BİTLİS EREN ÜNİVERSİTESİ
FEN BİLİMLERİ DERGİSİ

ISSN: 2147-3129 / e-ISSN: 2147-3188

Article Type	: Research Article	**Year**	: 2025
Received	: December 8, 2024	**Volume**	: 14
Revised	: May 13, 2025	**Issue**	: 2
Accepted	: May 29, 2025	**Pages**	: 806-837
DOI	: 10.17798/bitlisfen.1598152		

GEAR TRAIN DESIGN SOLUTIONS VIA HYPERSPHERE DYNAMICS DRIVEN ADVANCED PARTICLE SWARM OPTIMIZATION

İclal GÖR [1]

[1] *Aydın Adnan Menderes University, Department of Mathematic, Aydın, Türkiye, iclal@adu.edu.tr*

ABSTRACT

This work presents an advanced version of the Particle Swarm Optimization (PSO) algorithm, a well-known optimization algorithm for the solution of the global optimization problems, called PSO with Hypersphere Dynamics and Mutation (PSO-HDM), to deal with the optimization obstacles. The novel method employs a novel technique where the particles' positions are updated using the rotation of the hyperspheres, providing for better exploration of the search space. In addition, two new mutation techniques, Jitter and Gaussian, are used to keep away from the local optima and enhance the solution variety. Dynamic modifications of the classical PSO's parameters, such as cognitive and social coefficients, also improve the algorithm's achievement. The PSO-HDM optimization algorithm is evaluated with utilizing some benchmark functions and compared to classical PSO, getting better values in determining the optimal solutions. Gear train design problems are selected as an engineering design problem to show the effectiveness of the new suggested method. The obtained results present the capability of the proposed method. This proposed optimization algorithm could be seen as an alternative method to other optimization algorithms proposed in the literature.

Keywords: Optimization, Metaheuristic algorithms, Particle swarm optimization, Mutation, Hypersphere, Gear train design problem.

1 INTRODUCTION

Many engineering design problems are expressed with mathematical expressions and models, illustrating a significant area of various fields [1]. These engineering problems are also considered difficult problems because of their nature as real-world challenges. They are categorized as hard problems due to their combinatorial complexity. Non-Linear Programming

(NLP) is utilized to model these real-world problems, using both variables and constraints to represent complex relationships precisely [2]. Metaheuristic algorithms are used to solve NLP and are often chosen as an important alternative to the classical traditional methods because they do not use derivative information. Traditional methods have difficulty finding solutions, and even when they get successful solution, they typically only get local optimal solution, and they also do not guarantee of reaching the global optimum solution [3]. The reason for the using for metaheuristic algorithms is their capability to keep an effective balance between local search (exploitation) and global search (exploration) [4]. However, it is not possible for all metaheuristic algorithms to be achieved in all optimization problems. According to No Free Lunch Theorem (NFL), no optimization algorithm is universally superior to other optimization algorithms across all problems [5]. No free lunch theorem emphasizes that there is no "one-size-fits-all" solution in optimization problem. It is important the selection of the optimal algorithm should be based on the problem.

In the literature, there are several metaheuristic algorithms include Particle Swarm Optimization (PSO) [6], Modified Grey Wolf Optimization Algorithm (MGWO) [7], Tunicate Search Algorithm (TSA) [8], Salp Swarm Algorithm (SSA) [9], Multi-Verse Optimizer (MVO) [10], Grey Wolf Optimizer (GWO) [11], Improved Gray Wolf Optimization (IGWO) [12], Genetic-Particle Swarm Optimization (GPSO) [13], Ant Lion Optimizer (ALO) [14], Cuckoo Search Algorithm (CS) [15], Mine Blast Algorithm (MBA) [16], Interior Search Algorithm (ISA) [17], Genetic Algorithms (GA) [18], Artificial Bee Colony Algorithm (ABC) [19], Genetic Adaptive Search (GA) [20], Augmented Lagrange Multiplier (ALM) [21], Moth-Flame Optimization Algorithm (MFO) [22], Dingo Optimization Algorithm (DOA) [23], Flow Direction Algorithm (FDA) [24] and Crow Search Algorithm (CSA) [25]. These metaheuristic algorithm has own working principle. In this paper, we advanced PSO by using hypersphere and mutation approaches the performance of the proposed algorithm is compared with that of the given optimization algorithms for the solution of the gear train design problem.

Many metaheuristic optimization algorithms are not only being introduced as novel algorithms, but variants of these algorithms are also being improved. For the classical PSO algorithm, different approaches have been suggested in the literature to advance the algorithm. Duan et al. [26] propose the Chaos Adaptive Particle Swarm Optimization (CAPSO), employing the adaptive control over the inertia weight and the acceleration coefficients using chaotic theory. By using a chaotic search factor, CAPSO is the capable of the enhancing adaptability, supporting effective global and local search for the aim of the prevent the local

optima. Zhao and Wang [27] introduces the Elite-Ordinary Synergistic PSO (EOPSO) to solve the issue of the population diversity loss. EOPSO uses the particles as elite and ordinary groups, where the elite particles are utilized for the global exploration, while the other particles focus on local exploitation. Yang et al. [28] propose the Differential Elite Learning PSO (DELPSO), utilizing elite and non-elite group division to develop the diversity and adaptability. Non-elite particles learn from differential elite exemplars to improve search diversity, while dynamic parameter adjustments optimize the capability the exploration and the exploitation. Kassoil et al. [29] presents Exponential Particle Swarm Optimization (ExPSO), where the population includes three subgroups based on exponential-based search strategy that capables considerable jumps in the search space. The method ExPSO integrates the dynamic control over the particle velocity and a cognitive parameter that modifies over time, favoring large exploration leaps initially and focused improvements later. Solano-Rojas et al. [30] presents Micro Evolutionary Particle Swarm Optimization (MEPSO), which advances PSO by using the evolutionary mutations and crossovers instead of the classical velocity updates. In the MEPSO algorithm, these mutations and crossovers values give the probabilistic nature of the algorithm. Wang et al. [31] proposes Adam-LGQPSO, a method of the Quantum-Inspired Particle Swarm Optimization (QPSO) variant designed to reduce the premature convergence. This new method incorporates a Length of Potential Well (LPW) guiding strategy, a Gaussian random vector to improve information sharing, a perturbation structure to stop stagnation, and a diversity function to enhance adaptability. Zhu et al. [32] propose the Binary Restructuring Particle Swarm Optimization (BRPSO) algorithm, an adaptation of the classical Restructuring Particle Swarm Optimization (RPSO) for discrete optimization. Unlike other binary metaheuristics in the literature, BRPSO removes the use of a transfer function and it uses a random number for the particle updates. Moreover, it uses a novel perturbation term to advance the effectiveness of position updates. Bhargavi et al. [33] present a new PSO algorithm called Enhanced Particle Swarm Optimization (EPSO). In contrast to random deployments, the researches use EPSO to strategically position nodes, removing clustering and reducing coverage gaps. The researches also create individual swarms for each dimension, and the algorithm iteratively updates node positions, significantly reducing computational complexity compared to classical N-dimensional swarm methods in EPSO. Moreover, EPSO uses adaptive inertia weights and acceleration factors to fine-tune node placement, ensuring comprehensive monitoring of each target area. Gong et al. [34] propose a Quantum Particle Swarm Optimization (QPSO) algorithm that uses a diversity approach mechanism to improve the search performance in high-dimensional optimization problems. QPSO chooses migrating particles based on fitness and

population positions, replacing those that deviate from a central range, which in turn advances convergence and stability compared to classical optimization algorithms. Fusic and Sitharthan [35] propose Advanced Self-Adaptive Learning Particle Swarm Optimization (ISALPSO), modifing its parameters, like the inertia weight, the acceleration coefficients, the learning coefficients, the mutation factor, and the swarm size, in response to the effectiveness of the generated path. The ISALPSO algorithm has a self-adaptation strategy, differentiating it from the traditional PSO. Yang et al. [36] present a Leader-Adaptive Particle Swarm Optimization Algorithm with a Dimensionality Reduction Strategy (LAPSO-DR) that advances classical PSO by using a hybrid initialization strategy for diverse populations, a leader-adaptive strategy for advanced exploitation, and an inter-particle learning strategy, which provide different dimensions to learn from various particles. Hu et al. [37] propose a multiple Adaptive Co-Evolved Particle Swarm Optimization (ACEPSO) algorithm, incorporating a multiple adaptive co-evolved strategy, improving exploration via population grouping, advancing diversity with co-evolution, and using an adaptive mutation mechanism to prevent local optima. Liu et al. [38] introduce an Adaptive Particle Swarm Optimization Method that uses an Information Interaction (APSOIIM). APSOIIM advances the optimization capabilities of the traditional PSO by utilizing a chaotic sequence to greater distribute particles during initialization process and promoting diversity by interacting with the best solutions from neighboring particles during the search. APSOIIM algorithm also uses both a chaotic sequence strategy and an interaction information strategy. Ranganna et al. [39] propose a novel optimization algorithm, called Fitness Sharing Chaotic Particle Swarm Optimization (FSCPSO). FSCPSO uses the combination of chaos theory and fitness sharing mechanisms. Ambuj et al. [40] propose a new reinforcement learning particle swarm optimization (RLPSO) algorithm presents an innovative method that integrates the concepts of Particle Swarm Optimization (PSO) with reinforcement learning techniques. Tian et al. [41] propose a diversity-guided PSO with a multi-level learning strategy, combining a high-layer learning mechanism for global exploration and a low-layer scheme for local fine-tuning. Long et al. [42] improve a modified particle swarm optimization algorithm (RNP-PSO), utilizing various approaches such as particle encoding, initial population construction, and fitness calculation. Tang and Meng [43] presents an advanced particle swarm optimization algorithm (VASPSO), which includes the velocity pausing, a terminal replacement mechanism, time-varying inertia coefficients, and symmetric cooperative swarm concepts. Wang et al. [44] propose an improved Particle Swarm Optimization-Cubature Kalman Particle Filter (PSO-CPF) by using a particle filter, a forgetting factor and a new fitness function. Tantu and Biramo propose a novel variant of the classical PSO algorithm, called Adaptive PSO

(APSO), integrating the adaptive strategies that automatically tune algorithm parameters [65]. Cui and Seng proposed a new algorithm, Guaranteed PSO (GPSO), ensures the stable and the efficient convergence by dynamically adjusting particle velocities with a novel approach [66].

Research on developing the traditional PSO algorithm is studied rapidly in recent years and is not limited to the works mentioned in the previous paragraph. Numerous algorithms related to classical PSO can be found in the literature. In addition to the improvements to the PSO algorithm, the literature includes different engineering problems that have been solved using the PSO and its variants. The PSO algorithm is the capable of across various application areas, including engineering design optimization, such as the optimal gearbox layout design [45] and planetary gearbox optimization [46]. In addition to, various studies in the literature focus on PSO in the different application areas. Recent research shows that PSO is a method not only for improving its optimization capabilities but also for solving a wide range of engineering problems. This underscores the importance of our study as well for the solution of the engineering problems. In this study, a solution has been explored for the gear train design problem [47]. We get better results for the solution of the gear train design problem. Furthermore, as a future work, a new study could be conducted on solving other engineering problem in the literature by using the proposed new PSO algorithm called PSO-HDM from our research.

