Iranian Journal of Fuzzy Systems

Volume 23, Number 4, (2026), pp. 145-170

Original paper

University of Sistan and Baluchestan

A competitive framework for a structured particle swarm optimization using fuzzy logic

S. Vatsa [1], N. Singhal [2] and A. Tripathi [3]

[1,2,3] *Department of Mathematics, Jaypee Institute of Information Technology, Noida, 201309, Uttar Pradesh, India*

2404080001@mail.jiit.ac.in, neha.singhal@mail.jiit.ac.in, alka.tripathi@mail.jiit.ac.inm

Abstract

Particle Swarm Optimization (PSO) is one of the most popular metaheuristic algorithms used to solve complex real-world optimization problems to date. The major drawback of PSO is its weak exploration ability, which leads the algorithm prematurely to a local optimum. To address this issue, we propose an improved particle swarm optimization with an intelligent multi-swarm strategy (PSO-IMS), inspired by powerful PSO variants FHPSO, HPSO-ALS, and ALPSO. To ensure uniform dispersion of particles during initialization, a low-discrepancy Sobol sequence has been used in this article. PSO-IMS uses the worst-performing particle in the swarm, called g_{worst}, to enhance the algorithm's exploration ability in the initial stages of the search. The method integrates a fuzzy-logic-based parameter adaptation mechanism to manage uncertainty and dynamically adjust key control parameters, thereby reflecting a more realistic hierarchical decision process. To validate the performance of PSO-IMS, it has been tested on seventeen benchmark functions along with the CEC 2013 test suite, and compared against four powerful PSO variants, in which our proposed algorithm showed satisfactory results. As an application of the proposed algorithm, it has been applied to the traveling salesman problem with thirty-one cities as well as an Unmanned Aerial Vehicle (UAV) path planning problem.

Keywords: Particle swarm optimization, fuzzy logic, hierarchy strategy, global worst particle.

1 Introduction

Optimization has always been a major research focus due to its applicability to a wide range of real-world problems [28]. Whether in management science, engineering, or industry, efficient optimization remains a significant challenge. Conventional techniques such as direct methods (linear programming, quadratic programming, etc.) and gradient-based methods (Newton–Raphson, gradient descent, etc.) often fail when dealing with high-dimensional or discontinuous problems. These approaches typically search through the entire problem space, which makes them computationally expensive. NP Hard problems [27] fall into the class of problems that cannot be solved using conventional methods of optimization. In such scenarios, metaheuristic algorithms [32] offer a practical and efficient alternative by providing near-optimal solutions in reasonable computational time.

Metaheuristic algorithms [6, 9, 30] use a stochastic approach to tackle high-dimensional, multi-constraint problems and provide near-optimal solutions. They are based on trial and error and aim to balance exploration and exploitation. They are generally classified into two major categories: single solution based

Corresponding Author: N. Singhal
Received: February 2026; Revised: June 2026; Accepted: July 2026.
https://doi.org/10.22111/ijfs.2026.54604.9673

metaheuristics such as local search [14], simulated annealing [1], etc and population based metaheuristics such as genetic algorithm [11], particle swarm optimization [35], grey wolf optimization [24] etc. Among the wide range of metaheuristic techniques, particle swarm optimization (PSO) is widely recognized for its simplicity, flexibility, and robustness.

Particle swarm optimization (PSO) is a nature-inspired metaheuristic algorithm introduced in 1995 by James Kennedy, a social psychologist, and Russell Eberhart, an electrical engineer [17]. PSO is inspired by the collective movement of birds in a flock. In this algorithm, each individual in the swarm is referred to as a particle. The movement of each particle in the search space is guided by its personal best position (p_{best}), and the global best position (g_{best}) discovered by the swarm. The standard PSO incorporates acceleration coefficients in its learning mechanism that determine the relative influence of p_{best} and g_{best}. Although the standard PSO is simple and easy to implement, it tends to get stuck in local optima when faced with complex optimization problems. This limitation has motivated researchers to propose numerous PSO variants aimed at improving its exploration capability.

In this paper, an advanced adaptive approach has been presented. This paper models swarm behaviour using an analogy to hierarchical human organizations. Diversity is introduced during initialization to improve exploration, and low-performing particles are also incorporated to prevent premature convergence. The early search phase takes advantage of the exploratory tendencies of the g_{worst} particle. Each hierarchical level comprises trainers and trainees. Trainers update their positions based on their personal best, global best, and higher-level guidance, while trainees learn from their personal best and through trainers. Fuzzy systems [44] are widely recognized for their applications in real world problems [5]. In our work, parameter tuning is performed using the Mamdani fuzzy inference system to represent hierarchical social interactions. Thus, the contributions in this paper can be summarized as follows.

- In this paper, the pseudo-random sequences, namely Sobol sequences, have been used to initialize the population. This serves the purpose of uniform distribution of the population in the initialization stage compared to random initialization. Moreover, fuzzy logic is an excellent tool for dynamically adjusting parameters according to the needs of the situation. Hence, we use fuzzy logic to tune the inertia weight as well as the acceleration coefficients.

- The algorithm utilizes the exploration capabilities of the global worst particle g_{worst}, along with the exploitation abilities of a hierarchical structure. A parameter called '*improvement*' is used to balance the trade-off between exploration and exploitation. This parameter *improvement* measures the change in the total sum of personal best fitness when the algorithm moves from one generation to another. When a negligible change in *improvement* is observed, it means that the maximum capabilities of the g_{worst} particle have been utilized to lead the algorithm to unexplored areas of the search space. Consequently, the algorithm switches to hierarchical division of the population.

- Most hierarchical structures are often rigid, leading to premature convergence of the algorithm. Thus, the hierarchy proposed in this paper is made flexible so as to avoid local optima. For this, a *candidate* particle is introduced at each level of the hierarchy. Subsequently, this *candidate* particle is utilized to train members securing lower ranks in each level of hierarchy. Moreover, the position update strategy of the trainees is dynamic to speed up convergence.

- In this work, the proposed algorithm is rigorously tested against several well-known algorithms on 17 standard benchmarks as well as on the CEC 2013 test suite using powerful statistical tests, such as Friedman, Wilcoxon, Kendall's W and Cliff's δ along with Bootstrap Confidence Interval tests. To further validate the performance of the proposed algorithm, it is assessed on two real-world problems, which are the Unmanned Aerial Vehicles (UAV) path planning problem and the traveling salesman problem.

The rest of the paper is organized as follows: Section 2 presents the literature survey on PSO, Section 3 identifies the research gaps observed during the review. Section 4 describes the proposed algorithm in detail, while Section 5 provides the experimental comparisons with existing PSO variants. Section 6 discusses two case studies that demonstrate the execution of the proposed algorithm, and finally Section 7 concludes the paper.

Table 1: 17 basic benchmark functions

Function	**Mathematical Representation**	**Search Range**	**Category**
Sphere(f_1)	$\sum_{i=1}^{n} x_i^2$	$[-100, 100]$	Uni-modal
Schwefel 2.22(f_2)	$\sum_{i=1}^{n} \|x_i\| + \prod_{i=1}^{n} \|x_i\|$	$[-10, 10]$	Uni-modal
Rosenbrock(f_3)	$\sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-10, 10]$	Uni-modal
Noise(f_4)	$\sum_{i=1}^{n} i x_i^4 + rand(0, 1)$	$[-1.28, 1.28]$	Uni-modal
Sum Square(f_5)	$\sum_{i=1}^{n} i x_i^2$	$[-10, 10]$	Uni-modal
Step(f_6)	$\sum_{i=1}^{n} ([x_i + 0.5])^2$	$[-100, 100]$	Uni-modal
Quartic(f_7)	$\sum_{i=1}^{n} i x_i^4$	$[-1.28, 1.28]$	Uni-modal
Schwefel 2.26(f_8)	$418.98n - \sum_{i=1}^{n} x_i \sin(\sqrt{\|x_i\|})$	$[-500, 500]$	Multi-modal
Rastrigin(f_9)	$\sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12, 5.12]$	Multi-modal
Ackley(f_{10})	$-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)) + 20 + e$	$[-32, 32]$	Multi-modal
Griewank(f_{11})	$\frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-600, 600]$	Multi-modal
Penalized 1(f_{12})	$\frac{\pi}{n}\{10sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2)[1 + 10sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$	$[-50, 50]$	Multi-modal
Penalized 2(f_{13})	$\frac{1}{10}\{sin^2(\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + sin(3\pi x_{i+1})] + (x_n - 1)^2[1 + sin^2(2\pi x_{i+1})]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	$[-50, 50]$	Multi-modal
Levy(f_{14})	$\sum_{i=1}^{n-1}(x_i - 1)^2[1 + sin^2(3\pi x_{i+1})] + sin^2(3\pi x_i) + \|x_n - 1\|[1 + sin^2(3\pi x_n)]$	$[-10, 10]$	Multi-modal
Schaffer(f_{15})	$0.5 + \frac{sin\sqrt{\sum_{i=1}^{n} x_i^2} - 0.5}{1 + 0.001(\sum_{i=1}^{n} x_i^2))^2}$	$[-100, 100]$	Multi-modal
Alpine(f_{16})	$\sum_{i=1}^{n} \|x_i.sin(x_i) + 0.1\, x_i\|$	$[-10, 10]$	Multi-modal
Non-continuous Rastrigin(f_{17})	$\sum_{i=1}^{n} [y_i^2 - 10\cos(2\pi y_i) + 10]$	$[-5.12, 5.12]$	Multi-modal

2 Related work

The collective movement of birds in a swarm led to the development of the first PSO model by Kennedy and Eberhart [17]. Then, Shi and Eberhart formulated an enhanced variant of PSO [35], which we refer to as standard PSO (SPSO).

Standard PSO: The most commonly known PSO variant, i.e. the standard PSO (SPSO), proposed by Shi and Eberhart [35] have two main learning components, the personal best known as p_{best} and the global best known as the g_{best}. c_1 and c_2 are the acceleration coefficients that control the effect of p_{best} and g_{best} on particle's learning. The influence of its previously accumulated experience is controlled by the parameter inertia weight, w. The velocity and position update equations in SPSO are as follows:

$$v_i^d(t+1) = w(t) * v_i^d(t) + c_1 * r_1(p_{best_i}^d(t) - x_i^d(t)) + c_2 * r_2(g_{best}^d(t) - x_i^d(t)), \quad (1)$$

$$x_i^d(t+1) = v_i^d(t+1) + x_i^d(t). \quad (2)$$

In SPSO, c_1 and c_2 are fixed to 2. r_1 and r_2 are uniformly distributed random numbers.

In this study, PSO variants are classified into four categories: initialization techniques, parameter tuning, topology of the search space and hybrid variants. We discuss each category in detail as follows.

2.1 Initialization techniques

In early models, the initialization of the swarm in the search space was performed randomly. Later, some authors showed that using low-discrepancy sequences such as the Halton sequence and Sobol sequence for particle initialization in PSO yielded better results [39]. The prospect of a uniform dispersion of particles in the search space motivated the authors of [38] to utilize Sobol sequences for population initialization. To further strengthen PSO, the inertia weight update strategy as well as acceleration coefficients were improvised, and the particles were updated using the mutation strategy. Opposition-based learning [12] was another initialization technique in which every randomly generated particle was paired with its corresponding opposite particle. The performance of the randomly generated particle was compared with that of its opposite, and the better-performing particle was selected.

2.2 Parameter tuning

Parameter tuning in PSO has been studied widely by researchers. Shi and Eberhart studied the effect of inertia weight on algorithm's performance and introduced static inertia weight, random inertia weight, and linearly decreasing inertia weight [4, 35, 36]. Feng et al. [7] introduced chaotic inertia weight using logistic mapping. Chen et al. [3] used natural exponential functions as an inertia weight to increase the convergence rate of PSO. Ratnaweera et al. [33] introduced two novel PSO variants, HPSO-TVAC and MPSO-TVAC with time-varying acceleration coefficients. HPSO-TVAC had a hierarchical structure and MPSO-TVAC had particles with mutation properties. Melin et al. [23] used fuzzy logic to control the acceleration coefficients. Here, the Mamdani inference system was used to create three fuzzy systems and the effect of each system was studied on various benchmark functions.

2.3 Topology of the search space

How particles in PSO are connected in the search space has attracted the attention of many scholars. Kennedy [16] employed a ring topology in which, instead of the global best (g_{best}) particle, a locally best particle, referred to as l_{best}, was used in the velocity update. Kennedy and Mendes [18] evaluated the star, pyramid, ring, and Von Neumann topologies on five benchmark functions.

Liang et al. [21] presented a novel approach of selecting p_{best} to increase population diversity, called CLPSO. In CLPSO, a particle learns from the personal best experience of the whole swarm rather than just its own best. Inspired by CLPSO, Nasir et al. [25] proposed DNLPSO where each particle learns from the g_{best} as well as the p_{best} of its neighborhood. This neighborhood is dynamic in nature. In APSO [45], particles are distributed into four stages: exploration, exploitation, convergence, and jumping out. Fuzzy logic is used to modify parameters based solely on the stage in which a particle is operating.

2.4 Hybrid variants

Recently, an increasing trend of combining the strengths of two or more metaheuristics has been observed in the literature. These are referred to as hybrid algorithms. Early models proved effective when tested on standard benchmark functions; however, as the world moves towards a faster-paced environment, increasingly complex optimization problems have emerged. Consequently, hybrid algorithms are being developed to address such challenges.

The multi-swarm strategy is commonly used in hybrid algorithms. One such variant is PSO-DLS [43], in which the entire population is divided into multiple sub-swarms. Each sub-swarm is further subdivided into ordinary and communication particles. This division within each sub-swarm is dynamic.

