ISSN 2442-6571

An improved particle swarm optimization based on lévy flight and simulated annealing for high dimensional optimization problem

Samar Bashath [a,1], Amelia Ritahani Ismail [b,2,*], Ali A. Alwan [b,3], Amir Aatieff Amir Hussin [b,4]

[a] Department of Computer Science, Kulliyyah of Information and Communication Technology, International Islamic University Malaysia, P.O Box 10, 50728 Kuala Lumpur, Malaysia
[b] Second affiliation, Address, City and Postcode, Country (9pt)
[1] samarsalim7076@gmail.com; [2] amelia@iium.edu.my; [3] aliamer@iium.edu.my; [4] amiraatieff@iium.edu.my
* corresponding author

ARTICLE INFO

Article history
Received August 23, 2021
Revised December 9, 2021
Accepted January 14, 2022
Available online March 31, 2022

Keywords
Particle Swarm Optimization
Levy Flight
Optimization
Simulated Annealing
High Dimensions

ABSTRACT

Particle swarm optimization (PSO) is a simple metaheuristic method to implement with robust performance. PSO is regarded as one of the numerous researchers' most well-studied algorithms. However, two of its most fundamental problems remain unresolved. PSO converges onto the local optimum for high-dimensional optimization problems, and it has slow convergence speeds. This paper introduces a new variant of a particle swarm optimization algorithm utilizing Lévy flight-McCulloch, and fast simulated annealing (PSOLFS). The proposed algorithm uses two strategies to address high-dimensional problems: hybrid PSO to define the global search area and fast simulated annealing to refine the visited search region. In this paper, PSOLFS is designed based on a balance between exploration and exploitation. We evaluated the algorithm on 16 benchmark functions for 500 and 1,000 dimension experiments. On 500 dimensions, the algorithm obtains the optimal value on 14 out of 16 functions. On 1,000 dimensions, the algorithm obtains the optimal value on eight benchmark functions and is close to optimal on four others. We also compared PSOLFS with another five PSO variants regarding convergence accuracy and speed. The results demonstrated higher accuracy and faster convergence speed than other PSO variants. Moreover, the results of the Wilcoxon test show a significant difference between PSOLFS and the other PSO variants. Our experiments' findings show that the proposed method enhances the standard PSO by avoiding the local optimum and improving the convergence speed.

1. Introduction

Various practical fields rely on optimization mechanisms to achieve high performance. To solve optimization problems, optimization algorithms are utilized in systems in various domains, including science, engineering, manufacturing, and economics [1], [2]. For each optimization problem, there are a set of possible solutions known as the solution space [3]. The possible solution identified by the optimization algorithm is usually considered a global optimum if it is better than the other feasible solutions (perhaps a local optimum) [3]. Many engineering and scientific optimization applications contain many decision variables (known as dimensions) [2]. These optimization applications may be represented as a high-dimensional optimization problem [1], [2]. This may include problems such as sour water stripping plants [4], heat exchangers in thermal fields [5], clustering for data mining [6], scheduling with a large number of resources [7], and vehicle routing problems [8]. Computing a global solution for the high-dimensional optimization problem is not easy. In high-dimensional optimization problems, the search space grows exponentially, affecting the exploration performance of the algorithm [9]–[11]. Also, local optima solutions increase extensively. To a certain extent, the algorithm will reach a state where it will be stuck inside some local optima where it cannot further explore the search space and find its way to the optimal solution [12], [13]. Therefore, the algorithm effectively requires search strategies to explore and exploit the search space [14]. Kennedy and Eberhartin [15] introduced one of the most efficient optimization algorithms called the Particles swarm optimization algorithm (PSO). PSO shows robust performance in various applications making it attractive to many researchers and engineers [16]–[18]. PSO has been widely applied for solving real-world optimization problems such as efficient power utilization, path planning of mobile robots, and different scientific problems [19]–[21].

PSO has two significant issues when dealing with high-dimensional optimization problems. First, PSO is always trapped into some local optima due to inefficient exploration [22], [23] and lack of exploitation capability [24], [25]. Second, PSO exploration involves many fitness evaluations, which may affect the convergence speed [10], [26]. Although several modifications of PSO have been introduced to tackle the high-dimensional optimization problem, becoming trapped in the local optimum and problems with acquiring inefficient solutions are persisting [12], [27]. Moreover, these modifications do not appropriately handle the relationship between exploitation and exploration [19]. For that, a good balance between exploration and exploitation is required for PSO to achieve a global solution for any high-dimensional optimization problem [28], [29]. This paper attempts to resolve the high-dimensional optimization problem by proposing a new variant of particle swarm optimization utilizing Lévy flight-McCulloch, and fast simulated annealing (PSOLFS). PSOLFS is meant to acquire a global optimum solution and fast convergence speed for the high-dimensional optimization problem. We design PSOLFS to have a balanced approach between exploration and exploitation. We implement Lévy flight-McCulloch in the PSO position to maximize the algorithm's exploration of the large search space and create diversity in each exploration's starting point position. We implement Fast simulated annealing in the late iteration to make the algorithm select the most accurate solution.

The following points summarize the contributions of this paper: 1) We discuss the problem of computing a global solution for the high-dimensional optimization problem; 2) We conduct a comprehensive review of the modifications already done to improve PSO; 3) We introduce a new variant of particle swarm optimization based on levy flight-McCulloch and fast simulated annealing named PSOLFS. The main aim of PSOLFS is to obtain a global solution for a high-dimensional optimization problem; 4) We evaluate PSOLFS on sixteen benchmark functions based on convergence accuracy and speed; and 5) We compare the efficiency of PSOLFS with five PSO variants through several experiments. The experimental results demonstrate that PSOLFS outperforms the existing approaches in convergence accuracy and speed. We organize this paper as follows. Section 2 provides a detailed review of PSO, Levy flight, and fast simulated annealing. We review several variants of PSO in section 3. In section 4, we explain PSOLFS and present its pseudocode and steps. We discuss the experiments and results in section 5. Finally, we outline the conclusion and some future work recommendations in section 6.

2. Related Work

This section explains the three algorithms used to develop the proposed approach of computing the global solution for the high-dimensional optimization problem. We review PSO, levy flight, and fast simulated annealing. For each of these algorithms, we explain the algorithm and its equations in detail.

2.1. Particle Swarm Optimization (PSO)

Kennedy and Eberhart first introduced a swarm intelligence-based algorithm named particle swarm optimization algorithm (PSO) in 1995 [15]. PSO is a stochastic robust optimization method that depends on swarms' movement and intelligence [19], [30]. The swarm consists of many particles theorized to move toward a better solution during exploitation and exploration of the search space [16]. The particles in the search space present various potential solutions. Each particle represents a solution in the search space of a given problem. The particles have two best positions; (1) personal ($Pbest$) and; (2) global $Gbest$ [29]. $Pbest$ presents the best solution that is achieved by an individual particle. $Gbest$

is the best solution located by the entire swarm. The two best positions make the particle learn from its experience and the entire group [26]. PSO relies on two characteristics belonging to every particle: position and velocity. These poisons are updated based on the fitness value comparison between the current and the new position. This process is repeated until the swarm finds the global or optimal solution. The new velocity particle is updated using Eq. (1) [15].

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

where i is the particle index; t is the number of iterations; $V(t)$ is the current velocity of the particle; $Xi(t)$ is the current position of the particle; $Pbest$ represents the best previous position of particle i; $Gbest$ represents the best position among all particles; $r1, r2$ random numbers with values between (0,1); $c1$, and $c2$ are positive numbers called acceleration coefficients that guide the particle toward the particle best and swarm best positions.

The velocity updating formula consists of three parts [26]: 1) The previous velocity of the particle $V(t)$: It serves as a memory of the previous swarm direction. The memory is called 'momentum' and prevents the particle from changing its direction. The momentum makes the particle move in the same previous iteration direction; 2) The cognitive part $c1 * r1 * (Pbest - Xi(t))$: It presents the personal experience of the particle. The cognitive part pulls the particle toward its best individual position; and 3) The social part $c2 * r2 * (Pbest - Xi(t))$: It represents the cooperation of knowledge between the particles. The particle updates its position based on Eq. (2) [15].

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

where $V(t + 1)$ presents the new velocity of the particle that is calculated using the formula given in Eq. (1).

PSO has some drawbacks that prevent the algorithm from working effectively [16]. Researchers have solved this problem by certain modifications on the basic version, among them Shi and Eberhart who introduced the inertia weight (w) to control exploration and exploitation in Eq. (2). The new velocity equation is in Eq. (3).

$$V_i(t+1) = w * V_i(t) + c_1 * r_1 * \left(P_{best} - X_i(t)\right) + c_2 * r_2 * (G_{best} - X_i(t)) \quad (3)$$

2.2. Levy Flight

Lévy flight is a type of random walk introduced by Paul Lévy in 1937 and has the characteristic of intensive probability in its movement [30]–[33]. Yang, Ting, and Karamanoglu [31] implied that Lévy flight has an excellent capability to explore the search space. A new swarm intelligence-based algorithm Cuckoo Search (CS) introduced by Yang and Deb [32] contributed Lévy flight in the search strategy of the algorithm. Many researchers examined CS, and the result showed that it had an advanced performance [32]. Two algorithms are used to create the random steps of levy flight, including Mantegna algorithm and McCulloch algorithm [32].

2.2.1. Mantegna Algorithm

Mantegna algorithm generates random steps based on Lévy stable distribution. Mantegna's algorithm makes random steps by Eq. (4) [34].

$$s = \frac{u}{|v|\frac{1}{\beta}} \quad (4)$$

where u and v identified from normal distributions, $u \sim N(0, \sigma_u^2)$, $v \sim N(0, \sigma_v^2)$ and σ_u is given by Eq.(5), where $\Gamma(\beta)$ is a Gamma function.

$$\sigma_u = \{\frac{(\Gamma(1+\beta) sin\ (\Pi\beta/2))}{(\Gamma[(1+\beta)/2]\beta 2^{((\beta-1)/2)}}\}(1/\beta) \quad (5)$$

2.2.2. McCulloch Algorithm

The McCulloch algorithm was implemented by McCulloch [33], [35] to create stable random variables. The following formula is used to return random steps [36].

$$\frac{(cN_1\, N_2)}{D} + \tau \backsim S\_a\,(c, \beta, \tau) \tag{6}$$

where

$N_1 = sign[\alpha\varphi + tan^{-1}(\beta tan\,(\alpha\pi/2))]$;

$N_2 = (cos\,[(1-\alpha)\varphi - tan^{-1}\,(\beta\, tan\,(\alpha\pi/2))])^{(1/\alpha-1)}$; and

$D = (cos\,[(1-\alpha)\varphi - tan^{-1}\,(\beta\, tan\,(\alpha\pi/2))])^{(1/\alpha-1)}\,(\cos(\varphi))^{(1/\alpha)\, w\text{^}(1/\alpha-1)}$.

