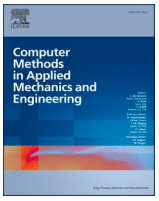
Computer Methods in Applied Mechanics and Engineering 446 (2025) 118318

Contents lists available at ScienceDirect

Computer Methods in Applied Mechanics and Engineering

journal homepage: www.elsevier.com/locate/cma



PDPSO: Priority-driven search particle swarm optimization with dynamic candidate solutions management strategy for solving higher-dimensional complex engineering problems

Gang Hu [a,b,*], Peidong He [a], Heming Jia [c], Essam H. Houssein [d], Laith Abualigah [e]

[a] Department of Applied Mathematics, Xi'an University of Technology, Xi'an 710054, PR China
[b] School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, PR China
[c] School of Information Engineering, Sanming University, Sanming 365004, China
[d] Faculty of Computers and Information, Minia University, Minia, Egypt
[e] Computer Science Department, Al al-Bayt University, Mafraq 25113, Jordan

ARTICLE INFO

Keywords:
Particle swarm optimization
Priority-driven search strategy
Dynamic candidate solution management strategy
Higher-dimensional complex engineering problems

ABSTRACT

Particle swarm optimization (PSO) is one of the most classical meta-heuristic algorithms (MAs), and the algorithm is extensively used in solving practical engineering application problems. To address the challenges of particle swarm optimization in high-dimensional complex engineering problems, including accuracy, stability, and resource utilization, we propose a PSO variant called PDPSO, which incorporates a priority-driven search strategy and a dynamic candidate solution management strategy. The priority-driven search strategy removes the inertia term, retaining the guidance of the individual optimal and high-priority candidate solutions, significantly enhancing algorithm stability, robust and execution efficiency; the dynamic candidate solution management strategy constructs an adaptive priority table, dynamically screening high-quality solution sets via a non-uniform subpopulation allocation mechanism, combined with a state rollback mechanism to prevent local stagnation, and integrating candidate solution elimination to achieve a self-balancing equilibrium between algorithm exploration and exploitation. We have not added any mutation strategy to the original algorithm, instead, the PDPSO algorithm balances the exploration and exploitation capabilities by whether or not to continue to learn the previous candidate solutions at the next update, which is a significant highlight of the algorithm and provides a new research direction to improve the robustness of other optimization algorithms for high-dimensional complex problems. Verified by CEC2017 high-dimensional testing (30, 50, and 100 dimensions) and 34 engineering examples, the PDPSO algorithm exhibits outstanding advantages in high-dimensional problems and excellent computational efficiency. When the test function dimension increases from 30D to 100D, the growth rate in time complexity is as low as 24.87%. The PDPSO algorithm demonstrates strong applicability in real-world engineering problems, with optimization results in various complex simulation scenarios approaching theoretical optimal solutions. PDPSO provides a high-precision, robust solution for high-dimensional engineering optimization, and its dimension-insensitive complexity characteristics have universal value. The source codes and supplementary materials of PDPSO are available at https://github.com/hepeidong1/PDPSO.

* Corresponding author.
E-mail address: hugang@xaut.edu.cn (G. Hu).

https://doi.org/10.1016/j.cma.2025.118318
Received 17 April 2025; Received in revised form 9 August 2025; Accepted 10 August 2025
Available online 25 August 2025

1. Introduction

Along with the development of society, human society is facing many problems and challenges, such as energy planning problems [1], autonomous electric vehicle system design [2], scheduling optimization problems [3], portfolio optimization [4], deep learning [5], path planning [6,7] and other engineering problems [8,9], These problems and challenges usually possess characteristics such as complex structure and difficult to be optimized. In the past, scientists have proposed many traditional methods to tackle these issues and challenges: random search method [10], pattern search method [11], convex optimization [12], linear programming [13], interior-point method [14], quadratic programming [15], trust-region method [16], Newton-Raphson method [17], steepest descent method [18], conjugate gradient method [19], etc.

However, in reality, most of the optimization problems have poor continuity and suffer from high computational costs, nonlinear constraints, non-convex search regions, and the presence of noise, which makes it difficult for traditional optimization methods to ensure their robustness and accuracy [20,21]. For these problems, a near-optimal solution can be utilized to supplant the ideal solution, whereas meta-heuristic algorithms are regularly utilized to discover a near-optimal solution for optimization issues [22]. As a random-based iterative optimization algorithm with simple parameters, The MAs do not rely on the information of gradient and are strong in robustness, so it is widely used to solve practical problems, even though they are not guaranteed to converge to the global optimal solution [23].

High-dimensional optimization problems present unique challenges due to their expansive search space, leading to sparse samples, numerous local extrema, gradient estimation difficulties, and high computational costs [24]. Current research focuses on using MAs to create efficient, low information dependent algorithms and employing random sparse sampling to minimize global computation.

Pan et al. enhanced the artificial bee colony algorithm by using a two-dimensional queue structure and hybrid guidance, improving solution distribution and convergence in high-dimensional optimization [25]. Shu et al. integrated multiple mutation strategies into MSFPSO, achieving high convergence accuracy and efficiency for complex real-world problems [26]. Salam et al. introduced a quadruple strategy-driven climbing optimization algorithm for medical data feature selection, showing superior accuracy and stability over traditional machine learning methods [27]. Chakraborty et al. developed the nwSOS algorithm, which focuses on non-linear payoff functions, weighted mutual vectors, and optimization computational burden, proposed the nwSOS algorithm, which demonstrates good global search capability and efficiency in solving engineering optimization problems [28]. Huo et al. proposed the SCWOA algorithm, which exhibits good local exploitation capability in solving complex reservoir scheduling optimization problems [29]. Zhang et al.'s HSAOA algorithm features a hierarchical framework and efficient sampling strategies for robust performance in complex optimization tasks [30]. Additional contributions to engineering optimization are noted [31–34].

Those discussed algorithm improvement strategies have significantly aided in solving complex engineering optimization problems. However, some seemingly perfect approaches may fall short in practice. For instance, while hierarchical proxy frameworks group populations based on characteristics, existing methods lack adaptability to specific optimization challenges. Some algorithms employ diversified mutation strategies, adjusting the update strategies of specific agents by assigning them different mutation probabilities, mutation magnitudes, or learning objectives. However, questions such as when individuals should mutate and whether the manner of mutation can be automatically adjusted based on the specific problem being addressed appear to have not been sufficiently considered. Despite employing multiple strategies to enhance algorithms, questions about generalizability, increased computational complexity, and achieving a '$1+1>2$' effect about strategies remain areas for further exploration.

Regarding the above issues, we propose a PSO variant based on priority-driven search strategy and dynamic candidate solution management strategy.

The main contributions include:

(1) We abandon the traditional PSO algorithm's velocity updating method and proposes a priority-oriented search strategy. This strategy aims to reduce the quantity of velocity-influencing factors to improve the algorithm's stability and efficiency
(2) We paper proposes a dynamic candidate solution management strategy, which aims to provide a set of rich and excellent candidate learning targets for the population and an efficient management method for the candidate learning targets to assist the algorithms in efficiently transforming the computational resource consumption into optimization capability. This strategy improves the diversity of the population through the idea of subpopulation optimization and avoids premature maturity of the algorithm. In this strategy, the population subgroups are generated spontaneously. The allocation of individuals among populations is non-uniform, adaptive and changing in real time, and the change process has a high degree of rationality and linkage.

In this paper, when a particle's fitness decreases while learning a target, it will continue to pursue that target with a certain probability. Conversely, if it fails repeatedly, the particle will gradually abandon the target and switch to others. When a particle switches to learning other learning targets, it generates a large number of scattered and discontinuous decision-making variables during the process of transforming the learning target, which increases the diversity of population positions. This process is similar to many PSO variants based on diversified mutation strategies. However, this approach employs a priority-driven search strategy that allows particles to switch candidate solutions, reducing computational complexity and making positional mutations time-varying and flexible.

Table 1
Classification and abbreviations of some MAs.

Eas:		Ref.	SI:		Ref.
Genetic algorithm	GA	[36]	Particle swarm optimization	PSO	[37]
			Artificial bee colony	ABC	[38]
PhAs:			Artificial hummingbird algorithm	AHA	[39]
Triangulation topology aggregation optimizer	TTAO	[40]	Artificial rabbits optimization	ARO	[41]
Attraction repulsion optimization algorithm	AROA	[42]	Dung beetle optimizer	DBO	[43]
Equilibrium optimizer algorithm	EO	[44]	Dwarf mongoose optimizer	DMO	[45]
Exponential-trigonometric optimization	ETO	[46]	Electric eel foraging optimization	EEFO	[47]
Fata morgana algorithm	FATA	[48]	Genghis khan shark optimizer	GKSO	[49]
Football team training algorithm	FTTA	[50]	Golden jackal optimization	GJO	[51]
Gradient-based optimizer	GBO	[52]	Grey wolf optimizer	GWO	[53]
Heap-based optimizer	HBO	[54]	Honey badger algorithm	HBA	[55]
Neural population dynamics optimization algorithm	NPD	[56]	Harris hawks optimization	HHO	[23]
Optical microscope algorithm	OMA	[57]	Marine predators algorithm	MPA	[58]
Weighted average algorithm	WAA	[59]	Nutcracker optimizer	NOA	[60]
			Whale optimization algorithm	WOA	[61]
HSBAs:			Arctic puffin optimization	APO	[62]
Teaching-learning-based optimization	TLBO	[63]	Artificial protozoa optimizer	APO2	[64]
Football game algorithm	FGA	[65]	Blood-sucking leech optimizer	BSLO	[66]
Socio evolution&learning optimization algorithm	SELO	[67]	Crested porcupine optimizer	CPO	[68]
Competition of tribes and cooperation of members algorithm	CTCM	[69]	Ivy algorithm	IVY	[70]
Escape	ESC	[71]	Piranha predation optimization algorithm	PPOA	[72]
Hiking optimization algorithm	HOA	[73]	Red-billed blue magpie optimizer	RBMO	[74]

2. Related work

2.1. Evolutionary algorithm research progress

MAs can be categorized into 4 classes based on their sources of inspiration. The first class is Evolutionary Algorithms (EAs), inspired by natural selection and survival of the fittest. The second class, Physical Law-based Algorithms (PhAs), draws from mathematical principles and natural phenomena. The third class is Human Social Behavior-based Algorithms (HSBAs). The most prevalent class is Swarm Intelligence (SI), which mimics collaborative foraging in animal populations, where groups work together to achieve common goals. They learn from their experiences, communicate a lot, and share what they know with each other [21]. Inspired by the self and collective behaviors of bird populations during collective feeding, Kennedy and Eberhart proposed the PSO algorithm in 1995, this algorithm mathematically models the behavior of each bird in the population, and it is a robust population-based randomized optimization algorithm [35].

Table 1 lists classic and new metaheuristic algorithms, grouped by their sources of inspiration.

According to the NFL theory, there is no universal algorithm for optimization problems. Choosing the right algorithm is essential and should depend on the specific problem and the algorithm's features [75]. Different optimization algorithms exhibit varying strengths and weaknesses when addressing distinct optimization problems. Selecting the appropriate algorithm can significantly enhance efficiency when tackling engineering optimization challenges[76]. Based on the differences in their new agent generation and update mechanisms, MAs can be categorized into three primary types[21]: The first category comprises solution-creation algorithms, characterized by the generation of new solutions through the combination of existing ones. Genetic Algorithm (GA, discrete) and Differential Evaluation (DE, continuous) are classic solution-creation algorithms. GA has a strong global search ability but underperforms in terms of stability and local optimization ability, and the gene chain's representation of individual populations also generates significant storage pressure. The DE algorithm not only retains the advantages of GA's strong global search ability, substantial variability, and adjustable learning factors but also addresses GA's shortcomings regarding storage pressure while offering greater robustness. However, weaknesses in stability and local optimization ability persist. The second category of algorithms is relatively rare, with AHA and ACO as representatives. Their characteristic is that agent individual updates rely on specific metric values, such as nectar quantity in the AHA algorithm and pheromone in the ACO algorithm. Taking the ACO algorithm as an example, compared with GA and DE algorithms, since ACO adopts the pheromone method to search for paths, and the pheromone system is a positive feedback regulation system, which can accelerate the algorithm's convergence in the late iteration stage, so this algorithm has an advantage in efficiency and optimization speed in the later stages. However, it is this positive feedback regulation system that requires some time for the algorithm to reach a stable state from its initial state, resulting in inferior optimization results in a short period. The third category of algorithms is Trajectory-based algorithms (TAs). The characteristic of this category of algorithms is that agent individuals exhibit specific behavioral trajectories during the update process; by controlling the relevant parameters of the trajectory, the algorithm's exploration and exploitation properties can be adjusted. PSO algorithms and most SI algorithms belong to the category of TAs. This type of algorithm appears to balance the above advantages and disadvantages in terms of performance; however, the solutions it generates have a single structure and cannot mutate, like GA and DE, making them highly vulnerable to local extremes. Based on the above reasons, this paper uses the PSO algorithm as a representative and proposes innovative improvements to address unresolved issues in engineering optimization, contributing to the enhancement of algorithms.

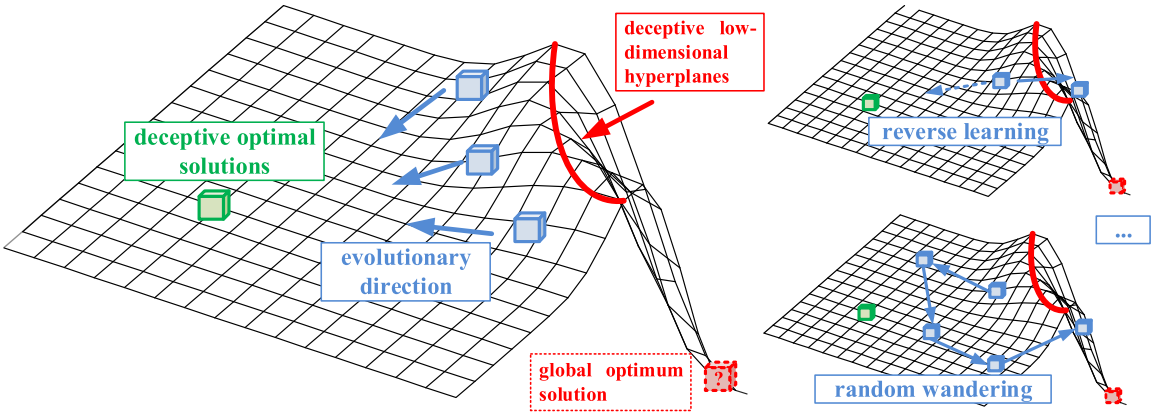


Fig. 1. A diagram of particle evolution process subject to deception and two improved strategies.

2.2. PSO algorithm research progress

In the initialization process of PSO, we use a random method to form the particle population $\boldsymbol{X}$. The initial population $\boldsymbol{X}$ is shown as:

$$\boldsymbol{x}_i = \boldsymbol{Lb} + \boldsymbol{r} \cdot (\boldsymbol{Ub} - \boldsymbol{Lb}), i = 1, 2, \cdots, N, \tag{1}$$

where $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$ denotes the particle population; N is the particle population size; $\boldsymbol{Ub}$ and $\boldsymbol{Lb}$ represent the upper and lower bounds of the D-dimensional problem, respectively; $\boldsymbol{r}$ is a D-dimensional random vector in [0,1].

