test the efficacy of the improved PSO algorithm against the classical PSO and another variant of the PSO. The results and analysis are detailed in sections “[Results and discussion](#)” and “[Conclusion](#)” concludes the paper.

Particle swarm optimization

The particle swarm optimization algorithm is one of the most widely used optimization algorithms. It has its applications in Engineering, Medical Science, Business, and Social Science, among others. The technique was put forward by two American scholars, Eberhart and Kennedy, in the year 1995. According to the researchers, there is an invisible communication within a flock of birds that are seen dispersed in their search for food making them able to find the best place in their search space. Inspired by this phenomenon, Eberhart and Kennedy simulated birds’ foraging behaviour and proposed particle swarm optimization. After their proposition, the PSO has seen exponential growth because of its simplicity and accuracy in determining optimal solutions. The PSO as a heuristic algorithm optimizes a problem by iteratively providing candidate solutions known as particles. At the initialization phase of the PSO, these particles are randomly chosen within specified boundary conditions together with two random values (r_1 and r_2), two acceleration constants (c_1 and c_2), and inertia weight (w) that helps in updating the velocity of each particle. The updating velocity function is shown in Eq. (1). Before each particle’s velocity is updated, the best position each particle knows as the best candidate solution (personal best) and the overall best position in the swarm of particles known as the global best are chosen when each particle runs through the modelled function of the given problem (objective function). The updated velocity is used to update each particle position using Eq. (2). This process is continued until the best position is found among the particles or until a stopping criterion is met (e.g., total number of iterations).

$$V_i(t + 1) = wV_i(t) + r_1 c_1 (P_{ibest} - X_i(t)) + r_2 c_2 (G_{best} - X_i(t)) \quad (1)$$

$$X_i(t + 1) = X_i(t) + V_i(t + 1) \quad (2)$$

where

- $V_i(t)$ is the velocity of particle i at time t .
- w is the inertia weight.
- c_1 , c_2 and c_3 are acceleration coefficients.
- r_1 , r_2 and r_3 are random numbers uniformly distributed between 0 and 1.
- $X_i(t)$ is the position of particle i at time t .
- P_{ibest} and G_{best} are the personal best and global best, respectively.

Pseudocode for PSO

```

Define Objective function  $f(x) = f(x_1, \dots, x_d)^T$ 
Define decision vector characteristics and PSO parameters.
Initialize particles positions and velocities
Evaluate solutions and find globalbest

Do while iter <= maxIter
  Update the velocities and positions of the particles
  Apply position and velocity limits
  Evaluate Solutions
  Update personal best and global best
  Apply weight damping factor
  Increase the number of iter

End while
Display results

```

K-means clustering algorithm

The k-means clustering algorithm is a hard computing or crisp clustering technique, widely used in pattern recognition and data analysis [25]. It was first introduced by MacQueen in 1967. Unlike soft computing clustering techniques such as the fuzzy c -means (FCM) algorithm, which allows each data point to belong to multiple clusters with varying degrees of membership, the k-means algorithm assigns each data point to exactly one cluster.

Given a set of d -dimensional data points, the objective function for the k -means algorithm is to minimize the sum of squared distances (SSD), which is defined as the sum of the squared Euclidean distances between each data point and its assigned cluster centre. The algorithm follows an iterative process to find optimal cluster centres and assign data points to the nearest clusters. The steps involved in the k-means clustering algorithm are as follows.

Initialization Choose the number of clusters, k and randomly initialize k cluster centres.

Assignment Step For each data point x_i , calculate the squared Euclidean distance to each cluster centre (Eq. 3) and assign the data point to the cluster with the nearest centre.

$$d(p, q) = \sum_{i=1}^n (p_i - q_i)^2 \quad (3)$$

where $d(p, q)$ = squared Euclidean distance between point p and centroid q . p_i = i th dimension of point p . q_i = i th dimension of centroid q . n = number of dimensions.

Update Step Recalculate the cluster centres by taking the mean of all data points assigned to each cluster.

Iteration Repeat the assignment and update steps until convergence or a termination criterion is met. Convergence can be determined by checking if the cluster centres remain unchanged or if the objective function value falls below a certain threshold.

The termination criterion typically used is when the change in cluster centres or the SSD between iterations is below a predefined threshold. The k-means clustering algorithm is computationally efficient and widely applicable for various clustering tasks. The process is summarized in the pseudocode below:

- Initialize the number of clusters 'k' and maximum number of iterations 'MaxIt'.
- Randomly initialize the centroids for each cluster.

Repeat the following steps until convergence or maximum iterations reached:

- Assign each data point to the nearest centroid based on squared Euclidean distance
- Recalculate the centroids by taking the mean of all data points assigned to each centroid.
- If the centroids have not changed significantly or the maximum iterations have been reached, exit the loop.
- Otherwise, go back to first step in this loop.

Return the final centroids and the cluster assignments for each data point.

Methodology

Testing benchmark functions and PSO parameters

Eighteen benchmark functions were used to evaluate the new PSO variant [26]. To test the algorithm's ability to exploit the search space, unimodal functions ($F1$ to $F6$) were utilized. Multimodal functions ($F7$ to $F12$) were employed to assess diversity and exploration. Noisy functions with Gaussian distribution noise ($F13$ to $F16$) were used to evaluate adaptiveness and robustness. Finally, variable dimension functions ($F17$ & $F18$) were applied to test the scalability of the algorithm. These benchmark functions are standard benchmark functions for assessing the performance of optimization algorithms. A comparative analysis was performed involving the enhanced PSO, the original PSO [27], PSO-AWDV (which incorporates an adaptive weighted delay strategy) [12], PSOEA (a variant combining Evolutionary Algorithm with PSO) [28], and Genetic Algorithm (GA) [29]. Table 1 shows the equations for the functions, while Table 2 shows the other specific details of the functions:

The parameters used at the initialization phase of the PSO and comparison between the algorithms are shown in Table 3. Also, the random values used $r1$, $r2$ and $r3$ used in the velocity updating function were generated using the random generator in MATLAB.

All five algorithms were executed within a MATLAB R2021a environment on a consistent computer setup: an Intel® Core™ i7-7500U with CPU running at 2.70 GHz to 2.90 GHz and equipped with 12 GB RAM. The comparative analysis was conducted using the following parameters: Optimum Cost, Mean Absolute Error (MAE), and Standard Deviation (SD). The calculations for these parameters are as follows.

Table 1 Function equations

No	Function name	Equation
F1	Rotated hyper-ellipsoid	$f(x) = \sum_{i=1}^d \sum_{j=1}^i x_j^2$
F2	Zakharov	$f(x) = \sum_{i=1}^n x_i^2 + \left(\sum_{i=1}^n 0.5 \cdot ix_i \right)^2 + \left(\sum_{i=1}^n 0.5 \cdot ix_i \right)^4$
F3	Dixon-price	$f(x) = (x_1 - 1)^2 + \sum_{i=2}^n i \cdot (2x_i^2 - x_{i-1})^2$
F4	Sum of different powers	$f(x) = \sum_{i=1}^d x_i ^{i+1}$
F5	Bohachevsky 1	$f(x) = x_1^2 + 2x_2^2 - 0.3 \cos(3\pi x_1) - 0.4 \cos(4\pi x_2) + 0.7$
F6	Matyas	$f(x, y) = 0.26(x^2 + y^2) - 0.48xy$
F7	Drop-wave function	$f(x) = \frac{1 + \cos\left(\frac{12\sqrt{x_1^2 + x_2^2}}{0.5(x_1^2 + x_2^2) + 2}\right)}{0.5(x_1^2 + x_2^2) + 2}$
F8	Rastrigin	$f(x) = 10n + \sum_i (x_i^2 - 10 \cos(2\pi x_i))$
F9	Ackley	$f(x) = 20e \left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2} \right) - e \left(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i) \right) + 20 + e$
F10	Levy	$f(x) = \sin^2(\pi w_1) + \sum_{i=1}^{n-1} (w_i - 1)^2 [1 + 10 \sin^2(\pi w_i + 1)] + (w_n - 1)^2 [1 + \sin^2(2\pi w_n)]$
F11	Griewank	$f(x) = 1 + \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right)$
F12	Styblinski-Tang function	$f(x) = \frac{1}{2} \sum_{i=1}^n (x_i^4 - 16x_i^2 + 5x_i)$
F13	Noisy sphere	$f(x) = \sum_{i=1}^n (x_i + \text{noise})^2$
F14	Noisy Rastrigin	$f(x) = 10n + \sum_{i=1}^n ((x_i + \text{noise})^2 - 10 \cos(2\pi(x_i + \text{noise})))$
F15	Noisy Ackley	$f(x) = 20e \left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i + \text{noise})^2} \right) - e \left(\frac{1}{n} \sum_{i=1}^n \cos(2\pi(x_i + \text{noise})) \right) + 20 + e$
F16	Noisy Griewank	$f(x) = \frac{1}{4000} \sum_{i=1}^n (x_i + \text{noise})^2 - \prod_{i=1}^n \cos\left(\frac{x_i + \text{noise}}{\sqrt{i}}\right) + 1$
F17	Rosenbrock	$f(x) = \sum_{i=1}^n [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$
F18	Powell's sum function	$f(x) = \sum_{i=1}^n x_i ^{i+1}$

$$MAE = \frac{1}{S} \sum_{i=1}^S |X_{oi} - X_i| \tag{4}$$

S represents the total number of cost samples. X_{oi} corresponds to the benchmark value of the test function, while X_i denotes the computed optimum value.

Table 2 Details of benchmark functions

No	Function	Search range	optimal value	Dimensions
1	Rotated hyper-ellipsoid	[-65.536, 65.536]	0	30
2	Zakharov	[-5, 10]	0	30
3	Dixon-price	[-10, 10]	0	30
4	Sum of different powers	[-1, 1]	0	30
5	Bohachevsky 1	[-100, 100]	0	2
6	Matyas	[-10, 10]	0	2
7	Drop-wave function	[-5.12, 5.12]	-1	2
8	Rastrigin	[-5.12, 5.12]	0	2
9	Ackley	[-32.768, 32.768]	0	100
10	Levy	[-10, 10]	0	100
11	Griewank	[-600, 600]	0	100
12	Styblinski-Tang function	[-5, 5]	-3916.16599	100
13	Noisy Sphere	[-100, 400]	0	30
14	Noisy Rastrigin	[5.12, 5.12]	0	30
15	Noisy Ackley	[-32, 32]	0	30
16	Noisy Griewank	[-600, 600]	0	30
17	Rosenbrock	[-5, 10]	0	Variable
18	Powell's sum function	[-1, 1]	0	Variable

Table 3 PSO parameters

Parameter	Value
Number of particles	150
Inertia weight (w_1)	1
Inertia weight damping ratio (wdamp)	0.99
Personal learning coefficient (c_1)	1.5
Global learning coefficient (c_2)	2
Murmuration coefficient (c_3)	1
Number of clusters (k)	5
Number of runs (cost samples)	50
Maximum iteration	1000

$$SD = \sqrt{\frac{\sum (X_i - \mu)^2}{S}} \quad (5)$$

μ is the mean of the total number of cost samples.

Proposed local best murmuration particle PSO variant

Rationale behind the proposed approach

The inspiration for the proposed variant comes from the natural phenomenon of bird murmuration. Birds move in coordinated patterns, maintaining a balance between individual movement and group dynamics by following a leader within their

vicinity. This behaviour allows the flock of birds to respond rapidly to changes in direction or threats, faster than if each bird moved on its own. Similarly, in the particle swarm optimization algorithm, particles (analogous to birds) can benefit from not only following their personal best (pbest) and the global best (gbest) but also by considering a local best leader within a subgroup or “murmuration cluster.”