This work focuses on the high-dimensional optimization problems. The PSO-HDM algorithm is a novel technique that enables for getting more effective exploration and solution diversity in high-dimensional search spaces. Jones et al. have used hyperbox methods in their work, as one of the suggested high dimensional search approaches in the literature, for solving optimization problems [61].

The proposed PSO-HDM algorithm also includes the dynamic adjustment of the coefficients. The concept of dynamically tuning coefficients has already been studied in the literature for different optimization problems. Duran and Caginalp have introduced a hybrid forecasting optimization algorithm for parameter optimization that dynamically adjusts coefficients using two sliding windows to optimize parameter selection [62]. Effective parameter selection has been shown to significantly influence optimization performance in mathematical models, as demonstrated by Tuncel and Duran [63]. Inspired by such studies, our PSO-HDM algorithm utilizes the dynamic parameter selection to the improvement of the exploration and the convergence behavior.

Another additional strategy used in the PSO-HDM algorithm is the usage of the hypersphere- based technique. In this manner, the optimization process in the Hyper-Spherical Search (HSS) algorithm proposed by Karami et al. focuses on the exploring the inner space defined between hypersphere centers and particles [64]. In our study, the novelty of the PSO-HDM algorithm is found in its implementation of the hyperspheres with dynamic rotation, which allows a more exploration of the search space. This dynamic mechanism enhances the algorithm's ability to solve optimization problems. Briefly, unlike the HSS algorithm which explores the search within a defined subregion, PSO-HDM does not define local search regions. PSO-HDM ensures the convergence by rotating particles on a hypersphere without the need for the specifying any local search areas.

The structure of this study is as follows: Section 2 mentions the suggested novel optimization algorithm, called PSO-HDM, including a detailed explanation of how it works. Section 3 introduce the experimental studies conducted in this work. The first subsection of Section 3, the different types of benchmark functions are solved and the next subsection of Section 3 introduces the gear train design problem and the solution of some optimization algorithms compared to the PSO-HDM. Finally, the conclusion of this study is summarized in the last section, also with a discussion of future work.

2 MATERIAL AND METHOD

Particle Swarm Optimization (PSO) is an optimization algorithm, inspired by the cooperative navigating of the birds and the social manners of the fishes [6]. This optimization algorithm is known as a swarm intelligence metaheuristic, where the particles, collaborate and learn from another particle's experiences to find optimal solutions in the search spaces. To success this, it is important to have information on both the particles' velocity and position. For this aim, the velocity vector given by Equation 1 and the position vector given by Equation 2 are used.

$$\vec{v}_i(t+1) = w * \vec{v}_i(t+1) + c_1 * r_1 \left(\vec{p}_{best_{i_i}}(t) - \vec{x}_i(t)\right) + c_2 * r_2\left(\vec{g}_{best}(t) - \vec{x}_i(t)\right) \quad (1)$$

where $\vec{v}_i(t+1)$ denotes the velocity vector of particle i, w is the inertia weight, c_1 and c_2 are the cognitive and social coefficients, r_1 and r_2 are random numbers (usually in the range [0,1]), $\vec{p}_{best_i}$ is the personal best position of particle i, $\vec{g}_{best}$ is the global best position among all particles, and finally, $\vec{x}_i(t)$ is the current position of particle i.

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1) \quad (2)$$

where $\vec{x}_i(t+1)$ is the new position of particle i, $\vec{x}_i(t)$ is the current position of particle i and $\vec{v}_i(t+1)$ is the updated velocity vector of particle i.

As given Equation 2, instead of a linear position change in the PSO algorithm, the proposed new algorithm, called PSO-HDM, changes the position by constructing two different hyperspheres. Rotation is then occurred on one of the hyperspheres constructed. In addition, two different mutations, Jitter and Gaussian, have been applied. This approach allows exploration of different points in the search space. In the next section, we explain the proposed method in detail.

2.1 Particle Swarm Optimization with Hypersphere Dynamics and Mutation (PSO-HDM)

In this paper, we create a new method inspired PSO algorithm, called Particle Swarm Optimization with Hypersphere Dynamics and Mutation (PSO-HDM). At the beginning, when the population is initialized, each particle's $\vec{p}_{best_i}$ values are the same as their positions, in other words, the $\vec{x}_i(t)$ vector and $\vec{p}_{best_i}$ have identical values. Hence, the particles update their positions initially in the same way as in the traditional PSO algorithm. Afterwards, the hypersphere construction procedure is carried out. We explain the movement of the particle $\vec{x}_i(t)$ in a 2D space to explain the working rule of the PSO-HDM algorithm. As seen in Figure 1, one of the hyperspheres has its center at the $\vec{g}_{best}$ position, and its radius is the distance between the $\vec{g}_{best}$ and the current position vector $\vec{x}_i(t)$. The other hypersphere is centered at the $\vec{p}_{best_i}$ position, and its radius is the distance between the $\vec{p}_{best_i}$ and $\vec{x}_i(t)$. When the hyperspheres are created, several different scenarios can develop. These scenarios will be explained in detail. The case shown in Figure 1 illustrates a situation where the hyperspheres intersect concentrically at the point $\vec{x}_i(t)$.

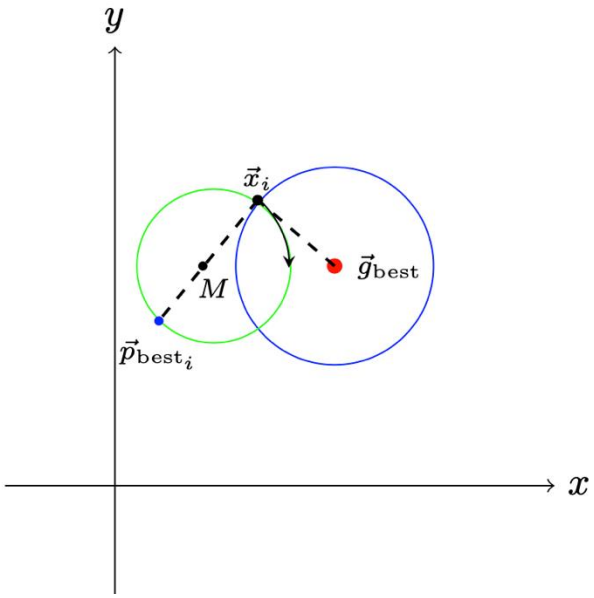

Figure 1. The movement of the position vector $\vec{x}_i(t)$ in the case of the overlap of the two inner circles.

In Figure 1, if $\vec{x}_i(t)$ performs a rotational motion on the green circle at a specific angle, as can be seen, $\vec{x}_i(t)$ will approach both the $\vec{p}_{best_i}$ value and the $\vec{g}_{best}$ value. Initially, the vector $\vec{x}_i(t)$ is given in cartesian coordinates as $(x_{i1}, x_{i2}, x_{i3}, \dots, x_{in})$, then it is transformed into spherical coordinates by using the radial distance ρ and the angular coordinates $(\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_{n-1})$, after which a rotation is applied by selecting randomized angle, such as α_1, and finally, the updated angular coordinates are converted back to cartesian coordinates, giving the new components $(x_{i1}, x_{i2}, x_{i3}, \dots, x_{in})$.

In order to provide a detailed explanation of the process outlined above, the position vector $\vec{x}_i(t)$ is defined in cartesian coordinates as $\vec{x}_i(t) = (x_{i1}, x_{i2}, x_{i3}, \dots, x_{in}) \in \mathbb{R}^n$. In order to determine the rotation angle, the system has been converted to angular coordinates as given in Equation 3.

$$
\begin{gathered}
x_{i1} = \rho \cos(\alpha_1) \\
x_{i2} = \rho \sin \alpha_1 \cos(\alpha_2) \\
\dots \\
x_{in-1} = \rho \sin \alpha_1 \sin \alpha_2 \dots \sin(\alpha_{n-2}) \cos(\alpha_{n-1}) \\
\text{and} \\
x_{in} = \rho \sin \alpha_1 \sin \alpha_2 \dots \sin(\alpha_{n-2}) \sin(\alpha_{n-1})
\end{gathered}
\tag{3}
$$

Hence, in angular coordinates $\vec{x}_i(t)$ can be expressed as $\vec{x}_i(t) = (\rho, \alpha_1, \alpha_2, \dots, \alpha_{n-1}) \in \mathbb{R}^n$. The axis of the rotation is selected randomly. In each iteration, a different axis is selected. The rotation occurs around the randomly chosen α_k axis. Thus, for $k \in \{1,2,\dots,n\}$, $\vec{x}_i(t) = (\rho, \hat{\alpha}_1, \hat{\alpha}_2, \dots, \hat{\alpha}_{k-1}, \alpha_k + \Delta\alpha, \hat{\alpha}_{k+1}, \dots, \hat{\alpha}_n)$ is obtained where $\Delta\alpha \in \left(0, \frac{\pi}{2}\right]$. Then, the position vector $\vec{x}_i(t) = (\hat{x}_{i1}, \hat{x}_{i2}, \hat{x}_{i3}, \dots, \hat{x}_{in}) \in \mathbb{R}^n$ is calculated in cartesian coordinates. After applying the inverse transformations, the angular coordinates of $\vec{x}_i(t)$ are given in Equation 4.

$$\begin{gathered}
\alpha_1 = \cot^{-1}\left(\frac{x_{i1}}{\sqrt{\sum_{j=i+1}^{n} x_j^2}}\right) \\
\alpha_2 = \cot^{-1}\left(\frac{x_{i2}}{\sqrt{\sum_{j=i+1}^{n} x_j^2}}\right) \\
\dots \\
\alpha_{n-2} = \cot^{-1}\left(\frac{x_{in-2}}{\sqrt{\sum_{j=i+1}^{n} x_j^2}}\right) \\
\text{and} \\
\alpha_{n-1} = 2\cot^{-1}\left(\frac{\sqrt{x_{n-1}^2 + x_n^2} + x_{n-1}}{x_n}\right)
\end{gathered} \tag{4}$$

Then, the cartesian coordinates of the position vector $\vec{x}_i(t)$ is recomputed using Equation 3. During the rotation, the selection of $\Delta\alpha$ is not crucial. Since the method is heuristic and a different α is selected in each iteration, the key point is that after the rotation, the current position vector converges to both the global best (gbest) and personal best (pbest) values. The order of the selection of α is also not crucial. While the choice of α may affect the convergence speed, it may not be possible to theoretically state this. This issue could be considered as a topic for further research.

To illustrate this randomization, we could present the situations where the yz, xy, and xz planes are randomly selected in three-dimensional space. If the situation shown in Figure 2 is randomly determined to be parallel to the yz-axis, we could say that when the $\vec{x}_i(t)$ position vector is rotated in both directions, it approaches both $\vec{p}_{best_i}$ and $\vec{g}_{best}$. Similarly, when the xy is randomly selected, as seen in Figure 3. Finally, xz is randomly selected, it can be observed that after the rotation of the $\vec{x}_i(t)$ position vector on the hypersphere, it goes both the $\vec{p}_{best_i}$ and $\vec{g}_{best}$ values.