A PSO variant featuring hierarchical division, called FHPSO, was presented by Wang et al. [41]. In this approach, the swarm is divided into four layers according to their fitness values. Each layer is further subdivided into two groups: high-energy particles (i.e., particles with fitness values lower than the average of the corresponding layer) and low-energy particles (with fitness values higher than the average fitness of the corresponding layer).

Another hybrid PSO, called PSO-CL [22] follows a crisscross learning strategy in which the population is divided into two sub-swarms: elite and ordinary sub-populations. Crossover-based learning strategies are devised for each sub-population.

Adaptive strategies are also prevalent among researchers for designing hybrid algorithms. Wang et al. proposed ALPSO [42] based on an adaptive learning strategy. The tolerance-based search direction adjustment strategy was proposed to avoid stagnation of the algorithm in a local optimum. Here, a self-learning-based *candidate* particle is generated and a competitive learning strategy is used between the *candidate* and the global best particle to select the new leader of the swarm.

Another notable variant is HPSO-ALS, proposed by Wang et al. [40]. In this variant, a chaotic opposition learning strategy is used for initialization and inertia weight is decreased using self-logical mapping. Inspired by the cask effect, HPSO-ALS takes into account the global worst particle(g_{worst}) into its velocity update formula.

Hybridization of different metaheuristics with in a single framework has also produced significant results. The authors in [10] combined PSO with DE, using hybrid penalty strategy. They aimed to provide an advanced algorithm for constrained optimization tasks, specifically the multi-UAV path planning problem.

Recently, a growing trend of merging metaheuristic with Artificial Intelligence (AI) based techniques [15, 19] have been observed. Reinforcement Learning (RL) is a paradigm of machine learning that combines Reinforcement Learning with metaheuristic algorithms has become a popular research area. For instance, authors in [20] integrated PSO with Q-Learning to solve a multi-objective problem in cargo and research operations (CRP). Another notable work where Q-Learning combined with PSO is done by [13] where authors have solved a scheduling problem arising in carbon-green certificate trading. In [8], authors have optimized artificial neural networks (ANN) using PSO and GA. They have used the model in optimizing production of biogas as well as its prediction.

3 Research gap

PSO has the drawback of converging prematurely to a local optimum. From the literature, it is clear that many researchers have devised strategies to balance exploration and exploitation in PSO. To strengthen the exploration and exploitation abilities of PSO and hence, contribute to the existing literature, we propose an algorithm that has the strengths of powerful PSO variants, HPSO-ALS [40], FHPSO [41], and ALPSO [42]. In the initial stage, the proposed algorithm focuses on exploring the search space thoroughly using the g_{worst} particle. Subsequently, the algorithm switches to the exploitation stage. When no improvement is observed in terms of fitness, a hierarchical division of the population is performed.

This stratification is inspired by the hierarchical structure of the corporate sector. Accordingly, each layer comprises two categories of particles: trainers and trainees. Particles whose fitness values are less than the average fitness value of the layer are designated as trainers, whereas the remaining particles are classified as trainees. To provide trainers with broader exposure to the workplace (search space), their learning process is guided by the global best particle, their personal best positions, and the trainers in the immediate upper layer. In contrast, trainees represent relatively weaker particles that require performance enhancement; therefore, their learning is influenced by their personal best experiences as well as the trainers within the same layer.

Furthermore, fuzzy parameter tuning is employed to adaptively regulate the algorithmic parameters, thereby mimicking the dynamics of human social structures.

4 Proposed algorithm

In this section, we propose an improved PSO variant called PSO-IMS, which is mainly inspired from powerful PSO variants, FHPSO [41], HPSO-ALS [40], and ALPSO [42]. We discuss the intricacies of the proposed algorithm as follows:

4.1 Swarm initialization

Random initialization results in low diversity, leading to premature convergence of the swarm to a local optimum. To improve exploration ability of the algorithm, low-discrepancy quasi-random sequences have proven to be fruitful. Uy et al. [39] investigated the effects of initialization using three popular low-discrepancy quasi-random sequences: Halton sequence, Sobol sequence and Faure sequence on six benchmark

functions. Among them, the Sobol sequence obtained the best results overall, with the Halton sequence securing the second best position.

We evaluated the initialization of the proposed algorithm using the Sobol sequence against the Halton sequence on seventeen benchmark functions (Table 1) using the Wilcoxon Rank Sum test. The test shows a significant difference with Sobol sequence performing better on four benchmark functions which are Rosenbrock, Schaffer, Penalised 1, and Penalised 2. For the remaining thirteen benchmark functions, there is no significant difference between the two. Therefore, we propose initialization using the Sobol sequence. It is to be noted that among the unimodal functions used in this study, Rosenbrock function, having a steep banana shaped valley, is the most difficult to optimize. Moreover, optimization algorithms show tendency of getting stuck to a local optimum in Schaffer function. Penalised 1 and 2 are multi-modal benchmarks, increasing problem complexity with penalties.

4.2 Learning strategy

The proposed algorithm, particle swarm optimization with an intelligent multi-swarm strategy (PSO-IMS) resembles the human corporate sector. Social stratification in any organization is an essential element in improving productivity. It creates a hierarchy based solely on the performance of individuals. Those showing similar capabilities are grouped. They not only learn from each other, but also from their superiors. This promotes inter-level information sharing. On the other hand, those who need improvement are placed under the guidance of their superiors.

Sometimes, even the worst-performing individual gives some valuable ideas. In that case, a strict hierarchical division might not be able to incorporate the valuable information coming from the worst-performing individual because of its placement at the bottom of the hierarchy. We propose an algorithm which follows a hierarchical division based on individual's performance, but it is also flexible enough to incorporate the global worst particle, g_{worst} to improve exploration ability of PSO. Thus, the whole algorithm is divided into two phases.

In the exploratory stage, the whole population learns from their personal best experience p_{best}, the global best particle, g_{best} and the global worst particle, g_{worst} [40]. This learning mechanism is depicted in Equations 3 & 4 as follows :

$$\begin{aligned} v_i^d(t+1) = w(t) * v_i^d(t) + c_1 * r_1(p^d_{best_i}(t) - x_i^d(t)) + c_2 * r_2(g^d_{best}(t) - x_i^d(t)) \\ + c_3 * r_3(g^d_{worst}(t) - x_i^d(t)), \end{aligned} \tag{3}$$

$$x_i^d(t+1) = v_i^d(t+1) + x_i^d(t). \tag{4}$$

Here, v_i^d is the velocity of the i^{th} particle in dimension d, and x_i^d is the particle's position. w is inertia weight, c_1, c_2 and c_3 are the acceleration coefficients. p_{best} is the particle's best position; g_{best} and g_{worst} denote the global best and worst particles, respectively.

When the algorithm stagnates and no improvement can be seen, it switches to hierarchical division of the population to start exploitation stage. How the algorithm measures improvement is an important question. To analyze whether the algorithm has achieved considerable growth between generations, measuring difference in global best fitness is less comprehensive approach as it measures progress on a global scale only which can lead to premature convergence in the end. True growth can be measured when we analyze difference in fitness per particle. In PSO-IMS, the difference between the sum of current p_{best} fitness to that of the previous p_{best} fitness of the whole population at a point in time is how it measures improvement [42]. Its mathematical depiction is shown as follows:

$$improvement(t) = \sum_{i=1}^{n} f(p_{best_i})^{t-1} - \sum_{i=1}^{n} f(p_{best_i})^{t}. \tag{5}$$

Here n is the population size and $f(p_{best_i})^t$ denotes fitness at p_{best} position of i^{th} particle at iteration t. When $|improvement| < 10^{-4}$ (Equation 5), the algorithm switches to hierarchical division of the population [41] consisting of a total of four layers.

The whole population is sorted by fitness. The top 10% particles of the population with the least fitness values form the top layer of hierarchy, followed by 20% particles in the second layer, 30% in the third and

the remaining 40% particles with maximum fitness in the bottom layer. Moreover, each layer is further segregated into two kinds of particles: trainers and trainees. The trainers of each layer are the particles having fitness values less than the average fitness value of that layer. The remaining particles of that layer are the trainees. The trainers and the trainees have different learning mechanism. The trainers learn from their personal best experience, the global best particle as well as the particles from their immediate upper layer.

For this, we define E_M which is the average of all positions of the M^{th} layer, that is:

$$E_M = \frac{1}{p}\sum_{i=1}^{p} x_{M_i}, \tag{6}$$

where, p is the size of M^{th} layer and x_{M_i} denotes the position of i^{th} particle in M^{th} layer. The learning mechanism of trainers of M^{th} layer is shown in Equations 7 & 8:

$$\begin{aligned} v_i^d(t+1) = w(t) * v_i^d(t) + c_1 * r_1(p_{best_i}^d(t) - x_i^d(t)) + c_2 * r_2(g_{best}^d(t) - x_i^d(t)) \\ + alpha * r_3(E_{M-1} - x_i^d(t)), \end{aligned} \tag{7}$$

$$x_i^d(t+1) = v_i^d(t+1) + x_i^d(t). \tag{8}$$

The trainees are expected to learn from their personal best experience and the trainers from the same layer. To provide proper guidance to the trainees from all the trainers of that layer, a new *candidate* particle is constructed. This *candidate* is constructed using the personal best experience of the trainers of that layer. Here, the d^{th} dimension of the *candidate* learns from the d^{th} dimension of a randomly selected trainer of the respective layer. All layers have their own *candidate* particle constructed in this manner. The pseudo code for the construction of the *candidate* is shown in Algorithm 1. Equation 9 shows the velocity update formula for trainees of the M^{th} layer.

$$v_i^d(t+1) = w(t) * v_i^d(t) + c_1 * r_1(p_{best_i}^d(t) - x_i^d(t)) + c_2 * r_2(candidate^d(t) - x_i^d(t)). \tag{9}$$

To speed up convergence, the position update strategy of trainees is made dynamic. For this, we utilize a parameter λ [40] which is defined as follows:

$$\lambda = \frac{\exp f(x_i)}{\exp(\frac{1}{p}\sum_{i=1}^{p} f(x_i))}. \tag{10}$$

When λ is large enough for a particle i, it is closer to the average fitness of the layer it belongs to. This indicates that this particle i has a better chance of converging faster to the optimum as compared to the other particles with lesser values of λ. Thus, it should focus on local exploitation. On the other hand, particles with lesser values of λ should focus on exploration. This adaptive strategy is shown in Equation 11.

$$x_i^d(t+1) = \begin{cases} w * x_i^d(t) + (1-w) * v_i^d(t+1), & \lambda > rand \\ x_i^d(t) + v_i^d(t+1), & otherwise \end{cases} \tag{11}$$

Algorithm 1 Construction of *candidate* particle for m^{th} layer

```
Calculate number of trainers, q in layer m
for i = 1 to D do
    k = randi([1, q])
    candidate(i) = p_best(k, i)
end for
```

Algorithm 2 Learning with g_{worst} particle

```
for each particle i do
    Evaluate velocity and position using Equation 3 & 4
    Update fitness, p_best, g_best, g_worst, p_best fitness
end for
```

Algorithm 3 Hierarchical division when tolerance is reached

```
Hierarchical division of population in m layers
for Each layer m do
    Calculate average fitness of layer m
    Divide into trainers and trainees
    Construct candidate using Algorithm 1
    for trainer i do
        Update velocity and position using Equation 7 & 8
        Update fitness, p_best fitness, p_best, g_best
    end for
    for trainee i do
        Update velocity using Equation 9
        Calculate λ using Equation 10
        Update position using Equation 11
        Update fitness, p_best fitness, p_best, g_best
    end for
end for
```

Algorithm 4 Proposed Algorithm Framework

```
Initialize positions using Sobol sequence, initialize velocity
Evaluate fitness, p_best fitness
Set p_best, g_best, g_worst
Initialize iteration, t
Calculate improvement
while stopping criterion not met do
    if |improvement| >tolerance then
        Implement Algorithm 2
    else
        Implement algorithm 3
    end if
end while
```

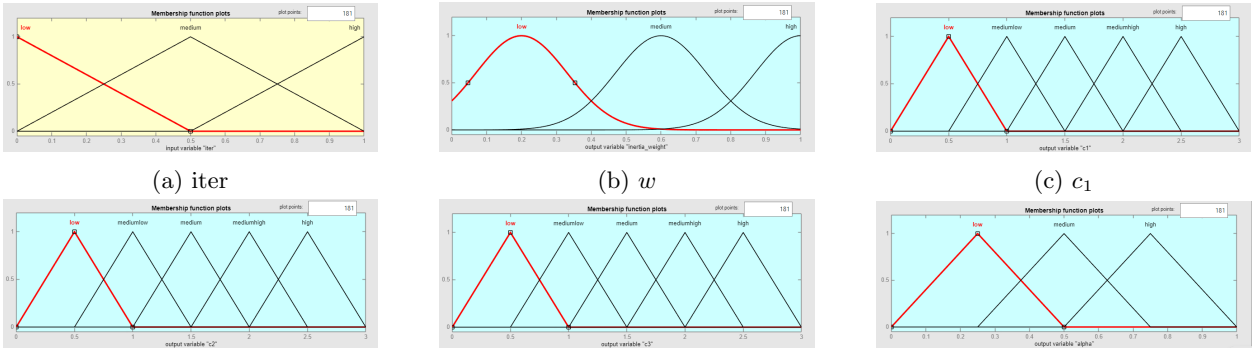
4.3 Parameter tuning

Parameter tuning is an important task in any optimization process. The proposed algorithm, PSO-IMS is inspired by the human corporate sector which is a real-world phenomenon. Hence, linearly varying the parameters will not do justice to it. Thus, to resemble a real-world scenario, we design a fuzzy system to tune our parameters, based on the Mamdani fuzzy inference system [23, 41].