Theoretical foundation

The original particle swarm optimization (PSO) algorithm relies on particles updating their velocity based on pbest and gbest. However, this approach can sometimes lead to slower convergence or getting trapped in local optima, especially in complex, high-dimensional search spaces. By introducing a local best murmuration particle (Mbest) within each cluster, the proposed method enhances the exploration–exploitation balance, allowing particles to follow a more dynamic path towards the global optimum. This approach aligns with swarm intelligence principles, where the collective behaviour of decentralized systems leads to improved problem-solving capabilities.

Algorithmic design

The proposed variant modifies the velocity update function to incorporate three components: personal best (pbest), global best (gbest), and local best murmuration (Mbest). The Mbest is determined using the k -means clustering technique, where particles are grouped into clusters, and the best-performing particle in each cluster is identified as the local leader. This clustering is performed at each iteration to adapt to the changing search space, ensuring that the Mbest is dynamically updated.

The modified velocity update function can be expressed as:

$$V_i(t+1) = wV_i(t) + r_1c_1(P_{ibest} - X_i(t)) + r_2c_2(G_{best} - X_i(t)) + r_3c_3(M_{best} - X_i(t)) \quad (6)$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (7)$$

where

- $V_i(t)$ is the velocity of particle i at time t .
- w is the inertia weight.
- c_1 , c_2 and c_3 are acceleration coefficients.
- r_1 , r_2 and r_3 are random numbers uniformly distributed between 0 and 1.
- $X_i(t)$ is the position of particle i at time t .
- P_{ibest} , G_{best} and M_{best} are the personal best, global best, and local best positions, respectively.

In Eq. (6), the updating velocity function of each particle is proposed to move according to a new velocity update function. In this equation, the personal best, global best, and local murmuration best are determined to update the velocity of each particle before their positions are updated using Eq. (7). The local best murmuration particle is found using the k -means clustering technique to classify each particle into a cluster known in this work as the murmuration cluster and the best particle in

the cluster will be chosen as the local best murmuration particle. Thus, each particle in each murmuration cluster will move towards the global best using their personal best, their cluster local murmuration best particle, and the global best for the entire swarm. This modification is based on the idea that birds always move in murmuration by looking at a leader in its vicinity. As a result, the 'ripple' through a flock is three times faster than could be explained if the birds were just moving individually. Scientists have shown several reasons why birds would always not fly alone but look at a cheerleader to move in murmurations. Birds murmur together for safety, and warmth among others. According to this contribution, a particle will not only look at its personal best position and the global best to move towards the global best position of the swarm but also consider a local murmuration's best position in order to murmur together towards the global best position. The introduction of the Mbest component is to achieve faster convergence and avoid premature convergence by enhancing the swarm's ability to explore the search space more effectively. The dynamic clustering ensures that particles adapt to local changes, reducing the likelihood of being trapped in suboptimal regions. Additionally, this approach maintains diversity within the swarm, as each cluster's particles follow a different Mbest, contributing to a more robust search process. Compared to the original PSO, the proposed local best murmuration PSO variant is expected to perform better in complex, multimodal optimization problems. The clustering-based approach helps in maintaining a good balance between exploration and exploitation, which is often a challenge in conventional PSO algorithms. The flowchart of the proposed model is shown in figure below (Fig. 1).

1. *Start*: Start of the algorithm.
2. *Enter the Objective Function and set PSO parameters*: The objective function $f(x)$ represents the problem to be optimized. This step defines the function and its parameters, which will be used to evaluate the fitness of each particle. Parameters for the particle swarm optimization (PSO) algorithm is also defined in this step. This includes defining the number of particles, the maximum number of iterations, inertia weight, cognitive and social coefficients, and any constraints on the decision vector.
3. *Initialize particles, evaluate solutions and determine the gbest*:
 - Each particle's position is initialized randomly within the feasible solution space.
 - Each particle's velocity is initialized randomly. This velocity dictates how a particle moves in the search space.
 - Each particle's initial position is evaluated using the objective function, determining their fitness. The particle with the best fitness is set as the initial global best (gbest).
4. *Set Iteration Counter*: Initialize the iteration counter to 1. This counter controls the number of times the main loop will run.
5. *While iter < = maxIter*: This loop iterates as long as the iteration counter is less than or equal to the maximum number of iterations. Within this loop:
 - *Cluster particles and find best in each group*: Use k-means clustering to group particles into 'n' clusters. This step helps identify diverse solutions within different

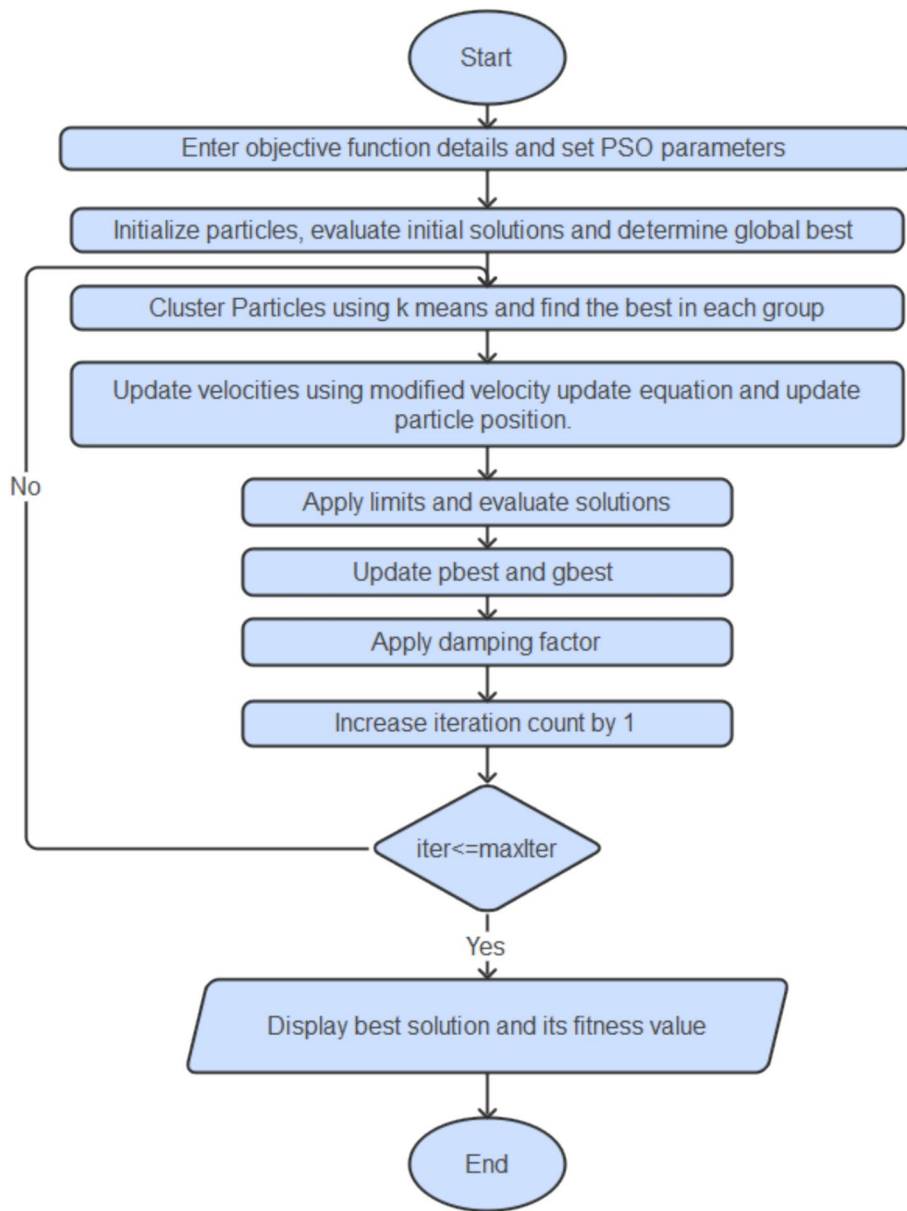


Fig. 1 Flowchart of proposed model

regions of the search space. For each cluster, identify the particle with the best fitness value. These particles represent the best solutions found within each cluster.

- *Update Velocities using modified velocity equation and update positions:* Update each particle’s velocity using a modified PSO velocity update equation. This equation considers each particle’s personal best position, the best position in its cluster, and the global best position. Update each particle’s position based on its new velocity.
- *Apply Limits and evaluate solutions:* Enforce position and velocity limits to be within the defined search space. Re-evaluate the fitness of each particle at its new position.

- *Update Pbest and Gbest*: Update each particle's personal best if its new position is better. Also, update the global best if the best particle in any cluster has a better fitness than the current global best.
 - *Apply Weight Damping*: Gradually decrease the inertia weight by multiplying damping factor with the current inertia weight.
 - *Increment Iteration Counter*: Increase iter by 1 to continue to the next iteration.
6. *End While Loop*: While loop ends
 7. *Display Results*: After completing the iterations, display the best solution found and its corresponding fitness value.

Results and discussion

Unimodal functions (F1 to F6)

Unimodal functions test the algorithm's ability to exploit the search space effectively. PSOMbest achieved the optimal solution in 3 out of the 6 unimodal functions (*F4*, *F5*, and *F6*), outperforming PSO, which achieved optimal values in 2 functions, and PSO-AWDV, PSOEA, and GA, which each achieved optimal solutions in 1, 2, and 1 function, respectively (see Table 4). PSOMbest also had the best optimal solution in all six functions. This superior performance in unimodal functions indicates that the proposed variant is highly effective at local search and convergence, thus addressing to some degree the issue of local entrapment.

Furthermore, in terms of MAE, PSOMbest recorded the lowest values in 3 of the 6 functions, demonstrating its consistency and precision in finding the optimal solution. Additionally, PSOMbest exhibited better standard deviation (SD) values in 3 out of the 6 unimodal functions, indicating less variability in the solutions across multiple runs (Table 5). This consistency further highlights the algorithm's reliability in simpler landscapes where exploitation is crucial.

Multimodal functions (F7 to F12)

Multimodal functions present a greater challenge due to the presence of multiple local optima. From Table 6, PSOMbest achieved the best optimum values in all 6 multimodal functions. In comparison, PSO achieved the best results in 2 functions, PSO-AWDV in

Table 4 Optimum values (Unimodal functions)

No	Function	Benchmark value	PSO	PSO-AWDV	PSOEA	GA	PSOM _{best}
1	Rotated hyper-ellipsoid	0	4.0218E−98	6.06252	5.1859E−100	0.00180	6.603E−236
2	Zakharov	0	7.50504E−10	1.98923	1.90473E−09	1.28608	4.11752E−27
3	Dixon-price	0	0.66667	1.63586	0.66667	0.66671	0.18241
4	Sum of different powers	0	1.9856E−171	2.6034E−18	3.1923E−171	1.63753E−17	0
5	Bohachevsky 1	0	0	0	0	0	0
6	Matyas	0	0	4.75199E−68	0	3.2249E−239	0

Bold values in each table is significant since it shows which algorithm is the best in each function

Table 5 MAE and SD (unimodal functions)

No	Function	Algorithm	MAE	SD
1	Rotated hyper-ellipsoid	PSO	4.296E-92	2.45705E-91
		PSO-AWDV	76.98422	85.49406
		PSOEA	3.89785E-92	1.88719E-91
		GA	13.91806	30.9640
		PSOMbest	2.93273E-08	2.07375E-07
2	Zakharov	PSO	6.66621E-08	1.15117E-07
		PSO-AWDV	3.75342	1.14183
		PSOEA	6.79923E-08	1.43314E-07
		GA	3.07840	1.45745
		PSOMbest	2.31505E-06	1.18901E-05
3	Dixon-price	PSO	0.66667	2.73792E-16
		PSO-AWDV	7.93845	5.73753
		PSOEA	0.66667	2.48253E-16
		GA	0.85748	0.70548
		PSOMbest	0.71922	0.90817
4	Sum of different powers	PSO	5.1162E-145	3.6008E-144
		PSO-AWDV	5.49909E-12	2.18341E-11
		PSOEA	3.9635E-150	2.5084E-149
		GA	3.50266E-11	1.34234E-10
		PSOMbest	0	0
5	Bohachevsky 1	PSO	0	0
		PSO-AWDV	0	0
		PSOEA	0	0
		GA	0	0
		PSOMbest	0	0
6	Matyas	PSO	0	0
		PSO-AWDV	6.56244E-44	4.6403E-43
		PSOEA	0	0
		GA	2.03137E-18	1.06667E-17
		PSOMbest	0	0