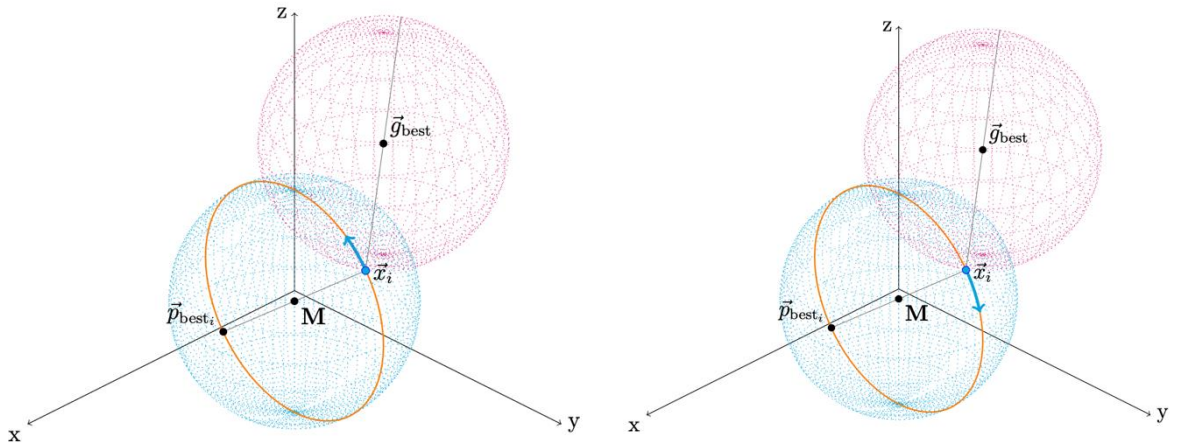

Figure 2. Rotation operation occurring when the yz-axis is randomly determined.

In the higher dimension, a similar situation occurs. After the random selection of the α_1, if the vector $\vec{x}_i(t)$ is rotated by Δα radians around the α_1-axis, it is moved to a new point such as $\vec{x}_i(t+1)$, where t is the iteration number in the angular coordinates.

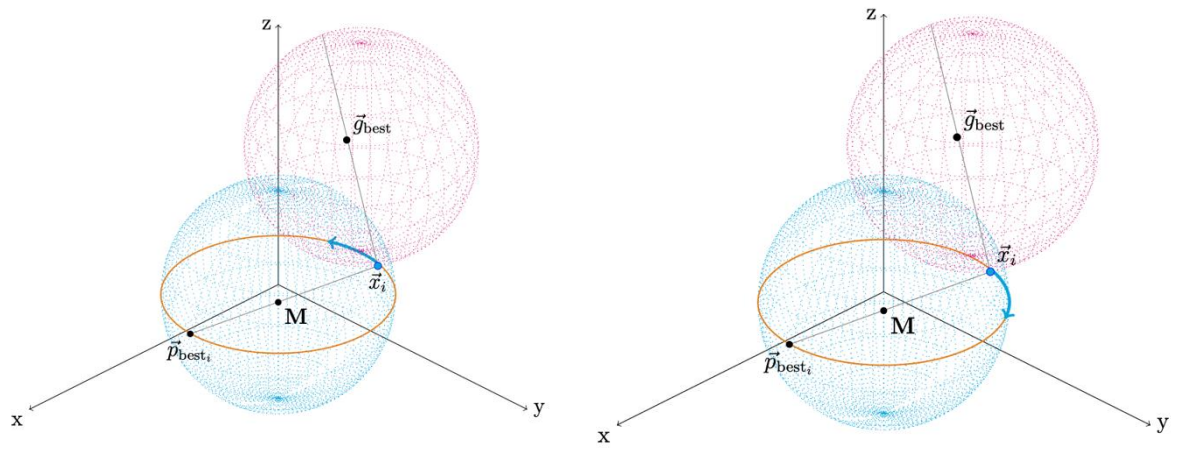

Figure 3. Rotation operation occurring when the xy-axis is randomly determined.

One of the another situation is confirming whether the $\vec{x}_i(t)$, $\vec{p}_{best_i}$, and $\vec{g}_{best}$ values are linear. If they are linear, the situations of being respectively internally and externally tangent in a d-dimensional space, as shown in Figure 4 and Figure 5, need to be evaluated. For this aim, the distance between $\vec{g}_{best}$ and $\vec{p}_{best_i}$, as well as the distance between $\vec{g}_{best}$ and $\vec{x}_i(t)$, need to be compared. When the condition given by Equation 5 is occurred, it is seen that the circles are internally tangent as shown in Figure 3. If the points are linear and internally tangent, the $\vec{x}_i(t)$ point is the point of intersection, and the rotation angle and direction is not important. Even

with a very small rotation of the $\vec{x}_i(t)$ position vector, it will approach both the $\vec{p}_{best_i}$, and $\vec{g}_{best}$. This situation suggests that the particle $\vec{x}_i(t)$ is being moved to both the personal best position and global best positions during its rotation.

$$\left\|\vec{g}_{best}, \vec{p}_{best_i}\right\| < \|\vec{g}_{best}, \vec{x}_i(t)\| \tag{5}$$

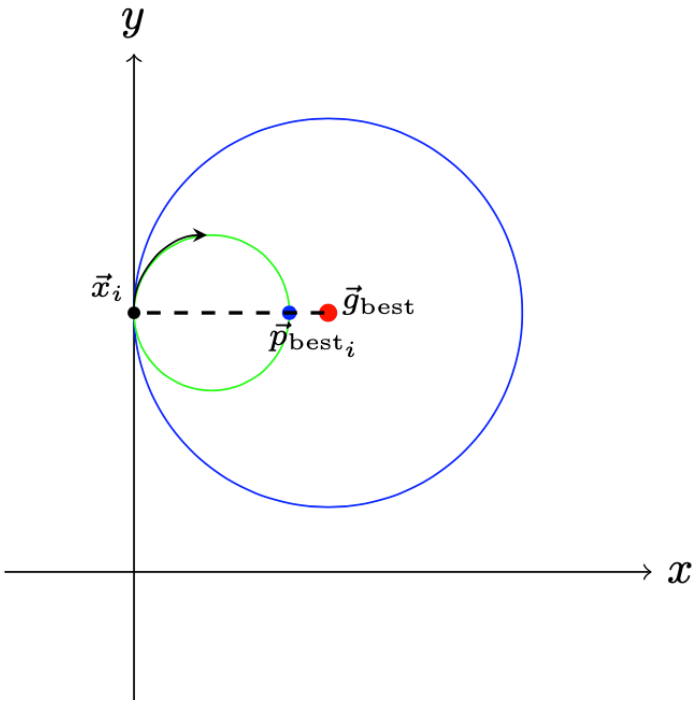

Figure 4. The movement of the position vector $\vec{x}_i(t)$ in the case of the $\vec{x}_i(t)$, $pbest_i$, and $gbest$ values are linear the circles are internally tangent.

Alternatively, in the case where the internally condition given by Equation 5 is not occurred, as shown in Figure 4, the circles are externally tangent. In this case, the situation described by Equation 6 occurs.

$$\left\|\vec{g}_{best}, \vec{p}_{best_i}\right\| > \|\vec{g}_{best}, \vec{x}_i(t)\| \tag{6}$$

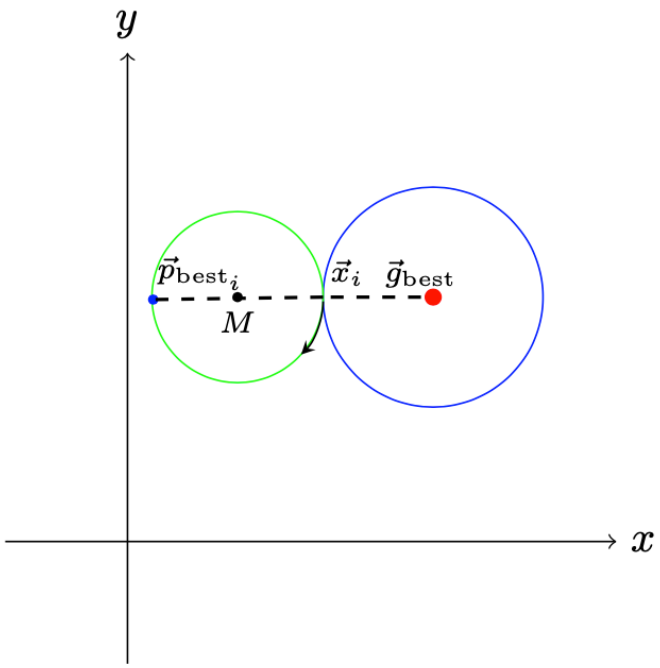

Figure 5. The movement of the position vector $\vec{x}_i(t)$ in the case of the $\vec{x}_i(t)$, $\vec{p}_{best_i}$, and $\vec{g}_{best}$ values are linear the circles are externally tangent. In this case, classical PSO is used without the rotation operation.

As seen in Figure 5, when the $\vec{x}_i(t)$ position vector rotates in any direction on the circle, it goes away from the $\vec{g}_{best}$. To tackle this, the traditional PSO algorithm has been used in the case where the circles are externally tangent as seen in Figure 5.

A modification added to the PSO in the PSO-HDM algorithm is the dynamic of coefficients. For this aim, the coefficients c_1 and c_2 are determined as iteration-dependent variables in the algorithm. In the literature, the coefficients $c_1 = 2$ and $c_2 = 2$ are commonly selected and are known to provide valid results [37]. However, some other values have also been used. For instance, some researchers use the lower coefficients, such as c_1=1.05 and c_2=1.05, or slightly higher values, like c_1=2.05 and c_2=2.05, to advance the performance of the algorithm [38]. In this work, the values of c_1 and c_2 are dynamically adjusted. The c_1 value is initially determined as 1.5 and reduces to 1.2, while the c_2 value starts at 2.0 and decreases to 1.5. The values of c_1 and c_2 are selected based on the current iteration t and the maximum iteration T, using Equations 7 and 8, respectively. These equations help adjust the coefficients gradually in the optimization.

$$c_1 = 1.2 + 0.3\frac{t}{T} \tag{7}$$

$$c_2 = 1.5 + 0.5\frac{t}{T} \tag{8}$$

In the mutation stage, the noise is added to decide the positions for the algorithm at distinct points in the search space [48-51]. The mutation operators, also known as noise, presented as by Jitter, given by Equation 9, and Gaussian, given by Equation 10, respectively.

$$\mu = (\vec{r} - 0.5)\mu_F \tag{9}$$

where $\mu_F = 0.05 + 0.45\left(1 - \frac{t}{T}\right)$ is a Jitter factor and $\vec{r}$ is in in the range $[0,1]^d$ such that d is the dimension of the problem.

$$\lambda = 0.1\vec{g} \tag{10}$$

where $\vec{g}$ is in in the range $[0,1]^d$ such that d is the dimension of the problem.

When the Jitter mutation is implemented, the noise μ given by Equation 8 is inserted to the position vector. Similarly, when Gaussian noise is implemented, the noise value λ given by Equation 9 is added to the position vector. The equations for Jitter and Gaussian noise are given with Equation 11 and Equation 12, respectively.

$$\vec{X} = \vec{x}_i(t) + \mu \tag{11}$$

$$\vec{X} = \vec{x}_i(t) + \lambda \tag{12}$$

In Equation 11 and in Equation 12, $\vec{X}$ represents the new position of the $\vec{x}_i(t)$ after inserting noise.

The mutation equations given by Equation 9 and Equation 10 are implemented the position vector to obtain a new position. If the solution at this new possible position is greater, it is selected as the new solution. Thus, the solutions are explored at different points in the search space. The probability of mutation is set to 50%. For instance, if the number of iterations is 1000, the mutation is applied in around 500 iterations. With a 50% probability, the mutation process generates 1000 new candidate solutions for each particle. The best solution within these candidate solutions, if it advances possible upon the current one, is selected and updated as the new solution.