The equations (6) return a n x m matrix of random numbers. Based on the equation, the required parameters are characteristic exponent α, skewness parameter β, scale c, and location parameter τ. The minimum value of α is 0.1 because of the non-negligible possibility of overflow. When an input is not in the valid range, the resultant matrix contains NaNs. The algorithm returns random numbers using Eq. (7).

$$x = \frac{(cos\,((1-\alpha)\varphi)/w)^{(1/\alpha-1)}\ sin\,(\alpha\varphi))/}{(cos\,(\alpha)\,)^{(1/\alpha)}} + \mu \tag{7}$$

where w, φ are independent random variable variables, α is an index of stability $\alpha \epsilon$ [0,2], σ scale parameter σ>0 and a location parameter $\mu \epsilon R$.

The McCulloch algorithm is a faster and more accurate mechanism than Mantegna [32], [33].

2.3. Fast Simulated Annealing

Fast simulated annealing (FSA) is a modified version of the simulated annealing algorithm (SA) [37]. SA has advantages over other local search methods, including taboo search and gradient descent [35]. SA can eliminate the local optimum and find the global optimum, and it has efficient exploitation capability [38]. FSA has a better performance than SA and can accelerate the creation of the neighbor's solutions [39]. FSA has two main factors: temperature reduction and neighbor points creation. FSA uses Eq. (8) to create random neighbor points [34].

$$G(x) = \frac{T}{((\Delta x^2 + T^2)^{(D+1)/2)}} \tag{8}$$

D is the dimensions of the search space. FSA uses Eq. (9) to reduce the temperature [34].

$$T_k = \frac{T^0}{K} \tag{9}$$

T_k is the current temperature, and T_0 is the initial temperature.

FSA starts its process by creating random points. Then, the algorithm determines the best solution by comparing the existing and nearest solutions. If the new solution has a better value than the previous one, the algorithm accepts the new solution. Otherwise, the algorithm determines the new solution by Eq. (10).

$$p = e^{\frac{(\Delta E)}{K}} > r \tag{10}$$

ΔE is the difference between the current and new point, and K is the number of iterations.

2.4. Modification Of the Velocity/Position Update Equations

Typically, PSO constitutes two equations: velocity and positions, as we explain in section 2.1. Several studies have introduced improvements in PSO by modifying the velocity and positions [29]. The work in [40] introduced PSO with adaptive inertia weight inertia to enhance the convergence speed of the particles. The size of the inertia weight was adjusted dynamically to control the speed of particles. The algorithm showed a slight improvement over PSO. The same idea was discussed in [41]; however, a population diversity function was used to maintain the inertia weight. The work proposed in [42] mutated the velocity of the particles. The mutation was performed on $Pbest$ and $Gbest$ to help the particles change their position when falling into the local optimum. The mutation used three mechanisms: Gauss, Lévy, and Cauchy probability [43]. Cauchy and Lévy were used to mutate the particles during the exploration, while Gauss mutated the particles during the exploitation. The algorithm showed improvement in convergence accuracy in some functions. The works contributed by [43] combined PSO with Lévy flight. A limit value was set for each particle. If the particle cannot improve its best position, the limit value is increased. If the particle exceeded the limit value, the particle updated the position based on the PSO equation; otherwise, the particle updated the position of particles based on Lévy flight [44]. The algorithm presented a soupier performance in the evaluation experiments. However, the dimensions of all experiments were quite small.

The work contributed by [26] attempted to enhance the algorithm proposed in [43]. In [24], the velocity, not the position, was updated using levy flight. The work introduced in [44] utilized three strategies to improve PSO: chaotic optimization, adaptive inertia weight, and dynamic linear acceleration factor. The chaotic method was performed to initialize PSO, the adaptive inertia weight was used to decrease the inertia weight during the iteration process, and the acceleration factor was performed to change the value of the acceleration in each iteration dynamically. The study [45] proposed a new modification of PSO using two strategies: dividing the particles into groups and exponential inertia weight. The proposed approach was based on dividing the particles into groups, each group using different acceleration factors to update the particle's velocity. The inertia weight was updated based on the exponential function. The proposed algorithm was still far from the global optimum. The work reported in [46] enhanced PSO using Lévy flight. The idea of the work was quite similar to the work in [40]. A random value was defined. The particle's position was updated based on the Lévy flight if the random value was larger than a specified number. Otherwise, the position was updated based on PSO equations. The approach introduced in [47], [48] mutated the particles' position of PSO with the help of Gaussian distribution. The proposed mechanism created a mutation in the selected particles' position. Combining PSO with Gauss was to make diversity on the swarm and improve the convergence accuracy. The work in [49] improved the exploration ability of PSO. The particles were derived into oscillatory trajectories based on several equations. The equations also updated the cognitive and social learning factors and the inertia weight settings. Lastly, The paper [50] introduced a multi-swarm strategy to adjust swarm size during the optimization process. Besides, an adaptive exploitation strategy is employed to maintain the fitness differences between exemplars and updated particles. In this way, the velocity of the particles is updated effectively. Although the algorithm appears to have high accuracy compared with other algorithms, it is long and complicated.

2.5. Hybrid PSO with Other Search Techniques

Hybrid PSO with other search techniques is one of the robust strategies that has attracted many researchers [51], [52], [53], [54]. The study presented in [55] proposed PSO with chaos. Chaos is a powerful search method characterized by high accuracy and high speed. The work presented in [56] improved the work in [49] by mutating the particles to narrow the search space. The work reported in [57] proposed a hybrid particle swarm optimization with Simulated Annealing (SA). SA was applied in the small area of search space. Also, the work in [52] introduced a hybrid PSO and SA with the same mechanism of the algorithm presented by the work in [51]. However, the work in [58] changed the temperature equation of SA.The approach in [44], [48] enhanced the local search ability of PSO by hybridizing PSO and grid search. The grid search method in this work searched for the global solution separately in each dimension. The work presented in [52] introduced a hybrid PSO with an artificial bee colony (ABC). The two algorithms searched for the global solution individually. The two algorithms worked simultaneously, and the information was outflowed between the two algorithms.

The work contributed by [28], [29] hybridized the artificial bee colony (ABC) and PSO to balance exploration and exploitation. PSO was performed for the exploitation mechanism, while the artificial bee colony performed the exploration. The work reported in [59] proposed a hybrid of PSO and harmony search (HS). In this study, PSO took advantage of the harmony search and harmony memory for searching for the global solution. The work presented in [23] announced a hybrid of particle swarm and genetic algorithm, in which the dimensionality reduction procedure was applied by dividing the swarm population into sub-swarms using the crossover operator of a genetic algorithm. Also, the mutation operator of the genetic algorithm was utilized in the swarm population to create diversity in the swarm. The paper of [24] introduced a new hybrid of PSO and SA. The algorithm worked by sharing the mechanism between the two algorithms. Last, the work in [53] proposed a hybrid PSO with an adaptive learning strategy. The algorithm applied a self-learning-based candidate generation strategy to enhance the exploration capability.

3. Method

We proposed a Particle Swarm Optimizations Based On Levy- Mcculloch Flight And Fast Simulated Annealing (PSOLFS) based on the two techniques: Lévy flight-McCulloch and fast simulated annealing. Fig.1 shows a flowchart of PSOLFS. We use these techniques to achieve a balance between exploration and exploitation.

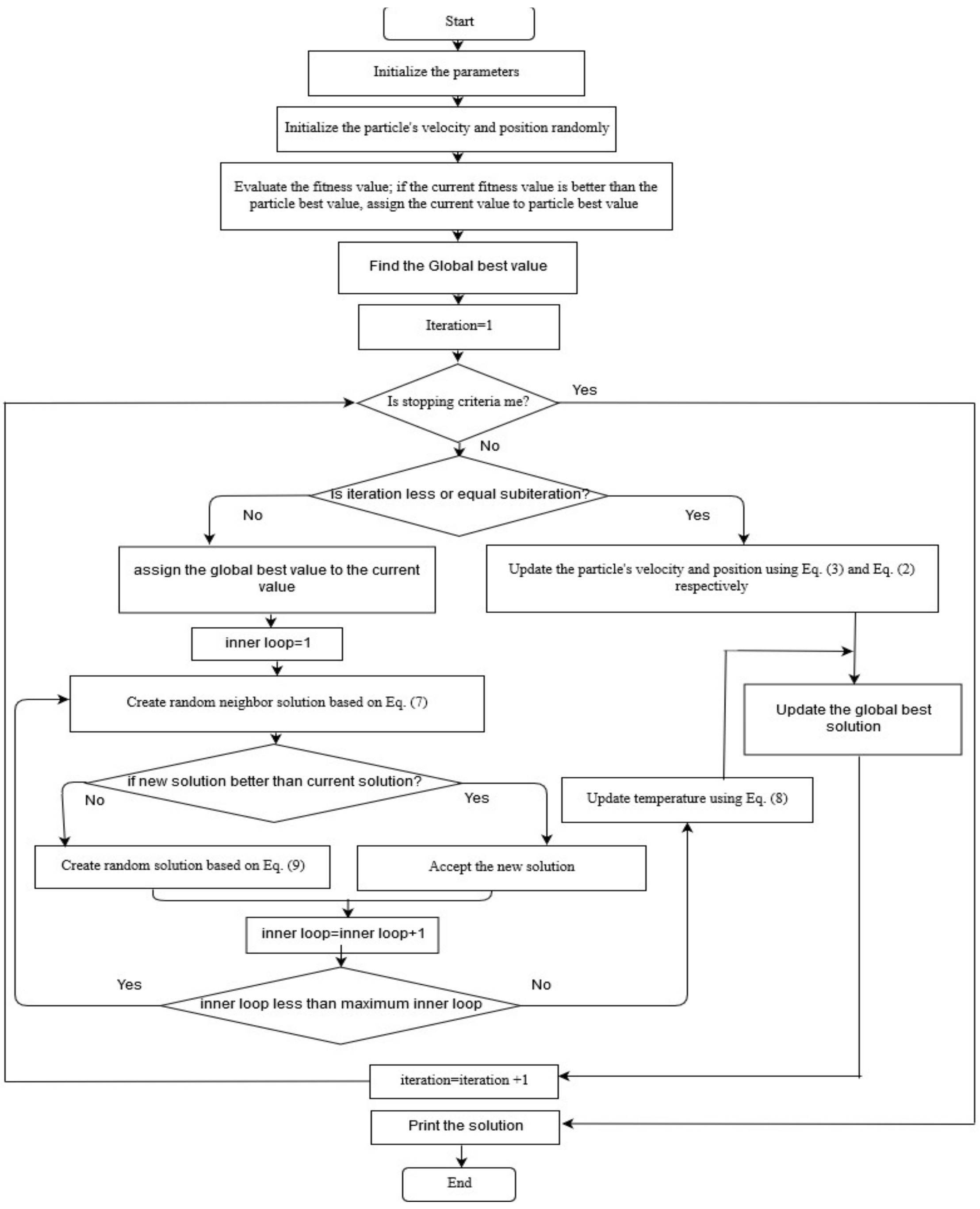

Start
Initialize the parameters
Initialize the particle's velocity and position randomly
Evaluate the fitness value; if the current fitness value is better than the particle best value, assign the current value to particle best value
Find the Global best value
Iteration=1
Is stopping criteria me?
Yes
No
Is iteration less or equal subiteration?
No
Yes
assign the global best value to the current value
Update the particle's velocity and position using Eq. (3) and Eq. (2) respectively
inner loop=1
Update the global best solution
Create random neighbor solution based on Eq. (7)
if new solution better than current solution?
No
Yes
Update temperature using Eq. (8)
Create random solution based on Eq. (9)
Accept the new solution
inner loop=inner loop+1
Yes
No
inner loop less than maximum inner loop
iteration=iteration +1
Print the solution
End