The ith particle position after the k+1st iteration is shown as:

$$\boldsymbol{x}_i^{k+1} = \boldsymbol{x}_i^k + \boldsymbol{v}_i^{k+1}, \ i = 1, 2, \cdots, N, \tag{2}$$

where N is the particle population size, $v_{i,j}^{k+1}$ denotes the velocity of the ith particle in the k+1st iteration, obtained from Eq. (3).

$$\boldsymbol{v}_i^{k+1} = \omega \boldsymbol{v}_i^k + c_1 r_1 \left(\boldsymbol{pb}_i^k - \boldsymbol{x}_i^k\right) + c_2 r_2 \left(\boldsymbol{gb}^k - \boldsymbol{x}_i^k\right), \ i = 1, 2, \cdots, N, \tag{3}$$

where r_1 and r_2 are random values between (0,1); N is the particle population size; ω denotes the inertia weight, which indicates that the particle velocity is affected by the inertia velocity; c_1 and c_2 indicate the individual acceleration coefficients and global acceleration coefficients, respectively; $\boldsymbol{v}_i^k$ denotes velocity; $\boldsymbol{pb}_i^k$ denotes individual optimal position; $\boldsymbol{gb}^k$ denotes global optimal position. In practice, ω normally adjusted between 0.4 and 0.9, and c_i (i=1,2) be also modified based on the actual problem [77].

Swarm intelligent optimization systems generally require populations to move toward decreasing fitness values to approach optimal solutions [78]. However, this approach can be misleading, as Xia et al. noted that fitness-value-based algorithms are often affected by deceptive local optima [79]. In complex optimization problems, low-dimensional hyperplanes can misguide population movement, steering them away from the true optimal solution. This situation typically occurs when tackling high-dimensional complex problems, as shown in Fig. 1 schematically.

Fig. 1 shows an example of locating a minimum point. The global optimal solution is in the red dashed square, while the other squares represent individuals in the population. In Fig. 1, the blue individuals are deceived by the green individuals and choose the wrong evolutionary direction, thus making it difficult to find the true minimum point. Strategies such as reverse learning, random wandering, random mutation [80,81], stochastic learning [82], and other strategies can make the population move without considering the fitness value, extending the search space by sacrificing the capacity of the neighborhood utilize to maintain a distance from the population in a local optimum. Long et al. proposed RNP-PSO, which combines reverse learning, neighborhood adjustment, and particle perturbation to enhance PSO's performance in high-dimensional optimization problems [83]. Hakli et al. integrated Levy flight with PSO to enable larger particle jumps, improving the algorithm's global search capabilities [84]. Song et al. increased population diversity using tent chaotic mapping and a random wandering strategy to boost exploitation [85]. Li et al. introduced a subpopulation division method and a co-evolutionary mechanism to improve convergence accuracy [86]. Yang et al. proposed an evolutionary state-driven multiple swarms collaborative particle swarm optimization algorithm (ESD-PSO); this algorithm utilizes an adaptive multiple swarm collaborative mechanism, which involves the random division of the population into multiple equal sub-populations [87]. It also describes a trigger-stopping compensation strategy that improves population diversity. An idea to avoid falling into local optimum is given here, that is, to avoid local optima, it suggests considering not just fitness values but also a rational assessment of the candidates' exploitation rationality when selecting learning candidates.

Another way to improve the optimization capability of an algorithm is to provide a dynamic method of parameter adjustment. In the updating process of traditional PSO, by controlling the values of the three learning factors, the exploration and exploitation ability can be balanced. Taking the inertia weight ω as an example, in the process of particle searching, setting a larger value of ω helps the

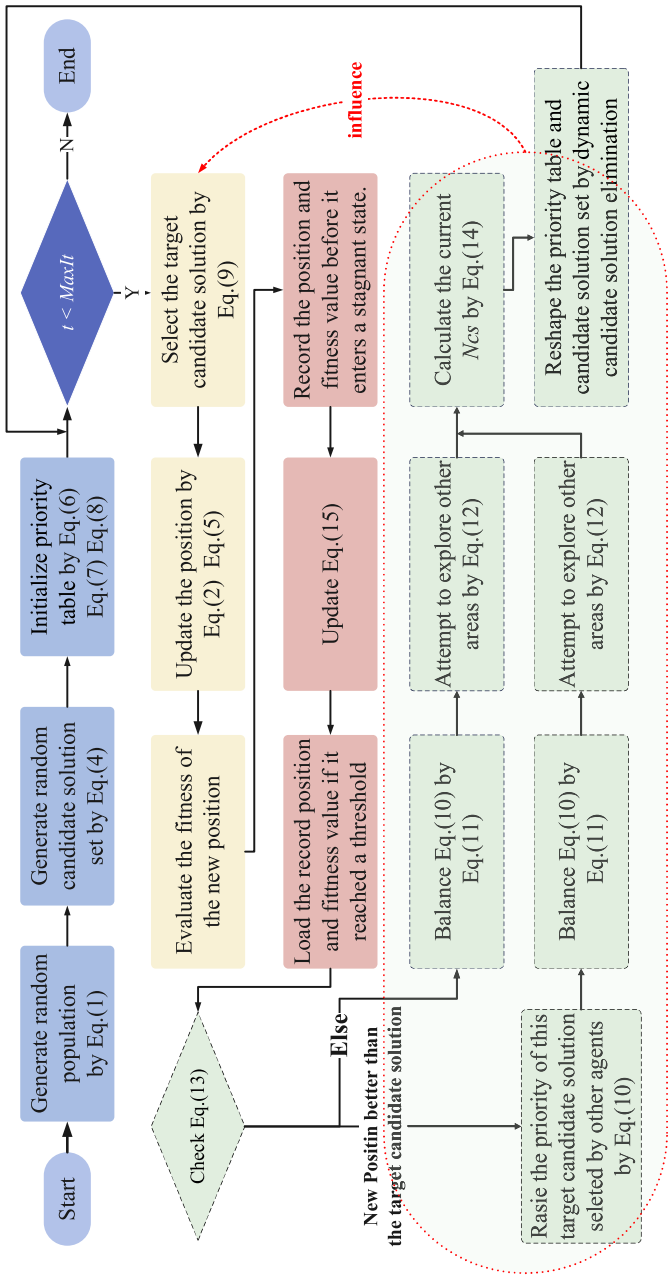


Fig. 2. The flow chart of PDPSO.

particle to search globally, helping the particle to move further from the current position and escape from the local optimum. Instead, setting a smaller ω helps the particle to locally explore and further optimize the currently obtained optimal solution [88]. Currently, common ideas for inertia weight improvement are to use adaptive parameter adjustment, such as Madani et al. sampling randomly over a given range so that the particles always maintain the potential for exploitation and exploration [89], Elaziz et al. adopting the exponential linear decreasing method of inertia weight [90], Han et al. adopting the inertia weight time-varying linear decreasing method of inertia weights [91], etc. The main purpose of these parameter adjustment strategies is to enable the particles to carefully explore the solution range at first and skillfully exploit the approximate optimal solution subsequently.

In the PSO algorithm's iterative process, a single particle's information is limited, and simply adjusting weight parameters doesn't overcome the traditional PSO's inability to produce only continuous decision variables, resulting in minimal the above improvements. Long et al. highlight that the PSO algorithm lacks memory of the solution transformation process, making it susceptible to local optima [83]. CLPSO addresses this by employing diverse learning strategies to enhance population diversity. However, the introduction of diverse evolutionary modes and dynamic parameter adjustments can lead to instability [92]. Hao Wu et al. noted that while CLPSO minimizes the risk of particles falling into local extremes, significant distance between particles in successive iterations can cause excessive update velocities, resulting in a "jumping" phenomenon [93]. The traditional CLPSO algorithm's inertial velocity update formula exacerbates this issue, hindering the effectiveness of the comprehensive learning strategy. To improve this, the FVCLPSO algorithm removes inertial velocity using a velocity forgetting strategy, reducing the "jump" phenomenon and enhancing convergence speed [94]. The above analysis shows that increasing algorithm complexity is not a long-term solution for improving performance; appropriate strategies are needed for fundamental improvements.

2.3. Main contributions

In fitness-based evolution, learning targets are static, individuals evaluate independently, and particles in high-dimensional problems can get misled by deceptive low-dimensional hyperplanes, causing wasted evaluation time in optimization. Most PSO variants target specific engineering optimization problems, lacking effective strategies for complex and unknown cases. Although it is feasible to increase the algorithm's complexity to solve complex optimization problems, this is not a long-term solution, as engineering optimization problems are already too complex. Therefore, we propose a new PSO variants. PDPSO offers several advantages over basic PSO and its variants:

(1) It adapts the population division in real time based on the optimization problem, thereby enhancing generalizability.
(2) It dynamically switches search modes according to the optimization problem, improving adaptability.
(3) Its search focus is not merely limited to one or more fixed areas but also to evaluate various factors, such as the suitability of areas for continued search, adjustments in search frequency, and decisions on abandoning or repositioning areas.
(4) The simplified velocity update formula boosts optimization accuracy and execution efficiency in high-dimensional problems.
(5) Experimental results demonstrate that PDPSO effectively identifies optimal or near-optimal solutions in various engineering optimization problems.

The paper is organized as follows: Section 3 provides the detailed steps of the PDPSO algorithm. To verify its effectiveness, Section 4 conducts several sets of numerical experiments. These two sections validate the algorithm's improved effectiveness in terms of theoretical and experimental results.

3. PDPSO algorithm

During the PSO algorithm update process, particles are subjected to three reference elements: inertial velocity and individual and global optimal positions. Although some of the previously mentioned research works focus on increasing the number of reference elements appropriately, these works do not manage these additional reference elements appropriately, which makes these additional used computational resources not efficiently transformed into the optimization ability of the algorithm. This paper introduces a priority-driven search strategy and a dynamic candidate solution management strategy to the traditional PSO algorithm.

3.1. The process of PDPSO algorithm

PDPSO is a particle swarm algorithm with hybrid priority-driven search and dynamic candidate solution management to efficiently and stably use more reference elements so that the computational resources are effectively transformed into the optimization ability of an algorithm.

Fig. 2 is the flowchart of the PDPSO. The blue represents the original PSO framework, yellow is for the priority-driven search strategy, red is for the state rollback strategy, and green represents other aspects of the dynamic candidate solution management strategy, excluding state rollback.

3.1.1. Priority-driven search strategy

In the PDPSO algorithm, we guide the particle search direction by maintaining a high-quality candidate solution set. Denote this candidate solution set as $\boldsymbol{CS}$, then

$$\boldsymbol{CS} = \left\{ \boldsymbol{CS}_j^k \middle| j = 1, 2, \cdots, Ncs^k \right\}, \tag{4}$$

where $\boldsymbol{CS}_j^k$ denotes some positions in the solution space at the kth iteration; and Ncs^k denotes the capacity of the candidate solution set at the kth iteration.

From the perspective of each individual in the population, the priority of selecting and using these candidate solutions varies. This strategy is termed a priority-driven search strategy because the priority is recorded and updated separately, with details on priority recording and updating discussed later. The high-priority candidate solution is the one with the highest priority for each individual in the population. High-priority candidate solutions vary by individual and can change over time, but all are part of the same set that shares global information.

The candidate solution is assigned to various positions in the solution space during updates, sharing information across all populations. This allows it to function like the global optimum in the original PSO algorithm. Thus, we replaced the global optimum with high-priority candidate solutions from the candidate solution set. In the original PSO algorithm, population diversity is enhanced through inertial velocity, allowing variation among agent individuals. In PDPSO, diversity comes from multiple candidate solutions, with variation achieved by selecting different candidates between iterations, eliminating the need for inertial velocity. Additionally, multiple velocity reference items create uncertainty in population optimization, as varying vector directions and magnitudes can lead to velocity vectors that are either too small or too large. To maintain algorithm stability, we eliminate the inertial velocity from Eq. (3) and focus only on the individual optimal position and the high-priority candidate solution in the velocity calculation. The velocity of the ith particle in the k+1st iteration based on the priority-driven search strategy is denoted as:

$$\boldsymbol{v}_i^{k+1} = c_1 \boldsymbol{r}_1 \left(\boldsymbol{pb}_i^k - \boldsymbol{x}_i^k \right) + c_2 \boldsymbol{r}_2 \left(\boldsymbol{CS}_{i(k)}^k - \boldsymbol{x}_i^k \right), \ i = 1, 2, ..., N, \tag{5}$$

where r_1 , r_2 is a random value between (0,1); c_1 c_2 indicates the individual acceleration coefficients and global acceleration coefficients, respectively; N is the particle population size; $\boldsymbol{pb}_i^k$ denotes the individual optimal position of the ith particle after the kth iteration; $\boldsymbol{CS}_{i(k)}^k$ denotes the high-priority candidate solution that the ith particle chooses to learn at the kth iteration. To reinforce the effect of high-priority candidate solutions, in this paper, $\boldsymbol{r}_1 = \sqrt{\boldsymbol{r}}$ and $\boldsymbol{r}_2 = 1 - \boldsymbol{r}_1$, where $\boldsymbol{r}$ is a D-dimensional random vector in [0,1].

3.1.2. Dynamic candidate solution management strategy

This paper proposes a dynamic candidate solution management strategy for managing this candidate solution set.

A. **Dynamic candidate solution management strategy initialization**

Before the PDPSO iteration begins, a dynamic candidate solution management strategy generates an initial set of candidate solutions, each with position and fitness value information. The capacity of this set, defined by Eq. (15), varies over time, starting from an initial value given by Eq. (6). The initial capacity must balance two key factors: setting it too high slows down priority updates and disperses the population's collective optimization capabilities, reducing optimization efficiency, while setting it too low will degrade that algorithm's performance to the baseline level of the traditional PSO algorithms.

The initial capacity of the candidate solution set is denoted as:

$$Ncs^0 = \min\left\{ N, \sqrt{T} \right\}, \tag{6}$$

where N denotes the population number; and T is the maximum iteration number.

Due to the algorithm's inability to gather useful information during initialization, we employs a random method to create the initial candidate solution set, denoting the jth candidate solution as $\boldsymbol{CS}_j^0$, the initialization method for $\boldsymbol{CS}_j^0$ is consistent with Eq. (1).

A. **Management of the candidate solution set**

The DCSMS selects the candidate solutions to learn by a priority table, denoted $\boldsymbol{P}$, then

$$\boldsymbol{P}^k = \begin{pmatrix} P_{1,1}^k & P_{1,2}^k & \cdots & P_{1,Ncs}^k \\ P_{2,1}^k & P_{2,2}^k & \cdots & P_{2,Ncs}^k \\ \vdots & \vdots & \ddots & \vdots \\ P_{N,1}^k & P_{N,2}^k & \cdots & P_{N,Ncs}^k \end{pmatrix}_{N \times Ncs}, \tag{7}$$

where $P_{i,j}^k$ denotes the priority of the jth candidate solution for the ith particle at the kth iteration.

Each particle's candidate solution priorities are independent. During initialization, candidate solutions are assigned to particles with equal probability:

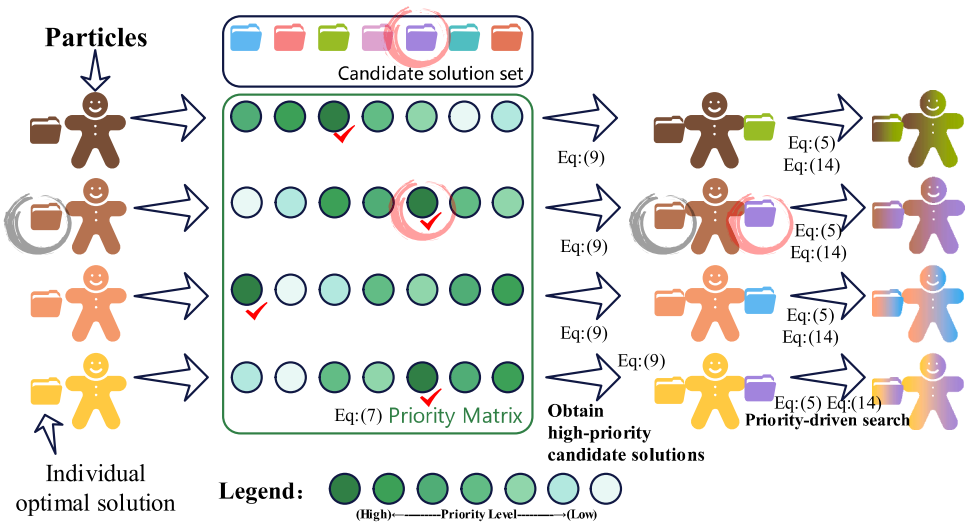


Fig. 3. Particles learning process based on priority-driven search strategies.