The pseudocode of the PSO-HDM (Particle Swarm Optimization with Hypersphere Dynamics and Mutation) algorithm is provided in Algorithm 1.

Algorithm 1. The pseudocode of the PSO-HDM.

1. Initialize the arbitrary parameters of PSO as the population size, the lower and the upper of the decision variables, the maximum iteration, the population szie, the particel positions.
2. Update the velocity of the particle using Eqution 1.
3. Update the position of the particle using Eqution 2.
4. Evaluate the fitness values of particle $\vec{x}_i(t)$
5. For each particle: Update the personal best ($\vec{p}_{best_i}$) and the global best ($\vec{g}_{best}$).
    1. Repeat the stopping criteria has been reached
    2. Update the acceleration coefficients (c_1, c_2) dynamically.
    3. For each particle:
    4. If the position unchanged: update $\vec{v}_i$ apply limits.
    5. Else: determine the center and radius of the hyperspheres and construct two hyperspheres which overlapped at the point $\vec{x}_i(t)$.
    6. Determine the angular of the rotation as $\vec{\alpha}$
    7. If the positions of the particle $\vec{x}_i(t)$, the personal best ($\vec{p}_{best_i}$) and the global best ($\vec{g}_{best}$) are collinear: update the velocity of the particle using Equation 1.
    8. Else: adjust position with angular derivatives, apply limits.
    9. Evaluate $f(\vec{x}_i)$, update $pbest_i$, gbest.
    10. Apply Gaussian or Jitter mutation if enabled.
    11. Update the value of inertia coefficients via a predetermined damping ratio
    12. Until the stopping criteria has been reached.
6. Return the position of the global ($\vec{g}_{best}$) and the fitness value at the global best position.

In the following section, the effectiveness of the novel proposed algorithm has been executed by comparing its solutions for two distinct types of benchmark functions, including unimodal and multimodal functions, with those of the classical PSO. In the following subsection, the solutions for a gear train design problem, a well known engineering design problem in the literature, is analyzed and compared some other optimization algorithms and the proposed PSO-HDM.

3 EXPERIMENTAL RESULTS AND COMPARISONS

As shown in the previous works in the literature, in order to get the performance of the proposed novel algorithm, both benchmark functions and engineering design problem solutions are generally used. The arbitrary parameter values is selected 100 for the number of population and also 100 maximum the number of iterations for the solution of all benchmark functions. The codes are executed 25 times.

3.1 Benchmark Functions

In this section, benchmark function solutions is initially used to compare the performance of the algorithms. Solutions have been obtained for two categories of classic benchmark functions, includes unimodal and multimodal functions as seen in Table 1 and Table 2 [52].

Table 1. Unimodal benchmark functions.

Function	Dim	Limits	f_{min}
$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	[-100,100]	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[-10,10]	0
$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j \right)^2$	30	[-100,100]	0
$f_4(x) = max_i\{\lvert x_i \rvert, 1 \leq i \leq n\}$	30	[-100,100]	0
$f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_{i-1})^2]$	30	[-30,30]	0
$f_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	30	[-100,100]	0
$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1)$	30	[-1,28,1.28]	0

Table 2. Multimodal benchmark functions.

Function	Dim	Limits	f_{min}
$f_8(x) = \sum_{i=1}^{n} [x_i{}^2 - 10\cos(2\pi x_i) + 10]$	30	[-5.12, 5.12]	0
$f_9(x) = -20\exp\left(\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i{}^2}\right) - \exp\left(\frac{1}{n}\sqrt{\sum_{i=1}^{n}\cos(2\pi x_i)}\right) + 20 + e$	30	[-32,32]	0
$f_{10}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i{}^2 - \prod_{i=1}^{n}\left(\cos\left(\frac{x_i}{\sqrt{i}}\right)\right) + 1$	30	[-600,600]	0
$f_{11}(x) = \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin 2(\pi y_{i+1})] + (y_n - 1)^2]\}$ $y_i = 1 + \frac{x_i+1}{4}, u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	30	[-50,50]	0
$f_{12}(x) = 0.1\{\sin_2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + \sin_2(3\pi x_1 + 1)] + (x_n - 1)^2[1 + \sin_2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	[-50,50]	0

The benchmark functions is solved using both PSO and PSO-HDM. The results are shown in Table 3, including the best scores, worst scores, mean scores, and their standard deviations obtained over 20 runs. Instead of relying on results obtained from a single run, evaluations were performed based on the outcomes of 20 runs. This approach ensures the reliability of the method can be assessed. As seen in Table 3, the PSO-HDM algorithm gets better solutions compared to the traditional PSO in solving benchmark functions. It has been observed that the PSO-HDM algorithm shows better results in terms of the best cost, the worst cost, and the average best cost values.

Table 3. The results of benchmark function solutions.

Function		PSO	PSO-HDM
F1	Best cost	1.9260	$1.0680e-02$
	Worst cost	$1.1273e+01$	$7.5747e-02$
	Mean of best costs	5.8350	$3.5353e-02$
	Standart deviation	2.2990	$1.9729e-02$
F2	Best cost	$4.5419e-01$	$5.4673e-02$
	Worst cost	1.261758	$1.1622e-01$
	Mean of best costs	$8.0060e-01$	$8.2368e-02$
	Standart deviation	$2.2200e-01$	$2.1008e-02$
F3	Best cost	$5.3871e+02$	$5.7507e+01$
	Worst cost	$1.9077e+03$	$1.3852e+03$
	Mean of best costs	$1.0171e+03$	3.8099e+02
	Standart deviation	$3.6656e+02$	2.8227e+02
F4	Best cost	3.5317	1.5223
	Worst cost	6.3676	5.6225
	Mean of best costs	4.9234	3.3375
	Standart deviation	$8.4048e-01$	1.1845
F5	Best cost	$7.0853e+01$	$2.4930e+01$
	Worst cost	$8.0734e+02$	$1.8769e+02$
	Mean of best costs	$2.4703e+02$	$6.2656e+01$
	Standart deviation	$1.7541e+02$	$4.7316e+01$
F6	Best cost	1.9806	$1.4106e-02$
	Worst cost	$1.5122e+01$	$1.0168e-01$
	Mean of best costs	5.1202	$4.4010e-02$
	Standart deviation	2.7843	$2.4874e-02$

Table 3. (continued). The results of benchmark function solutions.

Function		PSO	PSO-HDM
F7	Best cost	$1.2079e-02$	$3.7678e-03$
	Worst cost	$6.2254e-02$	$1.9575e-02$
	Mean of best costs	$2.8147e-02$	$1.2222e-02$
	Standart deviation	1.2069	$3.6393e-03$
F8	Best cost	$2.8477e+01$	$2.5785e+01$
	Worst cost	$7.5196e+01$	$6.0864e+01$
	Mean of best costs	$5.3386e+01$	$3.9440e+01$
	Standart deviation	$1.3798e+01$	$1.0622e+01$
F9	Best cost	$9.4141e-01$	$6.2463e-02$
	Worst cost	2.3180	2.3288
	Mean of best costs	1.5880	1.4219
	Standart deviation	$4.0456e-01$	$6.9459e-01$
F10	Best cost	$8.0581e-01$	$1.5818e-01$
	Worst cost	1.0859	$5.575824e-01$
	Mean of best costs	1.0354	$2.8687e-01$
	Standart deviation	$5.8554e-02$	$1.1308e-01$
F11	Best cost	$3.9844e-03$	$1.4814e-04$
	Worst cost	$4.5164e-01$	1.3024
	Mean of best costs	$1.5542e-01$	$1.3819e-01$
	Standart deviation	$1.4207e-01$	$2.9692e-01$
F12	Best cost	$3.1564e-01$	$2.5100e-03$
	Worst cost	3.1483	$2.535093e-02$
	Mean of best costs	$9.0645e-01$	$9.8414e-03$
	Standart deviation	$6.3325e-01$	$6.4765e-03$
F13	Best cost	$3.0917e-04$	$3.0768e-04$
	Worst cost	$1.4002e-03$	$1.2231e-03$
	Mean of best costs	$5.2679e-04$	$3.9998e-04$
	Standart deviation	$3.0932e-04$	$2.8152e-04$

Additionally, Figure 6 shows the benchmark functions with their best cost values and showing a comparison of the solutions PSO and PSO-HDM.

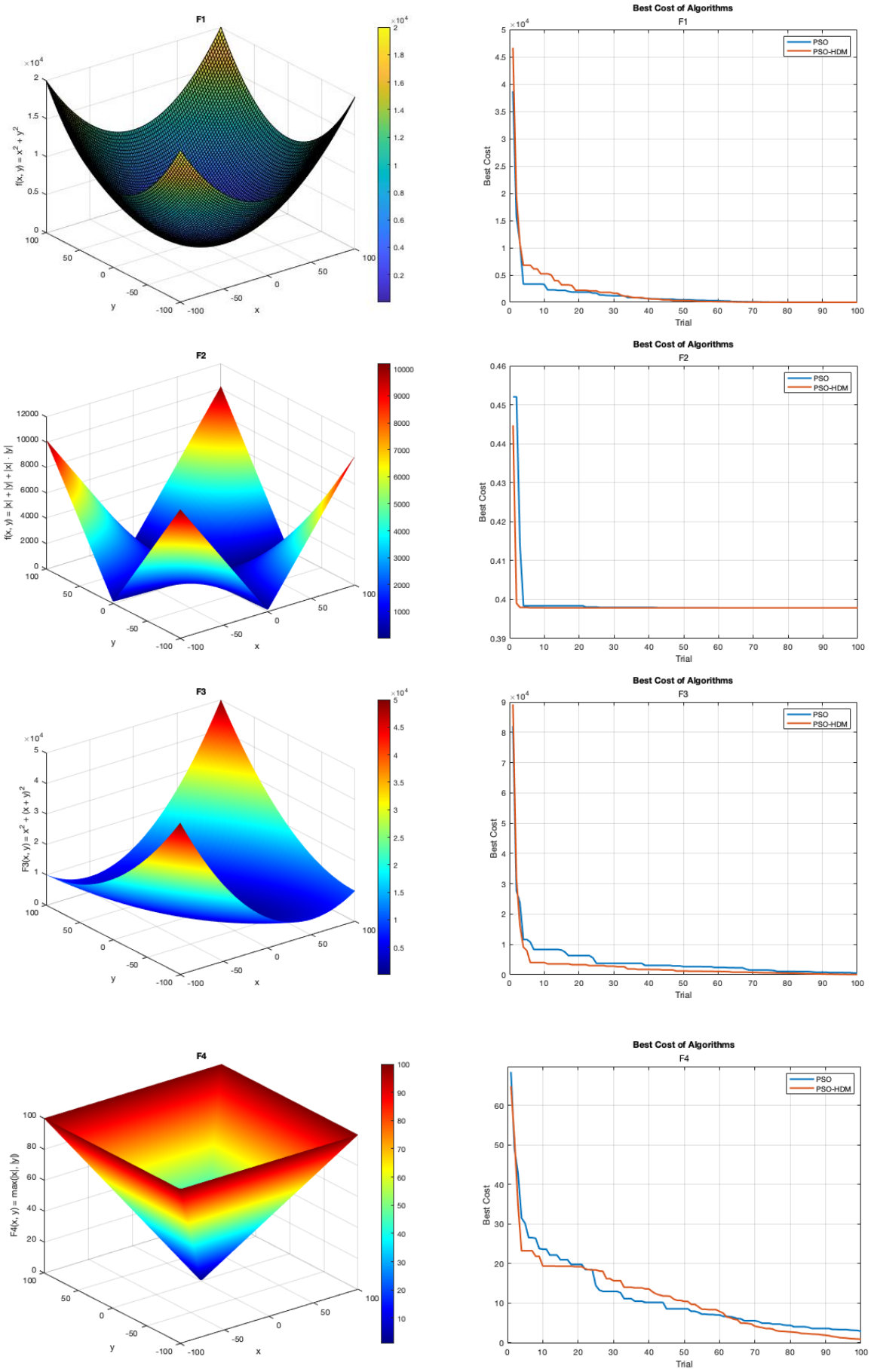

Figure 6. The comparison of the best cost values of benchmark function solutions.