Fig. 1. Flowchart of PSOLFS algorithm

At the beginning of the iteration, high exploration gives a high chance to find close to the optimal solutions. While at the end of the iteration, high exploitation gives the particles a chance to find the most accurate solution within the promising area. Using Lévy flight- McCulloch, the particles have two merits over PSO. First, the particles can explore a wide search area and explore the search space with a significant jump size to find a promising region. This procedure takes less time due to the mechanism of McCulloch over another type of Lévy Mantegna. Second, the particles have diversity in their position, which they gain from the various values of a Lévy variance. By doing so, the particles do not lose their ability to update their position. After many iterations, PSOLFS calls the FSA algorithm. FSA is utilized to refine the global area, so FSA help to find the most accurate results. FSA is applied in the promising area founded by PSO- Lévy flight- McCulloch. As the original PSO, the particles are randomly initialized in the search space. The particles' fitness values are evaluated to determine the personal best fitness value $Pbest$ and swarm best fitness value $Gbest$. The particle updates its velocity using the original PSO equation given by Eq. (3). The particle updates its position using Eq. (11) instead of Eq. (2).

$$Xi(t+1) = Vi(t+1) + \text{Lévy}\,flight\,(Xi(t)) \tag{11}$$

where Lévy flight $(X(t)) = Xi(t) + step \oplus$ random (size $(Xi(t))$, and the step is determined by Eq. (6), $\oplus$ which presents element-by-element multiplication, and random is the random number for all dimensions' sizes. Once the algorithm reaches a defined sub-iteration number, the algorithm applies fast simulation in the founded global area. The algorithm creates random points using Eq. (8) and evaluates the points. If the previous solution is better than the new solution, the algorithm creates points based on Eq. (10) and decreases the temperature using Eq. (9). The algorithm creates the point, evaluates the solution, and decreases the temperature until it reaches the stopping criteria. With the help of FSA, PSOLFS can exploit the global area to find the most accurate solution.

4. Results and Discussion

This section presents and discusses the evaluation experiments of the PSOLFS algorithm. We consider various algorithms for the comparison experiments. We carry out the experiments on a PC with an Intel Core i5 M460, 2.5 GHz CPU, 4GB RAM, and Windows 10, 64-bit operating system. Subsection A outlines the benchmark functions considered in this work. We present the convergence accuracy and algorithm convergence speed in subsections B and C. Section D shows the statistical analysis.

4.1 Benchmark Functions

This section lists the benchmark functions used in this work to validate the proposed algorithm. The functions considered in this work are the most popular functions used in most of the previous works, including these studies [54], [60], [61]. The first six functions are unimodal, while the functions from $f_7 - f_{16}$ are multimodal. The modality presents the number of local optima in functions. The unimodal functions have one local optimum, while the multimodal functions have one or more local optimums [62]. Multimodal functions constitute the most difficult class of benchmark functions [63]. The unimodal functions are utilized to prove the solution quality and convergence of an algorithm; however; the multimodal functions are used to determine how effectively an algorithm avoids local optimum solutions [64]. All the tests functions have a minimum value at $x^{\wedge} *= (0,0,\dots,0)$ for $i = 1,2,3,..D$. Table 1 summarizes the details of each benchmark function considered in this work. The table presents the function name, the corresponding formula, the range of the dimension, and the modality of the function.

Table 1. The benchmark functions used in our experiments

Function Name	Formula	Range	Modality
Rosenbrock	$\boldsymbol{f_1(x) = \sum_{i=1}^{D-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]}$	[-5,5]	Unimodal
Sphere	$f_2(x) = \sum_{i=1}^{D} x_i^2,$	[-100,100]	Unimodal
Step	$f_3(x) = \sum_{i=1}^{D} (x_i + 0.5)^2$	[-100,100]	Unimodal

Table 1. (Cont.)

Function Name	Formula	Range	Modality
Quatic	$f_4(x) = \sum_{i=1}^{D} ix_i^4 + random[0,1]$	[-1.28,1.28]	Unimodal
Quadric	$f_5(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j\right)^2$	[-1.28,1.28]	Unimodal
Ellipitic	$f_6(x) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} \times x_i^2$	[-1,1]	Unimodal
Rastrigin	$f_7(x) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	[-5.12,5.12]	Multimodal
Ackley	$f_8(x) = 20 + \exp(1) - 20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x^2}\right) - \exp\,(1/D \sum_{i=1}^{D} \cos\,(2\pi x_i)$	[-5,5]	Multimodal
Griewank	$f_9(x) = \sum_{i=1}^{D} xx_i^2/4000 - \prod_{i=1}^{D} \cos(x_i/sqrt(i)) + 1$	[-600,600]	Multimodal
Schwefel2.4	$f_{10}(x) = \sum_{i=1}^{D} (x_i - 1)^2 + (x_1 - x_i^2)^2$	[0,10]	Multimodal
Generalized penalized 1	$f_{11}(x) = \frac{\pi}{D}\left\{10(sin(\pi y_1))^2 + \sum_{i=1}^{D-1} (y_i - 1)^2 \left[1 + 10(sin(\pi y_{i+1}))^2\right] + (y_D - 1)^2\right\} + \sum_{i=1}^{D} u(x_i, 10,100,4)$ Where $y_i = 1 + \frac{(x_i+1)}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - \alpha)^m \;\; x_i > \alpha \\ 0 - \alpha \le x_i \le \alpha \;, \quad i = 1, \ldots, D \\ k(-x_i - \alpha)^m \;\; x_i < -\alpha \end{cases}$	[-50,50]	Multimodal
Generalized penalized 2	$f_{12}(x) = \sum_{i=1}^{D} 0.1\left\{sin(3\pi x_i)^2 + \sum_{i=1}^{D-1} (x_i - 1)^2 \left[1 + (sin(3\pi x_{i+1}))^2\right] + (x_D - 1)^2\left[1 + (sin(2\pi x_D))^2\right]\right\} + \sum_{i=1}^{D} u(x_i, 5,100,4)$	[-50,50]	Multimodal
Qing	$f_{13}(x) = \sum_{i=1}^{D} (x_i^2 - i)^2$	[-100,100]	Multimodal
Salmon	$f_{14}(x) = 1 - cos\left(2\pi\sqrt{\sum_{i=1}^{D} x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{D} x_i^2}$	[-100,100]	Multimodal
Quintic	$f_{15}(x) = \sum_{i=1}^{D} \lvert x_i^5 - 3x_i^4 + 4x_i^3 + 2x_i^2 - 10x_i^1 - 4\rvert$	[-10,10]	Multimodal
Alpine	$f_{16}(x) = \sum_{i=1}^{D} \lvert x_i sing(x_i) + 0.1x_i\rvert$	[-10,10]	Multimodal

4.2 Convergence Accuracy Results on 500 and 1000 dimensions

This section presents and discusses the convergence accuracy results in terms of mean and standard deviation. For each 16 test functions, the six algorithms PSO, PSO-SA, LFPSO, PSOLF, LFAPSO, and PSOLFS are run independently for about 20 times on 500 and 1000 dimensions. Table 2 demonstrates the convergence accuracy results in terms of the mean and standard deviation of 20 runs with 20000 fitness function evaluations (FEs) of sixteen benchmark functions on 500 dimensions for the six algorithms (PSO, PSO-SA, LFPSO, PSOLF, LFAPSO, and PSOLFS). We can see from the result in Table 2 that PSOLFS outperforms other PSO variants in all functions for both mean and standard deviation. PSOLFS has an optimum value in 14 out of 16 functions. PSOLFS has an optimum value in all six unimodal functions f_1, f_2, f_3, f_4, f_5, and f_6 and in eight multimodal functions f_7, f_8, f_9, f_{10}, f_{11}, f_{14}, f_{15} and f_{16}. However, PSOLFS doesn't obtain the optimal value for functions f_{11} and f_{12}. We can also notice that among the other PSO variants, PSOLF has a better-quality solution than the others; it achieves the optimal results in the three functions. PSO, PSO-SA, LFPSO, and LFAPO algorithms obtain a better outcome for f_{13} and f_{14} comparing to the other functions.