Bold values in each table is significant since it shows which algorithm is the best in each function

Table 6 Optimum values (Multimodal functions)

No	Function	Benchmark Value	PSO	PSO-AWDV	PSOEA	GA	PSOMbest
7	Drop-Wave Function	-1	-1	-0.99999	-1	-1	-1
8	Rastrigin	0	0	0	0	0	0
9	Ackley	0	1.15463E-14	0.00712	7.99361E-15	0.00059	4.44089E-15
10	Levy	0	2.175409076	8.27561	0.54385	2.86289	1.49976E-32
11	Griewank	0	3.96683E-13	10.34546028	3.6926E-13	26.97701	0
12	Styblinski-Tang function	-3916.166	-3421.831	-3001.808	-3407.695	-3455.488	-3916.166

Bold values in each table is significant since it shows which algorithm is the best in each function

Table 7 MAE and SD (Multimodal functions)

No	Function	Algorithm	MAE	SD
7	Drop-wave function	PSO	0.02	0
		PSO-AWDV	0.02014	0.00012
		PSOEA	0.02	0
		GA	0.02893	0.02474
		PSOMbest	0.02255	0.01262
8	Rastrigin	PSO	0	0
		PSO-AWDV	0	0
		PSOEA	0	0
		GA	0	0
		PSOMbest	0	0
9	Ackley	PSO	2.08544E-14	6.80507E-15
		PSO-AWDV	0.02165	0.01172
		PSOEA	2.00018E-14	4.91493E-15
		GA	0.00087	0.00012
		PSOMbest	0.00727	0.05135
10	Levy	PSO	10.71004	3.37357
		PSO-AWDV	18.72230	5.40355
		PSOEA	11.07623	4.11081
		GA	3.83042	0.44650
		PSOMbest	3.70392E-09	1.79588E-08
11	Griewank	PSO	0.12136	0.18319
		PSO-AWDV	19.94082	4.63812
		PSOEA	0.09148	0.11219
		GA	36.62218	5.94389
		PSOMbest	5.29789E-11	2.15226E-10
12	Styblinski-Tang function	PSO	660.20023	53.81486
		PSO-AWDV	1114.70001	74.99359
		PSOEA	642.65524	55.46988
		GA	697.45748	80.36825
		PSOMbest	10.24644	46.62574

Bold values in each table is significant since it shows which algorithm is the best in each function

Table 8 Optimum values (Noisy functions)

No	Function	Benchmark Value	PSO	PSO-AWDV	PSOEA	GA	PSOMbest
13	Noisy Sphere	0	0.0012633	1.54593591	0.00157898	0.01615	0.000593
14	Noisy Rastrigin	0	21.289979	30.1634412	20.44469693	2.559060	0.112073
15	Noisy Ackley	0	0.0308622	1.50390775	0.035410354	2.665462	0.020417
16	Noisy Griewank	0	7.55207E-05	0.55838062	0.00012165	1.72972	2.70531E-05

Bold values in each table is significant since it shows which algorithm is the best in each function

1, PSOEA in 2, and GA in 2 functions. This gives an indication that PSOMbest's local best murmuration mechanism enhances its ability to escape local optima, leading to superior exploration capabilities.

Table 9 MAE and SD (Noisy functions)

No	Function	Algorithm	MAE	SD
13	Noisy sphere	PSO	0.00121559	0.001280966
		PSO-AWDV	6.130958766	13.12464091
		PSOEA	0.001373176	0.002050239
		GA	3.336978416	10.93430651
		PSOMbest	0.003408802	0.023208555
14	Noisy Rastrigin	PSO	19.0844968	19.0844968
		PSO-AWDV	19.10981999	19.10981999
		PSOEA	17.26557704	17.26557704
		GA	3.084261342	3.084261342
		PSOMbest	0.252170126	0.252170126
15	Noisy Ackley	PSO	0.428796032	1.462144248
		PSO-AWDV	1.009873388	0.654539306
		PSOEA	0.038415831	0.198708018
		GA	1.553110144	0.718214117
		PSOMbest	0.019118491	0.011711702
16	Noisy Griewank	PSO	0.005105199	0.375443752
		PSO-AWDV	0.417872321	0.167354619
		PSOEA	0.005398832	0.012615807
		GA	1.380169938	0.887211018
		PSOMbest	0.07289616	0.013269013

Bold values in each table is significant since it shows which algorithm is the best in each function

Table 10 Optimum values (Variable dimension functions)

No	Function	Benchmark Value	PSO	PSO-AWDV	PSOEA	GA	PSOMbest
17	Rosenbrock (2)	0	0	6.22777E-24	0	8.56965E-19	0
	Rosenbrock (30)		0.593594803	32.90125955	0.11054394	1.45339704	0
	Rosenbrock (100)		126.9879929	3393.706633	71.6360264	604.1686149	0
18	Powell's Sum Function (2)	0	0	5.19977E-18	0	9.88369E-11	0
	Powell's Sum Function (30)		18.96820774	30.37178684	8.15080702	26.5396838	0
	Powell's Sum Function (100)		81.50860794	908.6016601	91.2217011	95.98033087	0

Bold values in each table is significant since it shows which algorithm is the best in each function

Moreover, in terms of MAE, PSOMbest outperformed the other algorithms in 5 of the 6 multimodal functions, which further emphasizes its effectiveness in navigating complex landscapes. PSOMbest also demonstrated better SD values in 5 out of the 6 multimodal functions (F8, F9, F10, F11, and F12) (see Table 7), which indicates a higher degree of consistency in finding the global optimum across multiple runs, reducing the risk of suboptimal convergence.

Noisy functions (F13 to F16)

Noisy functions, characterized by Gaussian distribution noise, evaluate the robustness and adaptiveness of algorithms. PSOMbest outperformed the competing algorithms by achieving the best results in all 4 noisy functions (see Table 8).

The MAE values further confirm PSOMbest’s superiority, as it achieved the lowest MAE in 3 of the 4 noisy functions, demonstrating its ability to effectively handle noisy environments. This attests to the robustness of the proposed variant. Additionally, PSOMbest exhibited the lowest SD in 3 noisy functions (F14, F15, F16), further indicating its consistent performance even when faced with noise, as well as its ability to maintain a stable search process across different runs (see Table 9).

Table 11 MAE and SD (Variable dimension functions)

No	Function	Algorithm	MAE	SD
17	Rosenbrock(2 dimensions)	PSO	0	0
		PSO-AWDV	8.59134E−11	9.519E−10
		PSOEA	0	0
		GA	0.000108437	0.000836554
		PSOMbest	0	0
17	Rosenbrock (30 dimensions)	PSO	13.04416821	24.57905489
		PSO-AWDV	47.88386943	62.50468094
		PSOEA	12.8030527	23.70867439
		GA	0.970510359	0.520100848
		PSOMbest	5.8795E−06	6.57347E−05
17	Rosenbrock (100 dimensions)	PSO	63.53987079	
		PSO-AWDV	2551.497442	2983.610568
		PSOEA	65.67604382	60.31844995
		GA	366.677693	296.9643339
		PSOMbest	32.30026951	49.41550314
18	Powell’s sum function (2 dimensions)	PSO	0	0
		PSO-AWDV	2.77658E−12	2.63792E−11
		PSOEA	0	0
		GA	0.000847392	0.004507696
		PSOMbest	0	0
18	Powell’s sum function (30 dimensions)	PSO	11.40941039	20.23738586
		PSO-AWDV	25.36109884	30.91413765
		PSOEA	7.876864475	3.324209896
		GA	10.92232852	0.696991482
		PSOMbest	0.489048421	2.986028171
18	Powell’s sum function (100 dimensions)	PSO	42.37149273	29.5871517
		PSO-AWDV	453.8217618	214.3659764
		PSOEA	44.09560408	23.84112572
		GA	542.0518502	5472.462358
		PSOMbest	39.11061438	0.745742797

Bold values in each table is significant since it shows which algorithm is the best in each function

Table 12 Worst values

No	Function	Algorithm	Worst values
1	Rotated hyper-ellipsoid	PSO	1.73539E-90
		PSO-AWDV	378.5818023
		PSOEA	1.33379E-90
		GA	151.7558672
		PSOMbest	1.46636E-06
2	Zakharov	PSO	5.57142E-07
		PSO-AWDV	6.108257572
		PSOEA	9.66256E-07
		GA	6.788905131
		PSOMbest	8.09531E-05
3	Dixon-price	PSO	0.666666667
		PSO-AWDV	24.76677769
		PSOEA	0.666666667
		GA	4.413501022
		PSOMbest	6.68136002
4	Sum of different powers	PSO	2.5463E-143
		PSO-AWDV	1.12244E-10
		PSOEA	1.7744E-148
		GA	8.94607E-10
		PSOMbest	0
5	Bohachevsky 1	PSO	0
		PSO-AWDV	0
		PSOEA	0
		GA	0
		PSOMbest	0
6	Matyas	PSO	0
		PSO-AWDV	3.28119E-42
		PSOEA	0
		GA	7.05859E-17
		PSOMbest	0
7	Drop-wave function	PSO	-1
		PSO-AWDV	-0.9993919
		PSOEA	-1
		GA	-0.9362453
		PSOMbest	-0.9362453
8	Rastrigin	PSO	0
		PSO-AWDV	0
		PSOEA	0
		GA	0
		PSOMbest	0
9	Ackley	PSO	3.9968E-14
		PSO-AWDV	0.058225798
		PSOEA	3.9968E-14
		GA	0.001114711
		PSOMbest	0.363106893

Table 12 (continued)

No	Function	Algorithm	Worst values
10	Levy	PSO	18.63381944
		PSO-AWDV	38.17312859
		PSOEA	19.54244632
		GA	5.111238465
11	Griewank	PSOMbest	9.07663E-08
		PSO	0.793765033
		PSO-AWDV	30.48733926
		PSOEA	0.468569005
		GA	50.76553979
12	Styblinski-Tang function	PSOMbest	1.41017E-09
		PSO	-3195.64389
		PSO-AWDV	-2624.39726
		PSOEA	-3153.23374
		GA	-3067.8354
13	Noisy sphere	PSOMbest	-3680.43650
		PSO	0.099155026
		PSO-AWDV	49.17726443
		PSOEA	0.010186423
		GA	44.63780751
14	Noisy Rastrigin	PSOMbest	0.005507762
		PSO	85.92131535
		PSO-AWDV	69.87803908
		PSOEA	69.66203281
		GA	14.63966182
15	Noisy Ackley	PSOMbest	2.299103273
		PSO	3.724887065
		PSO-AWDV	4.039111779
		PSOEA	0.937067222
		GA	5.063922483
16	Noisy Griewank	PSOMbest	0.076904616
		PSO	0.07289616
		PSO-AWDV	1.290203231
		PSOEA	0.051755699
		GA	5.394507826
17	Rosenbrock (2 dimensions)	PSOMbest	0.046925553
		PSO	0
		PSO-AWDV	4.25887E-09
		PSOEA	0
		GA	0.003385736
17	Rosenbrock (30 dimensions)	PSOMbest	0
		PSO	74.82296694
		PSO-AWDV	314.860825
		PSOEA	76.90164891
		GA	3.637892802
		PSOMbest	0.000293975

Table 12 (continued)

No	Function	Algorithm	Worst values
17	Rosenbrock (100 dimensions)	PSO	229.168419
		PSO-AWDV	13,570.16771
		PSOEA	313.5026471
		GA	1685.645411
		PSOMbest	137.0706187
18	Powell's sum function (2 dimensions)	PSO	0
		PSO-AWDV	1.18448E-10
		PSOEA	0
		GA	0.017839829
		PSOMbest	0
18	Powell's sum function (30 dimensions)	PSO	76.71101223
		PSO-AWDV	146.907062
		PSOEA	25.53414442
		GA	29.35884125
		PSOMbest	8.150807023
18	Powell's sum function (100 dimensions)	PSO	195.5733344
		PSO-AWDV	1637.580727
		PSOEA	148.9283369
		GA	24,586.70819
		PSOMbest	98.98273584

Bold values in each table is significant since it shows which algorithm is the best in each function

Variable dimension functions (F17 & F18)

Variable dimension functions assess an algorithm's scalability by testing it across different dimensional spaces. From Table 10, it can be seen that PSOMbest outperformed the other algorithms in all selected dimensions (2, 30, and 100) for both functions *F17* and *F18*. The ability of PSOMbest to scale effectively is evident in its superior performance in both low (2D) and high-dimensional (100D) settings. Specifically, PSOMbest achieved the best optimum values in both functions at all dimensional levels, surpassing PSO, PSO-AWDV, PSOEA, and GA, which exhibited diminished performance as dimensionality increased.