The inertia weight w, and acceleration coefficients c_1, c_2, c_3 and *alpha* are the required outputs of the fuzzy system. Designing this system using fuzzy logic depends on whether the algorithm is in the exploration stage or the exploitation stage. In the initial stage, the algorithm focuses on exploration of the search space, thus inertia weight and acceleration coefficients are kept high. As the number of iterations increases, the algorithm enters the exploitation stage in which inertia weight and the acceleration coefficients are kept low.

A great deal of investigation is needed to decide upon an input metric of the required fuzzy system. Wang et al. [41] studied three fuzzy systems with varying fuzzy metrics. These metrics were iteration number, diversity and error. Here, diversity is the measure of dispersion of particles from the global best position in the search space and error is the deviation of fitness values from the optimal value found so far. It was found that fuzzy system with iteration number as input showed best results on several benchmarks. Hence, in our work, we choose the iteration number as a fuzzy input metric. The normalized iteration number, referred as *iter* is chosen as the input parameter as shown in Equation 12:

$$iter = \frac{current\ iteration}{maximum\ iterations}. \tag{12}$$



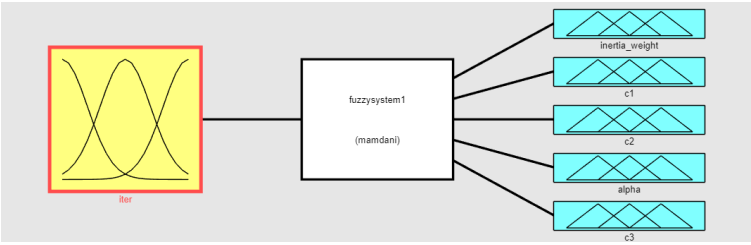
(a) iter (b) w (c) c_1

(d) c_2 (e) c_3 (f) *alpha*

Figure 1: Membership functions

The input parameter *iter* lies in the range $[0, 1]$ and has a triangular function. The output parameter inertia weight lies in the range $[0, 1]$ having a Gaussian membership function and the acceleration coefficients c_1, c_2, c_3 and *alpha* are in the range $[0, 3]$ having triangular function. During the exploration phase, outputs w, c_1, c_2 and c_3 are utilized (Equation 3) and *alpha* remains inactive. Similarly, the outputs w, c_1, c_2 and *alpha* are utilized during the exploitation phase (Equation 7 & 9) and c_3 remains inactive.

The membership functions for input and output parameters are shown in Fig. 1 and the schematic of the fuzzy system is shown in Fig. 2 . The flow chart demonstrating the working of our algorithm is shown in Fig. 3. Moreover, the pseudo-code for the same is provided in Algorithms 1 – 4.



1. If (iter is low) then (inertia_weight is high)(c1 is high)(c2 is high)(alpha is high)(c3 is high) (1)
2. If (iter is low) then (inertia_weight is high)(c1 is high)(c2 is high)(alpha is high)(c3 is high) (1)
3. If (iter is low) then (inertia_weight is high)(c1 is mediumhigh)(c2 is mediumhigh)(alpha is high)(c3 is mediumhigh) (1)
4. If (iter is medium) then (inertia_weight is medium)(c1 is mediumhigh)(c2 is mediumhigh)(alpha is medium)(c3 is mediumhigh) (1)
5. If (iter is medium) then (inertia_weight is medium)(c1 is medium)(c2 is medium)(alpha is medium)(c3 is medium) (1)
6. If (iter is medium) then (inertia_weight is medium)(c1 is mediumlow)(c2 is mediumlow)(alpha is medium)(c3 is mediumlow) (1)
7. If (iter is high) then (inertia_weight is medium)(c1 is mediumlow)(c2 is low)(alpha is low)(c3 is low) (1)
8. If (iter is high) then (inertia_weight is low)(c1 is low)(c2 is mediumlow)(alpha is low)(c3 is low) (1)
9. If (iter is high) then (inertia_weight is low)(c1 is low)(c2 is medium)(alpha is low)(c3 is low) (1)

Figure 2: Schematic of the fuzzy system

4.4 Computational complexity

For parameter tuning, the fuzzy system is initialized at the beginning; hence. Its time complexity is $O(1)$. The trade-off between exploration and exploitation is balanced by the parameter, *improvement* having time complexity $O(1)$. The time complexity of exploration using the g_{worst} particle is $O(N.D)$. Here N denotes population size and D is problem dimension.

Exploitation is performed using a hierarchy. Firstly, sorting the population for forming different learning structures (layers) has time complexity $O(NlogN)$. The energy gathering has time complexity $O(N)$, while the *candidate* particle has time complexity $O(D)$. Moreover, the parameter λ has time complexity $O(1)$.

Thus, exploitation stage has time complexity $O(NlogN + D + N + ND)$. Thus, we conclude that the time complexity of the proposed algorithm is $O(NlogN + N + D + 2ND)$, which is approximately $O(NlogN + N + ND)$.

Moreover, additional experiments have been conducted to calculate the mean runtime of the proposed algorithm across several benchmarks with varying dimension and population size over a span of 20 runs per benchmark. For this, runs have been performed on an Acer Aspire Go 14 laptop with an Intel Core Ultra 5 125H processor, 16 GB DDR5 RAM, and integrated Intel Arc Graphics in MATLAB R2025a. It is found that runtime scales approximately linearly for the proposed algorithm.

Table 2: Comparative results on 17 benchmarks after 20 runs ($f_1 - f_6$)

Algorithm	Test Stats	f_1	f_2	f_3	f_4	f_5	f_6
	Mean	3.9721E-246	1.8040E-115	7.6250E+00	4.0070E-01	1.0450E-201	0.0000E+00
	Std	0.0000E+00	7.2970E-115	7.8290E+00	3.2218E-01	0.0000E+00	0.0000E+00
Proposed	Best	3.8500E-248	3.8700E-124	3.4100E-04	2.5300E-05	4.9900E-247	0.0000E+00
	Worst	1.8600E-245	3.2700E-114	1.6100E+01	9.9500E-01	2.0900E-200	0.0000E+00
	Median	2.6455E-246	1.2040E-122	7.1414E+00	2.9340E-01	1.0406E-231	0.0000E+00
	Mean	8.7500E-74	1.3921E-17	2.1725E+01	5.0784E-01	5.5050E-32	0.0000E+00
	Std	3.9131E-73	6.1923E-17	1.5037E+01	3.1093E-01	2.2356E-31	0.0000E+00
FHPSO	Best	4.8300E-199	3.6300E-100	1.7800E+01	5.7200E-02	4.2100E-190	0.0000E+00
	Worst	1.7500E-72	2.7700E-16	8.5600E+01	9.4900E-01	1.0000E-30	0.0000E+00
	Median	7.1665E-147	2.7077E-71	1.8450E+01	5.8220E-01	1.8978E-133	0.0000E+00
	Mean	3.9425E-03	6.1467E-02	1.8862E+01	2.8480E-02	1.3476E-03	0.0000E+00
	Std	4.8220E-03	4.7413E-02	8.1238E-02	1.8131E-02	1.1659E-03	0.0000E+00
HPSO-ALS	Best	1.0642E-04	1.9866E-03	1.8701E+01	2.6616E-03	1.2324E-04	0.0000E+00
	Worst	1.8150E-02	1.8151E-01	1.8990E+01	6.8607E-02	4.5657E-03	0.0000E+00
	Median	1.7816E-03	5.5305E-02	1.8837E+01	2.3098E-02	1.2752E-03	0.0000E+00
	Mean	0.0000E+00	0.0000E+00	1.1788E+01	2.9207E-03	0.0000E+00	0.0000E+00
	Std	0.0000E+00	0.0000E+00	8.8354E+00	2.3365E-03	0.0000E+00	0.0000E+00
Sobol PSO	Best	0.0000E+00	0.0000E+00	2.9772E-03	2.7803E-04	0.0000E+00	0.0000E+00
	Worst	0.0000E+00	0.0000E+00	1.8713E+01	7.7331E-03	0.0000E+00	0.0000E+00
	Median	0.0000E+00	0.0000E+00	1.7779E+01	2.1393E-03	0.0000E+00	0.0000E+00
	Mean	5.0237E-01	6.2515E-01	2.8605E+01	4.4565E-03	7.3498E-02	9.1500E+00
	Std	5.1519E-01	2.8646E-01	2.2536E+01	2.0581E-03	9.0819E-02	3.8835E+00
SPSO	Best	1.2300E-02	2.4400E-01	1.3600E+01	8.6100E-04	7.7500E-03	2.0000E+00
	Worst	2.0700E+00	1.4000E+00	8.9700E+01	9.5800E-03	3.8500E-01	1.5000E+01
	Median	3.9793E-01	5.3113E-01	1.9282E+01	4.1333E-03	5.1529E-02	9.0000E+00

Table 3: Comparative results on 17 benchmark functions after 20 runs ($f_7 - f_{12}$)

Algorithm	Test Stats	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}
	Mean	0.0000E+00	3.3100E+03	0.0000E+00	4.4409E-16	0.0000E+00	5.8707E-05
	Std	0.0000E+00	2.5112E+03	0.0000E+00	0.0000E+00	0.0000E+00	5.8482E-05
Proposed	Best	0.0000E+00	1.5800E+03	0.0000E+00	4.4409E-16	0.0000E+00	1.6900E-06
	Worst	0.0000E+00	1.0100E+04	0.0000E+00	4.4409E-16	0.0000E+00	1.9400E-04
	Median	0.0000E+00	2.3688E+03	0.0000E+00	4.4409E-16	0.0000E+00	4.0037E-05
	Mean	5.6000E-02	3.4747E+03	5.8850E+00	3.1107E-15	0.0000E+00	2.0101E-03
	Std	2.5044E-01	3.3694E+03	8.7376E+00	1.9555E-15	0.0000E+00	1.3043E-03
FHPSO	Best	0.0000E+00	2.8300E-03	0.0000E+00	4.4400E-16	0.0000E+00	3.2500E-04
	Worst	1.1200E+00	8.3100E+03	3.2100E+01	7.5500E-15	0.0000E+00	5.6100E-03
	Median	0.0000E+00	2.3769E+03	0.0000E+00	3.9968E-15	0.0000E+00	1.6300E-03
	Mean	7.9867E-20	2.1000E+03	1.1412E-03	3.1516E-03	2.7287E-03	7.1967E-02
	Std	3.5344E-19	1.0961E+03	1.9009E-03	7.5939E-03	4.6489E-03	1.1101E-01
HPSO-ALS	Best	3.9407E-48	9.6905E+01	6.8083E-08	6.4127E-07	5.0049E-13	5.9313E-03
	Worst	1.5814E-18	4.2160E+03	5.7536E-03	2.9107E-02	1.5531E-02	5.0679E-01
	Median	1.0118E-26	2.3701E+03	1.1832E-04	4.8407E-05	5.2643E-04	4.1151E-02
	Mean	0.0000E+00	2.3928E+03	0.0000E+00	4.4409E-16	0.0000E+00	6.9136E-04
	Std	0.0000E+00	6.2753E+01	0.0000E+00	0.0000E+00	0.0000E+00	5.7220E-04
Sobol PSO	Best	0.0000E+00	2.3688E+03	0.0000E+00	4.4409E-16	0.0000E+00	5.4624E-05
	Worst	0.0000E+00	2.6087E+03	0.0000E+00	4.4409E-16	0.0000E+00	1.9493E-03
	Median	0.0000E+00	2.3688E+03	0.0000E+00	4.4409E-16	0.0000E+00	6.5484E-04
	Mean	7.2842E-08	2.4035E+03	2.4432E+01	2.1125E+00	4.3195E-01	5.2490E-01
	Std	2.0922E-07	5.6963E+02	8.1393E+00	5.6362E-01	1.5978E-01	5.1241E-01
SPSO	Best	8.1900E-10	1.3900E+03	9.7400E+00	1.1800E+00	1.1500E-01	1.0000E-02
	Worst	9.4500E-07	3.3400E+03	4.6800E+01	3.0600E+00	6.9300E-01	2.3100E+00
	Median	1.4320E-08	2.3725E+03	2.4553E+01	2.0547E+00	4.6130E-01	3.8429E-01

5 Experimental validation on benchmark functions

In this section, we compare the proposed algorithm against FHPSO [41], HPSO-ALS [40], Sobol PSO [39] and SPSO [35]. In Section 5.1, experiments are carried on 17 standard benchmark functions. Section 5.2 consists of an experimental analysis of our proposed algorithm against the aforementioned algorithms on the CEC 2013 test suite.