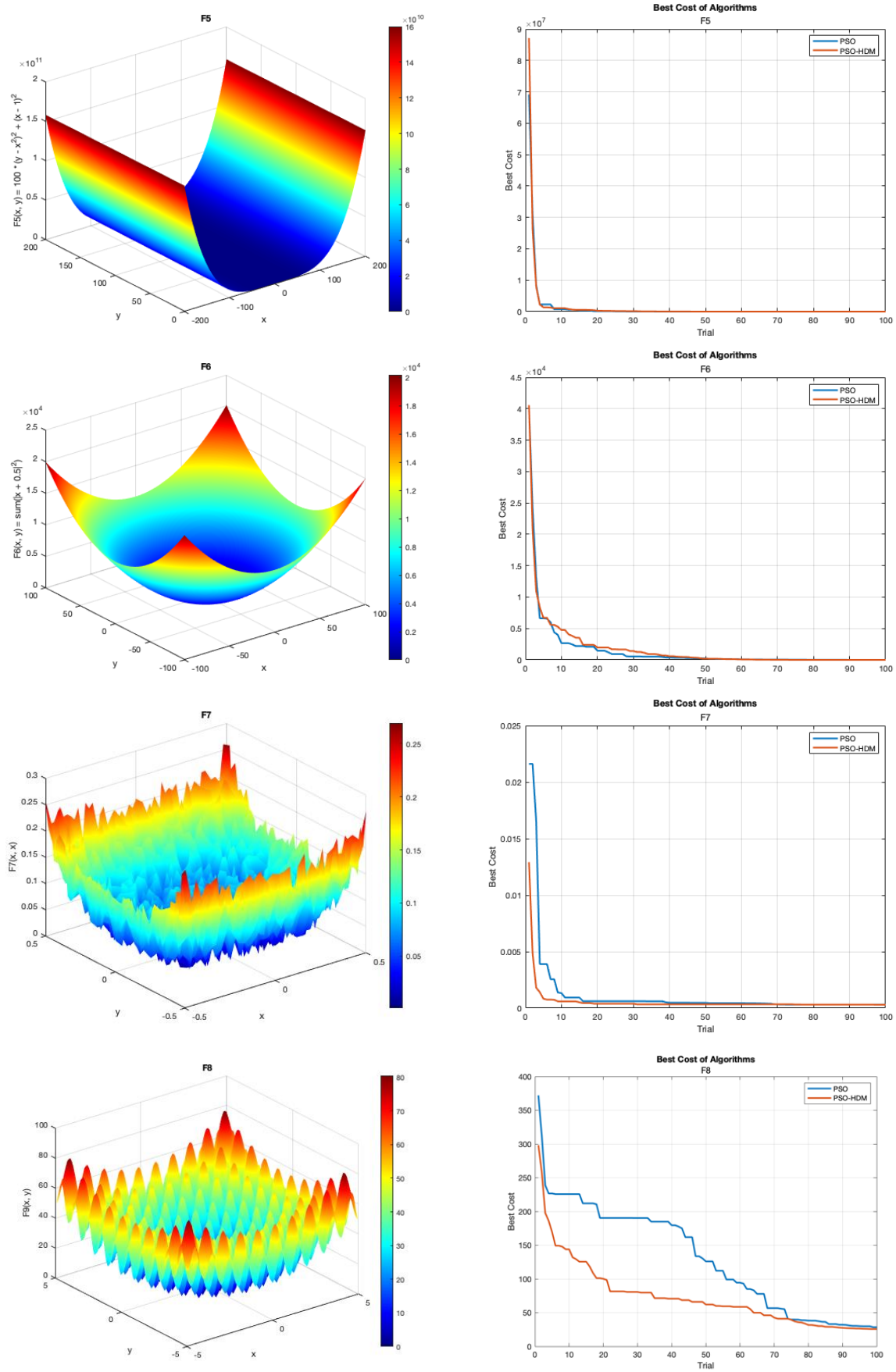

Figure 6. (continued). The comparison of the best cost values of benchmark function solutions.

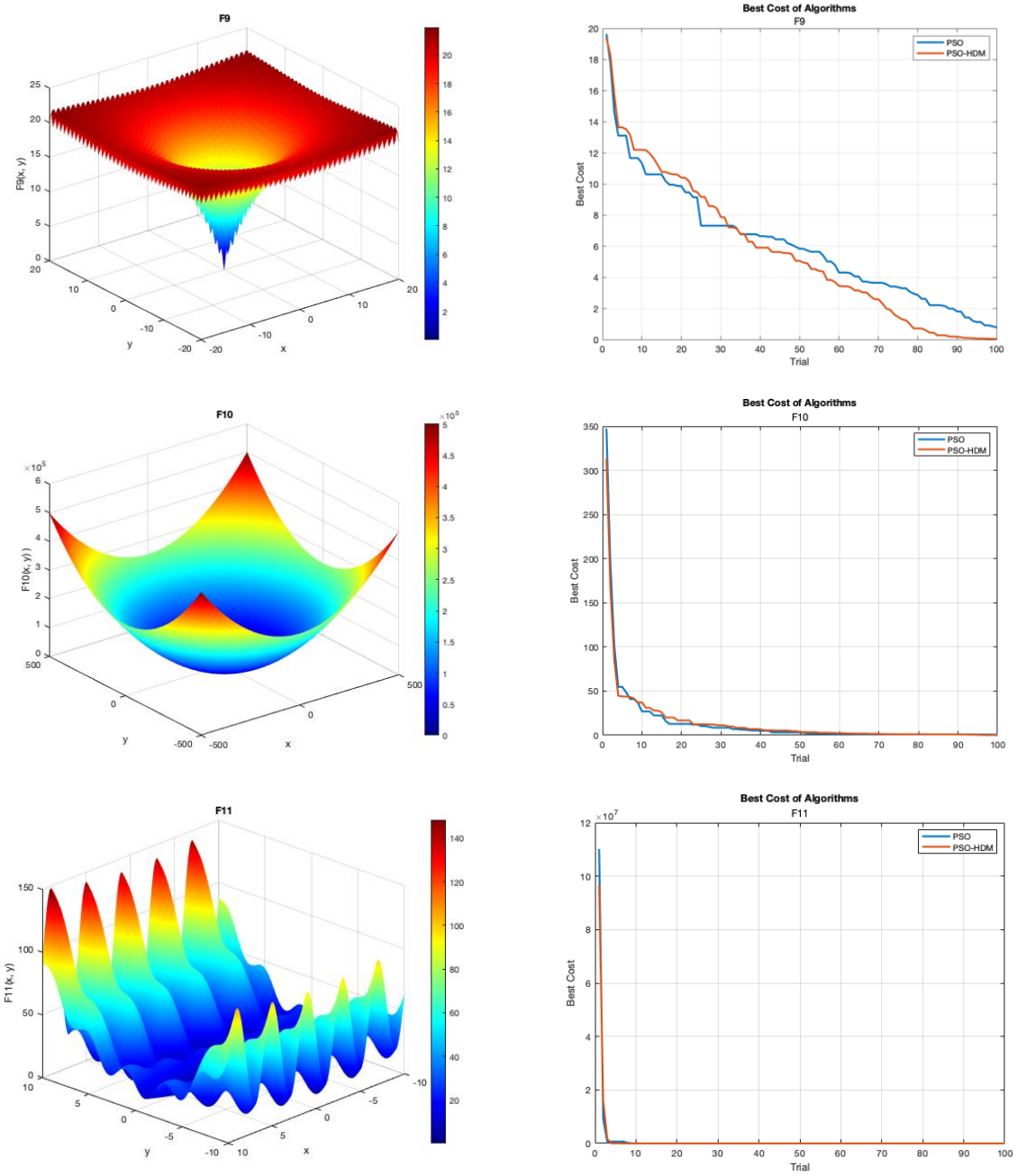

Figure 6. (continued). The comparison of the best cost values of benchmark function solutions.

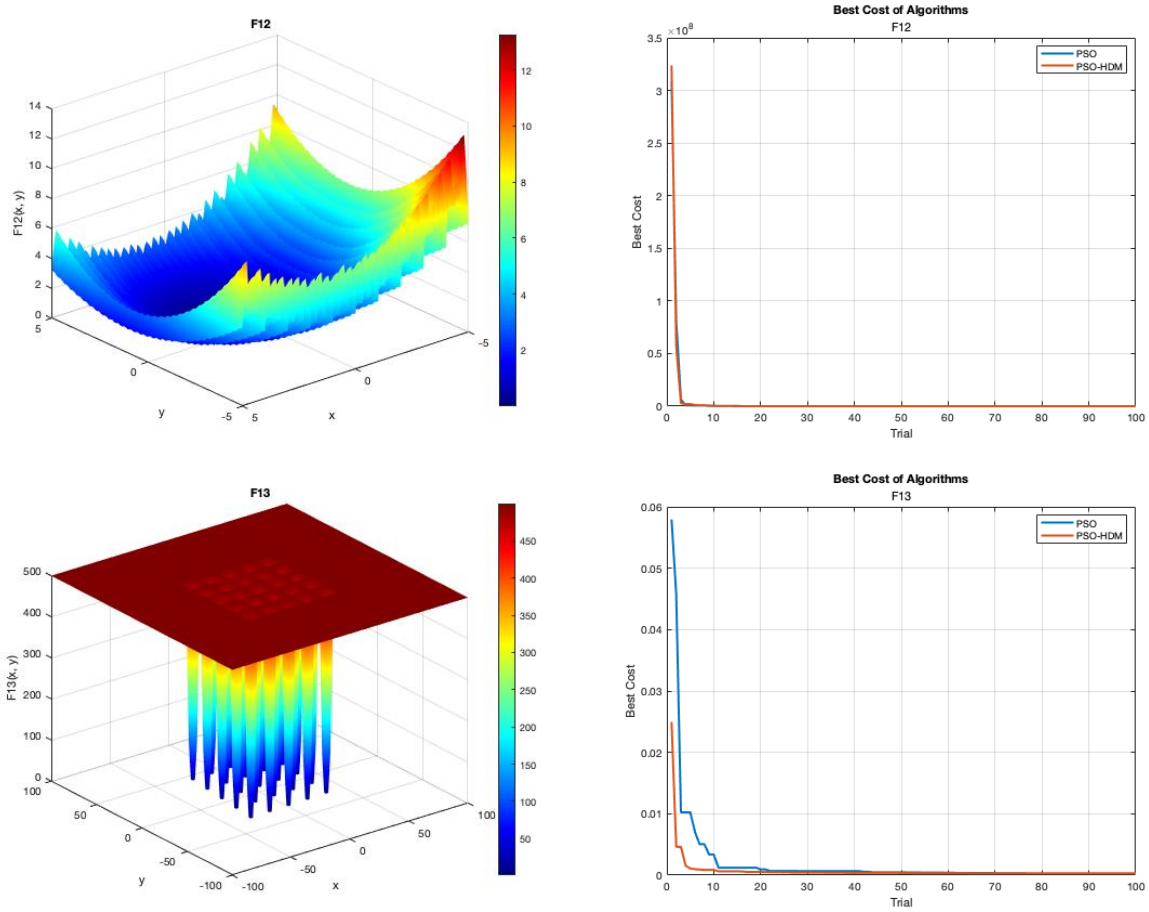

Figure 6. (continued). The comparison of the best cost values of benchmark function solutions.