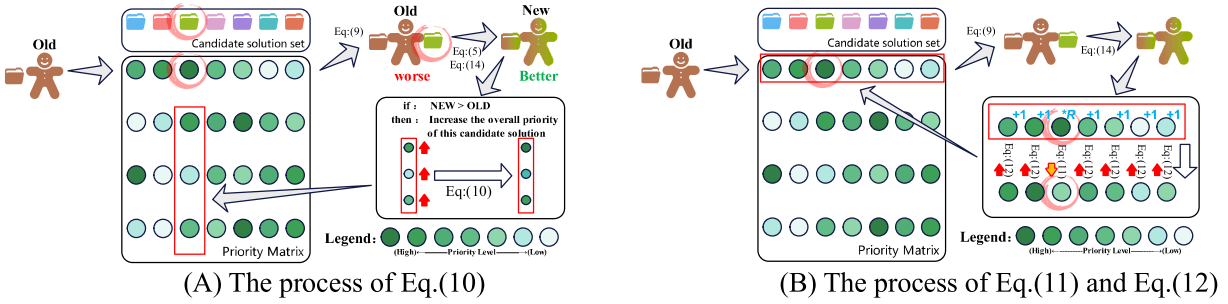


Fig. 4. Schematic diagram of the process of Eq.(10) Eq.(11) and Eq.(12) effects.

$$P_{i,j}^{0} = randi(1, Ncs), \quad i = 1, 2, \ldots, N; \quad j = 1, 2, \ldots, Ncs^{0}, \tag{8}$$

where $randi(a, b)$ denotes a random integer between a and b; N denotes the population number; Ncs^0 denotes the capacity of the initial candidate solution set.

At the kth iteration, before the ith particle executes the priority-driven search strategy, it will select a high-priority candidate solution from the set of candidate solutions in advance. The priority of this candidate solution is the maximum in the ith row of the priority table, so this candidate solution $\boldsymbol{CS}_{i(k)}^{k}$ is defined as the high-priority candidate solution for the ith particle, denoted as:

$$\boldsymbol{CS}_{i(k)}^{k} = \left\{ \boldsymbol{CS}_{j}^{k} \middle| P_{i,j}^{k} \geq P_{i,m}^{k}, \; m = 1, 2, \cdots, j-1, j+1, \cdots, Ncs^{k} \right\}, \tag{9}$$

where Ncs^k denotes the capacity of the candidate solution set at the kth iteration. If an individual selects different candidate solutions in two consecutive iterations, we call this switching search targets. This behavior leads to significant differences in the positions of proxy individuals, thereby increasing the diversity of population. So, we are primarily focusing on this behavior in this paper.

According to Eq. (5), particle updates rely on two positions: the particle's own best position and a high-priority candidate solution from the set. The process of particle learning based on the priority-driven search strategy is schematically shown in Fig. 3. In Fig. 3, the individual optimal solution is depicted on the left archive, and the selected candidate is shown on the right. The priority matrix indicates the importance of each candidate for each particle, with darker colours signifying higher priority.

After the ith particle learned the $i(k)$ candidate solution $\boldsymbol{CS}_{i(k)}^{k}$ in the candidate solution set $\boldsymbol{CS}^{k}$ through the priority-driven search strategy, if a new optimal position of a particle is found, it indicates that this solution is also valuable for other particles to learn from, thus increasing its priority. This can be illustrated in Fig. 4(A).

The process shown in Fig. 4(A) enables the particles to acquire the ability to exchange information about the states of candidate solutions, which is denoted as:

$$P_{h,i(k)}^{k} = P_{h,i(k)}^{k} + ad, h = 1, 2, \ldots, N, \tag{10}$$

where *ad* denotes a constant; $i(k)$ is the target sample point id of the ith particle selected according to Eq. (9); N is the particle population size.

When the priority of a candidate solution is raised by Eq. (10), it will attract other particles to learn that candidate solution through the priority-driven search strategy, and this process increases the probability of priority raising of that candidate solution to some extent. To prevent a candidate solution in the candidate solution set from being over-learned, when an ith particle learns the $i(k)$ candidate solution $\boldsymbol{CS}_{i(k)}^{k}$ in the candidate solution set $\boldsymbol{CS}^{k}$ through a priority-driven search strategy, it reduces that priority by a random percentage, which is denoted as:

$$P_{i,i(k)}^{k+1} = ceil\left(r \times P_{i,i(k)}^{k}\right), \tag{11}$$

where $i(k)$ is the sample point id of the ith particle selected according to Eq. (9), *ceil()* denotes round up to an integer, and r is a random number between (0,1).

PDPSO maintains the diversity of the algorithm population by periodically visiting those candidate solutions with low priority, thus reducing the possibility of the algorithm falling into a local optimum. This process is expressed as:

$$P_{i,j}^{k+1} = P_{i,j}^{k} + 1, j \in \left\{1, 2, \ldots, Ncs^{k}\right\}, j \neq i(k), \tag{12}$$

where Ncs^{k} denotes the capacity of the candidate solution set at the kth iteration; $i(k)$ is the sample point id of the ith particle selected according to Eq. (9).

Eq. (10) and (11) work together to control the priority range of candidate solutions, as depicted in Fig. 4(B). since Eq. (11) reduces the priority value corresponding to the candidate solution by a percentage each time the candidate solution is visited if the priority value cannot be replenished by Eq. (10) or by other ways, this candidate solution will slowly be abandoned. Meanwhile, Eq. (10) updates priorities based on a particle's better position, encouraging the population to focus on high-success solutions. Suppose a solution's priority is too high. In that case, it increases its selection probability by particles, resulting in a higher likelihood of priority reduction due to Eq. (11). This co-adjustment allows for maintaining priorities within a reasonable range.

The entire set of candidate solutions can be rationally managed by Eq. (10) and Eq. (11) acting together with Eq. (12). If a particle finds a better position, it attracts followers from other solutions (according to Eq. (10)). When the population stagnates, the priority of that solution decreases rapidly (according to Eq. (11)). Then, some particles will explore other candidates (according to Eq. (12)). Trying other less visited candidates generates diverse decision variables, shifting the search away from local optima. After a particle visits a candidate solution, the value of the corresponding position will be retained according to Eq. (11), so even if the population is stagnant for a while, some of the followers of a candidate solution will keep learning the candidate solution. Therefore, the candidate solution will only be regarded as worthless if it stagnates for an extended period, and particles will gradually abandon the exploitation of the point in the process and follow other candidate solutions with high exploitation priority.

A. Dynamic candidate solution elimination

The above analysis identifies two behaviors in agent individuals during the update process: forming small groups to develop fixed areas and exploring unknown regions by changing their development focus. By reducing the quantity of developable area, agents' opportunities for exploration can be limited, transitioning the algorithm from exploration to development.

Since the role of the candidate solution set in PDPSO is comparable to the global optimal position in PSO, the update of candidate solutions in the candidate solution set is denoted as:

$$\boldsymbol{CS}_{i(k)}^{k} = \begin{cases} \boldsymbol{x}_{i}^{k} & Cost\left(\boldsymbol{x}_{i}^{k}\right) \prec Cost\left(\boldsymbol{CS}_{i(k)}^{k}\right) \\ \boldsymbol{CS}_{i(k)}^{k} & else \end{cases}, \tag{13}$$

where $\boldsymbol{x}_{i}^{k}$ denotes the position of the ith particle after the kth iteration; and $Cost(\Delta)$ denotes the fitness value of Δ.

Eq. (13) for maintaining the optimal position is identical to the PSO algorithm, but PDPSO retains multiple global optimal positions (i.e., candidate solution positions). When the candidate solution set capacity is 1, PDPSO degenerates into the PSO algorithm without inertial velocity.

In the early stages of the algorithm, a larger candidate solution capacity enables the population to quickly approach the optimal solution while minimising incorrect searches. However, as the focus shifts from exploration to exploitation, this larger capacity can hinder optimization by over-exploiting single candidate solutions. To address this, we implement a dynamic candidate solution management strategy that balances exploration and exploitation by periodically adjusting the candidate solution set capacity.

The capacity of the candidate solution set is denoted as:

$$Ncs^{t} = round\left(1 + \left(Ncs^{0} - 1\right) \times \left(1 - \frac{t}{T}\right)\right), \tag{14}$$

where *round*() indicates that the number in parentheses is rounded to the nearest integer, Ncs^{0} indicates the starting capacity of the candidate solution set, t and T represent the current and maximum number of iterations, respectively.

When the capacity of the candidate solution set decreases, the candidate solution with the worst fitness value is deleted, denoting

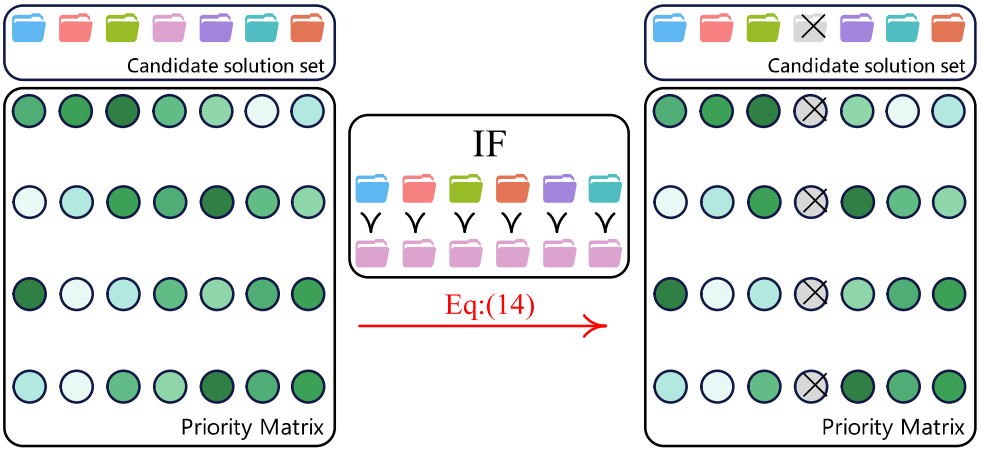


Fig. 5. Diagram of the dynamic candidate solution elimination process.

that this candidate solution is eliminated, and this process is shown in Fig. 5.

A. Particle state rollback

In practical optimization problems, solutions with better fitness values are typically clustered in some regions, which are denoted by color shading in Fig. 6. Here, we record the position of some of the regions through the candidate solution set and the population individuals search the neighboring regions of a candidate solution through the priority-driven search strategy based on the priority table. When an individual tries to learn those candidate solutions that are not frequently learned by this individual, a large number of non-continuous decision variables are generated, and this process is shown in Fig. 6(A). This process can change the original search pattern and make the particle jump out of the local optimum, as shown in Fig. 6(B, C).

In Fig. 6, the shaded areas indicate where solutions with better fitness values are concentrated, giving algorithms in these regions a higher chance of finding improved solutions. As the optimization progresses, candidate solutions improve, leading to smaller shaded regions. Furthermore, according to Eq. (14), the capacity of the candidate solution set decreases to 1, transitioning the algorithm from exploration to exploitation and ultimately resembling a PSO algorithm without inertial velocity. This reduction in capacity reduces the algorithm's ability to escape low-dimensional hyperplanes. To prevent particles from losing their target and being delayed in returning to the original solution area, we implement a particle state rollback strategy within the dynamic candidate management; this execution process is shown in Fig. 6(D).

When a particle switches the search target, an optimization stagnation counter for this particle is started, and the entire state of this particle is recorded. When the counter reaches the rollback threshold Ts_i^k, the particle returns to the previously recorded state. The previously recorded state refers to the agent's position and fitness value before it stagnated. Updating the agent will overwrite this record with its current position and fitness value. The Ts_i^k adaptively adjusted according to the optimization state of the particle, the expression of Ts_i^k is:

$$Ts_i^{k+1} = \begin{cases} 50 & t = 0 \\ \max\left(50, Ts_i^k \times tu\right) & Cost\left(\boldsymbol{x}_i^k\right) \prec Cost\left(\boldsymbol{x}_i^{k+1}\right), \\ Ts_i^k \times td & else \end{cases} \tag{15}$$

where tu and td are the particle rollback adjustment parameters; $\boldsymbol{x}_i^k$ denotes the position of the ith particle after the kth iteration; and $Cost(\Delta)$ denotes the fitness value of Δ.

For most algorithms, including PDPSO, the fitness value of the optimal solution decreases quickly in the early stages of optimization. As the process continues, unless the optimal solution is found early, the decline rate slows or stagnates due to algorithm limitations and problem difficulty. Thus, monitoring fitness value changes supports appropriate rollbacks. Since this strategy mainly operates in the later stages of execution, it is not sensitive to the initial parameter values. In Eq. (15), the number 50 can be appropriately selected to prevent the algorithm from triggering the rollback strategy so frequently. The rollback threshold affects the activity range of particles outside the candidate solution region, so the rollback tuning parameter adopts a standard boundary expansion contraction parameter. Herein, $tu = 1.05$, $td = 0.95$. After a particle triggers the state rollback strategy to return to a previously recorded state to prevent it from switching again to this high-priority candidate solution and thus searching the region where the last search failed again, it executes Eq. (11) after triggering the rollback, therefore decreasing the priority of this candidate solution.

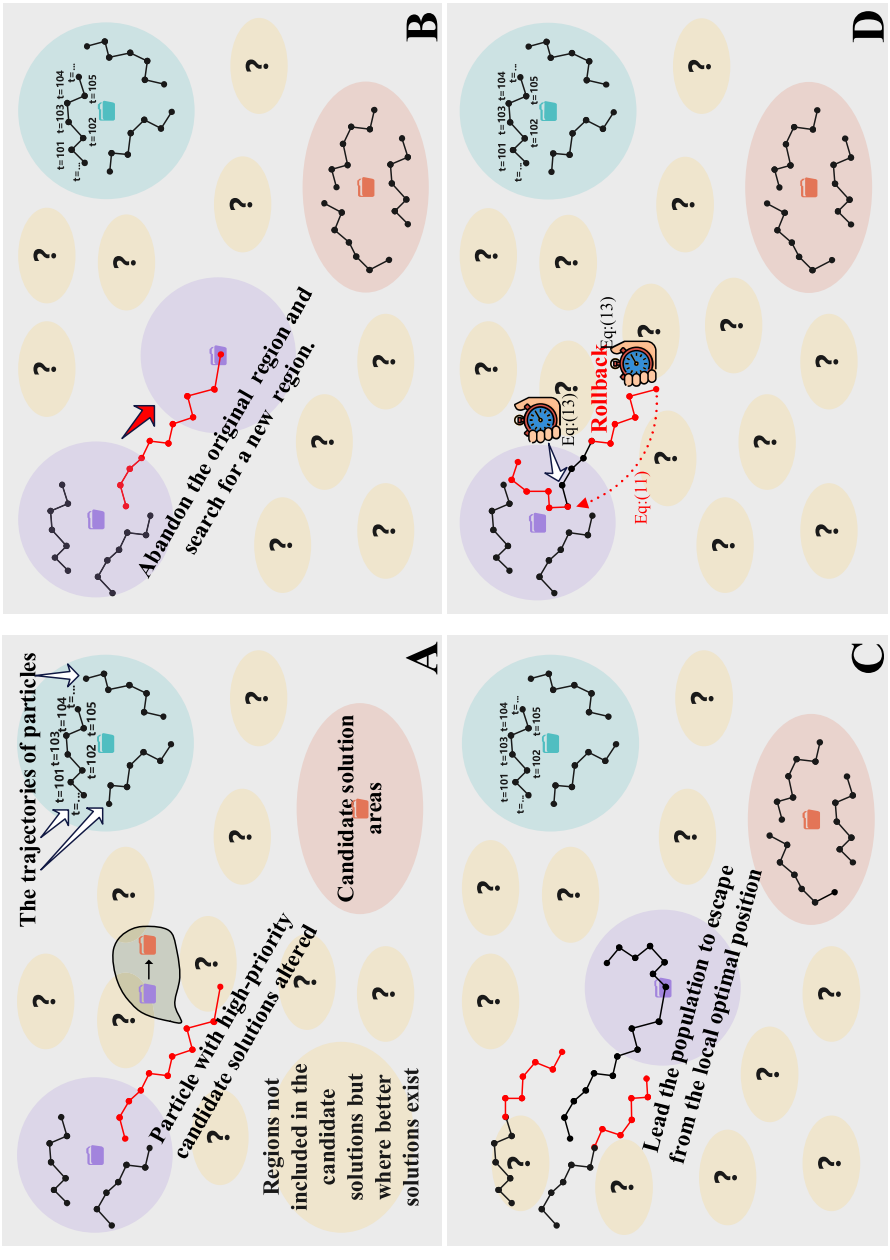


Fig. 6. Schematic of the particle trajectories around the candidate solution.