5.1 Comparison on 17 benchmark functions

In this study, a total of 17 benchmark functions ($f_1 - f_{17}$) have been used, of which 7 of them ($f_1 - f_7$) are uni-modal and the remaining ($f_8 - f_{17}$) are multi-modal benchmark functions. Table 1 provides a list of

benchmark functions. We have tested the proposed algorithm against some state-of-the-art PSO variants, HPSO-ALS [40], SPSO [35], Sobol PSO [39] and FHPSO [41] on these 17 benchmark functions, each with dimension size 20. For this, each algorithm is run 20 times independently on MATLAB R2025a. For testing PSO on single objective benchmarks, the maximum number of iterations often ranges from 500-1000 or more [34]. In order to deal with complex engineering problems, it is often suggested to keep the swarm size from 50-100 or more depending upon the complexity of the problem. Thus, we have fixed the maximum number of iterations to 1000 with a population size of 150 for each algorithm. Tables 2 – 4 show the median, mean, standard deviation, best, and worst fitness values on each benchmark separately. In terms of mean fitness value, the proposed algorithm gives superior results on multi-modal benchmark functions ($f_{10}, f_{12}, f_{13}, f_{14}$). Moreover, f_3 is the most complex uni-modal benchmark function, on which the proposed algorithm gives the best results. Sobol PSO shows best performance mostly on uni-modal benchmark functions (f_1, f_2, f_4, f_5) and one multi-modal function i.e f_{16}. HPSO-ALS outperforms rest of the algorithms on uni-modal function f_8, while FHPSO and SPSO outperforms others on none of the 17 benchmark functions. Moreover, on uni-modal f_6, no difference is observed among the algorithms. For uni-modal f_7 and multi-modal $f_9, f_{10}, f_{15}, f_{17}$ proposed algorithm and Sobol PSO have the same mean fitness value. On the multi-modal f_{11}, proposed algorithm, Sobol PSO and FHPSO yield identical results.

For fair comparison, the Friedman test has been used for statistical analysis. The median value is recorded for each algorithm on each benchmark. Friedman ranks for the proposed algorithm and Sobol PSO are 1.8824 and 1.8234 respectively. Hence, the difference between Sobol PSO and the proposed algorithm in terms of Friedman's rank is not large. Moreover, FHPSO ranks third and HPSO-ALS ranks fourth with average ranks 3.0294 and 3.5882 respectively. SPSO ranks fifth with an average rank of 4.6765.

To further investigate the performance of Sobol PSO against the proposed algorithm, we have performed the Wilcoxon signed-rank test. The test results show that Sobol PSO performs significantly better on 5 benchmarks out of which f_1, f_2, f_4, f_5 are uni-modal and f_{16} is multi-modal. The proposed algorithm performs significantly better on 4 benchmarks, out of which one is uni-modal i.e. f_3 and the rest are multi-modal which are f_{12}, f_{13} and f_{14}. The remaining 8 functions ($f_6, f_7, f_8, f_9, f_{10}, f_{11}, f_{15}, f_{17}$) show no significant difference. Thus, the proposed algorithm performs better than the Sobol PSO on multi-modal functions and it also shows superiority on complex functions like the Rosenbrock function,f_3.

Additionally, we have performed Kendall's W test and Cliff's δ along with Bootstrap Confidence Interval tests on median values. In our work, we have used Bootstrap CI after performing Cliff's δ test on each algorithm pair-wise with 10000 bootstrap resamples and 95% CI. In Kendall's W test, the test statistic obtained is 0.7439, showing strong concordance. Cliff's δ along with bootstrap CI points out that the proposed algorithm wins against FHPSO, HPSO-ALS and SPSO with large effect size on majority of functions. The proposed algorithm, when compared against Sobol PSO wins mostly on multi-modal functions. The effect size (δ), confidence interval (CI) and result obtained in each case have been provided in Table 5. In the result column, $'>'$ means that the proposed algorithm wins, $'<'$ means that the opponent wins and $'--'$ signifies that the test is inconclusive.
Thus, we can positively conclude that the proposed algorithm shows competitive results against Sobol PSO, SPSO, HPSO-ALS and FHPSO on 17 basic benchmark functions.

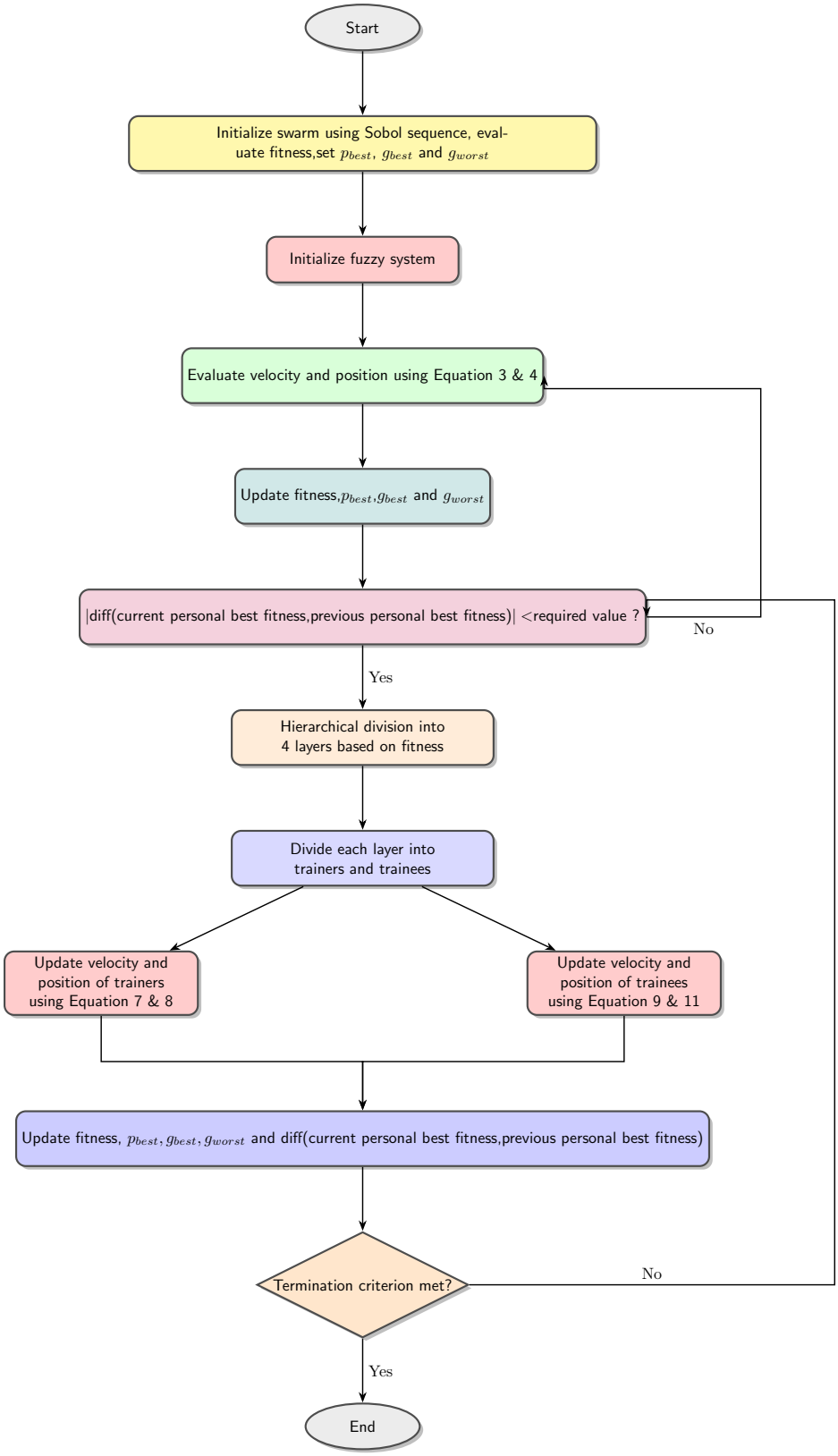


Figure 3: Flowchart of PSO-IMS

Table 4: Comparative results on 17 benchmarks after 20 runs ($f_{13} - f_{17}$)

Algorithm	Test Stats	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}
	Mean	2.8005E-03	1.2228E-07	0.0000E+00	3.01E-88	0.0000E+00
	Std	5.8102E-03	7.6049E-08	0.0000E+00	1.34E-87	0.0000E+00
Proposed	Best	1.0700E-05	4.0300E-09	0.0000E+00	1.81E-125	0.0000E+00
	Worst	2.2900E-02	2.5600E-07	0.0000E+00	6.01E-87	0.0000E+00
	Median	3.6118E-04	1.0881E-07	0.0000E+00	7.97E-115	0.0000E+00
	Mean	8.1740E-03	1.2868E-03	9.2664E-03	1.33E-03	4.2005E+00
	Std	6.7707E-03	1.2921E-03	2.0288E-03	3.2658E-03	6.3534E+00
FHPSO	Best	1.3500E-03	1.2200E-04	6.4700E-04	1.6200E-97	0.0000E+00
	Worst	2.1800E-02	4.3700E-03	9.7200E-03	1.0100E-02	2.1000E+01
	Median	5.0236E-03	5.4075E-04	9.7159E-03	1.1767E-71	0.0000E+00
	Mean	3.0447E-01	1.1879E-02	3.9174E-05	3.3671E-04	1.2872E-03
	Std	3.3574E-01	5.1127E-02	1.7243E-04	4.3947E-04	2.7509E-03
HPSO-ALS	Best	2.3111E-03	4.0410E-07	5.2423E-10	1.7168E-07	1.5339E-07
	Worst	1.1101E+00	2.2895E-01	7.7173E-04	1.4696E-03	1.0638E-02
	Median	1.6819E-01	7.0642E-06	7.8412E-08	1.9197E-04	1.3868E-04
	Mean	2.8887E-02	3.0821E-04	0.0000E+00	0.0000E+00	0.0000E+00
	Std	3.1385E-02	2.5821E-04	0.0000E+00	0.0000E+00	0.0000E+00
Sobol PSO	Best	2.0402E-04	2.3949E-06	0.0000E+00	0.0000E+00	0.0000E+00
	Worst	1.0251E-01	9.6339E-04	0.0000E+00	0.0000E+00	0.0000E+00
	Median	2.2574E-02	2.4691E-04	0.0000E+00	0.0000E+00	0.0000E+00
	Mean	3.2157E-01	1.1389E+00	1.2001E-01	4.7434E-01	2.3750E+01
	Std	1.9522E-01	1.2551E+00	3.3336E-02	4.9343E-01	7.4542E+00
SPSO	Best	2.2600E-02	1.5000E-04	7.8200E-02	4.8300E-02	1.4000E+01
	Worst	6.3700E-01	3.9600E+00	1.7800E-01	2.0200E+00	4.4000E+01
	Median	3.0793E-01	9.9084E-01	1.2699E-01	3.2608E-01	2.2000E+01

5.1.1 Convergence analysis

Among the compared algorithms, Sobol PSO achieves the best mean fitness on most uni-modal benchmark functions due to its simple PSO-based structure, which appears highly effective in relatively simple search spaces. However, its performance declines on more challenging landscapes, particularly the Rosenbrock function (f_3), where the proposed algorithm demonstrates superior capability in handling narrow valley-like geometries. A similar trend is observed for multi-modal functions. While Sobol PSO performs competitively on one function (f_{16}), the proposed algorithm consistently outperforms it on several others (f_{12}, f_{13}, f_{14}). This indicates that the proposed method is more robust in complex search spaces containing multiple local optima. The hierarchical division mechanism employed during exploitation helps the proposed algorithm maintain diversity and avoid premature convergence, leading to improved exploration–exploitation balance.

Although FHPSO also utilizes hierarchical division, its early emphasis on stratification appears to weaken exploration, limiting its overall effectiveness. HPSO-ALS shows strong performance only on f_8, likely due to its use of the global worst particle to enhance exploration. However, this strategy may reduce exploitation efficiency as the search approaches convergence. In conclusion, the proposed algorithm gives better results on 11 out of 17 benchmark functions, as shown in Tables 2 – 4. In the remaining 6 benchmarks, the proposed algorithm shows competitive results on 5 benchmarks. Hence, the credibility of the proposed algorithm is established.

Table 5: Cliff's δ along with bootstrap CI on 17 standard benchmarks

Fn.	Proposed vs Sobol PSO			Proposed vs FHPSO			Proposed vs HPSO-ALS			Proposed vs SPSO		
	δ	CI	Result	δ	CI	Result	δ	CI	Result	δ	CI	Result
f_1	-1.00	[-1.00,-1.00]	<	1.00	[1.00,1.00]	>	1.00	[1.00,1.00]	>	1.00	[1.00,1.00]	>
f_2	-1.00	[-1.00,-1.00]	<	1.00	[1.00,1.00]	>	1.00	[1.00,1.00]	>	1.00	[1.00,1.00]	>
f_3	0.65	[0.34,0.88]	>	1.00	[1.00,1.00]	>	1.00	[1.00,1.00]	>	0.95	[0.81,1.00]	>
f_4	-0.82	[-1.00,-0.59]	<	0.21	[-0.17,0.57]	−−	-0.72	[-1.00,-0.40]	<	-0.80	[-1.00,-0.51]	<
f_5	-1.00	[-1.00,-1.00]	<	1.00	[1.00,1.00]	>	1.00	[1.00,1.00]	>	1.00	[1.00,1.00]	>
f_6	0.00	[0.00,0.00]	−−	0.00	[0.00,0.00]	−−	0.00	[0.00,0.00]	−−	1.00	[1.00,1.00]	>
f_7	0.00	[0.00,0.00]	−−	0.70	[0.50,0.90]	>	1.00	[1.00,1.00]	>	1.00	[1.00,1.00]	>
f_8	-0.13	[-0.47,0.20]	−−	-0.10	[-0.46,0.26]	−−	-0.17	[-0.54,0.22]	−−	-0.07	[-0.42,0.35]	−−
f_9	0.00	[0.00,0.00]	−−	0.40	[0.20,0.60]	>	1.00	[1.00,1.00]	>	1.00	[1.00,1.00]	>
f_{10}	0.00	[0.00,0.00]	−−	0.70	[0.50,0.90]	>	1.00	[1.00,1.00]	>	1.00	[1.00,1.00]	>
f_{11}	0.00	[0.00,0.00]	−−	0.00	[0.00,0.00]	−−	1.00	[1.00,1.00]	>	1.00	[1.00,1.00]	>
f_{12}	0.89	[0.73,0.99]	>	1.00	[1.00,1.00]	>	1.00	[1.00,1.00]	>	1.00	[1.00,1.00]	>
f_{13}	0.76	[0.51,0.93]	>	0.72	[0.42,0.94]	>	0.98	[0.92,1.00]	>	1.00	[0.97,1.00]	>
f_{14}	1.00	[1.00,1.00]	>	1.00	[1.00,1.00]	>	1.00	[1.00,1.00]	>	1.00	[1.00,1.00]	>
f_{15}	0.00	[0.00,0.00]	−−	1.00	[1.00,1.00]	>	1.00	[1.00,1.00]	>	1.00	[1.00,1.00]	>
f_{16}	-1.00	[-1.00,-1.00]	<	0.98	[0.91,1.00]	>	1.00	[1.00,1.00]	>	1.00	[1.00,1.00]	>
f_{17}	0.00	[0.00,0.00]	−−	0.35	[0.15,0.55]	>	1.00	[1.00,1.00]	>	1.00	[1.00,1.00]	>