Furthermore, Table 4 shows the execution time of the optimization algorithm. In Table 4, T_0 shows the base execution time, T_1 is the mean execution time across all trials and $T_2 = \frac{(T_1 - \min(T))}{T_0}$ quantifies the normalized time variation. In Table 4, it is showed that the PSO-HDM algorithm runs slightly slower due to the inclusion of mutation stage and hypersphere generation steps. On the other hand, the results show an improvement over the PSO algorithm. Additionally, while the PSO-HDM algorithm might be slower, if it is success for the notable improvements in solving significant engineering problems, as mentioned in the introduction section, the additional execution time could be considered negligible. This makes the algorithm particularly valuable for tackling complex challenges in engineering optimization.

Table 4. The execution time of the methods for solving benchmark functions.

Function	Values	PSO	PSO-HDM
F1	T_0	$3.5087e-02$	$3.5087e-02$
	T_1	$5.4613e-01$	$2.2520e+01$
	T_2	2.0067	$2.9340e+01$
F2	T_0	$3.2064e-02$	$3.2064e-02$
	T_1	$5.3660e-01$	$2.5033e+01$
	T_2	1.5110	$5.0999e+01$
F3	T_0	$1.5580e-02$	$1.5580e-02$
	T_1	$6.8455e-01$	$5.7121e+01$
	T_2	9.1162	$1.9326e+02$
F4	T_0	$1.5326e-02$	$1.5326e-02$
	T_1	1.0827	$2.7865e+01$
	T_2	$4.7140e+01$	$1.3748e+02$
F5	T_0	$1.333480e-02$	$1.3334e-02$
	T_1	$5.5662e-01$	$2.6942e+01$
	T_2	6.8747	$1.6054e+02$
F6	T_0	$1.5346e-02$	$1.5346e-02$
	T_1	$4.2772e-01$	$2.3320e+01$
	T_2	4.9084	$9.8602e+01$
F7	T_0	$1.7332e-02$	$1.7332e-02$
	T_1	$5.6147e-01$	$5.1075e+01$
	T_2	7.0861	$1.0932e+02$
F8	T_0	$4.7453e-02$	$4.7453e-02$
	T_1	1.5013	$4.0612e+01$
	T_2	$1.8680e+01$	$1.2081e+02$
F9	T_0	$2.5330e-02$	$2.5330e-02$
	T_1	$8.2700e-01$	$2.7007e+01$
	T_2	$1.2866e+01$	$4.0047e+01$
F10	T_0	$1.3467e-02$	$1.3467e-02$
	T_1	$9.7158e-01$	$2.8222e+01$
	T_2	$3.7505e+01$	$9.7938e+01$
	T_0	$3.5087e-02$	$3.5087e-02$
F11	T_0	$1.6090e-02$	$1.6090e-02$
	T_1	$9.9783e-01$	$5.3426e+01$
	T_2	$1.7967e+01$	$3.2028e+02$
F12	T_0	$3.1564e-01$	$2.5100e-03$
	T_1	3.1483	$2.5350e-02$
	T_2	4.5949	$1.0017e+02$
F13	T_0	$1.5223e-02$	$1.5223e-02$
	T_1	$4.0601e-01$	$1.1586e+01$
	T_2	3.1791	$3.7192e+01$

In the following section, the gear train design problem will be analyzed in detail. The next section also contains the solution of this problem with and the original PSO and the proped new method PSO-HDM.

3.2 Gear Train Design Problem

The gear train optimization design problem, a well-known problem in mechanical engineering introduced by Sandgren [47] incorporates designing a gear train to decrease the input angular speed to a lower output speed, with the objective of minimizing the gear ratio, defined as the angular velocity ratio of the output shaft to the input shaft. The variables of the problem for this task involve the number of teeth on the gears A, B, C, and D, as demonstrated in Figure 10. The angular velocity ratio is shown seen in Equation 13, n is also known as two-gear transfer ratio.

$$n = \frac{w_0}{w_i} = \frac{t_i}{t_0} \tag{13}$$

In Equation 12, w_0 denotes the angular velocity of the output gear, while w_i is the angular velocity of the input gear. In the same way, t_0 and t_i correspond to the number of teeth on the output and input gears. Consequently, the transmission ratio is inversely related to the number of teeth on the gears. In Figure 7, the problem analyzed in this study, including two pairs of gears (a total of four gears) and seeks to achieve a transmission ratio as close as possible to 1/6.931. As a results, for this specific problem, the transmission Equation 13 can be reformulated as following Equation 14.

$$n = \frac{1}{6.931} = \frac{t_C t_B}{t_A t_D} = \frac{x_1 x_2}{x_3 x_4} \tag{14}$$

Sandgren [47] present that no gear in the system should have fewer than 12 teeth or more than 60 teeth. As a consequence, the gear train design problem includes choosing a set of gears (x_1, x_2, x_3 *and* x_4) such that the double reduction gear ratio is as close as possible to $\frac{1}{6.931}$ while satisfying the feasibility constraints. In particular, each design variable x_i must be an integer within the range [12, 60]. The gear train design problem can be expressed as following Equation 15:

$$\min f(x) = \left[\frac{1}{6.931} - \frac{x_1 x_2}{x_3 x_4}\right]^2 \qquad 12 \le x_i \le 60 \;\; i = 1,2,3,4 \tag{15}$$

The arbitrary parameters utilized in the experimental studies are presented in Table 5. This work uses two different iterations, such as 200 and 1000. Because the worst, the average, the best, and the standard deviation values obtained with 200 iterations are provided by Karami et al [24]. Additionally, the results of the $f(x)$ values obtained using x_1, x_2, x_3, and x_4 can be compared with those in Gopi [7], Duan et al. [13], and Mirjalili [14]. In these researches, the maximum number of iterations is set to 1000. Consequently, the problem was also solved with 1000 iterations to enable a fair comparison. The comparisons are demonstrated accordingly in Table 6 and Table 7, considering different maximum numbers of iterations. On the other hand, the other arbitrary parameters remain the same for both iterations.

Table 5. The arbitrary parameters values for the gear train design problem.

Parameters	Values
Number of Population	100
Number of Decision Variables	4
Lower Bound of Variables	12
Upper Bound of Variables	60
Maximum Number of Iterations	200, 1000

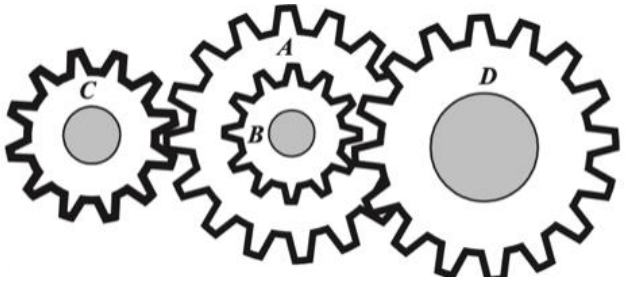

Figure 7. The Gear train design problem [14].

Figure 7 depicts, the number of teeth on gear A is represented by x_1, the number of teeth on gear B is represented by x_2, the number of teeth on gear C is represented by x_3, and the number of teeth on gear D is represented by x_4.

Table 6. The comparison results of the gear train problem with 1000 iteration.

Algorithms	x_1	x_2	x_3	x_4	$f(\vec{x})$
MGWO [7]	43.23451442	19.48161573	16.80898646	52.49664159	$9.444e-15$
TSA [8]	41.38908813	14.89380456	22.81969856	56.91350992	$1.153e-11$
SSA [9]	50.80890822	14.00937474	29.37997485	42.34395302	0.0022119
MVO [10]	30.42669549	15.51261593	12	42.40446052	$1.896e-12$
GWO [11]	60	30.57637562	15.04813329	53.1507917	$1.386e-12$
IGWO [12]	59.38921756	13.10291292	12.58277378	19.24141193	$2.135e-12$
ALO [14]	49	19	16	43	2.7009e − 12
CS [15]	43	16	19	49	2.7009e − 12
MBA [16]	43	16	19	49	2.7009e − 12
ISA [17]	N/A	N/A	N/A	N/A	2.7009e − 12
GA [18]	N/A	N/A	N/A	N/A	2.33e − 17
ABC [19]	19	16	44	49	2.78e − 11
GA [20]	33	14	17	50	1.362e − 09
ALM [21]	33	15	13	41	2.1469e − 08
APSO [65]	12	43.28379750	60	60	$6.7251e-20$
GCPSO [66]	15.37424096	24.26328913	59.24068481	43.64345879	$6.4739e-17$
CAPSO [26]	28.49091524	16.01777144	53.13400652	50.34126059	$6.9342e-04$
ExPSO [29]	16.75900322	14.97024322	43.58594282	41.73902679	$4.0598e-05$
PSO-HDM	12.75505496	34.87566866	56.02420707	55.03323691	$1.6366e-21$

Table 7. The comparison results of the gear train problem with 200 iteration.

Algorithms	**Worst**	**Average**	**Best**	**Standart deviation**
GPSO [53]	$2.7009e-12$	$5.5781e-10$	$2.7009e-12$	$8.7412e-10$
GPSO [54]	$2.7009e-12$	$9.0259e-10$	$1.1661e-10$	$1.9240e-09$
GPSO [55]	$2.7009e-12$	$1.9522e-10$	$8.8876e-10$	$4.1787e-10$
GPSO-PG [56]	$2.7009e-12$	$9.3414e-09$	$3.3667e-08$	$1.6663e-08$
OPSO [57]	$2.3078e-11$	$9.5187e-07$	$2.7265e-08$	$1.5744e-06$
GPSO [58]	$2.7009e-12$	$5.4215e-10$	$2.7009e-12$	$9.1924e-10$
DOA [23]	$2.7009e-12$	$3.6887e-10$	$1.1661e-10$	$7.2821e-10$
WOA [59]	$2.7009e-12$	$4.1501e-10$	$2.3078e-11$	$7.5615e-10$
GWO [11]	$2.7009e-12$	$4.5905e-10$	$2.3078e-11$	$1.2605e-09$
GPSO [60]	$2.3078e-11$	$1.8633e-07$	$2.3576e-09$	$3.5199e-07$
GPSO [13]	$2.7009e-12$	$5.1431e-11$	$2.7009e-12$	$8.8063e-11$
ABC [19]	N/A	$3.64e-10$	$2.70e-12$	$5.52e-10$
MBA [16]	$2.06e-08$	$2.47e-09$	$2.70e-12$	$3.94e-09$
CSA [25]	$3.18e-08$	$2.06e-09$	$2.70e-12$	$5.06e-09$
FDA [24]	$3.2999e-09$	$7.5614e-10$	$2.700857e-12$	$8.0465e-10$
APSO [65]	$7.910450e-12$	$6.791322e-13$	$3.631801e-18$	$1.923245e-12$
GCPSO [66]	$1.695473e-12$	$1.608118e-13$	$1.519431e-21$	$4.430094e-13$
CAPSO [26]	$1.471239e-01$	$7.657031e-03$	$2.859642e-06$	$2.913283e-02$
ExPSO [29]	$1.111886e-02$	$9.114819e-04$	$6.626667\ e-08$	$2.279462e-03$
PSO-HDM	$3.5723e-17$	$5.0208e-18$	$6.7296e-23$	$8.4829e-18$