Table 2
MSE and SNR statistics for different parameter combinations (D=50).

$c_1\downarrow$	$c_2\downarrow$	MSE $ad\rightarrow$ 0.15	SNR	MSE 0.17	SNR	MSE 0.19	SNR	MSE 0.21	SNR	MSE 0.23	SNR	MSE 0.25	SNR	MSE 0.27	SNR
0.3	2.7	0.475	0.425	0.465	0.421	0.455	0.430	0.509	0.395	0.478	0.419	0.463	0.415	0.472	0.417
	2.9	0.349	0.529	0.373	0.517	0.343	0.527	0.346	0.540	0.359	0.515	0.350	0.531	0.350	0.529
	3.1	0.308	0.577	0.300	0.589	0.269	0.619	0.342	0.544	0.312	0.579	0.320	0.562	0.350	0.536
	3.3	0.316	0.584	0.345	0.553	0.366	0.529	0.350	0.545	0.302	0.577	0.365	0.532	0.339	0.556
	3.5	0.406	0.499	0.430	0.471	0.435	0.472	0.465	0.445	0.399	0.504	0.450	0.460	0.450	0.451
0.4	2.7	0.458	0.425	0.433	0.458	0.436	0.456	0.464	0.424	0.430	0.465	0.419	0.466	0.416	0.472
	2.9	0.308	0.584	0.314	0.574	0.291	0.597	0.307	0.574	0.272	0.612	0.331	0.548	0.295	0.593
	3.1	0.271	0.630	0.273	0.621	0.291	0.602	0.284	0.614	0.275	0.621	0.257	0.643	0.241	0.646
	3.3	0.285	0.611	0.330	0.564	0.291	0.602	0.294	0.607	0.274	0.627	0.303	0.592	0.280	0.611
	3.5	0.384	0.516	0.381	0.514	0.379	0.513	0.356	0.537	0.354	0.544	0.362	0.529	0.361	0.534
0.5	2.7	0.377	0.512	0.391	0.494	0.392	0.498	0.379	0.504	0.374	0.519	0.395	0.491	0.382	0.513
	2.9	0.247	0.650	0.226	0.680	0.228	0.673	0.244	0.663	0.256	0.649	0.252	0.640	0.229	0.663
	3.1	0.243	0.653	0.225	0.681	0.230	0.670	0.235	0.665	0.236	0.660	0.257	0.642	0.237	0.669
	3.3	0.281	0.617	0.259	0.633	0.273	0.627	0.259	0.644	0.261	0.633	0.295	0.598	0.264	0.633
	3.5	0.352	0.541	0.359	0.533	0.360	0.528	0.384	0.505	0.374	0.514	0.370	0.521	0.339	0.547
0.6	2.7	0.325	0.574	0.340	0.556	0.333	0.562	0.352	0.541	0.309	0.580	0.337	0.548	0.351	0.537
	2.9	0.244	0.661	0.262	0.642	0.270	0.625	0.194	0.708	0.229	0.683	0.261	0.634	0.248	0.660
	3.1	0.195	0.714	0.232	0.675	0.235	0.666	0.258	0.636	0.242	0.657	0.244	0.660	0.271	0.637
	3.3	0.282	0.611	0.292	0.606	0.303	0.592	0.274	0.627	0.249	0.645	0.326	0.576	0.295	0.598
	3.5	0.363	0.523	0.394	0.497	0.403	0.492	0.404	0.482	0.360	0.533	0.416	0.469	0.389	0.494
0.7	2.7	0.271	0.623	0.324	0.578	0.287	0.604	0.295	0.596	0.260	0.632	0.273	0.631	0.285	0.628
	2.9	0.252	0.659	0.226	0.682	0.267	0.639	0.237	0.670	0.236	0.671	0.233	0.676	0.225	0.685
	3.1	0.249	0.667	0.284	0.614	0.266	0.640	0.282	0.616	0.262	0.648	0.255	0.650	0.253	0.655
	3.3	0.393	0.515	0.343	0.556	0.310	0.584	0.370	0.532	0.410	0.487	0.302	0.591	0.386	0.518
	3.5	0.551	0.359	0.499	0.402	0.554	0.366	0.517	0.393	0.552	0.359	0.522	0.378	0.553	0.356

Table 3
MSE and SNR statistics for different parameter combinations (D=100).

		MSE	SNR	MSE	SNR	MSE	SNR	MSE	SNR	MSE	SNR	MSE	SNR	MSE	SNR
$c_1\downarrow$	$c_2\downarrow$	$ad\rightarrow$ 0.15		0.17		0.19		0.21		0.23		0.25		0.27	
0.3	2.7	0.523	0.381	0.496	0.388	0.493	0.394	0.516	0.378	0.525	0.367	0.542	0.354	0.552	0.329
	2.9	0.343	0.537	0.387	0.484	0.382	0.498	0.365	0.507	0.375	0.505	0.359	0.507	0.359	0.502
	3.1	0.307	0.552	0.299	0.576	0.285	0.587	0.269	0.600	0.278	0.593	0.269	0.612	0.287	0.579
	3.3	0.309	0.563	0.278	0.603	0.307	0.561	0.292	0.583	0.294	0.583	0.297	0.580	0.279	0.597
	3.5	0.342	0.533	0.333	0.527	0.325	0.538	0.352	0.512	0.405	0.465	0.361	0.503	0.346	0.523
0.4	2.7	0.440	0.447	0.426	0.468	0.447	0.420	0.468	0.422	0.447	0.442	0.450	0.449	0.444	0.446
	2.9	0.283	0.596	0.264	0.625	0.274	0.607	0.277	0.599	0.283	0.599	0.298	0.585	0.280	0.608
	3.1	0.216	0.659	0.203	0.680	0.212	0.679	0.198	0.699	0.222	0.657	0.192	0.696	0.189	0.697
	3.3	0.229	0.648	0.223	0.655	0.219	0.659	0.197	0.697	0.201	0.690	0.211	0.673	0.224	0.663
	3.5	0.281	0.594	0.262	0.615	0.245	0.630	0.267	0.599	0.275	0.593	0.292	0.575	0.258	0.606
0.5	2.7	0.417	0.467	0.404	0.479	0.405	0.481	0.421	0.481	0.415	0.477	0.372	0.510	0.385	0.508
	2.9	0.238	0.653	0.244	0.641	0.246	0.641	0.266	0.619	0.235	0.646	0.245	0.642	0.242	0.649
	3.1	0.174	0.724	0.207	0.683	0.181	0.717	0.148	0.752	0.196	0.704	0.168	0.737	0.168	0.728
	3.3	0.188	0.719	0.233	0.663	0.180	0.718	0.185	0.701	0.200	0.695	0.182	0.707	0.188	0.718
	3.5	0.252	0.615	0.223	0.648	0.258	0.608	0.247	0.628	0.235	0.639	0.257	0.616	0.236	0.633
0.6	2.7	0.392	0.489	0.388	0.494	0.374	0.513	0.364	0.525	0.347	0.551	0.401	0.489	0.352	0.534
	2.9	0.234	0.659	0.232	0.663	0.218	0.673	0.216	0.675	0.219	0.683	0.201	0.699	0.212	0.680
	3.1	0.166	0.741	0.146	0.770	0.166	0.733	0.164	0.745	0.166	0.742	0.179	0.734	0.174	0.725
	3.3	0.183	0.699	0.208	0.676	0.197	0.681	0.193	0.696	0.178	0.714	0.188	0.711	0.184	0.704
	3.5	0.313	0.542	0.281	0.586	0.321	0.540	0.288	0.564	0.262	0.593	0.282	0.573	0.292	0.566
0.7	2.7	0.358	0.529	0.321	0.559	0.336	0.540	0.357	0.530	0.338	0.550	0.332	0.541	0.334	0.548
	2.9	0.218	0.662	0.201	0.683	0.231	0.656	0.209	0.669	0.194	0.711	0.203	0.697	0.222	0.669
	3.1	0.213	0.669	0.184	0.714	0.174	0.728	0.209	0.694	0.223	0.667	0.189	0.714	0.195	0.718
	3.3	0.250	0.622	0.289	0.591	0.262	0.615	0.255	0.619	0.256	0.638	0.279	0.599	0.252	0.620
	3.5	0.523	0.381	0.496	0.388	0.493	0.394	0.516	0.378	0.525	0.367	0.542	0.354	0.552	0.329

3.2. Complexity analysis

The computational complexity of PDPSO is related to the initialization process, candidate solution update, fitness value evaluation, and parameter adjustment. In the initialization process, the computational complexity is $O(N \times D)$, the fitness value is computed once for each position individually, and the computational complexity is $O(N)$. In the position update stage the computational complexity is $O(3 \times T \times N \times D)$. In the priority table update stage, the computational complexity is $O\left(T \times N \times \left(N + Ncs^0 - 1\right)\right)$. After each iteration, the candidate solution set capacity needs to be recalculated, and the computational complexity is $O(T)$. Therefore, the computational complexity of PDPSO is

$$O(\text{PDPSO}) = O\left(T \times N \times \left(N + 3 \times D + Ncs^0 - 1\right) + N(D+1) + T\right), \tag{16}$$

where N is the population size, T is the maximum number of iterations, D is the dimension, Ncs indicates the start capacity of the candidate solution set.

4. PDPSO numerical experiment

This section presents numerical simulation experiments using the PDPSO algorithm. To ensure a fair comparison, all experiments utilized the same settings: "Windows 10 (64-bit) 23H2," "i5-13600KF CPU@3.50 GHz 32GB." Each algorithm was run multiple times per experiment, and results were averaged to determine final performance metrics.

We chose the CEC2017 benchmark test suite for PDPSO numerical experiments because it includes 30 high-dimensional test functions (50 and 100 D) designed to simulate various real-world optimization problems. F2 was removed due to instability, leaving 29 functions categorized as follows: F1-F3 are unimodal functions that test solution accuracy; F4-F10 are multimodal functions focused on exploratory capabilities; F11-F20 are hybrid functions; and F21-F30 are composition functions that assess algorithm effectiveness in complex search environments.

There is no single standard for parameter settings in comparative experiments, although the CEC competition rules specify a maximum number of evaluations [42]. For extremely expensive optimization problems, standard evaluation limits include $\frac{11}{10} \times D$, 100, 300, 400, 1000, and 2000 [95–99], whereas normal engineer optimization problems often use 15,000 to 3,000,000 evaluations [40,42,43,45,52,100–107]. Duankhan et al. suggest that a population size of 30 is adequate for testing algorithm performance, with at least 30,000 fitness evaluations [42]. In this paper, we use a population size of 30 and 1,000 iterations for algorithm comparison.

4.1. Parameter adjustment experiment

This experiment seeks to enhance the performance of PDPSO by adjusting its key parameters: c_1, c_2, and ad. Before adjusting the parameters, we determine the upper range of their optimal values through extensive random parameter selection. We found that our algorithm performs best with $c_1 = 0.5$, $c_2 = 3.1$, and $ad = 0.21$. We then use grid search to identify optimal parameter values around these figures and verify our choices through cross-validation.

In this experiment, c_1 has values of [0.3 0.4 0.5 0.6 0.7], c_2 has values of [2.7 2.9 3.1 3.3 3.5], and ad has values of [0.15 0.17 0.19 0.21 0.23 0.25 0.27], resulting in 175 parameter combinations. Each combination was tested 30 times across 50 and 100 dimensions of CEC2017 to analyze the optimization capability and stability. The mean square error (Mse) and signal-to-noise ratio (Snr) were used to measure the performance since only parameter differences were present.

A smaller Mse results in a better optimization ability of the algorithm. The value of Mse in a set of experiments is denoted as:

$$mse = \frac{1}{n}\sum_{i=1}^{n}(y_i - u)^2 \tag{17}$$

where y_i denotes the optimization result in the ith experiment; u denotes the theoretical optimum of this problem; and n denotes the repetition times of the experiment.

The MSE of each algorithm on the test functions is normalized using Min-Max since only parameter differences were present. The average normalized MSE across 29 functions determines the overall score for this parameter combination.

A smaller Snr results in a better stability of the algorithm. The value of Snr in a set of experiments is denoted as:

$$snr = -10\log_{10}\left(\frac{1}{m}\sum_{j=1}^{m}\left(mse_j\right)^2\right) \tag{18}$$

where mse_j denotes the mean square error in the jth set of experiments and m denotes the number of replicate sets of experiments. The 30 experiment repetitions were split into 6 groups to calculate SNR values. The normalized SNR values, noted as SNR, were averaged across 29 test functions to determine the stability of the parameter combinations.

The MSE and SNR statistical results for different parameter combinations on the 50 and 100-dimensional test functions are shown in Table 2 and Table 3, where the bolded fonts denote the 20 results with the strongest optimization ability and the 20 results with the best stability.

Tables 2 and 3 show that the optimal value of parameter c_1 may be 0.5, 0.6, or 0.7, and the optimal value of parameter c_2 may be 3.1

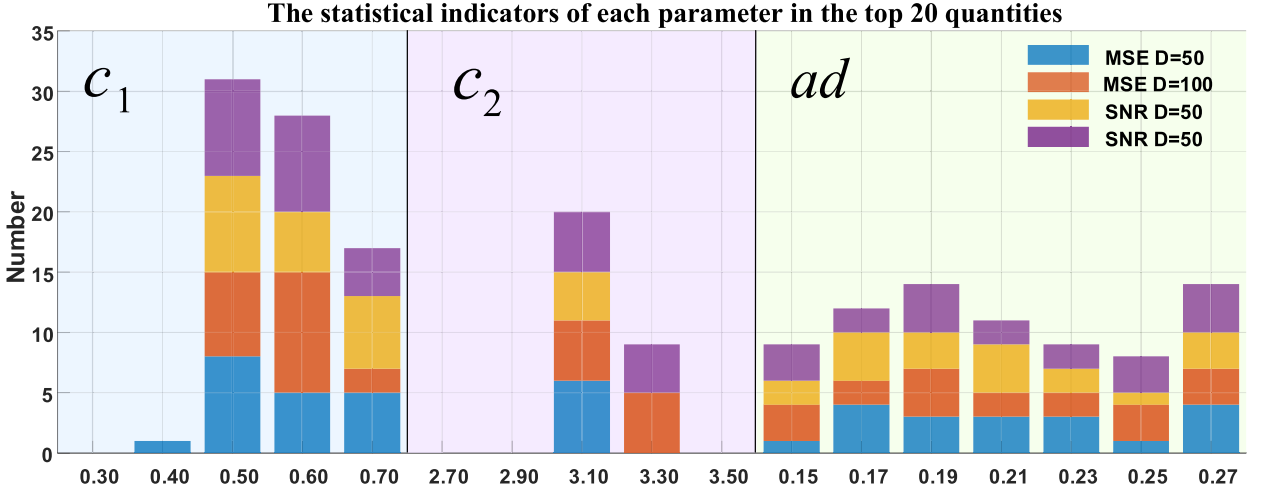


Fig. 7. Distribution of MSE and SNR values among the top 20.