Table 2. Convergence Accuracy of the Algorithms on 500 dimensions

Functions	Values	PSO	PSO-SA	LFPSO	PSOLF	LFAPSO	PSOLFS
$f1$	MEAN	6.84E+5	3.07E+5	2.07E+2	2.67E+2	1.64E+3	0
	STD	3.24E+5	1.45E+5	1.52E+2	9.51E+1	7.76E+2	0
$f2$	MEAN	4.87E+7	5.45E+6	4.85E-8	5.35E-19	4.85E+4	0
	STD	2.34E+6	1.34E+5	2.52E-10	7.41E-19	5.79E+2	0
$f3$	MEAN	9.53E+4	7.32E+3	5.63E-7	5.83E-29	5.53E+4	0
	STD	6.43E+4	4.31E+3	3.25E-8	3.48E-29	8.70E+3	0
$f4$	MEAN	1.34E+4	2.54E+4	5.63E+3	4.05E+3	4.56E+4	0
	STD	9.73E+3	3.96E+4	2.10E+3	6.68E+2	1.67E+4	0
$f5$	MEAN	7.32E+6	4.34E-9	3.20E-13	4.63E-20	4.47E-12	0
	STD	3.02E+5	3.87E-11	3.98E-16	6.81E-22	3.90E-13	0
$f6$	MEAN	9.64E+7	3.56E+7	7.97E-14	7.74E-17	5.38E-11	0
	STD	3.46E+6	7.99E+6	6.43E-15	4.19E-18	1.35E-11	0
$f7$	MEAN	7.34E+5	8.93E+3	5.76E+1	7.24E-56	6.53E+1	0
	STD	3.79E+4	5.76E+3	2.47E+2	7.81E-58	4.79E+2	0
$f8$	MEAN	4.67E+3	2.46E+3	1.45E+1	0	2.23E+1	0
	STD	5.33E+2	9.88E+2	1.42E+1	0	1.48E+1	0
$f9$	MEAN	9.60E+4	3.48E+1	3.69E-6	0	6.34E+1	0
	STD	3.32E+3	2.74E+1	9.65E-7	0	1.71E+2	0
$f10$	MEAN	4.30E+4	6.42E+4	1.65E+2	2.32E-6	4.34E+2	0
	STD	9.12E+3	2.34E+3	7.82E+1	5.29E-7	6.55E+2	0
$f11$	MEAN	8.32E+3	1.07E+4	9.98E-2	4.14E-4	7.92E+2	0
	STD	2.54E+3	7.23E+3	5.76E-2	7.37E-5	6.92E+2	8.05E-36
$f12$	MEAN	9.64E+4	7.43E+4	6.87E+2	8.47E-3	6.86E+4	1.76E-38
	STD	7.53E+4	6.07E+4	4.87E+2	2.83E-3	4.32E+4	7.65E-33
$f13$	MEAN	5.32E-5	5.31E-5	9.78E-13	3.45E-16	4.87E-5	9.85E-36
	STD	4.32E-4	3.76E-4	9.12E-14	9.25E-18	9.86E-8	0
$f14$	MEAN	3.27E-2	3.21E-9	4.45E-12	7.54E-13	5.43E-6	0
	STD	7.87E-1	5.37E-7	8.45E-12	3.73E-14	9.30E-8	0
$f15$	MEAN	7.35E+8	4.21E+4	3.76E-8	3.56E-6	5.78E+1	0
	STD	5.35E+8	3.70E+3	9.54E-10	6.87E-7	8.49E+1	0
$f16$	MEAN	7.89E+7	7.33E+5	1.43E+6	0	1.85E+3	0
	STD	3.23E+6	3.56E+5	8.56E+5	0	2.65E+3	0

As we can see from Table 3, PSOLFS maintains a stable performance despite the increment in the number of dimensions to 1000. Increasing the dimensions of the functions did not prevent the algorithm from obtaining the optimum value. Table 6 demonstrates the convergence accuracy results in terms of the mean and standard deviation of 20 runs with 50000 fitness evaluation function (FEs) of sixteen benchmark functions on 1000 dimensions for the six algorithms (PSO, PSO-SA, LFPSO, PSOLF, LFAPSO, and PSOLFS). PSOLFS still outperforms the five PSO variants. PSOLFS yields the optimum value on the functions: f_3 , f_5 , f_6 , f_7 , f_9 , f_{13}. PSOLFS still outperforms the five PSO variants. PSOLFS yields the optimum value on the functions f_3 , f_5 , f_6 , f_7 , f_9 , f_{13}, f_{14}, f_{15} and close to optimal results on f_2, f_{12}. It obtains high accuracy results on f_1, f_4, f_{10}, f_{11}. PSOLF outperforms PSOLFS in the two functions f_8 and f_{16}.

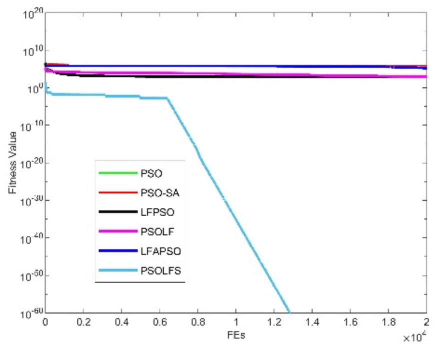
Table 3. Convergence Accuracy of the Algorithms on 1000 Dimensions

Functions	Values	PSO	PSO-SA	LFPSO	PSOLF	LFAPSO	PSOLFS
$f1$	MEAN	3.29E+8	9.14E+6	6.36E+5	7.84E+5	2.54E+6	9.542E-9
	STD	9.76E+7	2.45E+6	3.51E+5	1.24E+5	7.30E+5	6.65E-10
$f2$	MEAN	4.30E+5	4.39E+5	3.42E-4	9.67E-22	3.69E+3	9.74E-56
	STD	3.2E+5	1.86E+5	4.32E-6	8.05E-25	1.06E+3	3.22E-58
$f3$	MEAN	4.32E+5	6.34E+5	4.86E-6	5.31E-23	7.38E+4	0
	STD	3.39E+5	7.37E+5	7.48E-8	9.93E-25	3.23E+4	0
$f4$	MEAN	1.08E+5	3.97E+5	3.52E+4	4.43E+4	7.20E+5	3.37E-4
	STD	3.56E+5	9.46E+4	7.40E+3	9.26E+3	8.57E+4	7.29E-6
$f5$	MEAN	1.08E+5	3.93E+5	3.92E-14	4.13E-18	6.43E-10	0
	STD	3.56E+5	9.46E+4	3.51E-15	6.61E-18	8.97E-11	0
$f6$	MEAN	5.23E+7	9.38E+4	3.13E-12	4.82E-15	7.32E-10	0
	STD	3.12E+7	9.46E+4	3.87E-15	9.35E-17	7.20E-10	0
$f7$	MEAN	3.09E+5	1.29E+5	8.74E+1	0	2.46E+4	0
	STD	1.23E+5	3.54E+5	6.43E+1	0	4.74E+3	0
$f8$	MEAN	6.79E+8	2.37E+9	3.55E+1	4.34E-17	5.08E+1	2.45E-15
	STD	4.32E+8	9.87E+7	1.44E+1	3.65E-18	5.96E+1	6.04E-16
$f9$	MEAN	2.21E+4	6.34E+3	4.21E-7	0	6.33E+2	0
	STD	4.75E+4	6.53E+3	8.76E-6	0	5.28E+2	0
$f10$	MEAN	3.96E+6	4.79E+6	4.73E-3	5.91E-4	4.56E+2	6.17E-10
	STD	6.52E+6	3.31E+6	4.34E-3	2.01E-5	7.15E+2	4.55E-10
$f11$	MEAN	6.53E+5	6.76E+5	4.83E-3	6.98E-2	7.24E+2	7.24E-10
	STD	4.32E+5	1.39E+5	8.26E-3	6.56E-3	7.62E+2	8.68E-11
$f12$	MEAN	6.78E+2	3.06E+2	6.87E-23	7.88E-2	2.99E+4	8.70E-31
	STD	4.96E+2	1.84E+2	4.36E-23	8.46E-3	6.46E+3	1.89E-32
$f13$	MEAN	4.67E+5	2.05E+5	4.50E-10	6.26E-12	4.36E-7	0
	STD	2.90E+5	1.63E+5	2.19E-10	2.14E-12	7.42E-8	0
$f14$	MEAN	5.59E-4	3.28E-5	2.89E-9	4.73E-10	5.75E-7	0
	STD	2.94E-4	1.95E-5	7.73E-10	2.97E-10	9.58E-8	0
$f15$	MEAN	3.31E+8	5.12E+8	5.09E-6	6.29E-3	1.76E+3	0
	STD	1.98E+8	3.28E+8	3.68E-7	4.73E-4	9.35E+2	0
$f16$	MEAN	4.54E+4	8.52E+3	3.49E+6	4.10E-34	6.28E+3	4.21E-14
	STD	2.05E+3	5.64E+2	9.86E+6	8.19E-36	3.74E+3	6.49E-15

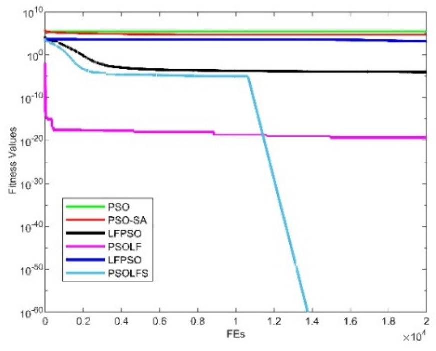
4.3 Convergence Speed Results on 500 and 1000 Dimensions

This section presents and discusses the results of the convergence speed of six algorithms on the 16 benchmark functions for 500 and 1000 dimensions. Fig. 2(a-h) plots the convergence speed of six algorithms on Rosenbrock, Sphere, Step, Quartic, Quadric, Elliptic Rastrigin, and Ackley for 500 dimensions. Fig. 3 (a-h) plots the convergence speed of the algorithms on Griewank, Schwefel 2.4, Generalized penalized 1, Generalized penalized 2, Qing, Salomon, Qunitic and Alpine for 500 Dimensions. Fig. 4 (a-h) plots the convergence speed of the six algorithms on Rosenbrock, Sphere, Step, Quartic, Quadric, Elliptic Rastrigin, and Ackley for 1000 dimensions. Fig. 5(a-h) plots the convergence speed results of the algorithms on Griewank, Schwefel 2.4, Generalized penalized 1, Generalized penalized 2, Qing, Salomon, Qunitic, and Alpine for 1000 dimensions.

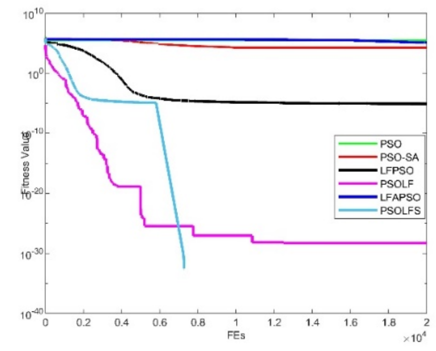
The x-axis represents the fitness evolution value (FEs). The x-axis indicates the convergence speed, and the y-axis represents the optimization value produced by each algorithm. On the Rosenbrock function, all the algorithms expect PSOLFS fall into the local optimum. PSOLFS converges to the global optimal at about 12763 FEs, as shown in Fig. 2(a). The three algorithms PSO-SA, and LFAPSO fall into the local optimum on the Sphere function, as shown in Fig. 2(b). PSOLF and LFPSO obtain a result that is closes to the global optimum. However, PSOLFS achieves the global optimum.