5.2 Comparison on CEC 2013 test suite

In this section, we compare the proposed algorithm against FHPSO [41], HPSO-ALS [40], Sobol PSO [39], and SPSO [35] on the CEC 2013 test suite consisting of 28 unconstrained benchmark functions ($F_1 - F_{28}$). It consists of 5 uni-modal, 15 multi-modal and 8 composition functions which are both shifted and rotated. A total of 50 runs is performed on each benchmark function. The dimension size used in this study is 30. The maximum number of iterations is 1000 and population size for each algorithm is fixed to 150. The mean, standard deviation, median, best and worst error values for each algorithm on every benchmark is recorded and presented in Tables 6 – 9. Fig. 4 depicts the convergence curve of each algorithm on functions F_1, F_5, F_{10}, and F_{28}. Comparing the mean data of all the algorithms, the proposed algorithm shows superior results on 20 out of 28 benchmark functions from CEC 2013 test suite. These functions are $F_1 - F_7, F_{10} - F_{15}$, $F_{19}, F_{22} - F_{25}, F_{27}$ and F_{28}. Sobol PSO shows the best performance on 5 functions which are $F_9, F_{16}, F_{17}, F_{20}$ and F_{21}. SPSO shows best performance on F_8 and F_{18}, with HPSO-ALS outperforming the others on F_{26}.

The Friedman test has been applied to median values after 50 runs for each benchmark in this study and it is found that the proposed algorithm ranks first with an average rank of 1.6429 and Sobol PSO ranks second with an average rank of 2.4286. SPSO and FHPSO rank third and fourth with average ranks 2.7500 and 3.4464 respectively. Among all evaluated algorithms, HPSO-ALS performs the worst, with an average rank of 4.7321. For further comparison of the top 2 algorithms i.e. the proposed algorithm and Sobol PSO (according to Friedman ranks), we use the Wilcoxon signed-rank test. It is found that the proposed algorithm is significantly better on 18 out of 28 benchmark functions of the CEC 2013 test suite which are F_1, F_2, F_3, F_5, F_6, F_7, F_{10}, F_{11}, F_{12}, F_{13}, F_{15}, F_{22}, F_{23}, F_{24}, F_{25}, F_{26}, F_{27}, and F_{28}. Sobol PSO performs significantly better on 5 benchmarks which are $F_9, F_{16}, F_{17}, F_{18}$, and F_{20}. The remaining 5 benchmark functions i.e. F_4, F_8, F_{14}, F_{19}, and F_{21} show no significant difference.

Table 6: Comparative results for CEC 2013 on $30D$ after 50 runs ($F_1 - F_7$)

Algorithm	Test Stats	F_1	F_2	F_3	F_4	F_5	F_6	F_7
Proposed	Mean	5.0698E-06	2.4220E+06	8.8351E+08	3.4177E+03	2.8679E-01	6.4743E+01	3.3083E+01
	Std	9.8535E-06	1.4251E+06	1.5944E+09	1.8973E+03	1.2554E+00	1.9982E+01	1.2554E+01
	Best	8.9490E-09	6.4744E+05	1.3187E+06	8.6877E+02	4.3912E-03	1.6552E+01	1.0021E+01
	Worst	5.2471E-05	7.1795E+06	8.4179E+09	9.2227E+03	8.9563E+00	1.0021E+02	7.3879E+01
	Median	9.8733E-07	2.1104E+06	2.2838E+08	2.9147E+03	8.6291E-02	7.6322E+01	3.0016E+01
FHPSO	Mean	2.1228E+03	5.3691E+07	1.3971E+10	1.9941E+04	3.0206E+02	2.2293E+02	3.7815E+06
	Std	6.7018E+03	1.0499E+08	5.4636E+10	1.8041E+04	2.9913E+02	4.5865E+02	1.8842E+07
	Best	1.0748E+01	6.8395E+06	1.9014E+08	1.2606E+03	4.0962E+01	3.1195E+01	1.4129E+01
	Worst	2.8577E+04	5.5220E+08	3.8189E+11	6.2856E+04	1.5750E+03	3.1449E+03	1.0542E+08
	Median	9.3265E+01	2.1873E+07	1.3488E+09	1.3154E+04	1.9144E+02	1.1370E+02	6.1439E+01
HPSO-ALS	Mean	2.4862E+03	9.9882E+07	3.0690E+10	3.7594E+04	1.0768E+03	4.7882E+02	1.5624E+02
	Std	1.3513E+03	5.5322E+07	2.1590E+10	1.0605E+04	4.0821E+02	2.3934E+02	4.0516E+01
	Best	1.2386E+03	2.7576E+07	9.1571E+09	1.6830E+04	3.5510E+02	2.0501E+02	9.7334E+01
	Worst	8.7986E+03	4.1628E+08	1.4851E+11	6.5834E+04	2.1231E+03	1.4616E+03	2.7879E+02
	Median	2.0182E+03	9.3101E+07	2.3616E+10	3.6667E+04	1.0152E+03	4.0371E+02	1.5753E+02
Sobol PSO	Mean	2.6052E+02	7.4093E+06	9.8779E+09	3.7608E+03	3.5234E+02	1.1885E+02	1.0544E+02
	Std	5.0949E+02	3.8617E+06	8.1122E+09	1.7796E+03	3.7256E+02	4.3601E+01	3.1245E+01
	Best	1.6171E+00	2.2015E+06	6.7843E+08	1.1634E+03	1.1175E+01	4.7381E+01	5.6915E+01
	Worst	2.4602E+03	2.3233E+07	4.6408E+10	8.5441E+03	2.3017E+03	2.2976E+02	1.8217E+02
	Median	9.0671E+00	6.8157E+06	8.5543E+09	3.3941E+03	2.8908E+02	1.1138E+02	9.9822E+01
SPSO	Mean	1.1307E+03	9.4594E+06	1.8123E+10	3.7214E+03	5.3127E+02	1.1212E+02	1.1795E+02
	Std	1.5243E+03	9.1006E+06	2.1007E+10	1.7457E+03	7.4944E+02	4.9091E+01	4.8223E+01
	Best	3.5461E+00	1.7429E+06	1.4334E+09	8.2568E+02	8.1075E+00	3.5916E+01	4.8675E+01
	Worst	7.3384E+03	6.0225E+07	9.8612E+10	8.7570E+03	2.4376E+03	2.4522E+02	2.9353E+02
	Median	8.0484E+02	7.3372E+06	1.0429E+10	3.4206E+03	1.7835E+02	9.9190E+01	1.0469E+02

In Kendall's W test, the test statistic obtained is 0.5849, showing moderate concordance. Key findings using Cliff's δ along with Bootstrap CI for each pair shows significant difference among algorithms and proposed algorithm outperforms others on the majority of functions with a large effect size. The effect size (δ), confidence interval (CI) and results obtained in each case have been provided in Table 10.

Hence, we conclude that the proposed algorithm shows promising performance on the CEC 2013 test suite among all the algorithms used in this study.

5.2.1 Convergence analysis

The proposed algorithm shows superior performance on all the 5 uni-modal functions of CEC 2013 test suite, 9 out of 15 multi-modal functions, and 6 out of 8 composition functions. Thus, it outperforms FHPSO, HPSO-ALS, Sobol PSO, and SPSO on a total of 20 benchmark functions. The sobol sequence for initialization disperses the population uniformly in the search space, incorporating the g_{worst} particle in the initial stages of the search space helps in exploring the remote areas of the search space. Using fuzzy logic to tune the parameters and having a flexible hierarchy during the exploitation stage enhances convergence accuracy. FHPSO also has a hierarchical division of population, but unlike the proposed algorithm, the low-level particles learn from their personal best and the average of the better-ranked particles in their respective layer. This makes the structure rigid and hence is likely to fall into local optima. On CEC 2013 test suite, FHPSO outperforms none of the algorithms. SPSO shows superior performance on 2 multi-modal functions because its simple structure gives it an advantage over others. Sobol PSO outperforms all others on 4 multi-modal functions and 1 composition function owing to its simple structure. Sobol sequences used

in initialization gives it an advantage over random initialization used by FHPSO, HPSO-ALS and SPSO. HPSO-ALS outperforms all others on 2 multi-modal functions. Here, the whole population learns from the global worst particle throughout the search process. It sometimes helps in avoiding local optima. But, the global worst particle also shows a tendency of leading the population away from the global optimum as the search progresses.

Table 7: Comparative results for CEC 2013 on $30D$ after 50 runs ($F_8 - F_{14}$)

Algorithm	Test Stats	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}
	Mean	2.0973E+01	2.6530E+01	2.4020E+01	8.6360E+01	8.1570E+01	1.5146E+02	3.4781E+03
Proposed	Std	4.8745E-02	2.3949E+00	9.9205E+00	2.1479E+01	2.2208E+01	2.6606E+01	1.0936E+03
	Best	2.0821E+01	2.1314E+01	7.0819E+00	4.4044E+01	3.7471E+01	8.3075E+01	1.8386E+03
	Worst	2.1046E+01	3.0556E+01	4.8080E+01	1.2887E+02	1.3467E+02	2.0924E+02	7.2121E+03
	Median	2.0985E+01	2.6482E+01	2.2642E+01	8.0687E+01	7.8298E+01	1.5089E+02	3.2177E+03
	Mean	2.0990E+01	3.3963E+01	1.2488E+02	2.2803E+02	3.9134E+02	3.2440E+02	5.7167E+03
	Std	4.5344E-02	1.3656E+01	1.5911E+02	2.9981E+02	4.1865E+02	3.5229E+02	1.3268E+03
FHPSO	Best	2.0857E+01	1.8839E+01	1.7675E+01	5.2982E+01	6.5714E+01	8.6507E+01	2.8458E+03
	Worst	2.1074E+01	5.5970E+01	7.2194E+02	1.1035E+03	1.0777E+03	1.1370E+03	8.4759E+03
	Median	2.0999E+01	2.7249E+01	7.4808E+01	1.3050E+02	1.2828E+02	1.6621E+02	5.5099E+03
	Mean	2.0977E+01	3.5766E+01	6.4787E+02	3.0231E+02	3.0538E+02	3.0813E+02	7.2185E+03
	Std	5.2769E-02	2.5252E+00	3.1648E+02	4.1413E+01	4.0781E+01	4.7433E+01	4.1844E+02
HPSO-ALS	Best	2.0767E+01	3.0255E+01	1.9652E+02	2.2802E+02	2.3338E+02	2.1956E+02	6.2139E+03
	Worst	2.1056E+01	4.0151E+01	2.3318E+03	3.8086E+02	4.0151E+02	4.4533E+02	7.9076E+03
	Median	2.0985E+01	3.5440E+01	5.8521E+02	3.0214E+02	2.9744E+02	3.0266E+02	7.1866E+03
	Mean	2.0965E+01	2.4897E+01	1.6333E+02	1.3049E+02	1.3787E+02	2.0092E+02	3.7566E+03
	Std	5.4910E-02	3.0564E+00	1.2808E+02	3.9307E+01	4.3321E+01	3.7587E+01	6.3849E+02
Sobol PSO	Best	2.0774E+01	1.7547E+01	2.6941E+01	5.8053E+01	5.6805E+01	1.2931E+02	1.6751E+03
	Worst	2.1038E+01	3.1627E+01	6.8236E+02	2.3296E+02	3.0687E+02	2.9569E+02	5.0498E+03
	Median	2.0978E+01	2.5035E+01	1.1617E+02	1.2728E+02	1.2570E+02	2.0243E+02	3.7150E+03
	Mean	2.0949E+01	2.6271E+01	1.9718E+02	1.3324E+02	1.3635E+02	2.0406E+02	3.8759E+03
	Std	5.8898E-02	3.0273E+00	1.5819E+02	3.7301E+01	3.1517E+01	4.4359E+01	6.1210E+02
SPSO	Best	2.0781E+01	2.1098E+01	5.8638E+00	6.6375E+01	8.3451E+01	1.3135E+02	2.3999E+03
	Worst	2.1043E+01	3.2361E+01	6.3133E+02	2.4231E+02	2.2379E+02	3.1089E+02	5.1466E+03
	Median	2.0951E+01	2.6016E+01	1.3884E+02	1.2919E+02	1.3509E+02	2.0962E+02	3.9467E+03

6 Case studies

To further validate the efficiency of the proposed algorithm, case studies on two engineering optimization problems have been conducted: the UAV path planning problem and the traveling salesman problem.