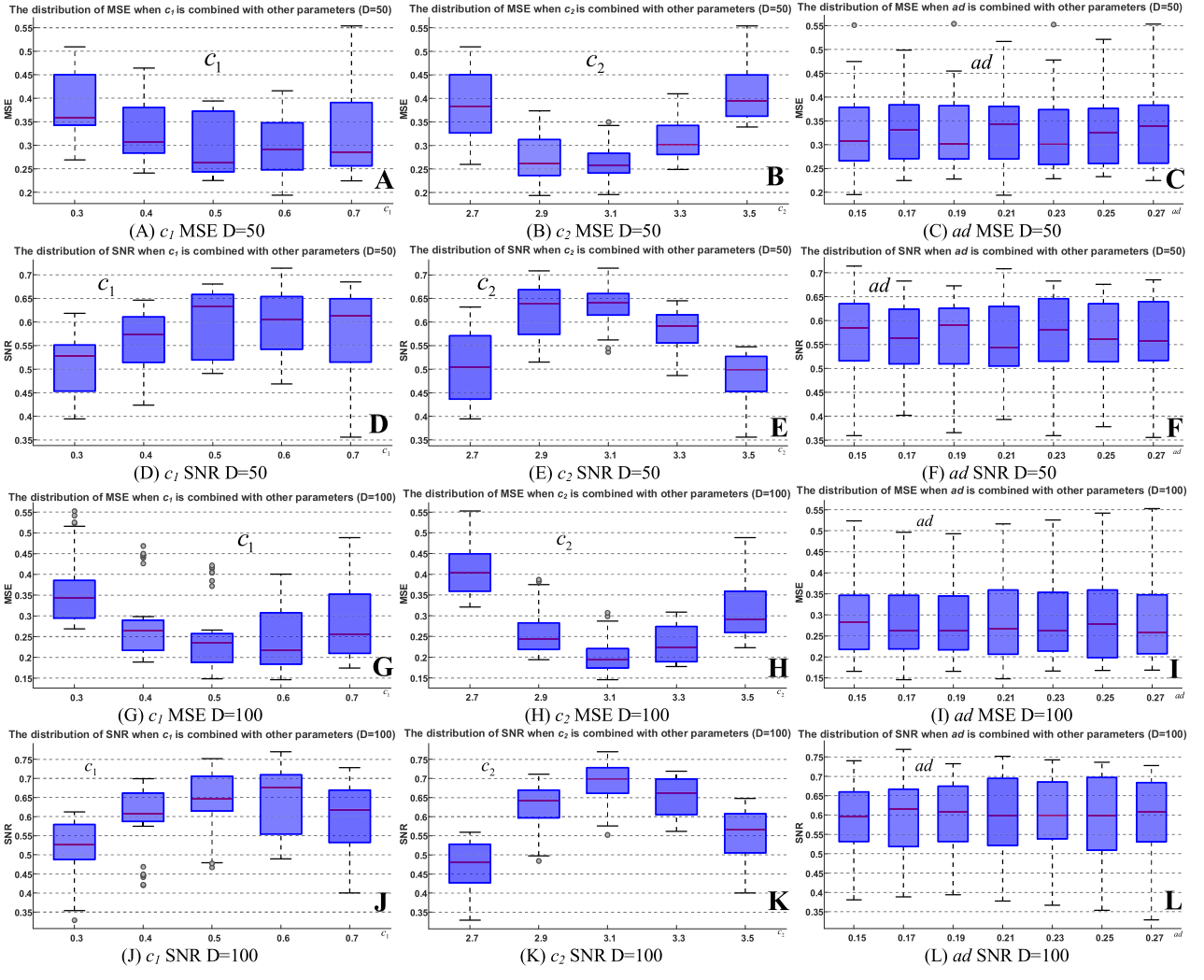


Fig. 8. Box plots of the distributions of MSE and SNR after fixing a certain parameter.

Table 4
The experimental data in ablation experiment (D=50).

		CEC01	CEC03	CEC04	CEC05	CEC06	CEC07	CEC08	CEC09	CEC10	CEC11
MSE	PSO	1.016E-06	3.906E-01	2.738E-04	1.026E-01	4.107E-02	2.283E-02	9.122E-02	1.119E-01	1.243E-01	2.612E-03
	PDPSO_El	8.195E-04	3.786E-01	7.783E-04	3.581E-01	4.303E-02	4.484E-02	3.646E-01	4.256E-03	9.027E-01	7.832E-03
	PDPSO	0	0	0	0	0	0	0	0	0	0
	PDPSOw	1	1	1	1	1	1	1	1	1	1
SNR	PSO	4.218E-01	3.457E-01	8.414E-01	5.263E-01	4.246E-01	6.525E-01	5.514E-01	2.458E-01	7.753E-01	5.983E-01
	PDPSO_El	2.172E-01	3.562E-01	7.474E-01	2.455E-01	4.184E-01	5.443E-01	2.416E-01	6.095E-01	5.554E-02	4.888E-01
	PDPSO	1	1	1	1	1	1	1	1	1	1
	PDPSOw	0	0	0	0	0	0	0	0	0	0
RunTime	PSO	10.47 s	10.42 s	10.84 s	13.41 s	23.13 s	13.67 s	14.19 s	13.32 s	16.07 s	11.43 s
	PDPSO_El	14.64 s	15.34 s	15.49 s	18.35 s	29.08 s	18.59 s	19.10 s	18.04 s	21.13 s	16.08 s
	PDPSO	14.38 s	14.23 s	15.19 s	17.65 s	28.04 s	18.13 s	18.45 s	17.02 s	20.08 s	15.40 s
	PDPSOw	19.05 s	17.61 s	18.26 s	21.66 s	30.23 s	22.32 s	22.54 s	21.78 s	24.79 s	19.46 s
p-value	PSO	7.066E-18	1.013E-17	2.954E-17	7.066E-18	7.066E-18	7.066E-18	1.075E-17	7.066E-18	3.122E-02	7.066E-18
	PDPSO_El	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18
	PDPSOw	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18

		CEC12	CEC13	CEC14	CEC15	CEC16	CEC17	CEC18	CEC19	CEC20	CEC21
MSE	PSO	5.627E-07	2.753E-12	2.845E-02	4.655E-12	9.955E-02	2.271E-02	1.852E-02	5.538E-11	1.394E-01	1.653E-01
	PDPSO_El	1.317E-04	1.402E-12	1.627E-01	0	4.555E-01	1.577E-01	5.855E-02	0	6.747E-01	4.441E-01
	PDPSO	0	0	0	1.830E-12	0	0	0	7.695E-11	0	0
	PDPSOw	1	1	1	1	1	1	1	1	1	1
SNR	PSO	7.608E-01	9.213E-01	8.881E-01	9.920E-01	6.807E-01	8.870E-01	4.840E-01	9.864E-01	7.237E-01	6.094E-01
	PDPSO_El	4.727E-01	9.415E-01	5.828E-01	1	2.639E-01	5.490E-01	3.450E-01	1	1.868E-01	3.049E-01
	PDPSO	1	1	1	9.966E-01	1	1	1	9.820E-01	1	1
	PDPSOw	0	0	0	0	0	0	0	0	0	0
RunTime	PSO	13.89 s	12.50 s	19.49 s	11.69 s	12.89 s	19.39 s	12.00 s	49.18 s	19.65 s	23.38 s
	PDPSO_El	18.46 s	16.34 s	24.80 s	15.76 s	17.93 s	23.90 s	17.89 s	53.48 s	24.19 s	28.73 s
	PDPSO	17.71 s	16.50 s	23.37 s	15.33 s	16.85 s	23.34 s	15.99 s	50.91 s	25.58 s	27.32 s
	PDPSOw	22.96 s	21.07 s	28.72 s	18.93 s	20.35 s	25.77 s	19.42 s	53.48 s	26.26 s	29.91 s
p-value	PSO	2.784E-17	9.679E-07	1.364E-03	7.020E-01	3.039E-08	2.680E-03	2.560E-15	6.824E-02	3.329E-04	7.504E-18
	PDPSO_El	7.066E-18	6.976E-13	1.366E-17	9.478E-01	7.066E-18	1.075E-17	7.066E-18	1.075E-01	1.212E-17	7.066E-18
	PDPSOw	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18

		CEC22	CEC23	CEC24	CEC25	CEC26	CEC27	CEC28	CEC29	CEC30	AVERAGE
MSE	PSO	2.888E-01	1.636E-01	3.181E-01	9.700E-04	1.069E-01	1.414E-01	5.917E-03	3.089E-02	1.007E-06	8.338E-02
	PDPSO_El	3.703E-01	3.590E-01	3.733E-01	2.694E-03	2.682E-01	5.311E-02	5.754E-03	1.540E-01	9.388E-05	1.947E-01
	PDPSO	0	0	0	0	0	0	0	0	0	2.717E-12
	PDPSOw	1	1	1	1	1	1	1	1	1	1
SNR	PSO	4.881E-01	6.746E-01	4.960E-01	9.371E-01	6.253E-01	6.639E-01	8.545E-01	8.118E-01	8.813E-01	6.810E-01
	PDPSO_El	4.030E-01	4.343E-01	4.381E-01	8.639E-01	3.928E-01	8.446E-01	8.573E-01	5.004E-01	5.993E-01	5.140E-01
	PDPSO	1	1	1	1	1	1	1	1	1	9.993E-01
	PDPSOw	0	0	0	0	0	0	0	0	0	0
RunTime	PSO	24.06 s	28.75 s	30.86 s	25.60 s	33.57 s	35.99 s	31.94 s	25.40 s	54.15 s	31.37 s
	PDPSO_El	30.48 s	34.26 s	35.98 s	30.35 s	38.48 s	40.97 s	37.09 s	30.21 s	58.65 s	36.52 s
	PDPSO	30.95 s	29.46 s	34.99 s	29.71 s	37.08 s	40.07 s	36.17 s	29.32 s	57.28 s	35.24 s
	PDPSOw	34.10 s	35.14 s	36.96 s	33.52 s	39.52 s	44.25 s	35.21 s	34.91 s	60.49 s	38.40 s

(*continued on next page*)

Table 4 (*continued*)

		CEC22	CEC23	CEC24	CEC25	CEC26	CEC27	CEC28	CEC29	CEC30	AVERAGE
p-value	PSO	1.031E-09	7.066E-18	7.066E-18	6.338E-17	4.653E-13	1.287E-17	7.066E-18	8.325E-08	9.540E-18	
	PDPSO_El	3.242E-04	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.271E-15	7.066E-18	7.066E-18	7.066E-18	
	PDPSOw	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	

or 3.3. Due to the large amount of data in Tables 2 and 3, to facilitate the analysis of the selection of the optimal parameter combinations, we counted the distributions of the top 20 of the MSE and SNR, and the distributions are shown in Fig. 7.

From Fig. 7, for parameters c_1 and c_2, the optimal value of parameter c_1 may be 0.5 or 0.6, and the optimal value of parameter c_2 may be 3.1 or 3.3 since its statistics perform best when these values are taken, and the number of MSE and SNR in the top 20 decreases significantly as the values are increased or decreased. For parameter *ad*, there is a significant bulge at 0.19 in the top 20 statistical indicators, which deserves our attention.

To finalize the parameter combinations, a cross-validation approach was used to validate each parameter's selection. This method analyzes the distribution of MSE and SNR for a fixed parameter combined with others. The box plot of this distribution is shown in Fig. 8, with gray points representing outliers.

For parameter c_1, although the MSE's box at $c_1 = 0.5$ is slightly higher than the case of $c_1 = 0.6$ and the SNR's box is slightly lower than the case of $c_1 = 0.6$, there are more outliers in 100 dimensions, and the MSE's outlier is higher than the maximum value when $c_1 = 0.6$, and the SNR's outlier is lower than the minimum when $c_1 = 0.6$, which means this algorithm may be unstable when $c_1 = 0.6$ is selected, and therefore we chose $c_1 = 0.5$. For parameter c_2, the algorithm performs best in terms of both optimization ability and stability when $c_2 = 3.1$, and even though some outliers exist, these outliers outperform the worst-case scenario when other parameters are selected. For parameter *ad*, Fig. 7 indicates that the algorithm may work better at $ad = 0.19$. However, since in Fig. 8(C), there are inferior outliers in MSE when $ad = 0.19$, while $ad = 0.21$ possesses a better minimum MSE value in both dimensions, along with a higher box for its SNR compared to $ad = 0.17$, so we chose $ad = 0.21$ as the final parameter choice. Ultimately, we chose $c_1 = 0.6$, $c_2 = 3.1$, and $ad = 0.21$ as the optimal parameter combinations.

4.2. Ablation experiment

To verify the effectiveness of the PDPSO algorithm components, we conduct an ablation experiment. In PDPSO, the priority-driven search strategy selects high-priority solutions, and both this strategy and the dynamic candidate solution management are interconnected, making separate ablation experiments unfeasible. We also experimented with the removal of inertial velocity, which is detailed in Section 3, as a separate strategy. As the dynamic candidate solution elimination operates independently, this experiment also isolates its impact. The same experimental metrics from the parameter adjustment experiments are used.

This experiment involves 50 and 100-dimensional functions running 50 iterations each, comparing PSO, PDPSO_El, and PDPSOw against PDPSO. PSO is the classical algorithm used to evaluate the priority-driven search strategy. PDPSO_El lacks candidate solution elimination, while PDPSOw incorporates inertial velocity. Both learning factors are set to 2, with inertia weights decreasing from 0.9 to 0.5. Results for MSE, SNR, total runtime over 30 repetitions, and Wilcoxon tests for significance at a 5% level are presented in Tables 4 and 5, where bold numbers indicate minimum MSE and maximum SNR values.

In comparing the MSE and SNR values of PSO and PDPSO_El, the results show that for 50-dimensional test functions, PDPSO had a smaller MSE than PSO, PDPSO_El, and PDPSOw in 93.1% of cases. Additionally, PDPSO exhibited a higher SNR than PSO and PDPSO_El in 93.1% of the test functions, with significant differences in most optimization results. For 100-dimensional test functions, PDPSO consistently outperformed the other algorithms in MSE and SNR across all tests. These findings indicate that the PDPSO algorithm demonstrates the best optimization performance and stability, benefiting significantly from candidate elimination and the removal of inertial velocity. Fig. 9 illustrates the total runtime for each algorithm across 30 runs, as detailed in Table 4 and Table 5.

Combining Tables 4, 5, and Figure 9, we find that PSO has the shortest runtime on the 50-dimensional test function. In comparison, PDPSO_El, PDPSO, and PDPSOw have average runtimes that increase by 16.4%, 9.2%, and 22.4%, respectively. For the 100-dimensional function, these values are 6.0%, 4.0%, and 8.8%. Overall, PDPSO adds minimal complexity while significantly improving optimization performance.

The experimental results indicate that PDPSO outperforms other algorithms in optimization capability and stability, confirming the effectiveness of its strategy combination. In contrast, PDPSOw performs the worst due to a contradiction in its search mechanisms, (Both inertial velocity and changing search targets boost population diversity, helping agents escape local optima) which leads to chaotic agent velocities and reduced performance. While comparing PDPSOEl with PSO, PSO generally performs better since PDPSOEl splits the population into smaller sub-populations with limited information exchange, resulting in an inferior overall effect. PDPSO and PDPSPEl can quickly identify regions of high-quality solutions due to multiple subgroups of candidate solutions, allowing them to bypass local optima. However, in later stages, PDPSPEl fails to reduce these subgroups effectively, limiting its ability to concentrate on high-quality solutions. Consequently, its overall performance is inferior to PSO and PDPSO. Thus, PDPSO is innovative in design, and its strategy combination both reasonable and essential.

Adding optimization strategies to an algorithm generally extremely increases time complexity. However, Tables 4, 5, and Fig. 9

Table 5
The experimental data in ablation experiment (D=100).