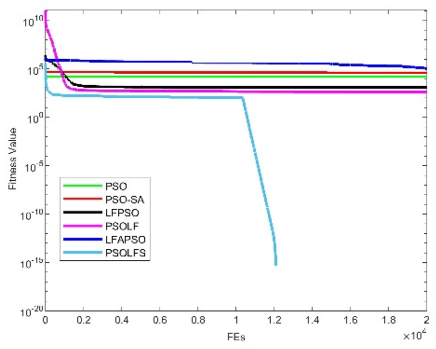
(a)Rosenbrock



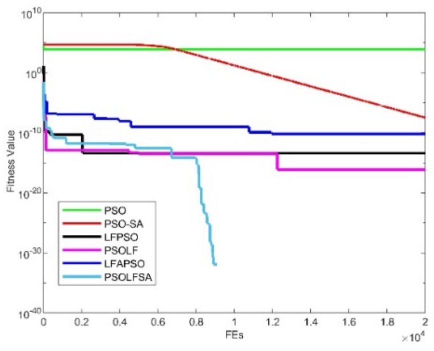
(b)Sphere



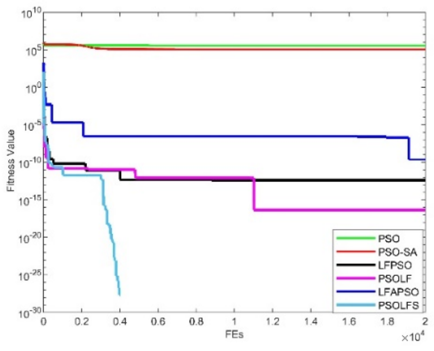
(c) Step



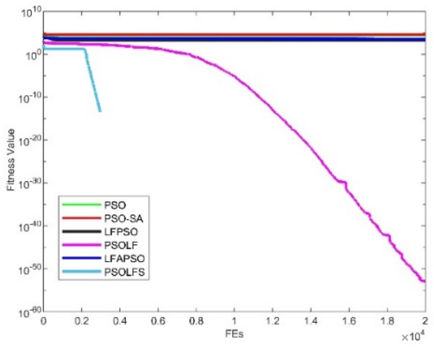
(d)Quartic



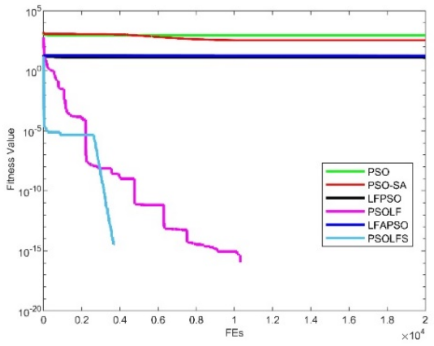
(e) Quadric



(f) Elliptic

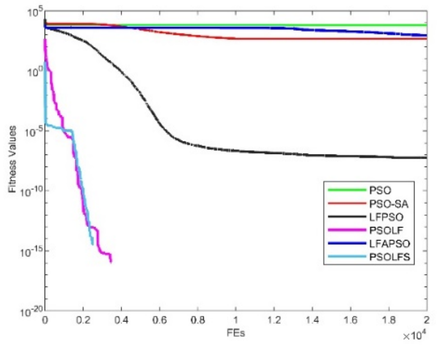


(g)Rastrigin

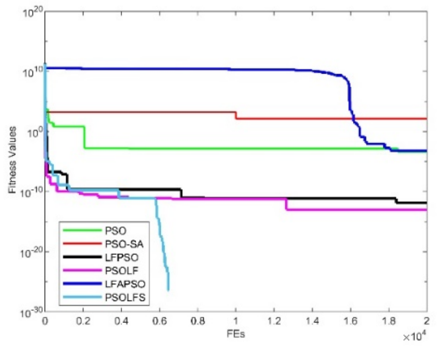


(h) Ackley

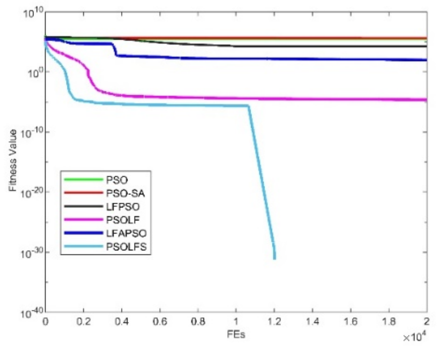
Fig. 2. Convergence Speed Results on Rosenbrock, Sphere, Step, Quartic, Quadric, Elliptic, Rastrigin and Ackley for 500 Dimensionss



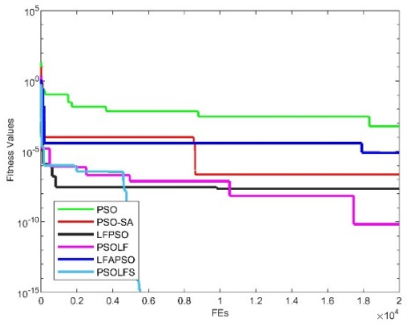
(a) Griewank



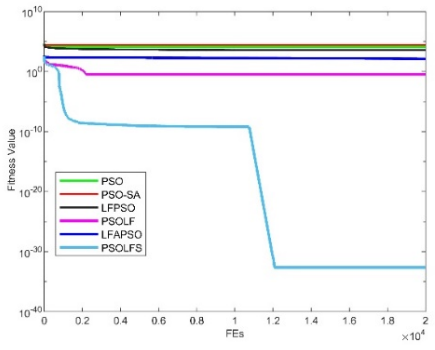
(e) Qing



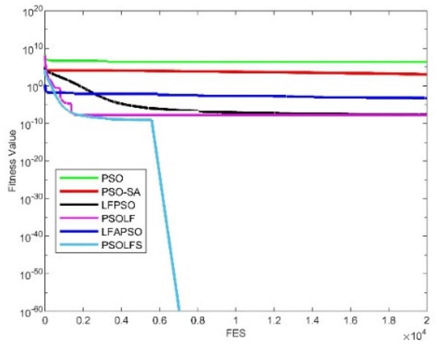
(b) Schwefel2.4



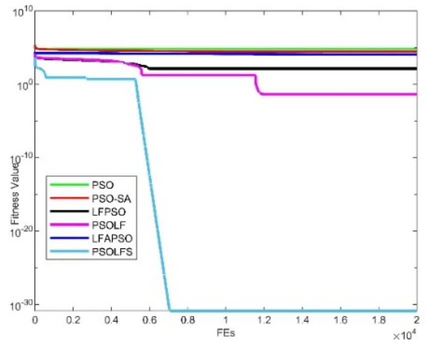
(f) Salmon



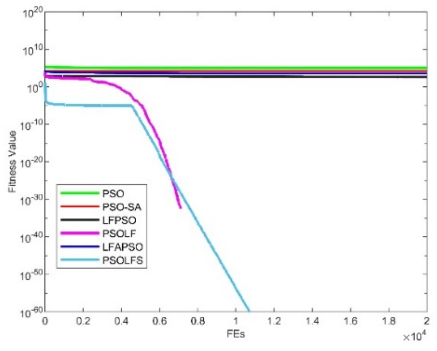
(c) Generalizedpenalized 1



(g) Qunitic



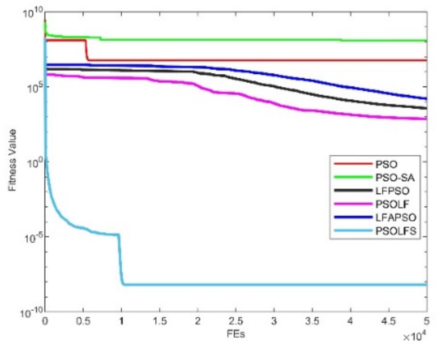
(d) Generalizedpenalized 2



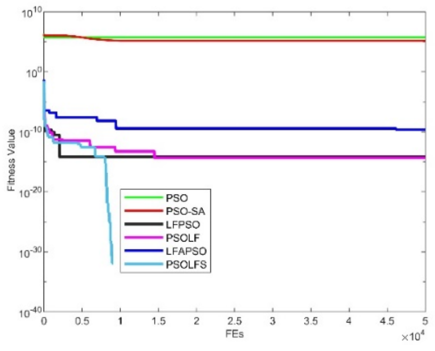
(h) Alpine

Fig. 3. Convergence speed results on Ackley, Griewank, Schwefel2.4, Generalized penalized 1, Generalized penalized 2, Qing, Salomon, Qunitic, and Alpine for 500 Dimensions

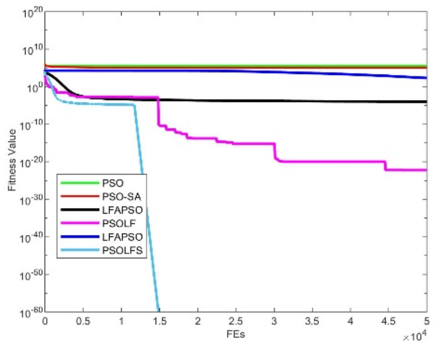
On the Step function, the algorithms PSO, PSO-SA, and LFAPSO fall into the local optimum, as seen in Fig. 2(c). LFPSO obtains better results than PSO, PSO-SA, LFAPSO. PSOLF reaches a solution, which is near to the optimal. PSOLFS reaches the optimum value 0 at 7742. PSOLFS gives an optimum value on the Quartic function at 12088, while all other algorithms fall into the local optimum, as shown in Fig. 2(d). PSO gives the worst results on the Quadric function among all the algorithms, as depicted in Fig. 2(e). PSOLF shows better results than PSO-SA, LFPSO, LFAPSO. PSOLFS converges quickly to the optimal solution at 9175.