In Table 6, it can be seen that the best function values obtained based on x_1, x_2, x_3, and x_4 values are achieved using the proposed PSO-HDM algorithm. Additionally, it is theoretically challenging to determine the impact of the random rotation on the performance of the PSO-HDM algorithm. However, since the method's approach toward the global best is nonlinear, the random rotation operation has helped the algorithm in preventing it from becoming restricted at the local minima. When the position vector and the fitness value, $f(x)$, at the point it represents in Table 6 are analyzed, it can be observed that x_1, x_2, x_3, and x_4 are close to distinct locations that can be considered local minimizers. Therefore, it can be observed that the algorithms, except for PSO-HDM, are restricted at the local minima. The performance improvement provided by PSO-HDM can be clearly seen in Table 6. The solution vector of the PSO-HDM algorithm is $\vec{x}_i(t) = (12.7551,34.8757,56.024255.0332)$. When the solution vectors of the other algorithms are examined, it could be observed that the nearest result is obtained by the ABC algorithm in terms of Euclidean distance. Given that the solution vector of the ABC algorithm is $\vec{x}_i(t) = (19,16,44,49)$, it could be interpreted that the other algorithms are restricted at the local minima. Among all optimization algorithms in Table 6, ABC is observed to be the algorithm that reaches the point closest to the solution found by PSO-HDM. In addition, in Table 7, the worst, average, best, and standard deviation values, as well as the best results, are get with the PSO-HDM algorithm.

For the purpose of the enhancing the exploitation capability of the algorithm and prevent it from getting stuck in local minima, the population diversity should be increased. For this reason, mutation operators are introduced in our paper. In addition to the methods used in our study, other mutation operators could also be used for this aim. In our study, Jitter and Gaussian mutations were selected for the mutation operation.

As an example, we solve Gear Train Design Problem with PSO-HDM algorithm without mutation operator. The results are presented in Table 8. Table 8 demonstrates that the mutation operator has a positive efect on the performance of the PSO-HDM algorithm. Therefore, for the future work, the mutation operator could be utilize in other optimization algorithms for solving other engineering problem.

Table 8. The comparison results of the gear train problem with 200 iteration and 1000 iteration with PSO-HDM and PSO-HDM without mutation operator

Algorithms	Mutation	Iteration	Worst	Average	Best	Standart deviation
PSO-HDM	Yes	200	$2.3903 - 10$	$2.6529e - 11$	$3.9580e - 19$	$5.8411e - 11$
PSO-HDM	No	200	$2.1147e - 10$	$1.2295e - 11$	$1.9444e - 16$	$4.2140e - 11$
PSO-HDM	Yes	1000	$9.8659e - 19$	$1.6866e - 19$	$2.9006e - 22$	$2.5088e - 19$
PSO-HDM	No	1000	$3.4730e - 09$	$1.4682e - 10$	$1.1524e - 18$	$6.9364e - 10$

4 CONCLUSION AND SUGGESTIONS

Particle Swarm Optimization (PSO) is an optimization algorithm for the solution of global optimization problems. In this work, we explore a different method for PSO called PSO-HDM, particle swarm optimization with Hypersphere Dynamics and Mutation. The position update is normally linear in original PSO, in this new PSO-HDM algorithm, it is succeeded by constructing two different hyperspheres and rotating them instead of using a linear velocity vector. Also, two different mutations, Jitter and Gaussian, is defined. This leads to advance compared to the classical PSO. For the purpose of the indicating the superiority of our novel proposed method, Gear Train Design problem are solved. Different categories of benchmark functions are also solved. In this study, the benchmark functions and the function to be minimized in the Gear Train Design problem are defined as the cost functions. For the future work, the proposed method could be applied to other well-known and important engineering design problems in the literature to further evaluate its effectiveness. Furthermore, the PSO-HDM could be combined into the other optimization algorithms in the literature to explore its potential.

Acknowledgements

We would like to thank Assoc. Prof. Dr. Korhan Günel for his valuable contributions.

Statement of Research and Publication Ethics

The study is complied with research and publication ethics.

Artificial Intelligence (AI) Contribution Statement

In this research, all aspects of the study, including its design, data analysis, and scientific contributions, were carried out solely by the authors. Only, ChatGPT® was used to improve the grammar and clarity of a few selected sentences.

Code Availability

The codes used in this study can be accessed at: https://github.com/iclalgor/PSO-HDM.

REFERENCES

[1] M.-H. Lin, J.-F. Tsai, N.-Z. Hu, and S.-C. Chang, "Design optimization of a speed reducer using deterministic techniques," *Mathematical Problems in Engineering*, vol. 2013, pp. 1–7, 2013. doi: 10.1155/2013/419043.

[2] L. Costa and P. Oliveira, "Evolutionary algorithms approach to the solution of mixed integer non-linear programming problems," *Computers and Chemical Engineering*, vol. 25, no. 2–3, pp. 257–266, 2001. doi: 10.1016/S0098-1354(00)00653-0.

[3] M. Braik, A. Sheta, and H. Al-Hiary, "A novel meta-heuristic search algorithm for solving optimization problems: capuchin search algorithm," *Neural Computing and Applications*., vol. 33, no. 7, pp. 2515–2547, 2021, doi: 10.1007/s00521-020-05145-6.

[4] E. Eker, M. Kayri, S. Ekinci and M.A. Kaçmaz, "Performance evaluation of PDO algorithm through benchmark functions and MLP training," *Electrica*, vol. 23, no. 3, pp. 597-606, 2023, doi: 10.5152/electr.2023.22179.

[5] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," in *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67-82, April 1997, doi: 10.1109/4235.585893.

[6] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," In *Proceedings of the MHS'95*. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 1995, pp. 39–43, 1995, doi: 10.1109/MHS.1995.494215.

[7] S. Gopi and P. Mohapatra, "A modified grey wolf optimization algorithm to solve global optimization problems," *OPSEARCH*, vol.62, pp. 337-367, 2025, doi: 10.1007/s12597-024-00785-x.

[8] S. Kaur, L. K. Awasthi, A. L. Sangal, and G. Dhiman, "Tunicate Swarm Algorithm: A new bio-inspired based metaheuristic paradigm for global optimization," *Engineering Applications of Artificial Intelligence*, vol. 90, no. 103541, pp. 103541, 2020.

[9] S. Mirjalili, A.H. Gandomi, A.H., S.Z. Mirjalili, S. Saremi, H. Faris, S.M., Mirjalili, "Salp swarm algorithm: a bio-inspired optimizer for engineering design problems," *Advances in Engineering Software*, vol. 114, pp. 163–191, 2017, doi: 10.1016/j.advengsoft.2017.07.002.

[10] S. Mirjalili, S.M. Mirjalili and A. Hatamlou, "Multi-verse optimizer: a nature-inspired algorithm for global optimization," *Neural Computing and Applications*. vol. 27, pp. 495–513, 2016, doi: 10.1007/s00521-015-1870-7.

[11] S. Mirjalili, S.M. Mirjalili and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*. vol. 69, 46–61, 2014, doi: 10.1016/j.advengsoft.2013.12.007.

[12] Y. Li, X. Lin and J. Liu, "An improved gray wolf optimization algorithm to solve engineering problems," *Sustainability*, vol. 13, no. 6, pp. 3208, 2021, doi: 10.1016/j.eswa.2020.113917.

[13] B. Duan, C. Guo, and H. Liu, "A hybrid genetic-particle swarm optimization algorithm for multi-constraint optimization problems," *Soft Computing*, vol. 26, no. 21, pp. 11695–11711, 2022, doi: 10.1007/s00500-022-07489-8.

[14] S. Mirjalili, "The Ant Lion Optimizer," *Advances in Engineering Software*, vol. 83, pp. 80-98, 2015, doi: 10.1016/j.advengsoft.2015.01.010.

[15] A.H. Gandomi, X.S. Yang and A.H. Alavi. "Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems," *Engineering Computations*, vol. 29, pp.17–35, 2013, doi: 10.1007/s00366-011-0241-y.

[16] A. Sadollah, A. Bahreininejad, H. Eskandar and M. Hamdi, "Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems," *Applied Soft Computing*, vol. 13, pp. 2592–612, 2013, doi: 10.1016/j.asoc.2012.11.026.

[17] A. H. Gandomi, "Interior search algorithm (ISA): a novel approach for global optimization," *ISA Transactions*, vol. 53, no. 4, pp. 1168-1183, 2014, doi: 10.1016/j.isatra.2014.03.018.

[18] S-J. Wu and P.-T. Chow. "Genetic algorithms for nonlinear mixed discrete-integer optimization problems via meta-genetic parameter optimization," *Engineering Optimization*, vol. 24, no. 2, pp. 137-159, 1995, doi: 10.1080/03052159508941187.

[19] D. Karaboga and B. Basturk. "Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems," In *Proceedings of International Fuzzy Systems Association World Congress*, pp.789–798, 2007, doi: 10.1007/978-3-540-72950-1_77.

[20] K. Deb and M. Goyal, "A combined genetic adaptive search (GeneAS) for engineering design," *Computer Science and Information*, vol. 26, pp.30–45, 1996.

[21] B. Kannan B and S.N. Kramer, "An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design," *Journal of Mechanical Design*, vol. 116, pp. 405–11, 1994, doi: 10.1115/1.2919393.

[22] S.Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowledge-Based Systems*, vol 89, pp. 228-249, 2015, doi: 10.1016/j.knosys.2015.07.006.

[23] H. Peraza-Vazquez, A. F. Pena-Delgado, G. Echavarria- Castillo, A. B. Morales- Cepeda, J. Velasco-Alvarez, F. Ruiz-Perez, "A bio-inspired method for engineering design optimization inspired by dingoes hunting strategies," *Mathematical Problems in Engineering*, vol. 202, pp. 11–19. 2021, doi: 10.1155/2021/9107547.

[24] H. Karami, M.V. Anaraki, S. Farzin and S. Mirjalili, "Flow Direction Algorithm (FDA): A Novel Optimization Approach for Solving Optimization Problems," *Computers and Industrial Engineering*, vol. 156, 107224, 2021, doi: 10.1016/j.cie.2021.107224.

[25] A. Askarzadeh, "A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm," *Computers and Structures*, vol. 169, pp. 1–12, 2016, doi: 10.1016/j.compstruc.2016.03.001.

[26] Y. Duan, N. Chen, L. Chang, Y. Ni, S. V. N. S. Kumar and P. Zhang, "CAPSO: Chaos Adaptive Particle Swarm Optimization Algorithm," in *IEEE Access*, vol. 10, pp. 29393-29405, 2022, doi: 10.1109/ACCESS.2022.3158666.

[27] S. Zhao, D. Wang, "Elite-ordinary synergistic particle swarm optimization" *Information Sciences*, vol. 609, pp. 1567-1587, 2022, doi: 10.1016/j.ins.2022.07.131.