		CEC01	CEC03	CEC04	CEC05	CEC06	CEC07	CEC08	CEC09	CEC10	CEC11
MSE	PSO	9.585E-04	4.870E-01	7.053E-04	2.436E-01	1.467E-01	4.268E-02	2.577E-01	2.389E-01	5.041E-01	1.222E-01
	PDPSO_El	6.248E-03	2.914E-01	1.146E-03	3.497E-01	8.496E-02	4.233E-02	3.393E-01	1.477E-02	9.393E-01	2.063E-01
	PDPSO	0	0	0	0	0	0	0	0	0	0
	PDPSOw	1	1	1	1	1	1	1	1	1	1
SNR	PSO	4.495E-01	3.682E-01	6.894E-01	3.636E-01	3.965E-01	6.005E-01	3.446E-01	2.223E-01	3.453E-01	3.621E-01
	PDPSO_El	3.283E-01	5.676E-01	6.445E-01	2.733E-01	5.036E-01	6.020E-01	2.768E-01	6.407E-01	3.487E-02	2.728E-01
	PDPSO	1	1	1	1	1	1	1	1	1	1
	PDPSOw	0	0	0	0	0	0	0	0	0	0
RunTime	PSO	19.91 s	19.90 s	20.63 s	25.68 s	45.98 s	25.98 s	27.00 s	25.73 s	30.38 s	22.38 s
	PDPSO_El	24.27 s	24.65 s	25.49 s	30.41 s	51.56 s	31.03 s	32.77 s	30.73 s	36.16 s	27.59 s
	PDPSO	23.10 s	23.71 s	24.31 s	29.38 s	51.17 s	30.25 s	31.19 s	29.51 s	34.60 s	26.00 s
	PDPSOw	27.71 s	27.77 s	28.42 s	33.37 s	53.74 s	34.15 s	35.37 s	34.14 s	40.05 s	30.49 s
p-value	PSO	7.066E-18	5.978E-17	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	6.687E-16	7.066E-18
	PDPSO_El	7.066E-18	4.695E-15	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18
	PDPSOw	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18

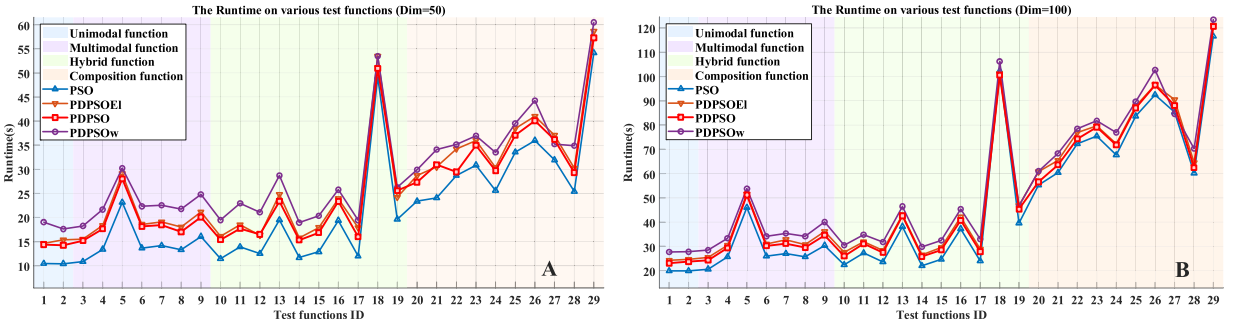
		CEC12	CEC13	CEC14	CEC15	CEC16	CEC17	CEC18	CEC19	CEC20	CEC21
MSE	PSO	6.417E-05	2.737E-12	6.807E-04	3.432E-11	1.027E-01	4.485E-04	1.034E-02	1.252E-12	3.010E-01	2.700E-01
	PDPSO_El	1.522E-03	3.520E-06	2.605E-03	1.360E-10	4.358E-01	2.264E-03	7.932E-02	1.254E-09	8.001E-01	3.653E-01
	PDPSO	0	0	0	0	0	0	0	0	0	0
	PDPSOw	1	1	1	1	1	1	1	1	1	1
SNR	PSO	5.600E-01	8.946E-01	9.222E-01	7.963E-01	6.966E-01	9.362E-01	5.255E-01	9.053E-01	5.735E-01	4.151E-01
	PDPSO_El	3.763E-01	4.226E-01	8.101E-01	7.508E-01	2.936E-01	8.091E-01	2.921E-01	6.785E-01	1.312E-01	3.253E-01
	PDPSO	1	1	1	1	1	1	1	1	1	1
	PDPSOw	0	0	0	0	0	0	0	0	0	0
RunTime	PSO	27.26 s	23.59 s	38.17 s	22.01 s	24.72 s	37.22 s	24.00 s	102.35 s	39.61 s	55.32 s
	PDPSO_El	31.89 s	28.40 s	43.03 s	26.50 s	29.50 s	42.11 s	28.86 s	101.31 s	46.86 s	60.73 s
	PDPSO	31.07 s	27.49 s	42.57 s	25.80 s	28.54 s	40.72 s	27.80 s	100.53 s	45.31 s	56.59 s
	PDPSOw	34.81 s	31.77 s	46.48 s	29.82 s	32.47 s	45.39 s	32.96 s	106.20 s	46.64 s	60.98 s
p-value	PSO	7.066E-18	3.525E-17	3.725E-11	2.469E-07	2.122E-11	3.828E-07	1.539E-17	5.455E-06	3.477E-05	7.066E-18
	PDPSO_El	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	1.450E-17	7.066E-18
	PDPSOw	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	8.462E-18	7.066E-18

		CEC22	CEC23	CEC24	CEC25	CEC26	CEC27	CEC28	CEC29	CEC30	AVERAGE
MSE	PSO	5.091E-01	1.983E-01	1.772E-01	2.342E-03	1.533E-01	5.278E-02	1.281E-02	2.562E-03	3.107E-08	1.323E-01
	PDPSO_El	9.420E-01	3.134E-01	1.618E-01	5.671E-03	1.625E-01	6.858E-02	1.301E-02	1.031E-02	5.854E-06	1.945E-01
	PDPSO	0	0	0	0	0	0	0	0	0	0
	PDPSOw	1	1	1	1	1	1	1	1	1	1
SNR	PSO	3.429E-01	5.653E-01	5.706E-01	7.572E-01	5.269E-01	7.290E-01	5.841E-01	8.870E-01	6.421E-01	5.853E-01
	PDPSO_El	3.356E-02	4.269E-01	5.943E-01	6.556E-01	5.125E-01	6.801E-01	5.820E-01	7.339E-01	4.475E-01	4.724E-01
	PDPSO	1	1	1	1	1	1	1	1	1	1
	PDPSOw	0	0	0	0	0	0	0	0	0	0
RunTime	PSO	60.39 s	72.23 s	75.44 s	67.61 s	83.69 s	92.39 s	85.29 s	60.20 s	116.58 s	76.91 s
	PDPSO_El	65.36 s	76.92 s	79.65 s	72.27 s	87.88 s	96.59 s	90.40 s	64.50 s	121.18 s	81.55 s
	PDPSO	63.62 s	74.29 s	79.11 s	71.80 s	87.02 s	96.43 s	88.05 s	62.45 s	120.64 s	80.00 s
	PDPSOw	68.29 s	78.48 s	81.73 s	76.95 s	89.67 s	102.67 s	84.60 s	70.30 s	123.43 s	83.71 s
p-value	PSO	7.486E-16	7.066E-18	7.066E-18	7.066E-18	1.349E-16	7.066E-18	7.066E-18	1.007E-14	7.066E-18	
	PDPSO_El	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	

(*continued on next page*)

Table 5 (*continued*)

	CEC22	CEC23	CEC24	CEC25	CEC26	CEC27	CEC28	CEC29	CEC30	AVERAGE
PDPSOw	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	7.066E-18	



(A) D=50 (B) D=100

Fig. 9. Total Runtime for each algorithm run 30 times on each test function.

Table 6
The parameters setting and related information of the algorithms (1).

Comparison algorithms	*Pop size*	Parameters
LSHADE_SPACMA	$18 \times D$	$Pbest = 0.01, H = 1.4, Arc_{rate} = 5, Fcp = 0.5, c = 0.8$
LSHADE_cnEpSin	$18 \times D$	$\mu F = \mu CR = 0.5, H = 5, freq = 0.5, ps = 0.5, pc = 0.4, NP\min = 4$
AGSK	100	$P = 0.05, Kw_p = [0.85, 0.05, 0.05, 0.05], c = 0.05$
MADDE	30	$P_{qBx} = 0.01, p = 0.08, A_{rate} = 2.23, NP_m = 2, F_0 = 0.2, Cr_0 = 0.2$

Table 7
The parameters setting and related information of the algorithms (2).

Algorithms	*Pop size*	Parameters
XPSO	30	$\omega = [0.9, 0.4], \mu_1 = \mu_2 = \mu_3 = 1.35, \sigma_1 = \sigma_2 = \sigma_3 = 0.1,$ $Stag_{GB} = 0, Stag_{\max} = 5, \eta = 0.2, p = 0.5$
FVICLPSO	30	$Gap = 4, w = 0, \lambda = 0.3$
TAPSO	30	$\omega = 0.7298, P_c = 0.5, P_m = 0.02, M = N/4$
CDLPSO	30	$\omega = [1, 0], c_2 = 2 \times \omega, c_1 = 2 - c_2, c_p = 0.75, c_s = 5 \times \omega$
DSPSO	30	$c_1 = 2.0, F_{\min} = 0.7$
EAPSO	30	$\omega = rand, \lambda_1 = rand, \lambda_2 = rand$
SRPSO	30	$\omega_I = 1.05, \omega_F = 0.5, c_1 = c_2 = 149445$
MFCPSO	30	$K = [N/20], P_i^t, c_{i,1}^t, c_{i,2}^i$Controlled by fuzzy logic control methods
HMPPSO	30	$\beta = 4, \zeta = 0.5, \omega = [0.9, 0.4], c_{1f} = 2.5, c_{1i} = 0.5$
QPSOL	30	$v_{\max} = 6, w_{\max} = 0.9, w_{\min} = 0.2, c_1 = c_2 = 2$
CEPSO	30	$r_1 = 0.129, r_2 = 1 - r_1, w_{\max} = 0.9, w_{\min} = 0.2, c_{\max} = 1.5, c_{\min} = 0.2$
GLS_MPA	30	$c_{\max} = 2500, \alpha = 30, FADs = 0.5, P = 0.5$
KOA_LEOBL	30	$Tc = 3, M0 = 0.1, \lambda = 15$

show the opposite for the PDPSO algorithm compared to the PDPSOw and PDPSO_El algorithm. For PDPSO, its priority-driven search strategy eliminates the inertial velocity computation, offsetting this complexity. Therefore, the combination of strategies alone is not the key to solving complex problems. Targeted and clever changes to address algorithm weaknesses are essential for effective improvement and practicality.

4.3. Algorithm comparison experiment

We evaluated PDPSO's performance against other state-of-the-art algorithms on CEC2017 using mean (Mean), the standard deviation (Std), the Friedman test results and rankings, Wilcoxon rank sum test results (p-value), and code runtime (Runtime). With a 5% significance level, we used PDPSO as the baseline for pairwise tests to assess its differences from other algorithms.

Table 8
The parameters setting and related information of the algorithms (3).

Algorithms	*Pop size*	Parameters
AROA [42]	30	$c = 0.95$,$fr_1 = 0.15$,$fr_2 = 0.6$,$p_1 = 0.2$,$p_2 = 0.8$,$e_f = 0.4$,$tr_1 = 0.9$,$tr_2 = 0.85$,$tr_3 = 0.9$
NRBO [120]	30	$DF = 0.6$
TTAO [40]	30	$r_0 = rand$,$r_1 = rand$,$r_2 = rand$,$r_3 = rand$,$r_4 = rand$
AHA [39]	30	Migrationcoefficient $= 2N$
EO [44]	30	$a_1 = 2$,$a_2 = 1$,$V = 1$,$GP = 0.5$
APO [62]	30	$F = 0.5$,$C = 0.5$
APO2 [64]	30	$np = 1$,$pf_{\max} = 0.1$
BSLO [66]	30	$m = 0.8$,$a = 0.97$,$b = 0.001$,$b_2 = 0.00001$,$t_1 = 20$,$t_2 = 20$
CPO [68]	30	$\alpha = 0.2$,$T_f = 0.8$,$N\min = 20$,$T = 2$
CTCM [69]	30	$m = 10, c_1 = 2, c_2 = 1, c_3 = 0.1$
ESC [71]	30	$\beta_{base} = 0.15, c = 0.15, h = 0.35, p = 0.5, exist = 5$
ETO [46]	30	$b = 1.55$
FATA [48]	30	$\alpha = 0.2$
GBO [52]	30	$\beta\min = 0.2, \beta\max = 1.2, pr = 0.5$
HBO [54]	30	$degree = 3$
HOA [73]	30	Angle $= [0, 50^{\circ}]$,$SF = [1, 3]$
IVY [70]	30	$\beta = rand([1, 1.5])$
NPD [56]	30	$a = 0.1, l = 0.7, d = 0.1$
OMA [57]	30	Nospecialparameters
PPOA [72]	30	$run\_step = 0.2, convert = 0.2$
RBMO [74]	30	$p = [2, 5], q = [10, n], CF = [0, 1], \varepsilon = 0.5$
WAA [59]	30	$\alpha = 10$

4.3.1. Comparison algorithm selection and parameter setting

To evaluate PDPSO's performance, several comparison algorithms are selected based on three criteria: high-performance algorithms from CEC competitions, improved versions of PSO algorithms, and newly published algorithms representing the latest advancements in optimization.

Algorithms based on the DE algorithm consistently perform well in CEC competitions [11]. This paper compares the PDPSO with four notable algorithms: LSHADE_SPACMA and LSHADE_cnEpSin, which excelled in the CEC2017 competition [108,109]; AGSK, which performed well in the CEC2020 competition [110]; and MADDE from the CEC2021 competition [111]. The parameter settings for all comparison algorithms align closely with the original studies, as detailed in Table 6.

This paper evaluates the effectiveness of the PDPSO algorithm by selecting 11 improved PSO algorithms, including XPSO [112], FVICLPSO [94], TAPSO [113], CDLPSO [86], DSPSO [82], EAPSO [114], SRPSO [115], MFCPSO [79], HMPPSO [91], QPSOL [116], and CEPSO [117]. Some of the algorithmic weaknesses have been explained in Sections 1 and 2. Additionally, two newer algorithms, GLS_MPA and KOA_LEOBL [118,119], are included. The parameter settings for all comparison algorithms align closely with the original literature, as detailed in Table 7.

Newly published optimization algorithms often demonstrate superior performance and incorporate the latest research advancements, making them popular in various practical applications. We selected 22 of these algorithms to test and compare against the PDPSO algorithm. The parameter settings for all comparison algorithms align closely with the original literature, as detailed in Table 8.

We observed that LSHADE_SPACM, LSHADE_cnEpSin, AGSK, and MADDE showed decreasing population sizes during iterations. To ensure a fair comparison, we adhered to the literature settings for LSHADE_SPACMA, LSHADE_cnEpSin, and AGSK, recording results after 30,000 evaluations. Given that *FESmax* was set to 30,000, we adjusted MADDE's population size to 30 to avoid detrimental effects on the algorithm.

4.3.2. Comparison with the CEC competition winner algorithm

Tables S.1, S.2, and S.3 in the Supplementary Material show the results of the PDPSO algorithm with LSHADE_SPACMA, LSHADE_cnEpSin, AGSK, and MADDE on the 30, 50, and 100 dimensions for the Mean, Std, and p-value, respectively. In Tables S.1, S.2, and S.3, bolded figures denote the values of mean minimum, std minimum, and Wilcoxon rank sum test results showing significant differences in optimization results using a significance level of 5% on each test function, respectively. According to Tables S.1, S.2, and S.3, on the 30D test functions, PDPSO consistently achieves lower Mean values than other algorithms on 10 functions, with an average ranking of 2.86. On CEC22, CEC25, and CEC27, it ranks 3rd, 4th, and 4th respectively, the error value of its results is only 0.1%, 1.6%, and 2.6% more than the error of the top-ranked algorithms. For the 50D test functions, PDPSO has the lowest Mean value on 11 functions, representing 37.9% of the total, and ranks second on another 11, improving its average rank to 2.04. Particularly, PDPSO obtains top 2 rankings on all combinatorial test functions except CEC27, indicating the dominance of the optimization ability of PDPSO in dealing with complex high-dimensional hybrid problems. PDPSO has an average Mean ranking of 1.45 on the 100D test functions, and the Mean value outperforms all the winning algorithms of the CEC competition on 20 test functions.