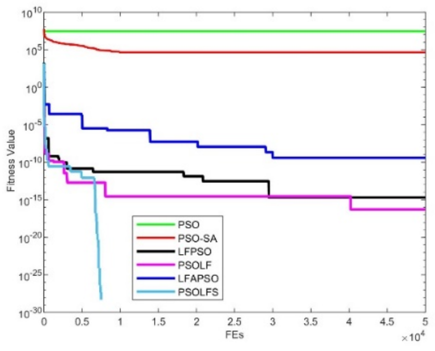
(a)Rosenbrock



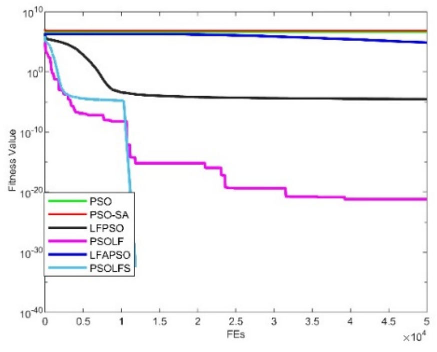
(e) Quadric



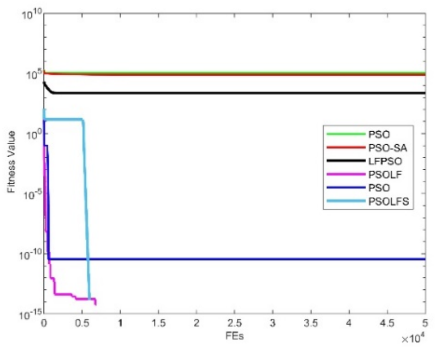
(b)Sphere



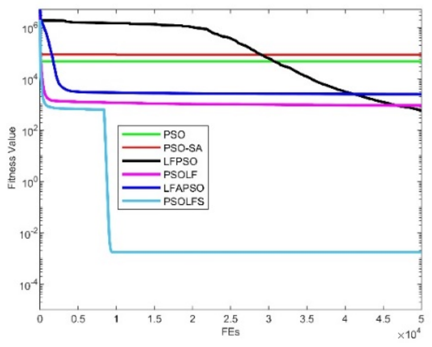
(f) Elliptic



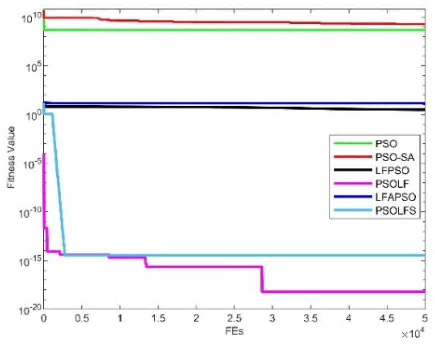
(c) Step



(g) Rastrigin

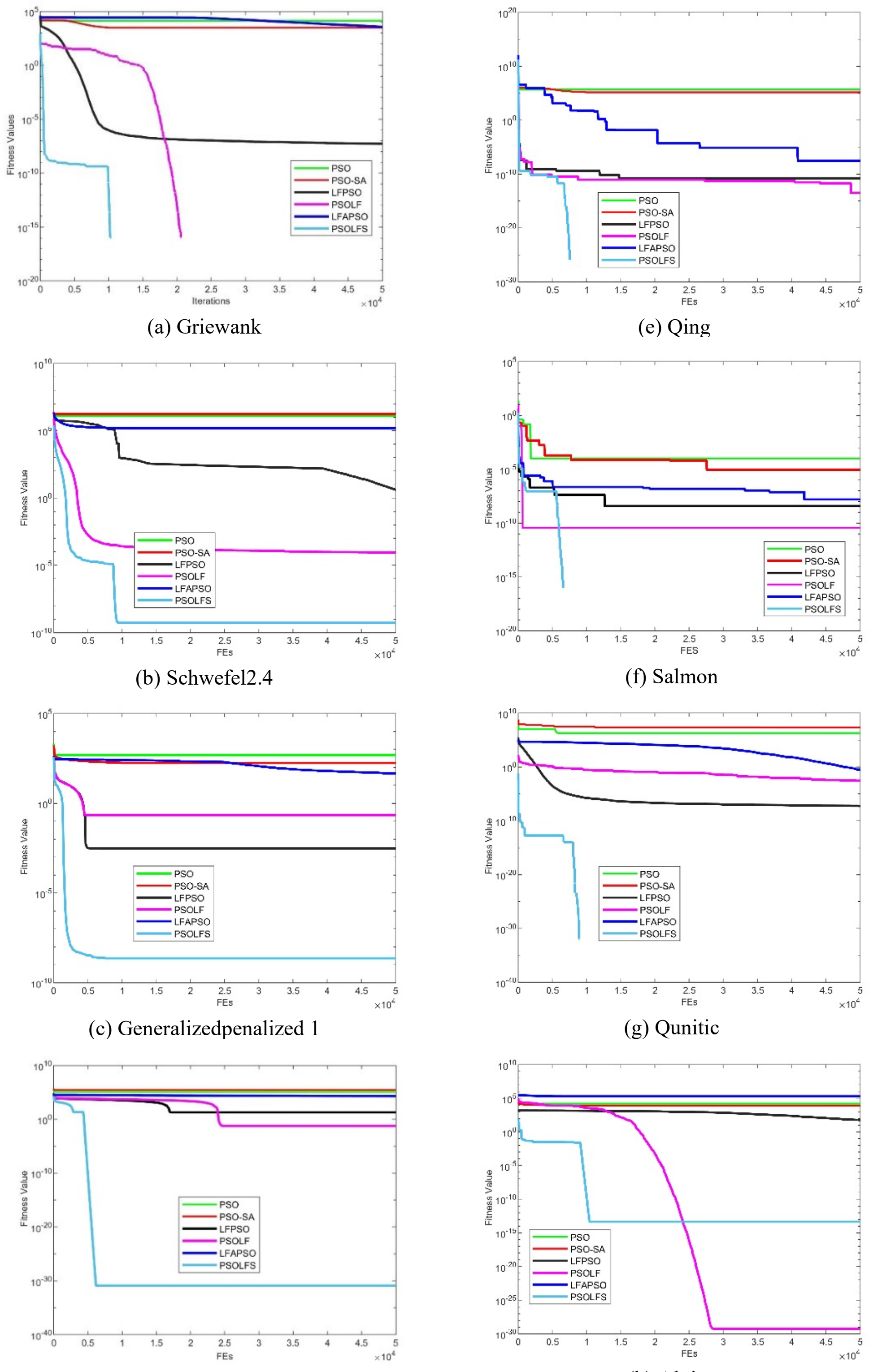


(d) Quartic



(h) Ackley

Fig. 4. Convergence Speed Results on Rosenbrock, Sphere, Step, Quartic, Quadric, Elliptic, Rastrigin and Ackley for 1000 Dimensions



(a) Griewank (e) Qing

(b) Schwefel2.4 (f) Salmon

(c) Generalizedpenalized 1 (g) Qunitic

(d) Generalizedpenalized 2 (h) Alpine

Fig. 5. Convergence speed results on Ackley, Griewank, Schwefel2.4, Generalized penalized 1, Generalized penalized 2, Qing, Salomon, Qunitic, and Alpine for 1000 Dimensions

The last unimodal function is Elliptic, where PSOLFS obtains the optimal solution in a small number of FEs, which is 4023, as illustrated in Fig. 2(f). For the multimodal functions: Rastrigin, Ackley and Griewank, the two algorithms PSOLF and PSOLFS converge to a global optimum, as demonstrated in Fig. 2(g), Fig. 2(h), and Fig. 3(a), respectively. The graph demonstrates that PSOLFS has faster convergence speed with low numbers of FEs. LFPSO obtains a better accuracy on Griewank than PSO, PSO-SA, and LFAPSO. From Fig. 3(b), we can see that PSOLFS gives a global optimum solution at about 12000 FEs, while other algorithms fail to give good results on Schwefel 2.4.

On Generalized penalized 1 and Generalized penalized 2, all the algorithms fail to obtain the global optimum, as shown in Fig. 3(c) and Fig. 3(d), respectively. PSOLFS algorithm achieve a result that is near to global optimal. For Qing, Salomon, and Quintic benchmark function, PSOLFS algorithm converges quickly to the optimal solution, as depicted in Fig. 3 (e), Fig. 3 (f), and Fig. 3 (g), respectively. Lastly, on the alpine function, PSOLF converges faster to the optimal solution than PSOLFS as shown in Fig. 3(h). The other four algorithms were unable to generate the global optimum. We observe from the results reported in Fig. 2 and Fig. 3 that PSOLFS always attempts to jump out of local optima and managed to generate the global optimum solution for all functions. It converges quickly to the global optimum during the convergence process.

From Fig. 4(a), On the Rosenbrock function, all the algorithms fall into the local optimum, but only PSOLFS gives better results in FEs number 11432. The three algorithms PSO, PSO-SA, LFAPSO converge into the local optimum on the Sphere and Step function. PSOLF obtains a close result to the global optimum, and PSOLFS achieves the global optimum, as depicted in Fig. 4(b) and Fig. 4(c). However, PSOLFS fails to find the global solution on Quartic as shown in Fig. 4(d), POSLFS gives the best solution compared with the other algorithms. The convergence speed of Quadric and Elliptic demonstrates that PSOLFS gives a global solution in less than 10000 FEs, as shown in Fig. 4(e) and Fig. 4(f). We also notice that the PSO and PSO-SA converge quickly into the local optimum, while other algorithms converge to a better-quality solution with many FEs. For Rastrigin, PSOLF and PSOLFS converge to the global optimum; however, PSOLFS shows a faster convergence speed with few FEs, as shown in Fig. 4(g). From the result given in Fig. 4(h), we observe that PSOLFS converges quickly to the global solution in less than 4,000 FEs; however, PSOLF outperforms PSOLFS in the accuracy solution. The other algorithm cannot converge to the global solution. Fig. 5(a) shows that PSOLF and PSOLFS converge to the global optimum on Griewank.

Fig. 5 (b) depicts that PSOLFS obtains better results in a few numbers of FEs. Fig. 5(c) and Fig. 5(d) show that the five previous algorithms achieve local optimum results on Generalized penalized 1 and Generalized penalized 2, while PSOLFS accomplishes a near-optimal result. Fig. 5(e), Fig. 5(f), and Fig. 5(g) prove that PSOLFS converges quickly to the optimal solution. The result presented in Fig. 5(h) shows that the PSOLF converges faster to the near-optimal solution than PSOLFS, while the other four algorithms are stuck in the local optimum. From the results of the experiment accomplished on the 1000 dimensions. We conclude that PSOLFS obtains the optimal solution in fewer fitness functions than other algorithms.

4.4 Non-parametric Test Results

Statistical tests have become essential measurements to evaluate the new algorithm's performance compared with other algorithms [65]. One of these tests is the Wilcoxon test. Wilcoxon test is a pairwise statistical test that works to find the statistical difference between two algorithms when applied on a set of problems [66]. For the Wilcoxon test, we state ten hypotheses describing what the test will incorporate. We hypothesize the following.

$H1_0$: The mean of the convergence accuracy achieved by the PSOLFS is equal to the mean of the convergence accuracy achieved by PSO for all benchmark functions on 500 dimensions after 20 runs.

$H2_0$: The mean of the convergence accuracy achieved by the PSOLFS is equal to the mean of the convergence accuracy achieved by PSO-SA for all benchmark functions on 500 dimensions after 20 runs.

$H3_0$: The mean of the convergence accuracy achieved by the PSOLFS is equal to the mean of the convergence accuracy achieved by LFPSO for all benchmark functions on 500 dimensions after 20, which is near to the optimal. PSOLFS reaches the optimum value 0 at 7742.

$H4_0$: The mean of the convergence accuracy achieved by the PSOLFS is equal to the mean of the convergence accuracy achieved by PSOLF for all benchmark functions on 500 dimensions after 20 runs.

H5$_0$: The mean of the convergence accuracy achieved by the PSOLFS is equal to the mean of the convergence accuracy by LFAPSO for all benchmark functions on 500 dimensions after 20 runs.

H6$_0$: The mean of the convergence accuracy achieved by the PSOLFS is equal to the mean of the convergence accuracy achieved by PSO for all benchmark functions on 1000 dimensions after 20 runs.

H7$_0$: The mean of the convergence accuracy achieved by the PSOLFS is equal to the mean of the convergence accuracy achieved by PSO-SA for all benchmark functions on 1000 dimensions after 20 runs.