The MAE values further validate this (see Table 11), with PSOMbest achieving the lowest MAE in all dimensions, particularly in the high-dimensional cases where other algorithms struggled significantly. Moreover, PSOMbest maintained the lowest SD in both functions across all dimensions except *F18* (30D), highlighting its consistency and adaptability, especially in high-dimensional spaces where the search process becomes more complex.

Analysis of worst-case performances

The worst values in Table 12 provide insight into the reliability of an algorithm, particularly in scenarios where it may fail to find optimal or near optimal solutions.

Analysing the table, PSOMbest consistently demonstrated lower worst-case values across most benchmark functions, indicating that even in the least favourable runs, it managed to avoid significantly poor solutions. The breakdown is as follows:

- PSOMbest achieved the lowest worst values in 3 out of 6 unimodal functions (*F4*, *F5*, and *F6*). This indicates the competitive consistent performance of PSOMbest.
- PSOMbest achieved the lowest worst values compared to the other algorithms in 4 out of 6 multimodal functions (*F8*, *F10*, *F11*, and *F12*). This confirms PSOMbest's ability to maintain exploration without falling into deep local minima.
- PSOMbest achieved the lowest worst values in 3 out of 4 noisy functions (*F13*, *F14*, and *F15*), demonstrating its robustness in noisy environments where traditional PSO and GA struggled, leading to higher worst-case errors.
- PSOMbest achieved the lowest worst values in both variable dimension functions (*F17* and *F18*) across all dimensions, especially in high-dimensional spaces (100D), where the worst-case values of other algorithms escalated significantly, indicating their difficulties in maintaining consistent performance as the problem complexity increased.

The ability of PSOMbest to maintain lower worst-case values across these diverse functions highlights its reliability and robustness.

Computational time

Table 13 shows the average computational time the various algorithms used in evaluating all 18 functions per run.

Table 13 reveals that PSOMbest has a longer average computational time per run (150.29 s) compared to other algorithms like GA (67.74 s), PSO (84.52 s), PSO-AWDV (91.86 s), and PSOEA (93.32 s). While this might seem like a disadvantage at first glance, the increased time can be viewed as a necessary trade-off for the superior accuracy, robustness, and scalability that PSOMbest offers.

In practical scenarios, especially in critical applications where the quality of the solution is paramount, the slightly longer computational time is often justified. For example, in engineering design optimization or complex system simulations, achieving the best possible solution can significantly outweigh the cost of additional computation time. Moreover, with advancements in computational power and parallel processing, the difference in time becomes less significant.

Table 13 Computational time

Algorithm	Average computational time per run
PSO	84.52 s
PSO-AWDV	91.86 s
PSOEA	93.32 s
GA	67.74 s
PSOMbest	150.29 s

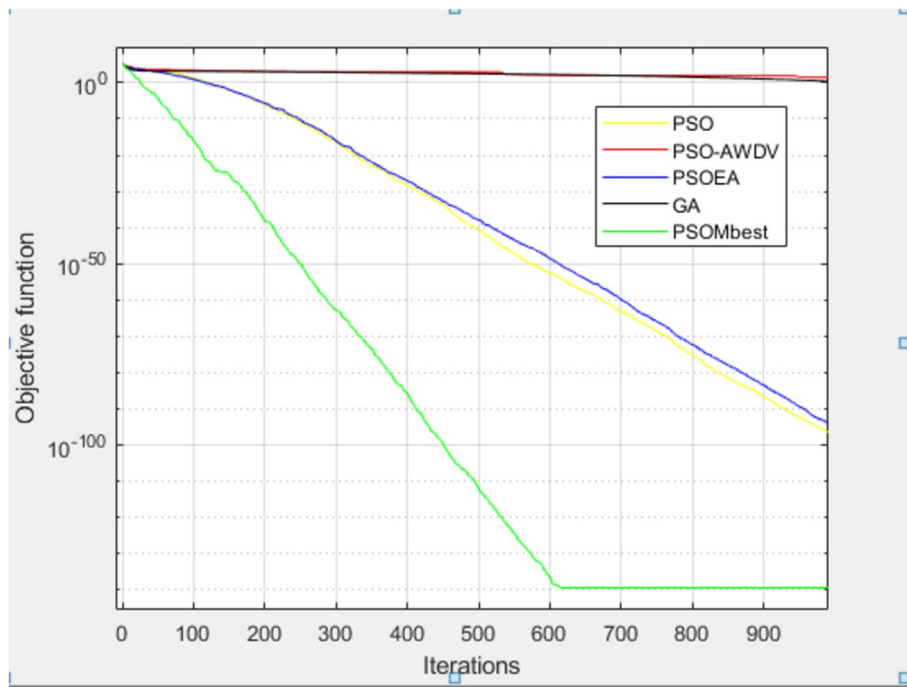


Fig. 2 Convergence curves for rotated hyper-ellipsoid (F1)

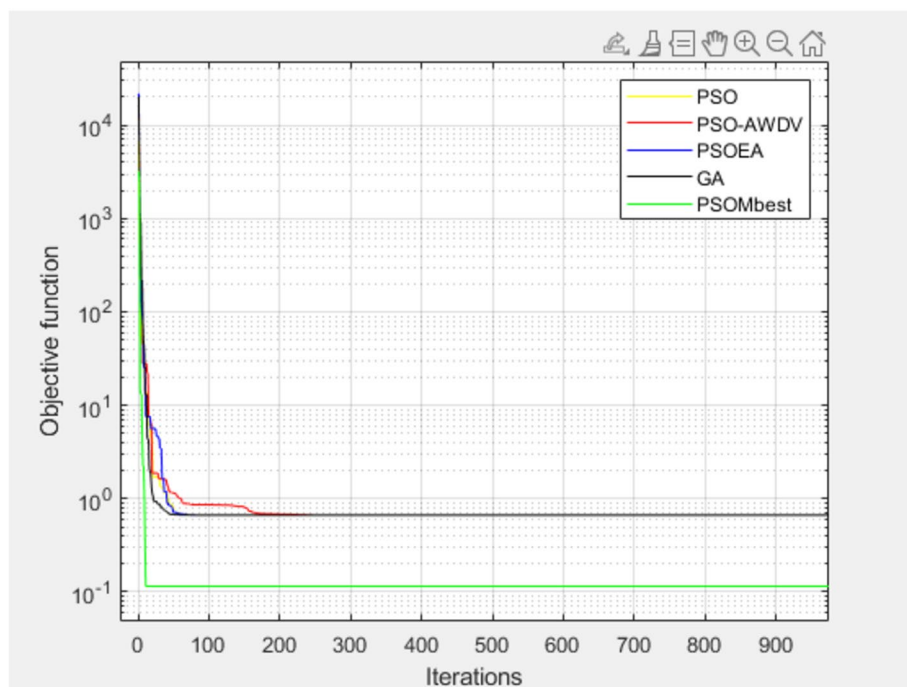


Fig. 3 Convergence curves for Dixon-price (F2)

Convergence rate

Figures 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23 show the convergence curves of the algorithms on each function. It can be observed that

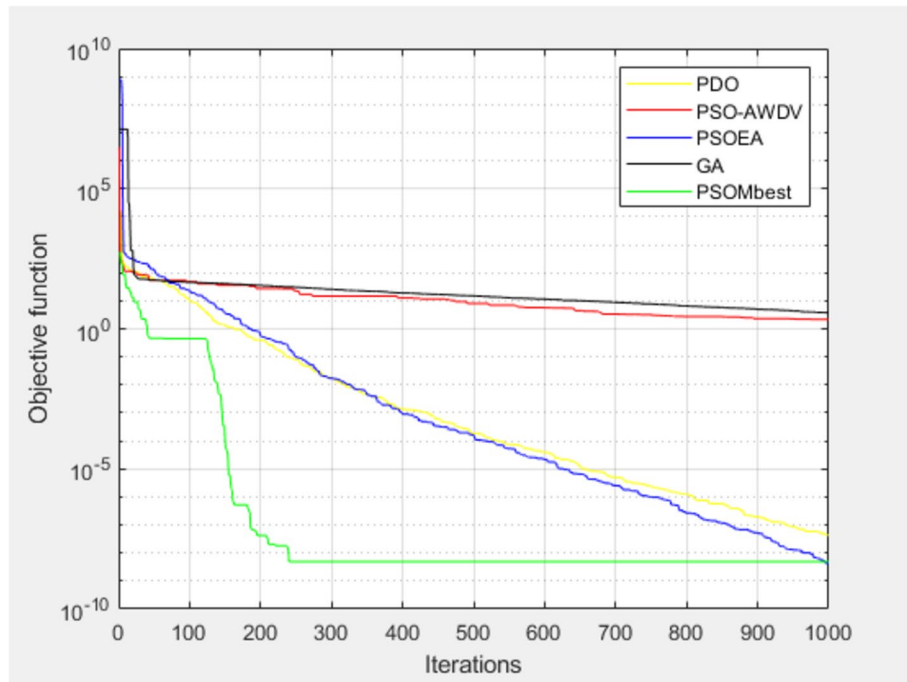


Fig. 4 Convergence curves for Zakharov (F3)

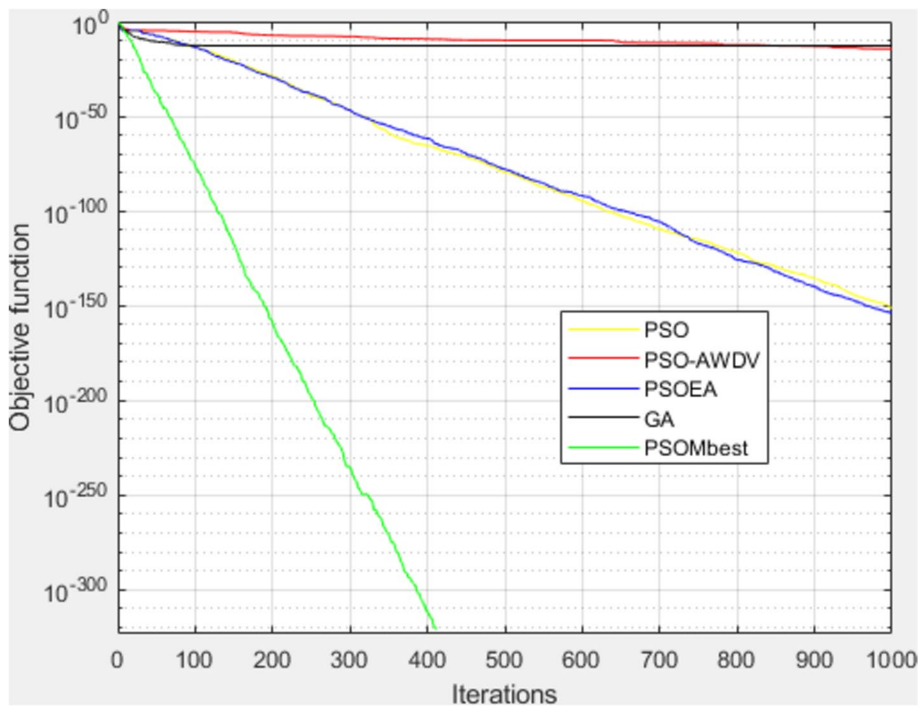


Fig. 5 Convergence curves for sum of different powers (F4)