According to Tables S.1, S.2, and S.3, LSHADE_cnEpSin exhibits the lowest standard deviation and the highest algorithmic stability across most test functions and environments. While the standard deviation of PDPSO is comparable to that of LSHADE_cnEpSin for many test functions, it demonstrates poorer stability in certain cases (such as CEC10 and CEC22 at 100D). This indicates that there is still potential for improving the stability of the PDPSO algorithm.

Table 9
Statistics of the results of the Friedman test (D=30, 50, 100) (1).

Algorithms	Dimension = 30				Dimension = 50				Dimension = 100			
	W	E	L	Friedman	W	E	L	Friedman	W	E	L	Friedman
LSHADE_SPACMA	14	4	11	3.10	21	2	6	3.83	25	2	2	4.14
LSHADE_cnEpSin	10	4	15	2.00	18	4	7	2.69	24	0	5	3.03
AGSK	19	5	5	4.43	23	3	3	3.72	27	2	0	3.41
MADDE	11	6	12	2.62	14	7	8	2.72	23	4	2	2.97
PDPSO	0	29	0	2.86	0	29	0	2.03	0	29	0	1.45

Table 9 shows the Friedman test results for five algorithms. It analyzes whether algorithms are "Significantly better than PDPSO (L)," "Not significantly different from PDPSO (E)," or "Significantly worse than PDPSO (W)," based on p-values and mean values from Tables S.1, S.2, and S.3. According to Table 9, for the 30D test functions, PDPSO ranks 3rd, trailing behind LSHADE_cnEpSin and MADDE, but it outperforms them on several functions. In the 50D and 100D test functions, PDPSO ranks 1st, significantly exceeding the performance of other algorithms.

Fig. 10 shows the Runtime statistics for PDPSO compared to the CEC competition winner algorithm. PDPSO demonstrates a shorter Runtime across all tested functions, indicating lower algorithmic complexity. Specifically, on the 30D CEC1 test function, PDPSO's Runtime is 45.06% of MADDE's and 55.27% of LSHADE_cnEpSin's, with similar patterns observed in other functions. On the 50D and 100D CEC3 functions, PDPSO's Runtime is 48.87% and 31.85% of LSHADE_cnEpSin's, respectively. As shown in Fig. 10, PDPSO consistently maintains leading algorithmic execution efficiency, regardless of changes in problem complexity, this efficiency highlights PDPSO's advantage in algorithmic complexity. LSHADE_cnEpSin ranks higher in Friedman on the 30D test function, but it has high algorithmic complexity, making it vulnerable to the optimization problem's complexity.

4.3.3. Comparison with the newest improved version of PSO and the new algorithm

In this section, we analyze the convergence, statistical properties, and computational complexity of the PDPSO algorithm through various comparisons. For ease of comparison, we include all improved PSO algorithms and recent optimization algorithms, totaling 35 algorithms.

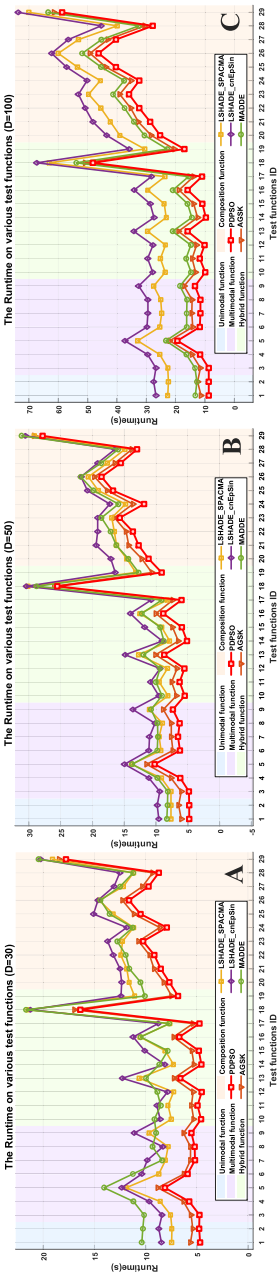
Tables S.4, S.5, and S.6 in the Supplementary Material present the PDPSO algorithm results alongside 35 comparison algorithms across 30, 50, and 100 dimensions, detailing the Mean, Std, and p-values. Bolded and unshaded figures indicate the mean minimum, std minimum, and significant differences through the Wilcoxon rank sum test at a 5% significance level. According to Tables S.4, S.5, and S.6, this paper counts the rankings of 36 algorithms in terms of Mean and Std values on 29 test functions, as shown in Fig. 11, where the x-axis of the graph is sorted in descending order of the average rankings. According to Fig. 11(A, B, C), the PDPSO algorithm has the highest average of 36 algorithms in Mean value ranking in all test dimensions and ranked in the top 10 in 100%, 97%, and 93% of the test functions, respectively. Hence, the PDPSO algorithm has a strong optimization ability. According to Fig. 11(D, E, F), in most scenarios, the PDPSO algorithm has a low Std value ranking, which means that the PDPSO algorithm maintains a relatively high stability.

According to Tables S.4, S.5, and S.6, considering whether the p-value is less than 0.05 and the size of the mean value, Table 10 counts the number of comparison algorithms that are "Significantly better than PDPSO (L)," "Not significantly different from PDPSO (E)" and "significantly worse than PDPSO (W)". The last column shows the average Friedman test results across 3 dimensions, which will guide the selection of algorithms for future experiments.

Table 10 shows that the PDPSO algorithm ranks highest in Friedman's test results across all three dimensions, significantly outperforming most other algorithms tested. Notably, XPSO, SRPSO, MFCPSO, GLS_MPA, AROA, APO2, and PPOA also achieved impressive results. Among the 35 algorithms compared, AROA ranked second but only surpassed PDPSO in 13.90%, 17.24%, and 20.70% of the tested functions.

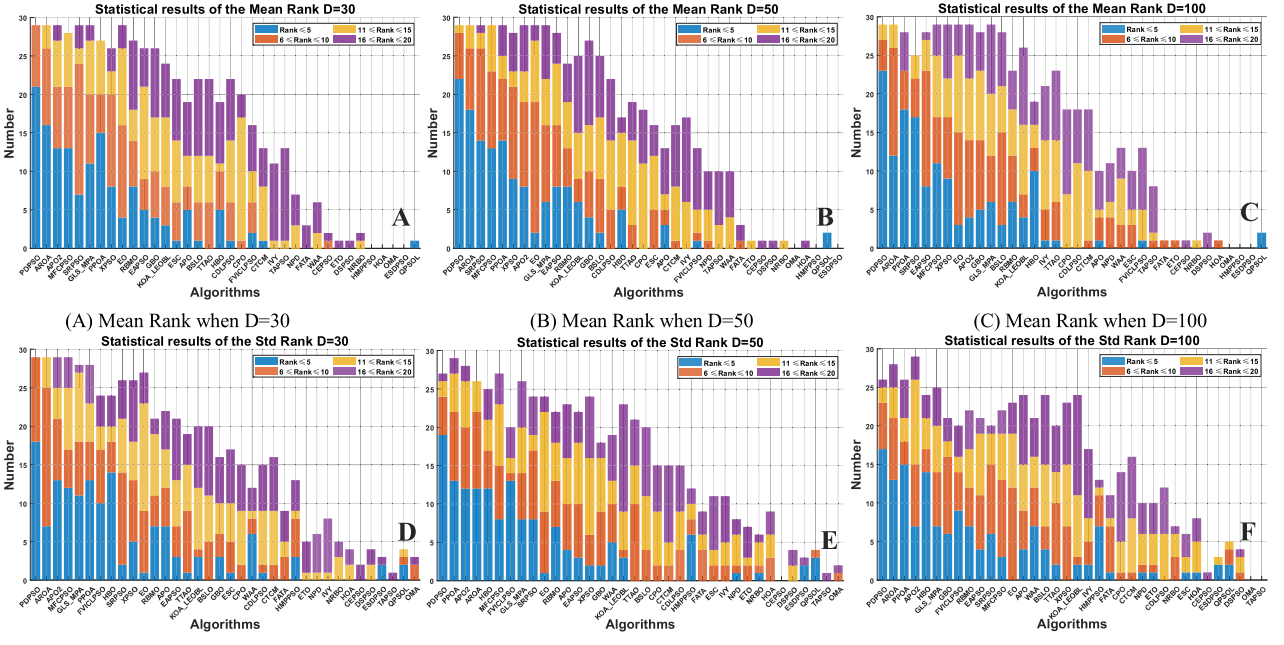
To analyze the convergence of the PDPSO algorithm, we plotted its average convergence curves alongside 35 comparative algorithms across various dimensions (see Fig. S.1, S.2, S.3 in the Supplementary Material). We selected 12 representative curves from four types of graphs from three dimensions, as shown in Fig. 12. In Fig. 12, The curves of most algorithms eventually flattened, indicating convergence. However, the optimal values from TAPSO, XPSO, NRBO, and NPD were significantly different, suggesting a limitation in escaping local optima, especially in high-dimensional problems. Initially, PDPSO did not show the steepest convergence but maintained a leading position. When other algorithms show slow convergence speed and optimization difficulties, the PDPSO can maintain a good convergence speed and eventually win in the final optimization results.

In the preliminary stage, the slower optimization speed of PDPSO does not indicate that the PDPSO algorithm has a poorer convergence ability. On the combined function image in Fig. 12(J, K, L), the convergence curve of the PDPSO algorithm possesses a clear turning part in the middle part; before the turning point, PDPSO does not dominate the optimization results of all algorithms. This phenomenon is primarily related to the capacity of the candidate solution set in the PDPSO algorithm. During the early stages of the algorithm, PDPSO maintains a relatively large number of candidate solutions. Each of these candidate solutions is developed by a portion of the agent individuals using a priority-driven search strategy. In essence, compared to the standard PSO algorithm, the PDPSO in its initial optimization phase resembles a multi-sub-populations PSO algorithm, where the total population is equivalent to that of the PSO. However, because these sub-PSO algorithms have limited information exchange among themselves, they cannot achieve the '1+1=2' effect, resulting in relatively low convergence for the PDPSO algorithm during these early stages. As the algorithm



(A) Runtime when D=30 (B) Runtime when D=50 (C) Runtime when D=100

Fig. 10. Total Runtime for each algorithm run 30 times on each test function.



(A) Mean Rank when D=30 (B) Mean Rank when D=50 (C) Mean Rank when D=100

(D) Std Rank when D=30 (E) Std Rank when D=50 (F) Std Rank when D=100

Fig. 11. Ranking statistics of 36 algorithms on 29 test functions in Mean and Std values.

Table 10
Statistics of the results of the Friedman test (D=30, 50, 100) (2).

Algorithms	D = 30				D = 50				D = 100				
	W	E	L	Friedman	W	E	L	Friedman	W	E	L	Friedman	Friedman(Average)
PDPSO	0	29	0	3.66	0	29	0	3.45	0	29	0	3.17	3.43
XPSO	20	8	1	9.97	20	7	2	8.97	22	6	1	9.76	9.56
FVICLPSO	26	2	1	17.28	27	2	0	19.90	29	0	0	21.79	19.66
TAPSO	27	2	0	22.76	26	3	0	22.72	27	2	0	23.62	23.03
CDLPSO	26	2	1	16.38	27	2	0	16.28	29	0	0	18.45	17.03
DSPSO	29	0	0	28.52	29	0	0	29.14	29	0	0	29.59	29.08
EAPSO	21	8	0	12.69	20	8	1	10.03	24	4	1	8.34	10.36
SRPSO	17	12	0	7.72	20	8	1	6.55	22	7	0	7.86	7.38
MFCPSO	17	8	4	7.62	19	5	5	7.03	21	6	2	8.62	7.76
ESDPSO	29	0	0	34.14	29	0	0	33.90	29	0	0	33.41	33.82
HMPPSO	29	0	0	31.55	29	0	0	32.21	29	0	0	31.48	31.75
QPSOL	28	0	1	34.31	27	0	2	33.66	27	0	2	33.62	33.86
CEPSO	29	0	0	27.66	29	0	0	28.38	28	1	0	29.45	28.49
GLS_MPA	20	1	8	7.76	19	5	5	9.79	23	3	3	11.45	9.67
KOA_LEOBL	21	5	3	13.62	21	3	5	13.00	25	0	4	13.55	13.39
AROA	16	9	4	5.59	15	9	5	5.52	21	2	6	6.17	5.76
NRBO	29	0	0	28.93	29	0	0	29.90	28	1	0	29.55	29.46
TTAO	24	5	0	16.07	27	2	0	16.69	28	1	0	16.48	16.41
EO	25	4	0	10.10	28	1	0	9.76	29	0	0	10.59	10.15
APO	23	3	3	15.97	24	4	1	19.28	28	0	1	20.14	18.46
APO2	15	10	4	7.24	20	7	2	9.00	26	2	1	10.90	9.05
BSLO	25	3	1	15.97	25	2	2	13.41	24	4	1	11.83	13.74
CPO	27	2	0	16.55	28	1	0	18.21	28	1	0	18.38	17.71
CTCM	26	3	0	19.90	26	3	0	19.45	27	2	0	18.83	19.39
ESC	27	2	0	15.69	28	1	0	18.34	28	1	0	21.76	18.60
ETO	29	0	0	27.72	29	0	0	26.93	28	0	1	26.48	27.05
FATA	29	0	0	25.55	29	0	0	25.86	28	0	1	26.38	25.93
GBO	21	5	3	13.14	23	3	3	13.07	24	3	2	11.28	12.49
HBO	22	5	2	16.24	24	4	1	16.48	22	4	3	14.24	15.66
HOA	29	0	0	31.66	29	0	0	31.69	28	0	1	30.79	31.38
IVY	28	1	0	22.55	29	0	0	19.83	27	1	1	15.76	19.38
NPD	29	0	0	23.03	28	1	0	22.41	28	1	0	20.28	21.91
OMA	29	0	0	32.90	29	0	0	31.59	29	0	0	31.24	31.91
PPOA	19	8	2	7.79	21	5	3	7.07	21	5	3	7.00	7.29
RBMO	18	5	6	11.45	23	2	4	12.79	25	0	4	13.17	12.47
WAA	29	0	0	26.34	29	0	0	23.72	27	2	0	20.59	23.55

progresses, it eliminates slower candidates and focuses on promising solutions, ultimately achieving faster convergence and better optimization results. In essence, PDPSO trades early convergence speed for the identification of high-quality candidates, enabling superior performance in later stages. After this turning point PDPSO algorithm is almost entirely superior to its comparative algorithms, which indicates that when dealing with complex problems PDPSO algorithm possesses an advantage of escape from local optimum.

The time complexity of a metaheuristic algorithm determines its practical applicability. This paper evaluates the time complexity of the PDPSO algorithm based on IEEE CEC2017 [121]. Table 11 presents the time complexity statistics for 36 algorithms. T_0 is the time for standard procedures, T_1 for evaluating an optimization problem, T_2 for optimizing a problem, and $(T_0\text{-}T_0)/T_0$ reflects the algorithm's time complexity. *Ur* indicates the increase in time complexity as the problem dimensions rise from 30 to 100.