H8$_0$: The mean of the convergence accuracy achieved by the PSOLFS is equal to the mean of the convergence accuracy achieved by LFPSO for all benchmark functions on 1000 dimensions after 20 runs.

H9$_0$: The mean of the convergence accuracy achieved by the PSOLFS is equal to the mean of the convergence accuracy achieved by PSOLF for all benchmark functions on 1000 dimensions after 20 runs.

H10$_0$: The mean of the convergence accuracy achieved by the PSOLFS is equal to the mean of the convergence accuracy by LFAPSO for all benchmark functions on 1000 dimensions after 20 runs.

Table 4. The *p*-value of the Wilcoxon test of the Six Algorithms

Number of Dimensions	PSOLFS vs PSO	PSOLFS vs PSO-SA	PSOLFS vs LFPSO	PSOLFS vs PSOLF	PSOLF vs LFAPSO
500	3.6749e-07	3.6749e-07	5.1893e-07	5.6836e-05	5.1893e-07
1000	2.9431e-06	3.6038e-06	1.7231e-04	0.0113	1.1728e-05

Table 4 shows the p-value of the Wilcoxon test for the five algorithms compared with PSOLFS. The experiment rejects al hypothesizes at the default significant level s=0.05, which is indicated by the p-values in Table 4. As seen in Table 4, the p-value is less than 0.05. The null hypothesis test is rejected, with a significant difference between PSOLFS and the other PSO variants. The results of hypothesis tests show that PSOLFS significantly outperforms the other variants of PSO.

5. Conclusion

This paper proposes a particle swarm optimization based on levy flight and fast simulated annealing, PSOLFS. PSOLFS attempts to obtain a global solution for a high-dimensional optimization problem. We use different sets of benchmark functions for 500 and 1000 dimensions. On 500 dimensions, the algorithm obtains the optimal value on 14 of the 16 functions. On 1000 dimensions, the algorithm obtains the optimal value on eight benchmark functions and is close to optimal on four functions. We also compare PSOLFS with the other five PSO variants in terms of convergence accuracy and speed. The results demonstrate that PSOLFS achieves higher accuracy and faster convergence speed than other PSO variants. Also, the results of the Wilcoxon test show a significant difference between PSOLFS and the other PSO variants. In future work, we are planning to investigate the efficiency of the proposed algorithm on different optimization problems, such as nurse scheduling and constrained engineering optimization problems

Acknowledgment

The authors thank the Ministry of Higher Education Malaysia for supporting this research through the Fundamental Research Grant Scheme.

Declarations

Author contribution. Samar Bashath and Amelia Ritahani Ismail conceived the presented idea, developed the theory, performed the computations and experiments, and wrote the manuscript. Ali A Alwan and Amir Aatieff Amir Hussin provided critical feedback and helped shape the research, analysis, and manuscript.

Funding statement. This research is financially supported by a grant from the Ministry of Higher Education Malaysia: Fundamental Research Grant Scheme (FRGS/1/2018/ICT02/UIAM/02/1).
Conflict of interest. The authors declare no conflict of interest.
Additional information. No additional information is available for this paper.

References

[1] A. F. J. Levi and S. Haas, Eds., "Global optimization algorithms," in *Optimal Device Design*, Cambridge: Cambridge University Press, 2008, pp. 262–276. doi: 10.1017/CBO9780511691881.010

[2] S. Mahdavi, M. E. Shiri, and S. Rahnamayan, "Metaheuristics in large-scale global continues optimization: A survey," *Inf. Sci. (Ny).*, vol. 295, pp. 407–428, Feb. 2015, doi: 10.1016/j.ins.2014.10.042.

[3] E. K. Nyarko, R. Cupec, and D. Filko, "A comparison of several heuristic algorithms for solving high dimensional optimization problems," *Int. J. Electr. Comput. Eng. Syst.*, vol. 5, no. 1., pp. 1–8, 2014. Available at: Google Scholar.

[4] N. Quirante and J. A. Caballero, "Large scale optimization of a sour water stripping plant using surrogate models," *Comput. Chem. Eng.*, vol. 92, pp. 143–162, Sep. 2016, doi: 10.1016/j.compchemeng.2016.04.039.

[5] H. Wei, X. Du, L. Yang, and Y. Yang, "Entransy dissipation based optimization of a large-scale dry cooling system," *Appl. Therm. Eng.*, vol. 125, pp. 254–265, Oct. 2017, doi: 10.1016/j.applthermaleng.2017.06.117.

[6] Y. Zhao, Y. Yuan, F. Nie, and Q. Wang, "Spectral clustering based on iterative optimization for large-scale and high-dimensional data," *Neurocomputing*, vol. 318, pp. 227–235, Nov. 2018, doi: 10.1016/j.neucom.2018.08.059.

[7] L. Sun, L. Lin, H. Li, and M. Gen, "Large scale flexible scheduling optimization by a distributed evolutionary algorithm," *Comput. Ind. Eng.*, vol. 128, pp. 894–904, Feb. 2019, doi: 10.1016/j.cie.2018.09.025.

[8] D. E. Mazzuco *et al.*, "A concept for simulation-based optimization in Vehicle Routing Problems," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1720–1725, 2018, doi: 10.1016/j.ifacol.2018.08.208.

[9] A. Singh and N. Dulal, "A Survey on Metaheuristics for Solving Large Scale Optimization Problems," *Int. J. Comput. Appl.*, vol. 170, no. 5, pp. 1–7, Jul. 2017, doi: 10.5120/ijca2017914839.

[10] X. Lai and Y. Zhou, "An adaptive parallel particle swarm optimization for numerical optimization problems," *Neural Comput. Appl.*, vol. 31, no. 10, pp. 6449–6467, Oct. 2019, doi: 10.1007/s00521-018-3454-9.

[11] B. Goertzel, "The Embodied Communication Prior: A characterization of general intelligence in the context of Embodied social interaction," in *2009 8th IEEE International Conference on Cognitive Informatics*, 2009, pp. 38–43, doi: 10.1109/COGINF.2009.5250687.

[12] W. Guo, C. Si, Y. Xue, Y. Mao, L. Wang, and Q. Wu, "A Grouping Particle Swarm Optimizer with Personal-Best-Position Guidance for Large Scale Optimization," *IEEE/ACM Trans. Comput. Biol. Bioinforma.*, vol. 15, no. 6, pp. 1904–1915, Nov. 2018, doi: 10.1109/TCBB.2017.2701367.

[13] S. Chen, J. Montgomery, and A. Bolufé-Röhler, "Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution," *Appl. Intell.*, vol. 42, no. 3, pp. 514–526, Apr. 2015, doi: 10.1007/s10489-014-0613-2.

[14] X.-S. Yang, "Swarm intelligence based algorithms: a critical analysis," *Evol. Intell.*, vol. 7, no. 1, pp. 17–28, Apr. 2014, doi: 10.1007/s12065-013-0102-2.

[15] J. Wen, H. Ma, and X. Zhang, "Optimization of the occlusion strategy in visual tracking," *Tsinghua Sci. Technol.*, vol. 21, no. 2, pp. 221–230, Apr. 2016, doi: 10.1109/TST.2016.7442504.

[16] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intell.*, vol. 1, no. 1, pp. 33–57, Oct. 2007, doi: 10.1007/s11721-007-0002-0.

[17] M. Juneja and S. K. Nagar, "Particle swarm optimization algorithm and its parameters: A review," in *2016 International Conference on Control, Computing, Communication and Materials (ICCCCM)*, 2016, pp. 1–5, doi: 10.1109/ICCCCM.2016.7918233.

[18] F. Zhu, D. Chen, and F. Zou, "A novel hybrid dynamic fireworks algorithm with particle swarm optimization," *Soft Comput.*, vol. 25, no. 3, pp. 2371–2398, Feb. 2021, doi: 10.1007/s00500-020-05308-6.

[19] D. Wang, D. Tan, and L. Liu, “Particle swarm optimization algorithm: an overview,” *Soft Comput.*, vol. 22, no. 2, pp. 387–408, Jan. 2018, doi: 10.1007/s00500-016-2474-6.

[20] M. N. Ab Wahab, S. Nefti-Meziani, and A. Atyabi, “A Comprehensive Review of Swarm Optimization Algorithms,” *PLoS One*, vol. 10, no. 5, p. e0122827, May 2015, doi: 10.1371/journal.pone.0122827.

[21] J. Ding, Q. Wang, Q. Zhang, Q. Ye, and Y. Ma, “A Hybrid Particle Swarm Optimization-Cuckoo Search Algorithm and Its Engineering Applications,” *Math. Probl. Eng.*, vol. 2019, pp. 1–12, Mar. 2019, doi: 10.1155/2019/5213759.

[22] Y. Ning, Z. Peng, Y. Dai, D. Bi, and J. Wang, “Enhanced particle swarm optimization with multi-swarm and multi-velocity for optimizing high-dimensional problems,” *Appl. Intell.*, vol. 49, no. 2, pp. 335–351, Feb. 2019, doi: 10.1007/s10489-018-1258-3.

[23] A. F. Ali and M. A. Tawhid, “A hybrid particle swarm optimization and genetic algorithm with population partitioning for large scale optimization problems,” *Ain Shams Eng. J.*, vol. 8, no. 2, pp. 191–206, Jun. 2017, doi: 10.1016/j.asej.2016.07.008.

[24] F. Javidrad and M. Nazari, “A new hybrid particle swarm and simulated annealing stochastic optimization method,” *Appl. Soft Comput.*, vol. 60, pp. 634–654, Nov. 2017, doi: 10.1016/j.asoc.2017.07.023.

[25] S. Bashath and A. R. Ismail, “Improved Particle Swarm Optimization By Fast Simulated Annealing Algorithm,” in *2019 International Conference of Artificial Intelligence and Information Technology (ICAIIT)*, 2019, pp. 297–301, doi: 10.1109/ICAIIT.2019.8834515.

[26] R. Jensi and G. W. Jiji, “An enhanced particle swarm optimization with levy flight for global optimization,” *Appl. Soft Comput.*, vol. 43, pp. 248–261, Jun. 2016, doi: 10.1016/j.asoc.2016.02.018.

[27] D. Sedighizadeh, E. Masehian, M. Sedighizadeh, and H. Akbaripour, “GEPSO: A new generalized particle swarm optimization algorithm,” *Math. Comput. Simul.*, vol. 179, pp. 194–212, Jan. 2021, doi: 10.1016/j.matcom.2020.08.013.

[28] Z. Li, W. Wang, Y. Yan, and Z. Li, “PS–ABC: A hybrid algorithm based on particle swarm and artificial bee colony for high-dimensional optimization problems,” *Expert Syst. Appl.*, vol. 42, no. 22, pp. 8881–8895, Dec. 2015, doi: 10.1016/j.eswa.2015.07.043.