6.1 UAV path planning problem

Path planning is a real-world problem [2, 26]. It consists of planning an optimal flight route for one or more UAVs. Each UAV has to maintain a safe distance to avoid colliding with any obstacle in its way. The cost required to avoid such collision is termed as obstacle cost. Moreover, the optimal path found consists of several waypoints joined together consisting of several sharp turns. Thus, path smoothing is required. This is included in the problem formulation as path smoothing cost. The modeling of the problem can be done in various ways [37]. The approach used in this paper is similar to that of Zhang et al. [46]. Our objective

is to plan the shortest path for a UAV while avoiding threat regions.

Table 8: Comparative results for CEC 2013 on $30D$ after 50 runs ($F_{15} - F_{21}$)

Algorithm	Test Stats	F_{15}	F_{16}	F_{17}	F_{18}	F_{19}	F_{20}	F_{21}
	Mean	3.5691E+03	2.6292E+00	1.8388E+02	1.8433E+02	9.6033E+00	1.2980E+01	4.0335E+02
Proposed	Std	1.0037E+03	2.8757E-01	2.6809E+01	1.3862E+01	2.6691E+00	6.1981E-01	6.5103E+01
	Best	1.6317E+03	1.9184E+00	1.0738E+02	1.4543E+02	6.1769E+00	1.1776E+01	3.0000E+02
	Worst	7.0257E+03	3.1531E+00	2.2601E+02	2.0820E+02	1.8240E+01	1.5000E+01	4.4354E+02
	Median	3.4695E+03	2.6631E+00	1.9386E+02	1.8678E+02	9.5287E+00	1.2928E+01	4.4354E+02
	Mean	5.1038E+03	2.1751E+00	3.4107E+02	2.9508E+02	1.6184E+04	1.4803E+01	7.1418E+02
	Std	1.2490E+03	7.1436E-01	3.0874E+02	2.4994E+02	4.0547E+04	6.8674E-01	5.8638E+02
FHPSO	Best	3.2070E+03	4.9202E-01	8.8754E+01	1.4322E+02	6.2143E+00	1.1762E+01	3.5051E+02
	Worst	7.7157E+03	3.1453E+00	9.5982E+02	9.3888E+02	2.0849E+05	1.5000E+01	2.1874E+03
	Median	4.7240E+03	2.4070E+00	1.9151E+02	2.0306E+02	1.4608E+01	1.5000E+01	4.5927E+02
	Mean	6.8751E+03	2.5962E+00	3.6709E+02	3.7119E+02	2.7145E+02	1.4931E+01	7.0815E+02
	Std	5.9944E+02	3.3252E-01	3.4957E+01	4.7769E+01	5.8331E+02	1.0151E-01	2.1183E+02
HPSO-ALS	Best	4.5490E+03	1.8836E+00	2.7826E+02	2.8600E+02	2.5173E+01	1.4625E+01	4.5968E+02
	Worst	7.6415E+03	3.3306E+00	4.5045E+02	5.3456E+02	3.4382E+03	1.5000E+01	1.2731E+03
	Median	6.9876E+03	2.6368E+00	3.6348E+02	3.6615E+02	7.4019E+01	1.4985E+01	6.5831E+02
	Mean	4.0846E+03	1.2579E+00	1.5020E+02	1.4990E+02	1.5273E+01	1.1987E+01	3.7636E+02
	Std	6.7467E+02	3.7864E-01	3.1689E+01	2.4198E+01	3.2872E+01	1.1095E+00	8.8553E+01
Sobol PSO	Best	2.7113E+03	4.1138E-01	9.9785E+01	9.5105E+01	4.5078E+00	9.6237E+00	1.5076E+02
	Worst	5.0877E+03	2.0708E+00	2.4941E+02	2.0190E+02	2.4127E+02	1.4500E+01	6.0815E+02
	Median	4.1503E+03	1.2209E+00	1.4633E+02	1.4979E+02	9.9948E+00	1.1874E+01	3.5586E+02
	Mean	4.2776E+03	1.4016E+00	1.5088E+02	1.4178E+02	7.1159E+02	1.2196E+01	3.7915E+02
	Std	7.7685E+02	3.8214E-01	2.1538E+01	2.8751E+01	2.8550E+03	1.1324E+00	1.2296E+02
SPSO	Best	2.7183E+03	6.3067E-01	1.0921E+02	9.0655E+01	6.7281E+00	9.7212E+00	1.9598E+02
	Worst	5.6962E+03	2.2219E+00	2.0759E+02	2.1757E+02	1.8851E+04	1.5000E+01	7.8968E+02
	Median	4.2094E+03	1.3850E+00	1.4621E+02	1.4149E+02	1.4400E+01	1.2066E+01	3.5065E+02

6.1.1 Results and discussion

We have solved the aforementioned problem using our proposed algorithm and compared the results with Sobol PSO [39], SPSO [35], HPSO-ALS [40], and FHPSO [41]. We have considered two cases, case 1 with 3 threat regions and case 2 with 6 threat regions. The coordinates of the start point of the search space, S are $(5, 5, 0)$ and the end point, T is $(500, 500, 0)$. The population size of all algorithms has been set to 150. The maximum number of iterations is 1000. Each algorithm has been run 20 times independently in MATLAB R2025a and the best results have been recorded (Tables 11– 12).

From Tables 11 – 12, it can be observed that that in case 1 with 3 threat regions, the proposed algorithm ranks first in terms of optimal results, while ranking third in terms of mean fitness. In case 2 with 6 threat regions, the proposed algorithm ranks first in both optimal as well as mean results. Hence, we can conclude that the proposed algorithm is best suited when the number of threat regions is increased.

From Fig. 5, it is easy to observe that optimal paths obtained using FHPSO and HPSO-ALS cross one or more threat regions. The optimal paths obtained by SPSO, Sobol PSO and the proposed algorithm do not cross any threat regions.

Table 9: Comparative results for CEC 2013 on $30D$ after 50 runs ($F_{22} - F_{28}$)

Algorithm	Test Stats	F_{22}	F_{23}	F_{24}	F_{25}	F_{26}	F_{27}	F_{28}
	Mean	3.4665E+03	3.4136E+03	2.5614E+02	2.8082E+02	3.2128E+02	6.8637E+02	3.6550E+02
Proposed	Std	8.6837E+02	6.4221E+02	1.1104E+01	1.0943E+01	3.7070E+01	1.2565E+02	2.9950E+02
	Best	1.5973E+03	2.0693E+03	2.3799E+02	2.5789E+02	2.0007E+02	4.3450E+02	1.0000E+02
	Worst	5.1623E+03	5.6262E+03	2.8459E+02	3.0556E+02	3.5262E+02	9.5184E+02	1.6271E+03
	Median	3.5130E+03	3.3413E+03	2.5677E+02	2.8031E+02	3.3013E+02	6.8650E+02	3.0000E+02
	Mean	6.0570E+03	6.4045E+03	4.1038E+02	3.7393E+02	3.4539E+02	1.0650E+03	1.7090E+03
	Std	2.0365E+03	1.6672E+03	2.3495E+02	8.0679E+01	8.0677E+01	1.4724E+02	2.1426E+03
FHPSO	Best	3.8755E+03	3.9249E+03	2.1628E+02	2.7956E+02	2.0026E+02	7.6728E+02	2.3491E+02
	Worst	1.0063E+04	9.5042E+03	8.9001E+02	4.8817E+02	6.5145E+02	1.4961E+03	7.4707E+03
	Median	5.1511E+03	5.8777E+03	2.8831E+02	3.2847E+02	3.6340E+02	1.0614E+03	8.2813E+02
	Mean	7.8085E+03	7.8579E+03	3.1551E+02	3.4655E+02	2.1982E+02	1.2579E+03	1.8875E+03
	Std	5.4218E+02	5.3229E+02	1.8075E+01	2.3223E+01	4.3769E+01	9.2082E+01	4.8704E+02
HPSO-ALS	Best	6.4983E+03	6.3536E+03	2.8179E+02	3.1358E+02	2.0318E+02	1.0860E+03	1.0478E+03
	Worst	8.7306E+03	8.5280E+03	3.6816E+02	4.2149E+02	4.0086E+02	1.4287E+03	3.5904E+03
	Median	7.8115E+03	7.9888E+03	3.1432E+02	3.4137E+02	2.0770E+02	1.2474E+03	1.8434E+03
	Mean	4.4493E+03	4.7440E+03	2.8302E+02	3.0011E+02	3.6802E+02	9.4532E+02	1.2290E+03
	Std	6.5485E+02	4.6880E+02	8.1294E+00	9.8152E+00	8.1222E+00	9.5614E+01	5.7374E+02
Sobol PSO	Best	2.6748E+03	3.6187E+03	2.6599E+02	2.8231E+02	3.5026E+02	7.0851E+02	4.3468E+02
	Worst	5.5198E+03	5.4707E+03	2.9837E+02	3.2425E+02	3.8597E+02	1.1234E+03	2.3243E+03
	Median	4.4264E+03	4.8037E+03	2.8241E+02	2.9960E+02	3.6843E+02	9.5434E+02	1.4169E+03
	Mean	4.1564E+03	4.8717E+03	2.8250E+02	2.9462E+02	3.0081E+02	1.0034E+03	1.2491E+03
	Std	7.9310E+02	9.0841E+02	9.5586E+00	8.5496E+00	7.9467E+01	8.7321E+01	6.3240E+02
SPSO	Best	2.5564E+03	2.5729E+03	2.6126E+02	2.7957E+02	2.0010E+02	7.6750E+02	1.7501E+02
	Worst	5.6582E+03	6.6124E+03	3.0414E+02	3.1092E+02	3.7690E+02	1.1816E+03	3.4557E+03
	Median	4.1116E+03	4.9005E+03	2.8182E+02	2.9381E+02	3.5558E+02	1.0029E+03	1.2700E+03

6.2 Traveling salesman problem

The traveling salesman problem (TSP) is a classical NP-Hard combinatorial problem. Many metaheuristic algorithms have been used previously to solve it [29, 31]. For a continuous problem to be solved using TSP, careful execution is needed. We have tried to discretize our algorithm to solve TSP. In discrete problems, one deals with permutations as particles. We start with randomly generated permutations. The velocity update here is nothing but swaps required to convert one permutation into another. Certain probabilities are assigned to the algorithm which decides whether swaps are to be executed or not. For instance, whether a particle will go through swaps to get converted into its personal best 'permutation' so far is decided upon the hands of $p_{best_{prob}}$. If this so-called $p_{best_{prob}}$ is high enough, then we perform swaps to convert its current position (permutation) into the p_{best} found so far.

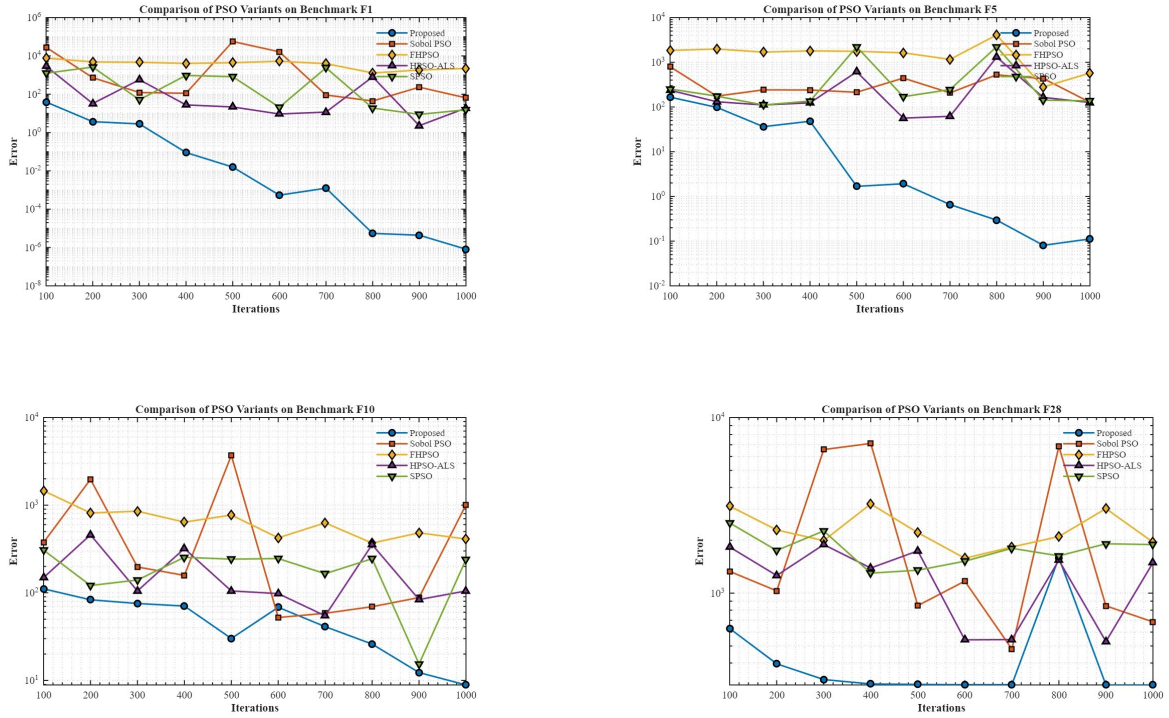


Figure 4: Convergence curves of proposed algorithm, HPSO-ALS, FHPSO, Sobol PSO and SPSO on CEC 2013 benchmark for functions F_1, F_5, F_{10}, and F_{28}

6.2.1 Results and discussion

A traveling salesman problem with 31 cities has been solved using the proposed algorithm. The obtained results have also been compared with SPSO [35] and HPSO-ALS [40]. The coordinates of the cities and the parameter settings of HPSO-ALS have been adopted from Wang et. al. [40]. For SPSO, $p_{best_{prob}}$ and $g_{best_{prob}}$ are set to $0.500 * rand$. For the proposed algorithm, $p_{best_{prob}}$ is set to $0.100 * rand$, $g_{best_{prob}}$ is set to $0.075 * rand$, $g_{best_{prob}}$ and $g_{worst_{prob}}$ are set to $0.075 * rand$ and $0.001 * rand$, respectively. Moreover, $alpha_{prob}$ as well as $candidate_{prob}$ are set to $0.500 * rand$. The swarm size is fixed at 500 for each algorithm, and a maximum of 2000 iterations is performed in each case.