PSOMbest consistently achieved superior convergence rates across all 18 functions, even in cases where all algorithms converged to optimum values (as shown in Figs. 6 and 9). PSOMbest exhibited rapid convergence before the 50th iteration in functions where

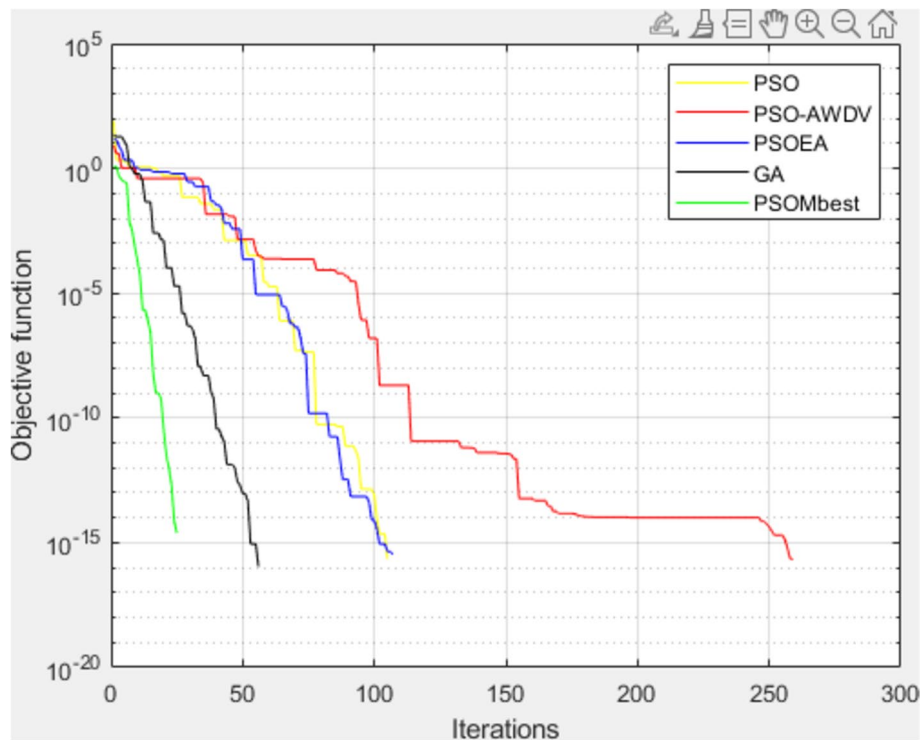


Fig. 6 Convergence curves for Bohachevsky 1 (F5)

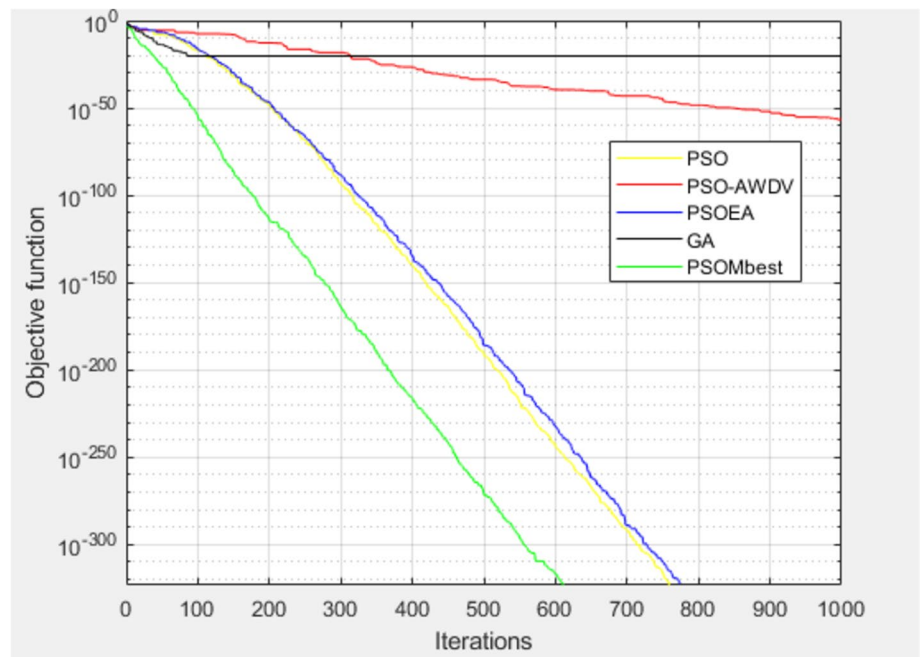


Fig. 7 Convergence curves for Matyas (F6)

it attained optimum cost (functions $F4$, $F5$, $F6$, $F7$, $F8$, $F11$, $F12$, $F17$ & $F18$), except for $F4$, where convergence was achieved just after the 400th iteration. Functions $F2$, $F9$, and $F10$, PSOMbest also demonstrated convergence within the first 50 iterations. These

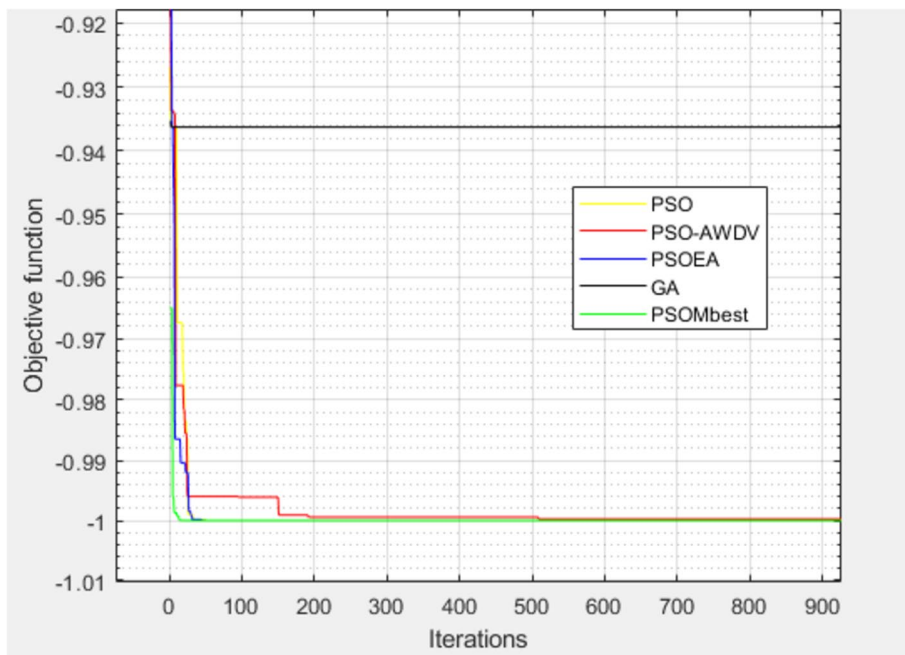


Fig. 8 Convergence curves for dropwave (F7)

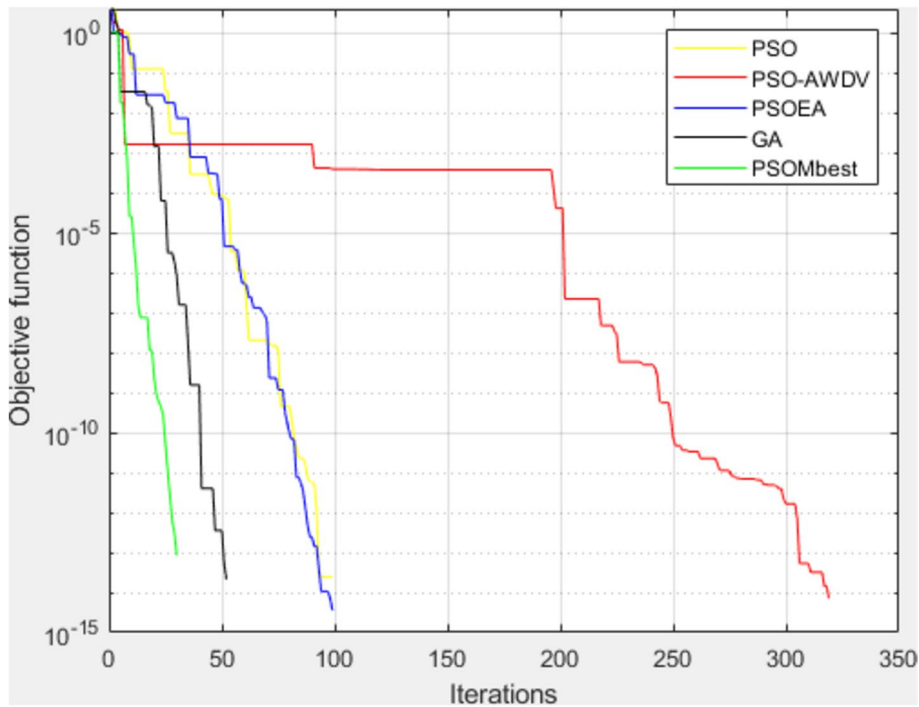


Fig. 9 Convergence curves for Rastrigin (F8)

findings highlight the unique ability of PSOMbest to converge swiftly without compromising accuracy.

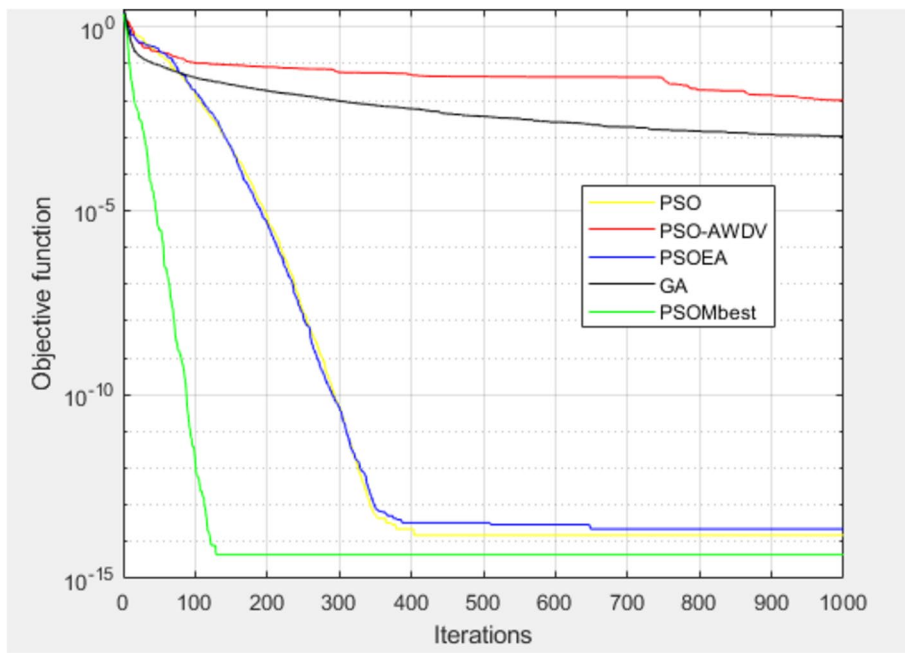


Fig. 10 Convergence curves for Ackley (F9)