[29] H. Zhang, J. Sun, T. Liu, K. Zhang, and Q. Zhang, “Balancing exploration and exploitation in multiobjective evolutionary optimization,” *Inf. Sci. (Ny).*, vol. 497, pp. 129–148, Sep. 2019, doi: 10.1016/j.ins.2019.05.046.

[30] M. R. Bonyadi and Z. Michalewicz, “Particle Swarm Optimization for Single Objective Continuous Space Problems: A Review,” *Evol. Comput.*, vol. 25, no. 1, pp. 1–54, Mar. 2017, doi: 10.1162/EVCO_r_00180.

[31] S. S. Dash, S. K. Nayak, and D. Mishra, “ABC Versus PSO: A Comparative Study and Analysis on Optimization Aptitude,” 2021, pp. 527–544. doi: 10.1007/978-981-16-0695-3_50

[32] H.-K. Jung, J. T. Kim, T. Sahama, and C.-H. Yang, Eds., *Future Information Communication Technology and Applications*, vol. 235. Dordrecht: Springer Netherlands, 2013. doi: 10.1007/978-94-007-6516-0

[33] A. R. Ismail, N. A. Aziz, A. M. Ralib, N. Z. Abidin, and S. S. Bashath, “A particle swarm optimization levy flight algorithm for imputation of missing creatinine dataset,” *Int. J. Adv. Intell. Informatics*, vol. 7, no. 2, p. 225, Jul. 2021, doi: 10.26555/ijain.v7i2.677.

[34] X.-S. Yang and Suash Deb, “Cuckoo Search via L&#x00E9;vy flights,” in *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, 2009, pp. 210–214, doi: 10.1109/NABIC.2009.5393690.

[35] M. Abdel-Basset, A.-N. Hessin, and L. Abdel-Fatah, “A comprehensive study of cuckoo-inspired algorithms,” *Neural Comput. Appl.*, vol. 29, no. 2, pp. 345–361, Jan. 2018, doi: 10.1007/s00521-016-2464-8.

[36] M. Leccardi, “Comparison of three algorithms for Levy noise generation,” in *Proceedings of fifth EUROMECH nonlinear dynamics conference*, 2005, pp. 1–14. Available at: Google Scholar.

[37] X.-S. Yang and S. Deb, “Cuckoo search: recent advances and applications,” *Neural Comput. Appl.*, vol. 24, no. 1, pp. 169–174, Jan. 2014, doi: 10.1007/s00521-013-1367-1.

[38] S. Chakraborty *et al.*, “Modified cuckoo search algorithm in microscopic image segmentation of hippocampus,” *Microsc. Res. Tech.*, vol. 80, no. 10, pp. 1051–1072, Oct. 2017, doi: 10.1002/jemt.22900.

[39] H. Soneji and R. C. Sanghvi, “Towards the improvement of Cuckoo search algorithm,” in *2012 World Congress on Information and Communication Technologies*, 2012, pp. 878–883, doi: 10.1109/WICT.2012.6409199.

[40] L. Ingber, “Very fast simulated re-annealing,” *Math. Comput. Model.*, vol. 12, no. 8, pp. 967–973, 1989, doi: 10.1016/0895-7177(89)90202-1.

[41] A. M. Zain, H. Haron, and S. Sharif, “Simulated annealing to estimate the optimal cutting conditions for minimizing surface roughness in end milling Ti-6Al-4V,” *Mach. Sci. Technol.*, vol. 14, no. 1, pp. 43–62, Feb. 2010, doi: 10.1080/10910340903586558.

[42] S. N. Kadry and A. El Hami, “A New Hybrid Simulated Annealing Algorithm for Large Scale Global Optimization,” *Int. J. Manuf. Mater. Mech. Eng.*, vol. 5, no. 3, pp. 24–36, Jul. 2015, doi: 10.4018/IJMMME.2015070102.

[43] A. Peng, “Particle Swarm Optimization Algorithm Based on Chaotic Theory and Adaptive Inertia Weight,” *J. Nanoelectron. Optoelectron.*, vol. 12, no. 4, pp. 404–408, Apr. 2017, doi: 10.1166/jno.2017.2033.

[44] S. N. Chegini, A. Bagheri, and F. Najafi, “PSOSCALF: A new hybrid PSO based on Sine Cosine Algorithm and Levy flight for solving optimization problems,” *Appl. Soft Comput.*, vol. 73, pp. 697–726, Dec. 2018, doi: 10.1016/j.asoc.2018.09.019.

[45] X. Shen, Z. Chi, J. Yang, C. Chen, and Z. Chi, “Particle Swarm Optimization with Dynamic Adaptive Inertia Weight,” in *2010 International Conference on Challenges in Environmental Science and Computer Engineering*, 2010, pp. 287–290, doi: 10.1109/CESCE.2010.16.

[46] R. K. Huda and H. Banka, “New efficient initialization and updating mechanisms in PSO for feature selection and classification,” *Neural Comput. Appl.*, vol. 32, no. 8, pp. 3283–3294, Apr. 2020, doi: 10.1007/s00521-019-04395-3.

[47] H. Wang, W. Wang, and Z. Wu, “Particle swarm optimization with adaptive mutation for multimodal optimization,” *Appl. Math. Comput.*, vol. 221, pp. 296–305, Sep. 2013, doi: 10.1016/j.amc.2013.06.074.

[48] T. Nishio, J. Kushida, A. Hara, and T. Takahama, “Adaptive particle swarm optimization with multi-dimensional mutation,” in *2015 IEEE 8th International Workshop on Computational Intelligence and Applications (IWCIA)*, 2015, pp. 131–136, doi: 10.1109/IWCIA.2015.7449476.

[49] E. H. Houssein, M. R. Saad, F. A. Hashim, H. Shaban, and M. Hassaballah, “Lévy flight distribution: A new metaheuristic algorithm for solving engineering optimization problems,” *Eng. Appl. Artif. Intell.*, vol. 94, p. 103731, Sep. 2020, doi: 10.1016/j.engappai.2020.103731.

[50] H. Haklı and H. Uğuz, “A novel particle swarm optimization algorithm with Levy flight,” *Appl. Soft Comput.*, vol. 23, pp. 333–345, Oct. 2014, doi: 10.1016/j.asoc.2014.06.034.

[51] L. Yi, “Study on an Improved PSO Algorithm and its Application for Solving Function Problem,” *Int. J. Smart Home*, vol. 10, no. 3, pp. 51–62, Mar. 2016, doi: 10.14257/ijsh.2016.10.3.06.

[52] “Autonomous Groups Particle Swarm Optimization Algorithm Based On Exponential Decreasing Inertia Weight,” *Rev. Tec. La Fac. Ing. Univ. Del Zulia*, vol. 39, no. 7, Nov. 2016, doi: 10.21311/001.39.7.36.

[53] W. Wang, J.-M. Wu, and J.-H. Liu, “A Particle Swarm Optimization Based on Chaotic Neighborhood Search to Avoid Premature Convergence,” in *2009 Third International Conference on Genetic and Evolutionary Computing*, 2009, pp. 633–636, doi: 10.1109/WGEC.2009.168.

[54] M. Chen, T. Wang, J. Feng, Y.-Y. Tang, and L.-X. Zhao, “A Hybrid Particle Swarm Optimization Improved by Mutative Scale Chaos Algorithm,” in *2012 Fourth International Conference on Computational and Information Sciences*, 2012, pp. 321–324, doi: 10.1109/ICCIS.2012.19.

[55] Z. Du, S. Li, Y. Sun, and N. Li, “Adaptive particle swarm optimization algorithm based on levy flights mechanism,” in *2017 Chinese Automation Congress (CAC)*, 2017, pp. 479–484, doi: 10.1109/CAC.2017.8242815.

[56] Z. Lin and Q. Zhang, “An effective hybrid particle swarm optimization with Gaussian mutation,” *J. Algorithm. Comput. Technol.*, vol. 11, no. 3, pp. 271–280, Sep. 2017, doi: 10.1177/1748301817710923.

[57] H. Shi *et al.*, “Oscillatory Particle Swarm Optimizer,” *Appl. Soft Comput.*, vol. 73, pp. 316–327, Dec. 2018, doi: 10.1016/j.asoc.2018.08.037.

[58] D. Li, W. Guo, A. Lerch, Y. Li, L. Wang, and Q. Wu, “An adaptive particle swarm optimizer with decoupled exploration and exploitation for large scale optimization,” *Swarm Evol. Comput.*, vol. 60, p. 100789, Feb. 2021, doi: 10.1016/j.swevo.2020.100789.

[59] M. S. Kıran and M. Gündüz, “A recombination-based hybridization of particle swarm optimization and artificial bee colony algorithm for continuous optimization problems,” *Appl. Soft Comput.*, vol. 13, no. 4, pp. 2188–2203, Apr. 2013, doi: 10.1016/j.asoc.2012.12.007.

[60] L. Idoumghar, M. Melkemi, R. Schott, and M. I. Aouad, “Hybrid PSO-SA Type Algorithms for Multimodal Function Optimization and Reducing Energy Consumption in Embedded Systems,” *Appl. Comput. Intell. Soft Comput.*, vol. 2011, pp. 1–12, 2011, doi: 10.1155/2011/138078.

[61] M. Basu, P. Deb, and G. Garai, “Hybrid of particle swarm optimization and simulated annealing for multidimensional function optimization,” *Int. J. Inf. Technol.*, vol. 20, no. 1, pp. 112–120, 2014. Available at: Google Scholar.

[62] Qian Wang, Pei-hong Wang, and Zhi-gang Su, “A hybrid search strategy based particle swarm optimization algorithm,” in *2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA)*, 2013, pp. 301–306, doi: 10.1109/ICIEA.2013.6566384.

[63] W. Phuchan, B. Kruatrachue, and K. Siriboon, “Hybrid multi-swarm with Harmony Search algorithm,” in *2017 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, 2017, pp. 541–544, doi: 10.1109/ECTICon.2017.8096294.

[64] F. Wang, H. Zhang, K. Li, Z. Lin, J. Yang, and X.-L. Shen, “A hybrid particle swarm optimization algorithm using adaptive learning strategy,” *Inf. Sci. (Ny).*, vol. 436–437, pp. 162–177, Apr. 2018, doi: 10.1016/j.ins.2018.01.027.

[65] Q. Cui *et al.*, “Globally-optimal prediction-based adaptive mutation particle swarm optimization,” *Inf. Sci. (Ny).*, vol. 418–419, pp. 186–217, Dec. 2017, doi: 10.1016/j.ins.2017.07.038.

[66] M. Gholamian and M. R. Meybodi, “Enhanced comprehensive learning cooperative particle swarm optimization with fuzzy inertia weight (ECLCFPSO-IW),” in *2015 AI & Robotics (IRANOPEN)*, 2015, pp. 1–7, doi: 10.1109/RIOS.2015.7270730.