The best fitness value obtained by SPSO on TSP is 18.814529 while for HPSO-ALS, it is 17.310666. Using the proposed algorithm, the best fitness value obtained is 16.588895. Hence, the proposed algorithm gives the best result when compared against SPSO and HPSO-ALS. The optimal paths obtained by all three variants are shown in Fig. 6.

7 Conclusion and future scope

In this work, we propose a hybrid variant of PSO that achieves a balance between exploration and exploitation through the use of the global worst particle (g_{worst}) and a hierarchical division strategy. To promote uniform dispersion of particles during initialization, we suggest using the Sobol sequence in place of random initialization. Incorporating the global worst particle in the algorithm ensures a thorough search of the problem space. To avoid getting stuck at a local optimum, hierarchical division is useful. A social hierarchy encourages all individuals of the population to perform effectively in synchronization. It is the backbone of human civilization. Moreover, using fuzzy logic to tune parameters helps mimic a real social structure.

Table 10: Cliff's δ along with bootstrap CI on CEC 2013

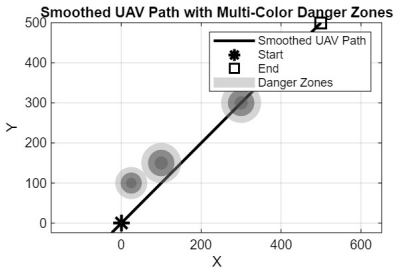
Fn.	**Proposed vs Sobol PSO**			**Proposed vs FHPSO**			**Proposed vs HPSO-ALS**			**Proposed vs SPSO**		
	δ	**CI**	**Result**	δ	**CI**	**Result**	δ	**CI**	**Result**	δ	**CI**	**Result**
F_1	1.00	[1.00,1.00]	>	1.00	[1.00,1.00]	>	1.00	[1.00,1.00]	>	1.00	[1.00,1.00]	>
F_2	0.88	[0.78,0.95]	>	1.00	[1.00,1.00]	>	1.00	[1.00,1.00]	>	0.81	[0.71,0.92]	>
F_3	0.93	[0.86,0.98]	>	0.62	[0.44,0.78]	>	1.00	[1.00,1.00]	>	0.95	[0.90,0.99]	>
F_4	0.14	[-0.09,0.38]	−−	0.78	[0.61,0.90]	>	1.00	[1.00,1.00]	>	0.14	[-0.07,0.37]	−−
F_5	1.00	[1.00,1.00]	>	1.00	[1.00,1.00]	>	1.00	[1.00,1.00]	>	1.00	[1.00,1.00]	>
F_6	0.83	[0.70,0.93]	>	0.76	[0.62,0.88]	>	1.00	[1.00,1.00]	>	0.66	[0.50,0.80]	>
F_7	0.99	[0.97,1.00]	>	0.75	[0.58,0.87]	>	1.00	[1.00,1.00]	>	0.99	[0.96,1.00]	>
F_8	-0.06	[-0.22,0.11]	−−	0.04	[-0.12,0.19]	−−	-0.04	[-0.20,0.14]	−−	-0.29	[-0.48,-0.11]	<
F_9	-0.33	[-0.53,-0.09]	<	0.15	[-0.07,0.38]	−−	1.00	[1.00,1.00]	>	-0.07	[-0.30,0.14]	−−
F_{10}	0.95	[0.89,0.99]	>	0.88	[0.78,0.96]	>	1.00	[1.00,1.00]	>	0.84	[0.68,0.96]	>
F_{11}	0.69	[0.52,0.83]	>	0.54	[0.35,0.72]	>	1.00	[1.00,1.00]	>	0.73	[0.58,0.86]	>
F_{12}	0.81	[0.69,0.91]	>	0.72	[0.57,0.84]	>	1.00	[1.00,1.00]	>	0.86	[0.75,0.94]	>
F_{13}	0.72	[0.57,0.85]	>	0.24	[0.03,0.46]	>	1.00	[1.00,1.00]	>	0.68	[0.52,0.82]	>
F_{14}	0.32	[0.10,0.53]	>	0.83	[0.70,0.94]	>	0.98	[0.92,1.00]	>	0.38	[0.16,0.58]	>
F_{15}	0.41	[0.20,0.60]	>	0.71	[0.56,0.84]	>	0.97	[0.92,1.00]	>	0.49	[0.30,0.67]	>
F_{16}	-1.00	[-1.00,-0.98]	<	-0.39	[-0.60,-0.17]	<	-0.06	[-0.29,0.16]	−−	-0.99	[-1.00,-0.98]	<
F_{17}	-0.60	[-0.77,-0.41]	<	0.12	[-0.12,0.35]	−−	1.00	[1.00,1.00]	>	-0.67	[-0.81,-0.49]	<
F_{18}	-0.76	[-0.89,-0.62]	<	0.35	[0.11,0.56]	>	1.00	[1.00,1.00]	>	-0.81	[-0.91,-0.67]	<
F_{19}	0.13	[-0.11,0.36]	−−	0.64	[0.44,0.80]	>	1.00	[1.00,1.00]	>	0.60	[0.41,0.76]	>
F_{20}	-0.57	[-0.76,-0.37]	<	0.85	[0.71,0.96]	>	0.95	[0.86,1.00]	>	-0.54	[-0.71,-0.34]	<
F_{21}	-0.07	[-0.30,0.16]	−−	0.74	[0.57,0.89]	>	1.00	[1.00,1.00]	>	-0.21	[-0.44,0.02]	−−
F_{22}	0.64	[0.47,0.80]	>	0.86	[0.75,0.94]	>	1.00	[1.00,1.00]	>	0.44	[0.23,0.64]	>
F_{23}	0.90	[0.79,0.98]	>	0.96	[0.91,1.00]	>	1.00	[1.00,1.00]	>	0.81	[0.68,0.92]	>
F_{24}	0.95	[0.89,0.99]	>	0.86	[0.72,0.96]	>	1.00	[1.00,1.00]	>	0.92	[0.84,0.98]	>
F_{25}	0.80	[0.68,0.91]	>	0.92	[0.85,0.98]	>	1.00	[1.00,1.00]	>	0.67	[0.51,0.81]	>
F_{26}	1.00	[0.99,1.00]	>	0.66	[0.45,0.83]	>	-0.73	[-0.90,-0.55]	<	0.28	[0.02,0.55]	>
F_{27}	0.90	[0.81,0.97]	>	0.98	[0.94,1.00]	>	1.00	[1.00,1.00]	>	0.97	[0.93,1.00]	>
F_{28}	0.93	[0.84,1.00]	>	0.87	[0.74,0.97]	>	0.98	[0.93,1.00]	>	0.84	[0.70,0.97]	>

Table 11: Comparison of optimal and mean results for Sobol PSO, SPSO, HPSO-ALS, FHPSO and the proposed algorithm for case 1

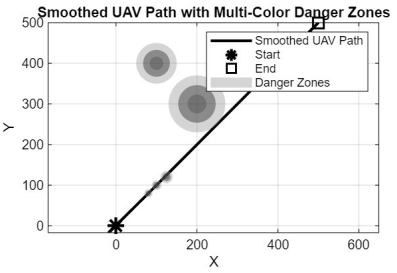
Algorithm	**Optimal Result**	**Mean Result**
Sobol PSO	715.956355	730.2148277
SPSO	714.59827	**723.7382709**
HPSO-ALS	1147.629711	1804.24856
FHPSO	959.144701	1305.892019
Proposed	**714.575055**	749.4361269

Table 12: Comparison of optimal and mean results for Sobol PSO, SPSO, HPSO-ALS, FHPSO and the proposed algorithm for case 2

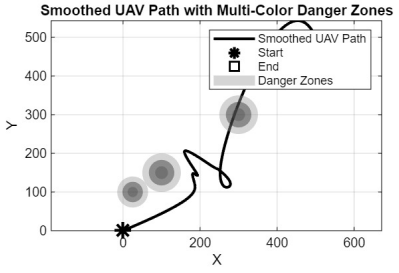
Algorithm	**Optimal Result**	**Mean Result**
Sobol PSO	722.344991	734.2601799
SPSO	722.417454	742.2925033
HPSO-ALS	1110.369207	1514.36069
FHPSO	937.38589	1286.191292
Proposed	**721.879877**	**728.5808532**



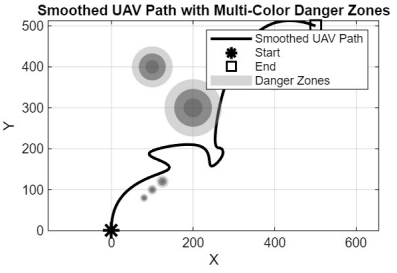
(a) HPSO-ALS with 3 threat regions



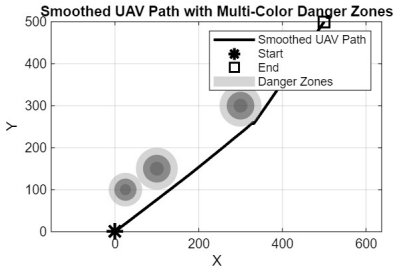
(b) HPSO-ALS with 6 threat regions



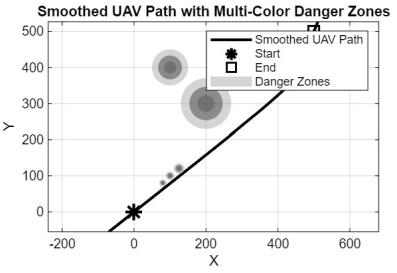
(c) FHPSO with 3 threat regions



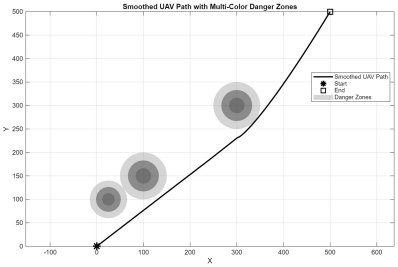
(d) FHPSO with 6 threat regions



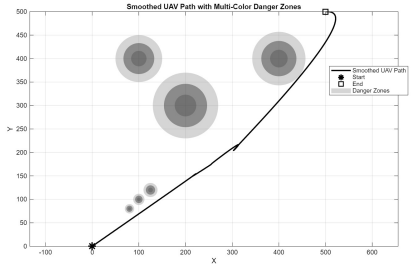
(e) SPSO with 3 threat regions



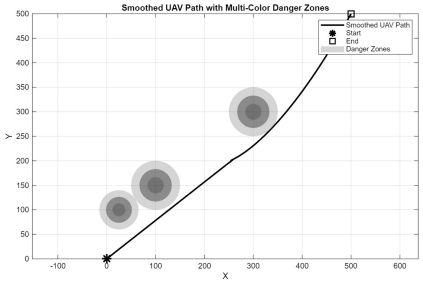
(f) SPSO with 6 threat regions



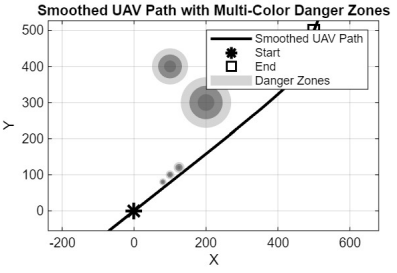
(g) Sobol PSO with 3 threat regions



(h) Sobol PSO with 6 threat regions

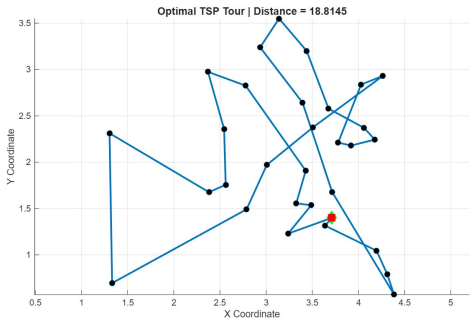


(i) Proposed with 3 threat regions

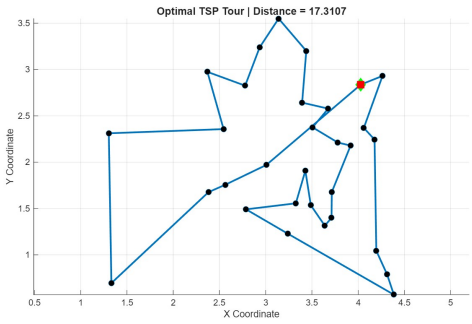


(j) Proposed with 6 threat regions

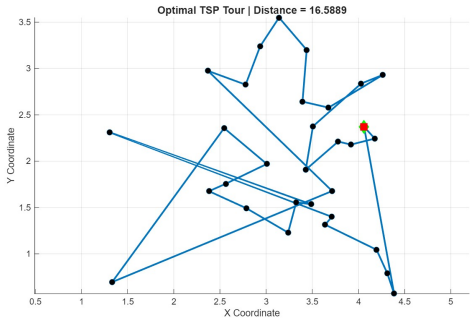
Figure 5: Optimal paths generated by HPSO-ALS, FHPSO, SPSO, Sobol PSO and Proposed algorithm in cases 1 and 2 respectively



(a) Optimal path for SPSO



(b) Optimal path for HPSO-ALS



(c) Optimal path for proposed algorithm

Figure 6: Optimal path for traveling salesman problem

The effectiveness of the proposed algorithm has been rigorously demonstrated through extensive experiments on 17 standard benchmark functions, the CEC 2013 test suite, and real-world optimization problems such as UAV path planning and the traveling salesman problem, in comparison with several powerful PSO variants. The proposed algorithm demonstrates better performance in terms of mean fitness value on 11 out of 17 basic benchmark functions. Similarly, on the CEC 2013 test suite, the proposed algorithm outperforms other PSO variants on 20 out of 28 benchmarks. For statistical validation, Friedman, Wilcoxon, Kendall's W, Cliff's δ along with Bootstrap Confidence Interval (CI) has been employed to validate our results obtained on 17 basic benchmarks as well as on CEC 2013 test suite. To further validate the efficiency of the proposed algorithm, it has been tested on two engineering problems namely the UAV path planning problem and the traveling salesman problem. In both cases, the proposed algorithm produced satisfactory results. In accordance with the No Free Lunch Theorem, no algorithm is universally optimal, and the proposed method also has scope for improvement. Future work may focus on exploring more advanced fuzzy membership functions, such as Gaussian or sigmoidal, and extending the validation to a broader range of engineering optimization problems.