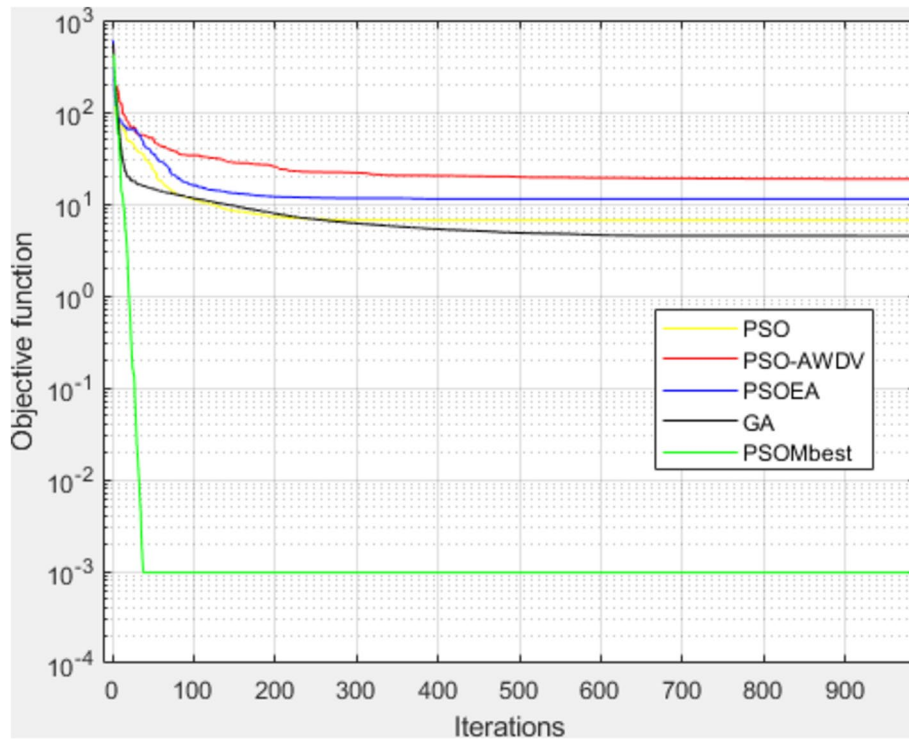


Fig. 11 Convergence curves for Levy (F10)

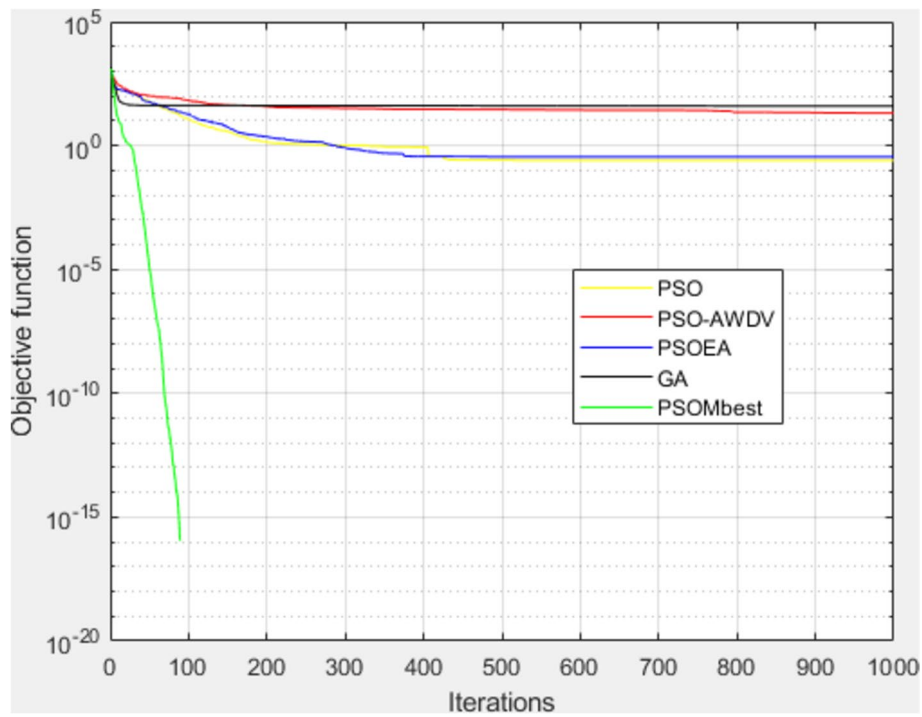


Fig. 12 Convergence curves for Griewank (F11)

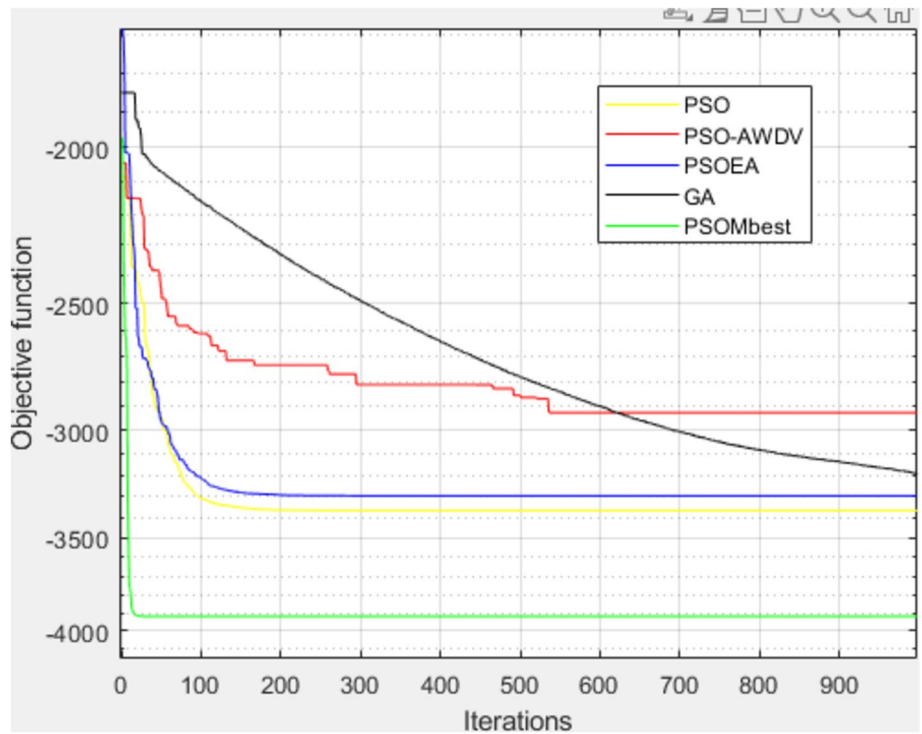


Fig. 13 Convergence curves for Styblinski-Tang (F12)

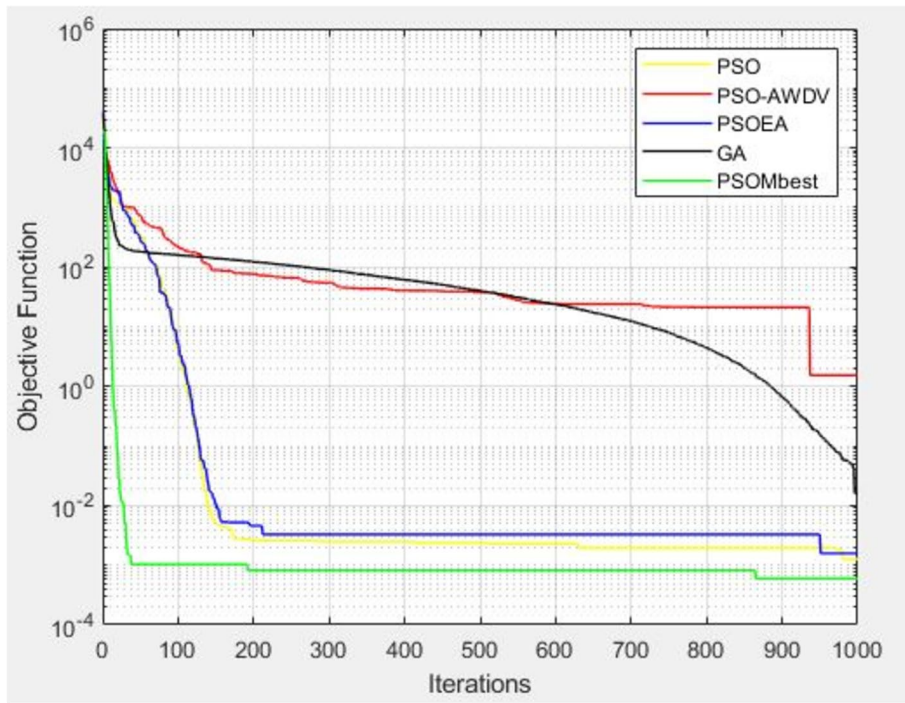


Fig. 14 Convergence curves for noisy sphere (F13)

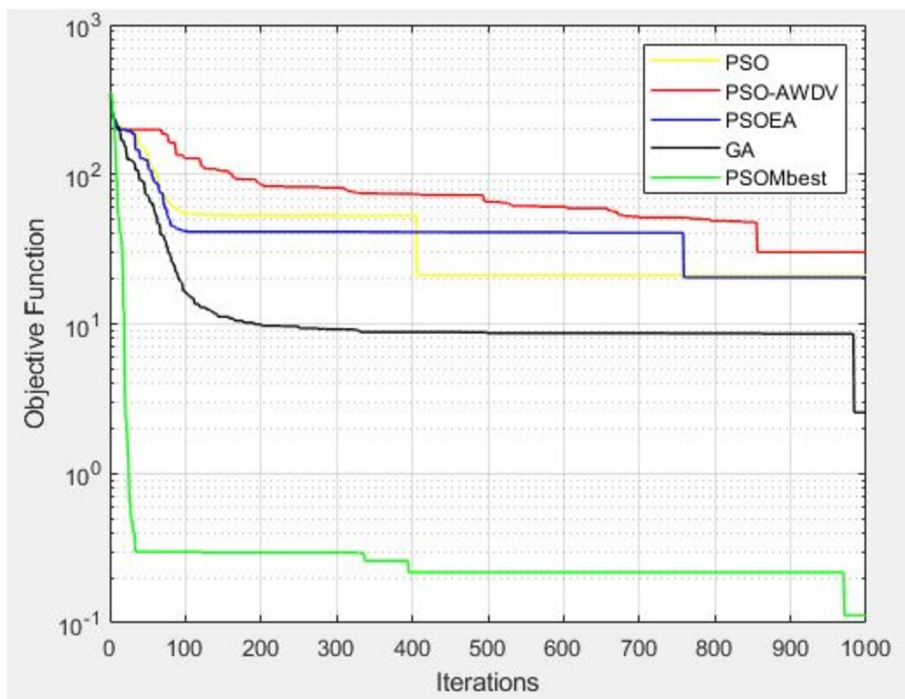


Fig. 15 Convergence curves for noisy Rastrigin (F14)

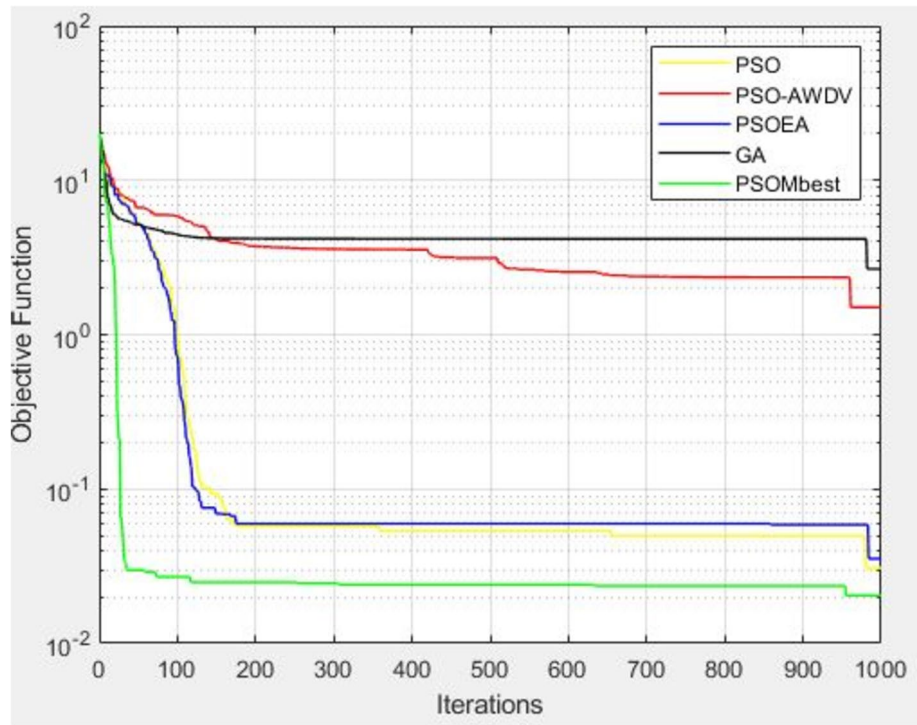


Fig. 16 Convergence curves for noisy Ackley (F15)