[28] Q. Yang, X. Guo, X.D. Gao, D.D. Xu and Lu, Z.Y. "Differential Elite Learning Particle Swarm Optimization for Global Numerical Optimization," *Mathematics*, vol. 10, no 8, pp. 1261, 2022, doi: 10.3390/math10081261.

[29] K. Kassoul, N. Zufferey, N. Cheikhrouhou and S. Brahim Belhaouari, "Exponential Particle Swarm Optimization for Global Optimization," in *IEEE Access*, vol. 10, pp. 78320-78344, 2022, doi: 10.1109/ACCESS.2022.3193396.

[30] B.J. Solano-Rojas, R.Villalón-Fonseca and R. Batres, "Micro Evolutionary Particle Swarm Optimization (MEPSO): A new modified metaheuristic," *Systems and Soft Computing*, vol 5, 200057, 2023, doi: 10.1016/j.sasc.2023.200057.

[31] C. Wang, Z. Wang, S. Zhang and J. Tan, "Adam-assisted quantum particle swarm optimization guided by length of potential well for numerical function optimization," *Swarm and Evolutionary Computation*, vol.79, 2023, doi: 10.1016/j.swevo.2023.101309.

[32] J. Zhu, J. Liu, Y. Chen, X. Xue and S.Sun, "Binary Restructuring Particle Swarm Optimization and Its Application," *Biomimetics*, vol. 8, no 2, pp. 266, 2023, doi: 10.3390/biomimetics8020266.

[33] K. V. N. A. Bhargavi, G. P. S. Varma, I. Hemalatha, and R. Dilli, "An enhanced particle swarm optimization-based node deployment and coverage in sensor networks," *Sensors (Basel)*, vol. 24, no. 19, pp. 6238, 2024, doi: 10.3390/s24196238.

[34] C. Gong, N. Zhou, S. Xia, and S. Huang, "Quantum particle swarm optimization algorithm based on diversity migration strategy," *Future Generation Computer Systems*, vol. 157, pp. 445–458, 2024, doi: 10.1016/j.future.2024.04.008.

[35] S. J. Fusic and R. Sitharthan, "Self-adaptive learning particle swarm optimization-based path planning of mobile robot using 2D Lidar environment," *Robotica*, vol. 42, no. 4, pp. 977–1000, 2024, doi: 10.1017/S0263574723001819.

[36] S. Yang, B. Wei, L. Deng, X. Jin, M. Jiang, Y. Huang and F. Wang, "A leader-adaptive particle swarm optimization with dimensionality reduction strategy for feature selection," Swarm and Evolutionary Computation, vol. 91, no. 101743, pp. 101743, 2024, doi: 10.1016/j.swevo.2024.101743.

[37] G. Hu, M. Cheng, G. Sheng, and G. Wei, "ACEPSO: A multiple adaptive co-evolved particle swarm optimization for solving engineering problems," *Advanced Engineering Informatics*, vol. 61, no. 102516, pp. 102516, 2024, doi: 10.1016/j.aei.2024.102516.

[38] R. Liu, L. Wei, and P. Zhang, "An adaptive particle swarm optimization with information interaction mechanism," *Machine Learning Science and Technology*, vol. 5, no. 2, pp. 025080, 2024, doi: 10.1088/2632-2153/ad55a5.

[39] P.K.K. Ranganna, S.G. Matt, C.-L. Chen, A.B. Jayachandra, and Y.-Y. Deng, "Fitness sharing chaotic particle swarm optimization (FSCPSO): A metaheuristic approach for allocating dynamic virtual machine (VM) in fog computing architecture," *Computers Materials and Continua*, vol. 80, no. 2, pp. 2557–2578, 2024, doi: 10.32604/cmc.2024.051634.

[40] Ambuj, H. Nagar, A. Paul, R. Machavaram, and P. Soni, "Reinforcement learning particle swarm optimization based trajectory planning of autonomous ground vehicle using 2D LiDAR point cloud," *Robotics and Autonomous Systems*, vol. 178, no. 104723, pp. 104723, 2024, doi: 10.1016/j.robot.2024.104723.

[41] D. Tian, Q. Xu, X. Yao, G. Zhang, Y. Li, and C. Xu, "Diversity-guided particle swarm optimization with multi-level learning strategy," *Swarm and Evolutionary Computation*, vol. 86, no. 101533, pp. 101533, 2024, doi: 10.1016/j.swevo.2024.101533.

[42] X. Long, W. Cai, L. Yang, and H. Huang, "Improved particle swarm optimization with reverse learning and neighbor adjustment for space surveillance network task scheduling," *Swarm and Evolutionary Computation*, vol. 85, no. 101482, pp. 101482, 2024, doi: 10.1016/j.swevo.2024.101482.

[43] K. Tang and C. Meng, "Particle swarm optimization algorithm using velocity pausing and adaptive strategy," *Symmetry (Basel)*, vol. 16, no. 6, pp. 661, 2024, doi: 10.3390/sym16060661.

[44] C. Wang, S. Wang, G. Zhang, P. Takyi-Aninakwa, C. Fernandez, and J. Tao, "An improved particle swarm optimization-cubature Kalman particle filtering method for state-of-charge estimation of large-scale energy storage lithium-ion batteries," *Journal of Energy Storage*, vol. 100, no. 113619, pp. 113619, 2024, doi: 10.1016/j.est.2024.113619.

[45] M. Hoseiniasl and J.J. Fesharaki, "3D Optimization of Gear Train Layout Using Particle Swarm Optimization Algorithm," *Journal of Applied Computational Mechanics*, vol. 6, no.4 ,pp. 823-840, 2020, doi:10.22055/JACM.2019.29093.1558.

[46] M. Sedak and M. Rosić, "Hybrid Butterfly Optimization and Particle Swarm Optimization Algorithm-based constrained multi-objective nonlinear planetary gearbox optimization," *Applied Sciences (Basel)*, vol. 13, no. 21, pp. 11682, 2023, doi: 10.3390/app132111682.

[47] E. Sandgren, "Nonlinear Integer and Discrete Programming in Mechanical Design Optimization," *Journal of Mechanical Design*, vol. 112, no 2, pp. 223-229, 1990, doi: 10.1115/1.2912596.

[48] H. Mutahira, V. Shin, U. Park and M. S. Muhammad, "Jitter noise modeling and its removal using recursive least squares in shape from focus systems," *Scientific Reports*, vol. 12, 14015, 2022, doi: 10.1038/s41598-022-18150-7.

[49] O. Bell. "Applications of Gaussian mutation for self adaptation in evolutionary Genetic Algorithms," *ArXiv*:2201.00285, 2022, https://api.semanticscholar.org/CorpusID:245650445.

[50] R.M. Zur, Y. Jiang and C.E. Metz, "Comparison of two methods of adding jitter to artificial neural network training," *International Congress Series*, vol. 1268, pp. 886–889, 2004, doi: 10.1016/j.ics.2004.03.238.

[51] K. Deb and D. Deb, "Analysing mutation schemes for real-parameter genetic algorithms," *International Journal of Artificial Intelligence and Soft Computing*, 2014, doi: 10.1504/IJAISC.2014.059280.

[52] M. Premkumar, Pradeep Jangir, R. Sowmya, Rajvikram Madurai Elavarasan, B. Santhosh Kumar, "Enhanced chaotic JAYA algorithm for parameter estimation of photovoltaic cell/modules," *ISA Transactions*, vol. 116, pp. 139-166, 2021, doi: 10.1016/j.isatra.2021.01.045.

[53] Z.Zhang, "Abnormal detection of pumping unit bearing based on extension theory," *IEEJ Transactions on Electrical and Electronic Engineering*, vol.16, no 12, pp. 1647–1652, 2021, doi: 10.1002/tee.23468.

[54] X.R. Zhao, Y.R. Zhou and Y. Xiang, "A grouping particle swarm optimizer," *Applied Intelligence*, vol. 49, no., 8, pp. 2862–2873, 2019, doi: 10.1007/s10489-019-01409-4.

[55] J.S. Guan, S. J. Hong, S. B. Kang, Y. Zeng, Y. Sun and C.M. Lin, "Robust adaptive recurrent cerebellar model neural network for non- linear system based on GPSO," *Frontiers in Neuroscience*, Switz, vol. 13, pp. 390. 2019, doi: 10.3389/fnins.2019.00390.

[56] W. Guo, C. Si, Y. Xue, Y. Mao, L. Wang, Q. Wu "A grouping particle swarm optimizer with personal-best-position guidance for large scale optimization," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 15, no. 6, pp. 1904–1915, 2018, doi: 10.1109/TCBB.2017.2701367.

[57] L. T. Al-Bahrani, J. C. Patra, "A novel orthogonal PSO algorithm based on orthogonal diagonalization," *Swarm and Evolutionary Computation*, vol. 40, pp. 1–23, 2018, doi: 10.1016/j.swevo.2017.12.004.

[58] M. X. Song, K. Chen and J. Wang, "Three-dimensional wind turbine positioning using Gaussian particle swarm optimization with differential evolution," *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 172, pp. 317–324. 2018, doi: 10.1016/j.jweia.2017.10.032.

[59] S. Mirjalili and A. Lewis "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016, doi: 10.1016/j.advengsoft.2016.01.008.

[60] S. A. K. Alrufaiaat and A. Q. J. "Robust decoding strategy of MIMO-STBC using one source Kurtosis based GPSO algorithm," *Journal of Ambient Intelligence and Humanized Computing*, vol.12, no. 2, pp. 1967–1980. 2021, doi: 10.1007/s12652-020-02288-1.

[61] D.R. Jones, C.D. Perttunen, B.E. Stuckman, "Lipschitzian optimization without the Lipschitz constant," *Journal of Optimization Theory and Application*, vol. 79, no.1, pp.157–181, 1993, doi: 10.1007/BF00941892.

[62] A. Duran and G. Caginalp, "Parameter optimization for differential equations in asset price forecasting," Optimization Methods and Software, vol. 23, no. 4, pp. 551–574, 2008. Issue: Mathematical programming in data mining and machine learning, doi: 10.1080/10556780801996178.

[63] M. Tuncel and A. Duran, "Effectiveness of grid and random approaches for a model parameter vector optimization," *Journal of Computational Science*, vol. 67, no. 101960, 2023, doi: 10.1016/j.jocs.2023.101960.

[64] H. Karami, M.J. Sanjari, and G.B. Gharehpetian, "Hyper-Spherical Search (HSS) algorithm: a novel meta-heuristic algorithm to optimize nonlinear functions," *Neural Computing and Applications*, vol. 25, pp. 1455–1465, 2014, doi: 10.1007/s00521-014-1636-7 .

[65] A. T., Tantu, and D.G. Biramo, "Power flow control and reliability improvement through adaptive PSO based network reconfiguration," *Heliyon*, vol. 10, no. 17, pp. e36668, 2024, doi: 10.1016/j.heliyon.2024.e36668.

[66] Z. Cui, and J. Zeng. "A Guaranteed Global Convergence Particle Swarm Optimizer," In: Tsumoto, S., Słowiński, R., Komorowski, J., Grzymała-Busse, J.W. (eds) Rough Sets and Current Trends in Computing. RSCTC 2004, *Lecture Notes in Computer Science*, vol. 3066. Springer, Berlin, Heidelberg, 2004, doi: 10.1007/978-3-540-25929-9_96.