References

[1] D. Bertsimas, J. Tsitsiklis, *Simulated annealing*, Statistical Science, **8**(1) (1993), 10-15. https://doi.org/10.1214/ss/1177011077

[2] H. Chen, T. Jia, J. Guo, L. Yang, *A multi-strategy enhanced black-winged kite algorithm for UAV path planning*, The Journal of Supercomputing, **81**(15) (2025), 1-32. https://doi.org/10.1007/s11227-025-07905-4

[3] G. Chen, H. Xinbo, J. Jia, Z. Min, *Natural exponential inertia weight strategy in particle swarm optimization*, 6th World Congress on Intelligent Control and Automation, **1** (2006), 3672-3675. https://doi.org/10.1109/WCICA.2006.1713055

[4] R. C. Eberhart, Y. Shi, *Tracking and optimizing dynamic systems with particle swarms*, Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546), **1** (2001), 94-100. https://doi.org/10.1109/CEC.2001.934376

[5] M. Eftekhari, A. Mehrpooya, F. Saberi-Movahed, V. Torra, *How fuzzy concepts contribute to machine learning*, Springer, Cham, Switzerland, (2022). https://doi.org/10.1007/978-3-030-94066-9

[6] R. Etesami, M. Madadi, F. Keynia, *Adaptive fuzzy swarm-based search algorithm (AFSSA) for complex engineering optimization*, Iranian Journal of Fuzzy Systems, **22**(6) (2025), 125-145. https://doi.org/10.22111/ijfs.2025.52217.9209

[7] Y. Feng, G. Teng, A. Wang, Y. Yao, *Chaotic inertia weight in particle swarm optimization*, Second International Conference on Innovative Computing, Informatio and Control (ICICIC 2007), (2007), 475-475. https://doi.org/10.1109/ICICIC.2007.209

[8] A. Ghadiri, M. Pazoki, S. Erfani, *Hybrid GA-PSO-optimized neural network for biogas production: Comparative evaluation of metaheuristic algorithms*, Renewable Energy, **262** (2026), 125432. https://doi.org/10.1016/j.renene.2026.125432

[9] A. Ghodousian, S. Zal, *A two-phase-ACO algorithm for solving nonlinear optimization problems subjected to fuzzy relational equations*, Iranian Journal of Fuzzy Systems, **21**(5) (2024), 151-174. https://doi.org/10.22111/ijfs.2024.49652.8760

[10] E. Guo, Y. Gao, C. Hu, *A two-stage evolutionary algorithm based on hybrid penalty strategy and its application to multi-UAV path planning*, Expert Systems with Applications, **298**(C) (2026), 129698. https://doi.org/10.1016/j.eswa.2025.129698

[11] J. H. Holland, *Genetic algorithms*, Scientific American, **267**(1) (1992), 66-73. https://www.jstor.org/stable/24939139

[12] H. Jabeen, Z. Jalil, A. R. Baig, *Opposition based initialization in particle swarm optimization (O-PSO)*, Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers, (2009), 2047-2052. https://doi.org/10.1145/1570256.1570274

[13] J. Jin, X. Pang, B. Wang, D. Wang, Z. Zheng, *Optimal scheduling method of carbon-green certificate trading virtual power plant via Q-learning-enhanced particle swarm algorithm*, Complex and Intelligent Systems, **12**(1) (2026), 51. https://doi.org/10.1007/s40747-025-02176-1

[14] D. S. Johnson, C. H. Papadimitriou, M. Yannakakis, *How easy is local search?*, Journal of Computer and System Sciences, **37**(1) (1988), 79-100. https://doi.org/10.1016/0022-0000(88)90046-3

[15] A. Karkadakattil, *AI and metaheuristic optimization in additive manufacturing of lightweight alloys: A critical review*, Journal of The Institution of Engineers (India): Series C, **107** (2026), 1063-1085. https://doi.org/10.1007/s40032-026-01355-4

[16] J. Kennedy, *Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance*, Proceedings of the 1999 Congress on Evolutionary Computation-CEC'99 (Cat. No. 99TH8406), **3** (1999), 1931-1938. https://doi.org/10.1109/CEC.1999.785509

[17] J. Kennedy, R. Eberhart, *Particle swarm optimization*, Proceedings of ICNN'95-International Conference on Neural Networks, **4** (1995), 1942-1948. https://doi.org/10.1109/ICNN.1995.488968

[18] J. Kennedy, R. Mendes, *Population structure and particle swarm performance*, Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02 (Cat. No. 02TH8600), **2** (2002), 1671-1676. https://doi.org/10.1109/CEC.2002.1004493

[19] P. A. Kowalski, S. Kucharczyk, J. Mańdziuk, *Constrained hybrid metaheuristic algorithm for probabilistic neural networks learning*, Information Sciences, **713** (2025), 122185. https://doi.org/10.1016/j.ins.2025.122185

[20] X. Li, Z. Yang, M. Li, W. Hong, *Integrated scheduling of cargo vessels, research vessels, and marine experiments in multifunctional ports using Q-learning enhanced PSO*, Swarm and Evolutionary Computation, **102** (2026), 102315. https://doi.org/10.1016/j.swevo.2026.102315

[21] J. J. Liang, A. K. Qin, P. N. Suganthan, S. Baskar, *Comprehensive learning particle swarm optimizer for global optimization of multimodal functions*, IEEE Transactions on Evolutionary Computation, **10**(3) (2006), 281-295. https://doi.org/10.1109/TEVC.2005.857610

[22] B. Liang, Y. Zhao, Y. Li, *A hybrid particle swarm optimization with crisscross learning strategy*, Engineering Applications of Artificial Intelligence, **105** (2021), 104418. https://doi.org/10.1016/j.engappai.2021.104418

[23] P. Melin, F. Olivas, O. Castillo, F. Valdez, J. Soria, M. Valdez, *Optimal design of fuzzy classification systems using PSO with dynamic parameter adaptation through fuzzy logic*, Expert Systems with Applications, **40**(8) (2013), 3196-3206. https://doi.org/10.1016/j.eswa.2012.12.033

[24] S. Mirjalili, S. M. Mirjalili, A. Lewis, *Grey wolf optimizer*, Advances in Engineering Software, **69** (2014), 46-61. https://doi.org/10.1016/j.advengsoft.2013.12.007

[25] M. Nasir, S. Das, D. Maity, S. Sengupta, U. Halder, P. N. Suganthan, *A dynamic neighborhood learning based particle swarm optimizer for global numerical optimization*, Information Sciences, **209** (2012), 16-36. https://doi.org/10.1016/j.ins.2012.04.028

[26] A. Ozlek, B. Ervural, B. Cayir Ervural, *A hybrid IRN-based BWM–COPRAS framework for electro-optic system selection in UAVs with heterogeneous evaluations*, Iranian Journal of Fuzzy Systems, **23**(2) (2026), 157-175. https://doi.org/10.22111/ijfs.2026.52261.9218

[27] D. Pal, H. K. Sharma, O. Prentkovskis, F. Chakraborty, L. Maskeliūnaitė, *A study of the multi-objective neighboring only quadratic minimum spanning tree problem in the context of uncertainty*, Applied Sciences, **14**(19) (2024), 8941. https://doi.org/10.3390/app14198941

[28] D. Pal, H. K. Sharma, O. Prentkovskis, F. Chakraborty, L. Maskeliūnaitė, *Multi-objective windy postman problem in a fuzzy transportation network*, Promet-Traffic and Transportation, **37**(4) (2025), 853-873. https://doi.org/10.7307/ptt.v37i4.1134

[29] P. Y. Pamungkas, N. M. E. Normasari, A. A. Fanani, *Modified gorilla troops optimizer with elite opposition-based learning and tangent flight operator to solve traveling salesman problem*, Journal of Intelligent and Fuzzy Systems, **50**(3) (2026), 771-787. https://doi.org/10.1177/18758967251360039

[30] P. Patro, K. Kumar, G. S. Kumar, A. K. Sahu, *Intelligent data classification using optimized fuzzy neural network and improved cuckoo search optimization*, Iranian Journal of Fuzzy Systems, **20**(6) (2023), 155-169. https://doi.org/10.22111/ijfs.2023.44767.7887

[31] A. Raj, P. Punia, P. Kumar, *An innovative fuzzy gravitational search algorithm (FGSA) with an adaptive swap mechanism for solving travelling salesman problem (TSP)*, International Journal of Information Technology, (2025), 1-17. https://doi.org/10.1007/s41870-025-02848-8

[32] K. Rajwar, K. Deep, S. Das, *An exhaustive review of the metaheuristic algorithms for search and optimization: Taxonomy, applications, and open challenges*, Artificial Intelligence Review, **56**(11) (2023), 13187-13257. https://doi.org/10.1007/s10462-023-10470-y

[33] A. Ratnaweera, S. K. Halgamuge, H. C. Watson, *Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients*, IEEE Transactions on Evolutionary Computation, **8**(3) (2004), 240-255. https://doi.org/10.1109/TEVC.2004.826071

[34] R. Sharma, J. S. Matharu, K. S. Parmar, *A survey on particle swarm optimization: Evolution, adaptations and practical implementations*, Applied Soft Computing, **186** (2025), 114016. https://doi.org/10.1016/j.asoc.2025.114016

[35] Y. Shi, R. C. Eberhart, *A modified particle swarm optimizer*, 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360), (1998), 69-73. https://doi.org/10.1109/ICEC.1998.699146

[36] Y. Shi, R. C. Eberhart, *Empirical study of particle swarm optimization*, Proceedings of the 1999 Congress on Evolutionary Computation-CEC'99 (Cat. No. 99TH8406), **3** (1999), 1945-1950. https://doi.org/10.1109/CEC.1999.785511

[37] H. Sun, J. Cao, X. Liang, C. Lan, Q. Zheng, X. Su, H. Li, X. Ding, *Return path planning for UAVs in mountainous power transmission line inspection based on an improved grey wolf optimization algorithm*, 2025 International Conference of Clean Energy and Electrical Engineering (ICCEEE), (2025), 1-6. https://doi.org/10.1109/ICCEEE63357.2025.11156523

[38] Z. Tian, *Path planning for mobile robots based on enhanced particle swarm optimization algorithm*, Journal of Electrical Engineering and Technology, (2026), 1-14. https://doi.org/10.1007/s42835-026-02747-3

[39] N. Q. Uy, N. X. Hoai, R. I. McKay, P. M. Tuan, *Initialising PSO with randomised low-discrepancy sequences: The comparative results*, 2007 IEEE Congress on Evolutionary Computation, (2007), 1985-1992. https://doi.org/10.1109/CEC.2007.4424717

[40] L. Wang, D. Tian, X. Gou, Z. Shi, *Hybrid particle swarm optimization with adaptive learning strategy*, Soft Computing, **28**(17) (2024), 9759-9784. https://doi.org/10.1007/s00500-024-09814-9

[41] Y. Wang, Z. Wang, G. Wang, *Hierarchical learning particle swarm optimization using fuzzy logic*, Expert Systems with Applications, **232** (2023), 120759. https://doi.org/10.1016/j.eswa.2023.120759

[42] F. Wang, H. Zhang, K. Li, Z. Lin, J. Yang, X. Shen, *A hybrid particle swarm optimization algorithm using adaptive learning strategy*, Information Sciences, **436-437** (2018), 162-177. https://doi.org/10.1016/j.ins.2018.01.027

[43] W. Ye, W. Feng, S. Fan, *A novel multi-swarm particle swarm optimization with dynamic learning strategy*, Applied Soft Computing, **61** (2017), 832-843. https://doi.org/10.1016/j.asoc.2017.08.051

[44] L. A. Zadeh, *Fuzzy sets*, Information and Control, **8**(3) (1965), 338-353. https://doi.org/10.1016/S0019-9958(65)90241-X

[45] Z. Zhan, J. Zhang, et. al., *Adaptive particle swarm optimization*, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), **39**(6) (2008), 1362-1381. https://doi.org/10.1109/TSMCB.2009.2015956

[46] B. Zhang, H. Duan, *Three-dimensional path planning for uninhabited combat aerial vehicle based on predator-prey pigeon-inspired optimization in dynamic environment*, IEEE/ACM Transactions on Computational Biology and Bioinformatics, **14**(1) (2015), 97-107. https://doi.org/10.1109/TCBB.2015.2443789