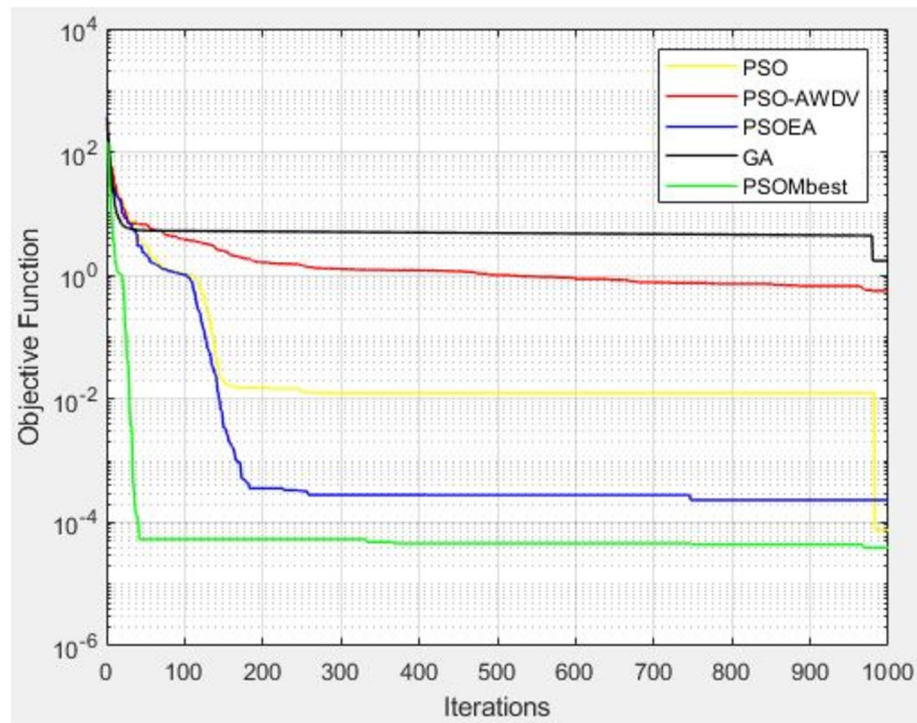


Fig. 17 Convergence curves for noisy Griewank (F16)

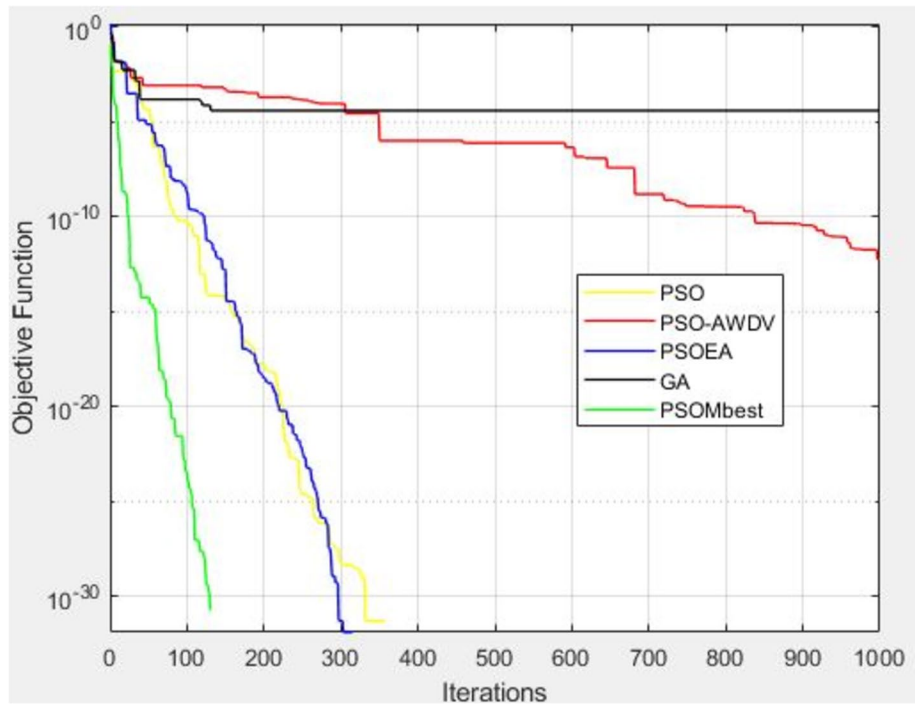


Fig. 18 Convergence curves for Rosenbrock (F17, 2D)

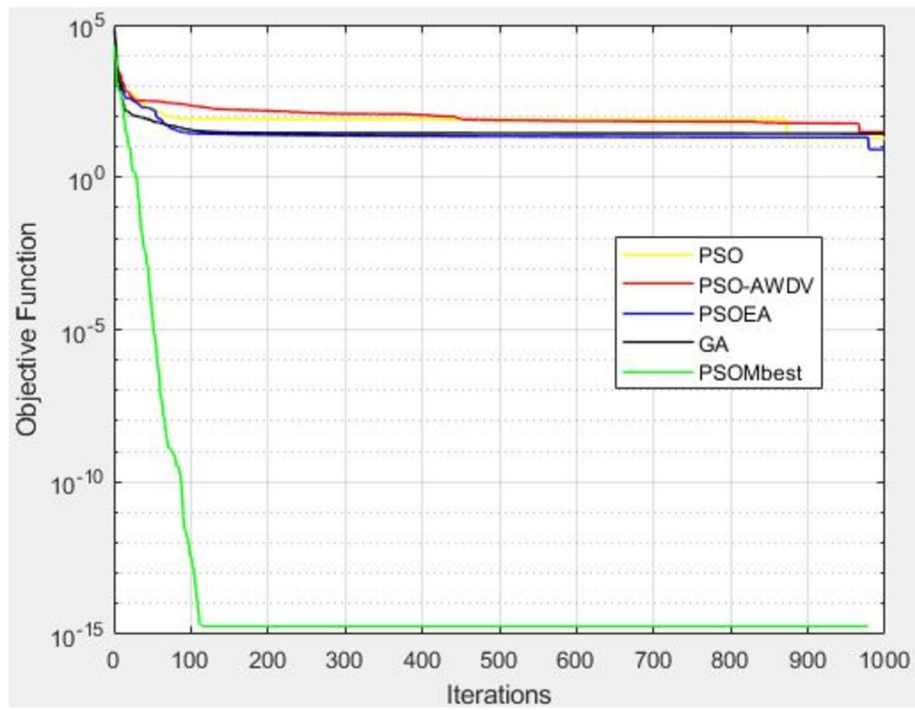


Fig. 19 Convergence curves for Rosenbrock (F17, 30D)

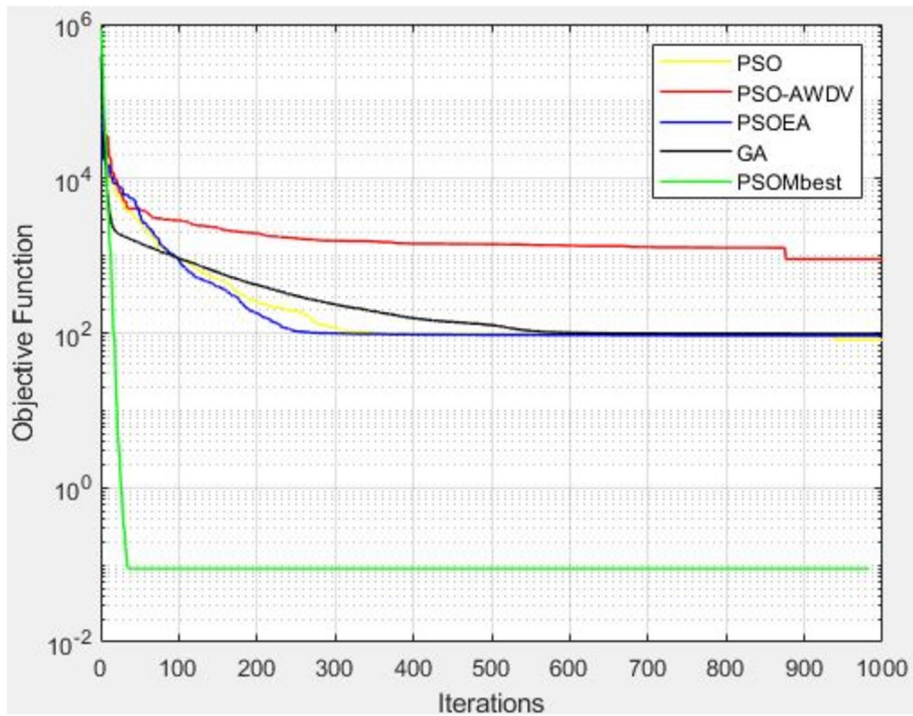


Fig. 20 Convergence curves for Rosenbrock (F17, 100D)

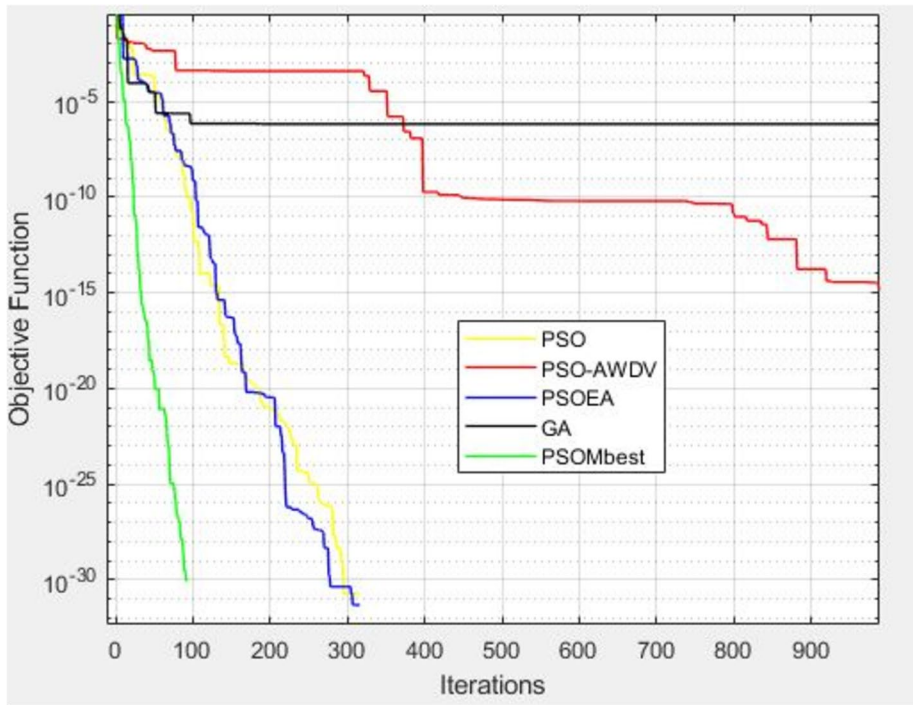


Fig. 21 Convergence curves for Powell's sum function (F18, 2D)

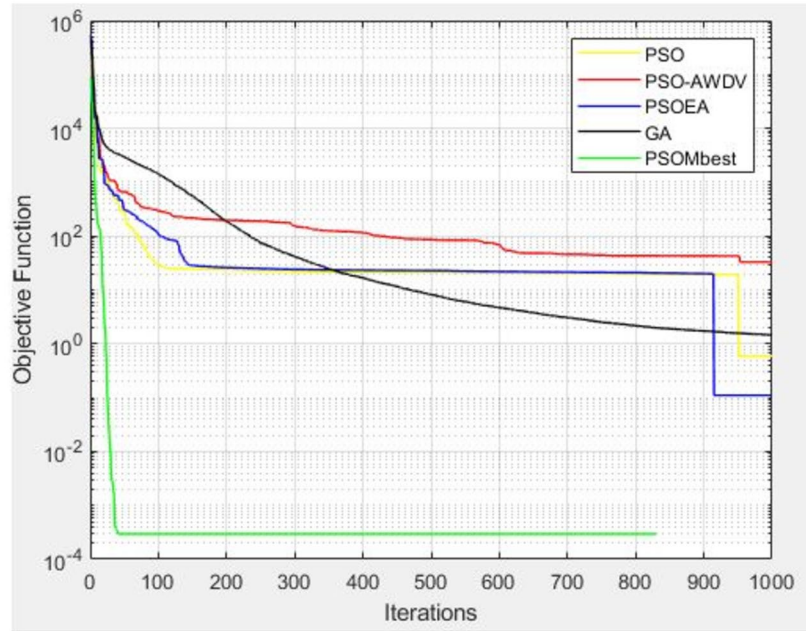


Fig. 22 Convergence curves for Powell's sum function (F18, 30D)

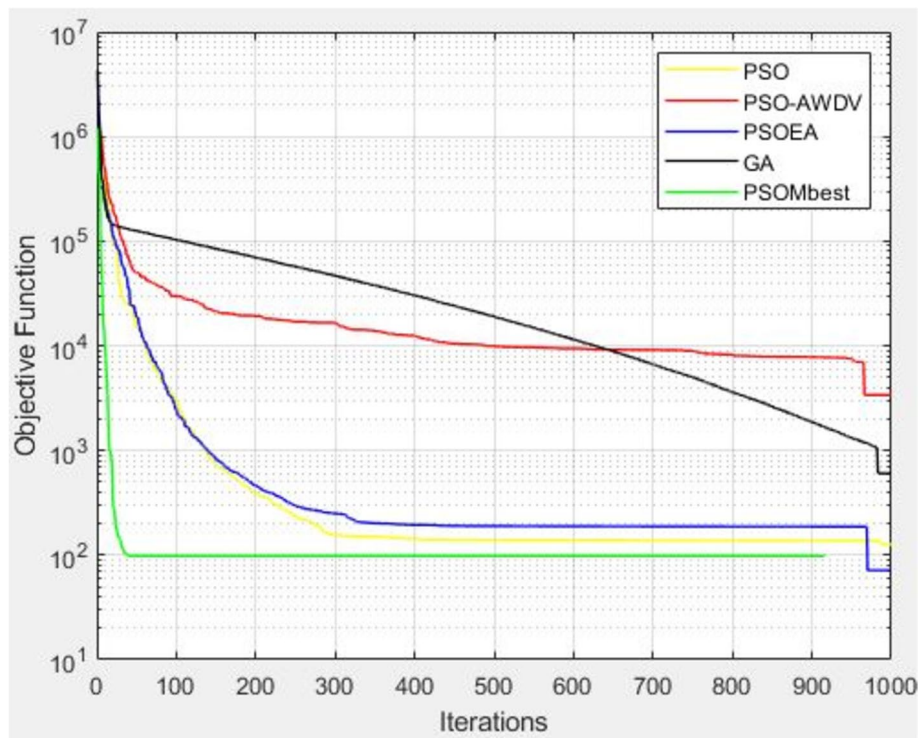


Fig. 23 Convergence curves for Rosenbrock (F18, 100D)

Conclusion

An improvement to the traditional PSO called PSOMbest has been made, the proposed improvement modifies the updating velocity function of the PSO, and it uses a local best murmuration particle which is found using the k-means clustering technique. A comprehensive analysis of the proposed PSOMbest variant across 18 benchmark functions reveals its superiority in terms of both optimal performance and reliability compared to traditional PSO, PSO-AWDV, PSOEA, and GA. The proposed improvement demonstrated superior exploration abilities by achieving the best optimum values in 15 out of 18 functions, particularly in the multimodal functions, where it achieved the best optimum value in all 6 cases. It also achieved the best worst-case values in 12 out of 18 functions, especially in the variable-dimension functions, where other algorithms showed significant escalation, indicating the proposed improvement's reliability and robustness. In terms of convergence, the proposed improvement exhibited the best convergence rate in all 18 functions. These findings highlight the impressive ability of the proposed improvement to converge swiftly without compromising accuracy. PSOMbest consistently achieved the best or near-best optimum values, demonstrating superior exploitation capabilities in unimodal functions, robust exploration in multimodal landscapes, adaptability in noisy environments, and scalability across varying dimensionalities. The statistical analysis, including MAE, SD, and worst-case values, further reinforces PSOMbest's robustness and consistency. It exhibited the lowest MAE and SD in a majority of the functions, alongside lower worst-case values, indicating its ability to deliver reliable performance across different optimization scenarios.

Acknowledgements

Not applicable.

Author contributions

Elvis Twumasi conceptualized the research idea, wrote the paper, and was involved in the simulation and analysis of the results. Emmanuel Asuming Frimpong was involved in the writing, reviewing, and editing of the work and fine-tuning the methodology. Nicholas Kwesi Prah II did the MATLAB simulation of the algorithm on the benchmark functions using the improved algorithms. David Boah Gyasi was involved in the coding of the improved particle swarm optimization algorithm.

Funding

Not applicable.

Availability of data and materials

The datasets used and/or analysed during the current study are available from the corresponding author on reasonable request. All data generated or analysed during this study are included in this published article.

Declarations

Competing interests

The authors declare that they have no known competing financial or non-financial interests that could have appeared to influence the work reported in this research.

Received: 7 August 2023 Accepted: 26 September 2024

Published online: 02 October 2024

References

1. Mohapatra S, Mohapatra P (2023) American zebra optimization algorithm for global optimization problems. *Sci Rep* 13(1):5211
2. Bäck TH, Kononova AV, van Stein B, Wang H, Antonov KA, Kalkreuth RT, Ye F (2023) Evolutionary algorithms for parameter optimization—thirty years later. *Evol Comput* 31(2):81–122
3. Ghasemi M, Kadkhoda Mohammadi S, Zare M, Mirjalili S, Gil M, Hemmati R (2022) A new firefly algorithm with improved global exploration and convergence with application to engineering optimization. *Decis Anal J* 5:100125

4. Prah II NK, Frimpong EA, Twumasi E (2022) Modified individual experience mayfly algorithm. *Carpathian J Electr Eng* 16(1)
5. Gad AG (2022) Particle swarm optimization algorithm and its applications: a systematic review. *Arch Comput Methods Eng* 29(5):2531–2561
6. Thomas J, Chaudhari NS (2014) A new metaheuristic genetic-based placement algorithm for 2D strip packing. *J Ind Eng Int* 10:47. <https://doi.org/10.1007/s40092-014-0047-9>
7. Haris PA, Gopinathan E, Ali CK (2010) Performance of some metaheuristic algorithms for multiuser detection in TTCM-assisted rank-deficient SDMA-OFDM system. *J Wirel Com Netw* 2010:473435. <https://doi.org/10.1155/2010/473435>
8. Fang J, Liu W, Chen L, Lauria S, Miron A, Liu X (2023) A survey of algorithms, applications and trends for particle swarm optimization. *Int J Netw Dyn Intell* 24–50
9. Wihartiko FD, Wijayanti H, Virgantari F (2018) Performance comparison of genetic algorithms and particle swarm optimization for model integer programming bus timetabling problem. In: *IOP conference series: materials science and engineering*, vol 332, p 012020, IOP Publishing
10. Zhang E, Nie Z, Yang Q, Wang Y, Liu D, Jeon SW, Zhang J (2023) Heterogeneous cognitive learning particle swarm optimization for large-scale optimization problems. *Inf Sci* 633:321–342
11. Vasanthi SVS, Babulal CBDC (2016) PSO with time varying acceleration coefficients for solving optimal power flow problem. *J Electr Eng* 16(3):10–10
12. Xu L, Song B, Cao M (2021) An improved particle swarm optimization algorithm with adaptive weighted delay velocity. *Syst Sci Control Eng* 9(1):188–197. <https://doi.org/10.1080/21642583.2021.1891153>
13. Kiani AT, Nadeem MF, Ahmed A, Khan IA, Alkhamash HI, Sajjad IA, Hussain B (2021) An improved particle swarm optimization with chaotic inertia weight and acceleration coefficients for optimal extraction of PV models parameters. *Energies* 14(11):2980
14. Yang W, Zhou X, Luo Y (2021) Simultaneously optimizing inertia weight and acceleration coefficients via introducing new functions into PSO algorithm. In: *Journal of physics: conference series*, vol 1754, No 1, p 012195, IOP Publishing
15. Farine DR (2022) Collective action in birds. *Curr Biol* 32(20):R1140–R1144
16. Vallee M (2021) Animal, body, data: starling murmurations and the dynamic of becoming in-formation. *Body Soc* 27(2):83–106
17. Azvolinsky A (2013) Birds of a feather... Track seven neighbours to flock together. *News at Princeton*
18. Richardson MJ, Chemero A (2014) Complex dynamical systems and embodiment. *The Routledge handbook of embodied cognition*, pp 39–50
19. Smaldino P (2023) *Modeling social behaviour: mathematical and agent-based models of social dynamics and cultural evolution*. Princeton University Press, Princeton
20. Papageorgiou D, Farine DR (2020) Group size and composition influence collective movement in a highly social terrestrial bird. *Elife* 9:e59902
21. Perinot E, Fritz J, Fusani L, Voelkl B, Nobile MS (2021) Characterizing the flying behaviour of bird flocks with fuzzy reasoning. In: *WILF*
22. Bajec IL, Heppner FH (2009) Organized flight in birds. *Anim Behav* 78(4):777–789
23. Benjamin D, Komlos D (2020) What leaders can learn from a flock of birds about the balance between leading and following. <https://www.forbes.com/sites/benjaminkomlos/2020/04/29/what-leaders-can-learn-from-a-flock-of-birds-about-the-balance-between-leading-and-following/?sh=3ca81a61c8a2>
24. Cavagna A, Cimarelli A, Giardina I, Parisi G, Santagati R, Stefanini F, Viale M (2010) Scale-free correlations in starling flocks. *Proc Natl Acad Sci* 107(26):11865–11870
25. Hamdan Ali H, Emad Kadhun L (2017) K-means clustering algorithm applications in data mining and pattern recognition. *Int J Sci Res ISSN* 6(8):1577–1584. <https://doi.org/10.21275/ART20176024>
26. Optimization Test Functions and Datasets. <https://www.sfu.ca/~ssurjano/optimization.html> (accessed Jun. 28, 2023)
27. Slowik A (2011) Particle swarm optimization. In: *The industrial electronics handbook-five volume set*, pp 1942–1948. https://doi.org/10.1007/978-3-319-46173-1_2
28. Chansamorn S, Somgiat W (2022) Improved particle swarm optimization using evolutionary algorithm. In: *2022 19th international joint conference on computer science and software engineering (JCSSE)*, pp 1–5. <https://doi.org/10.1109/JCSSE54890.2022.9836238>
29. Thengade A, Dondal R (2012) Genetic algorithm—survey paper. *MPCI national multi conference*, pp 975–8887

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.