According to Table 11, PDPSO has the second highest time complexity among improved PSO algorithms, following CEPSO, which excels in parallelizability but has slower convergence. PDPSO, however, offers lower time complexity and higher optimization accuracy. Algorithms GLS_MPA and KOA_LEOBL surpass PDPSO in optimization accuracy while being 30.55% and 8.96% faster in terms of time complexity for 30-dimensional problems. Nevertheless, their time complexity significantly increases with problem dimensions. In comparison, the newly published algorithms exhibit over a 100% increase in time complexity, except for AROA, NRBO, APO2, CPO, CTCM, HOA, and IVY, which have increase rates of 41.02%, 80.74%, 28.18%, 79.12%, 61.18%, and 34.73%, respectively. PDPSO's increase rate is only 24.87%, the lowest among these algorithms.

Overall, the PDPSO algorithm has advantages such as high optimization accuracy, relatively low time complexity, and time complexity that is not easily affected by the dimension of the problem. Furthermore, we also focus on EAPSO, EO, CPO, APO2, and AROA due to their advantages, such as high convergence accuracy, lower time complexity, or time complexity that is less affected by the dimension of the optimization problem.

4.4. Real-world high-dimensional complex problems

Real-world complex problems differ from test functions due to their practical complexity. Unlike test functions, which have explicit expressions and clear optimal solutions, real-world problems involve intricate relationships between variables and various challenges

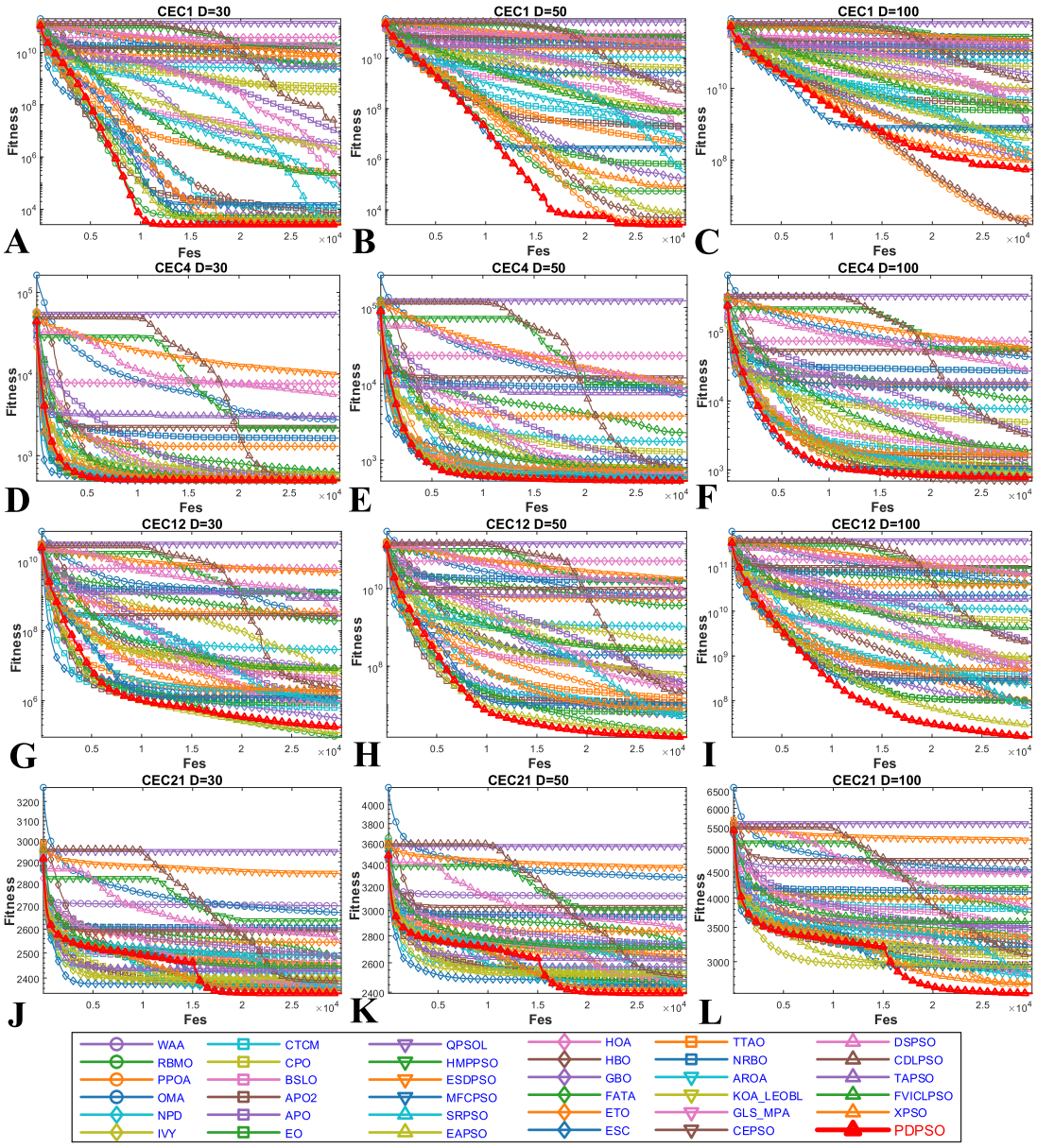


Fig. 12. Convergence curves for all algorithms in partial test functions.

such as noise, heterogeneous types, and dynamic constraints. Solving these problems requires a focus on robustness and practicality, enabling a more accurate assessment of algorithm performance in real-world scenarios rather than solely on mathematical optimality.

In this section, 35 real-world problems are optimized using PDPSO and XPSO, EAPSO, SRPSO, GLS_MPA, AROA, EO, APO2, and HBO as a way to test the ability of the PDPSO algorithm to solve real-world problems. These comparison algorithms were reasonably selected because they all obtained better average Friedman rankings (top 15, as in Table 10) and possessed smaller average time complexity (top 15, as in Table 11) or time complexity increase rates (top 15, as in Table 11). The performance of the algorithm to optimize real-world problems is reflected by the mean (Mean), standard deviation (Std), optimal value (Best), median (Median), and worst value (Worst) of the repeated experiment, to facilitate the description of the subsequent experiments, we will call these five indicators as "five fundamental indicators" herein.

Each engineering problem is defined as:

Minimize:

$$f(\overline{x}), \tag{20}$$

where $\overline{\boldsymbol{x}} = (x_1, x_2, \cdots, x_N)$, N denote the Real-world optimization problem dimension.

Table 11
Statistics of the results of the time complexity (D=30, 50, 100).

Algorithms	Dimension = 30				Dimension = 50				Dimension = 100				
	T_0	T_1	T_2	$(T_2\text{-}T_1)/T_0$	T_0	T_1	T_2	$(T_2\text{-}T_1)/T_0$	T_0	T_1	T_2	$(T_2\text{-}T_1)/T_0$	*Ur*
PDPSO	0.0082 s	0.3547 s	0.9962 s	78.1323	0.0082 s	0.5296 s	1.2202 s	84.1061	0.0082 s	1.5183 s	2.3193 s	97.5661	24.87%
XPSO	0.0082 s	0.3547 s	1.3819 s	125.1159	0.0082 s	0.5296 s	1.7462 s	148.1740	0.0082 s	1.5183 s	3.2291 s	208.3775	66.55%
FVICLPSO	0.0082 s	0.3547 s	3.7383 s	412.1303	0.0082 s	0.5296 s	3.9669 s	418.6632	0.0082 s	1.5183 s	5.1035 s	436.6832	5.96%
TAPSO	0.0082 s	0.3547 s	4.1181 s	458.3803	0.0082 s	0.5296 s	6.2784 s	700.1936	0.0082 s	1.5183 s	13.9906 s	1519.1234	231.41%
CDLPSO	0.0082 s	0.3547 s	1.6148 s	153.4887	0.0082 s	0.5296 s	1.8606 s	162.1129	0.0082 s	1.5183 s	2.9555 s	175.0507	14.05%
DSPSO	0.0082 s	0.3547 s	1.5218 s	142.1564	0.0082 s	0.5296 s	1.7710 s	151.2037	0.0082 s	1.5183 s	2.8738 s	165.1030	16.14%
EAPSO	0.0082 s	0.3547 s	1.4646 s	135.1930	0.0082 s	0.5296 s	1.6156 s	132.2749	0.0082 s	1.5183 s	2.1768 s	80.2123	-40.67%
SRPSO	0.0082 s	0.3547 s	1.7164 s	165.8521	0.0082 s	0.5296 s	1.9805 s	176.7158	0.0082 s	1.5183 s	3.1214 s	195.2586	17.73%
MFCPSO	0.0082 s	0.3547 s	5.8642 s	671.0575	0.0082 s	0.5296 s	7.7591 s	880.5451	0.0082 s	1.5183 s	13.0884 s	1409.2442	110.00%
ESDPSO	0.0082 s	0.3547 s	1.6966 s	163.4470	0.0082 s	0.5296 s	1.9546 s	173.5595	0.0082 s	1.5183 s	3.1251 s	195.7148	19.74%
HMPPSO	0.0082 s	0.3547 s	2.2458 s	230.3429	0.0082 s	0.5296 s	2.5131 s	241.5920	0.0082 s	1.5183 s	3.7108 s	267.0522	15.94%
QPSOL	0.0082 s	0.3547 s	1.3209 s	117.6878	0.0082 s	0.5296 s	1.5109 s	119.5188	0.0082 s	1.5183 s	2.4881 s	118.1297	0.38%
CEPSO	0.0082 s	0.3547 s	0.3984 s	5.3308	0.0082 s	0.5296 s	0.5942 s	7.8667	0.0082 s	1.5183 s	1.6337 s	14.0579	163.71%
GLS_MPA	0.0082 s	0.3547 s	0.8002 s	54.2657	0.0082 s	0.5296 s	1.1208 s	72.0073	0.0082 s	1.5183 s	2.4585 s	114.5205	111.04%
KOA_LEOBL	0.0082 s	0.3547 s	0.9387 s	71.1336	0.0082 s	0.5296 s	1.4065 s	106.8044	0.0082 s	1.5183 s	3.6684 s	261.8859	268.16%
AROA	0.0082 s	0.3547 s	0.9711 s	75.0831	0.0082 s	0.5296 s	1.2263 s	84.8494	0.0082 s	1.5183 s	2.3876 s	105.8827	41.02%
NRBO	0.0082 s	0.3547 s	1.0970 s	90.4108	0.0082 s	0.5296 s	1.4571 s	112.9599	0.0082 s	1.5183 s	2.8599 s	163.4102	80.74%
TTAO	0.0082 s	0.3547 s	1.1278 s	94.1685	0.0082 s	0.5296 s	1.5693 s	126.6265	0.0082 s	1.5183 s	3.7137 s	267.4047	183.96%
EO	0.0082 s	0.3547 s	0.6324 s	33.8209	0.0082 s	0.5296 s	0.8927 s	44.2180	0.0082 s	1.5183 s	2.0828 s	68.7563	103.30%
APO	0.0082 s	0.3547 s	1.4744 s	136.3870	0.0082 s	0.5296 s	2.0261 s	182.2677	0.0082 s	1.5183 s	4.2951 s	338.2204	147.99%
APO2	0.0082 s	0.3547 s	1.1481 s	96.6393	0.0082 s	0.5296 s	1.3932 s	105.1782	0.0082 s	1.5183 s	2.5353 s	123.8716	28.18%
BSLO	0.0082 s	0.3547 s	1.1733 s	99.7066	0.0082 s	0.5296 s	1.7754 s	151.7306	0.0082 s	1.5183 s	3.8741 s	286.9401	187.78%
CPO	0.0082 s	0.3547 s	0.5524 s	24.0802	0.0082 s	0.5296 s	0.7844 s	31.0328	0.0082 s	1.5183 s	1.8724 s	43.1317	79.12%
CTCM	0.0082 s	0.3547 s	1.0739 s	87.5980	0.0082 s	0.5296 s	1.3664 s	101.9181	0.0082 s	1.5183 s	2.5782 s	129.0992	47.38%
ESC	0.0082 s	0.3547 s	2.2642 s	232.5751	0.0082 s	0.5296 s	3.2370 s	329.7530	0.0082 s	1.5183 s	6.1173 s	560.1569	140.85%
ETO	0.0082 s	0.3547 s	1.3168 s	117.1814	0.0082 s	0.5296 s	2.0167 s	181.1235	0.0082 s	1.5183 s	4.3427 s	344.0166	193.58%
FATA	0.0082 s	0.3547 s	0.7235 s	44.9186	0.0082 s	0.5296 s	1.0998 s	69.4503	0.0082 s	1.5183 s	2.5307 s	123.3114	174.52%
GBO	0.0082 s	0.3547 s	1.2578 s	110.0016	0.0082 s	0.5296 s	1.7106 s	143.8377	0.0082 s	1.5183 s	3.3724 s	225.8318	105.30%
HBO	0.0082 s	0.3547 s	0.6017 s	30.0878	0.0082 s	0.5296 s	0.9009 s	45.2244	0.0082 s	1.5183 s	2.1363 s	75.2750	150.18%
HOA	0.0082 s	0.3547 s	0.4611 s	12.9626	0.0082 s	0.5296 s	0.6667 s	16.6995	0.0082 s	1.5183 s	1.6898 s	20.8941	61.19%
IVY	0.0082 s	0.3547 s	0.9897 s	77.3442	0.0082 s	0.5296 s	1.2514 s	87.9069	0.0082 s	1.5183 s	2.3738 s	104.2096	34.73%
NPD	0.0082 s	0.3547 s	1.8298 s	179.6753	0.0082 s	0.5296 s	2.4658 s	235.8257	0.0082 s	1.5183 s	4.5076 s	364.1043	102.65%
OMA	0.0082 s	0.3547 s	0.8652 s	62.1827	0.0082 s	0.5296 s	1.2826 s	91.7093	0.0082 s	1.5183 s	3.3554 s	223.7643	259.85%
PPOA	0.0082 s	0.3547 s	1.0571 s	85.5494	0.0082 s	0.5296 s	1.5442 s	123.5776	0.0082 s	1.5183 s	5.2541 s	455.0186	431.88%
RBMO	0.0082 s	0.3547 s	2.2049 s	225.3583	0.0082 s	0.5296 s	2.8505 s	282.6804	0.0082 s	1.5183 s	5.3435 s	465.9189	106.75%
WAA	0.0082 s	0.3547 s	3.7510 s	413.6730	0.0082 s	0.5296 s	6.0784 s	675.8418	0.0082 s	1.5183 s	12.3185 s	1315.4604	218.00%

Table 12
Results of PDPSO and comparative algorithms on CEC2020-RW.

		RC34	RC35	RC36	RC37	RC38	RC39	RC40	RC41	RC42	RC43	RC44	RC45
Mean	PDPSO	2.045E+11	2.792E+14	2.643E+14	2.371E+11	2.784E+11	2.546E+11	1.700E+14	2.780E+14	9.897E+13	1.092E+14	-5.677E+03	2.672E+07
	XPSO	6.187E+11	1.543E+14	1.608E+14	2.600E+11	2.423E+11	2.271E+11	1.223E+14	1.183E+14	3.426E+14	3.274E+14	-5.331E+03	5.520E+08
	EAPSO	2.949E+11	3.757E+14	3.752E+14	4.219E+11	4.496E+11	3.996E+11	4.402E+14	3.168E+14	2.929E+14	2.312E+14	-5.755E+03	2.179E+09
	SRPSO	3.114E+11	2.503E+14	2.646E+14	3.309E+11	2.891E+11	2.877E+11	2.179E+14	2.510E+14	2.555E+14	2.624E+14	-5.532E+03	1.171E+10
	GLS_MPA	8.633E+11	1.137E+14	1.245E+14	1.389E+11	1.471E+11	1.510E+11	1.729E+14	1.554E+14	1.726E+12	1.782E+12	-5.799E+03	3.083E+08
	AROA	1.307E+11	6.378E+13	4.518E+13	5.143E+10	5.483E+10	4.829E+10	1.015E+14	9.067E+13	9.125E+12	7.292E+12	-5.942E+03	2.557E+09
	EO	1.207E+11	3.171E+13	2.541E+13	3.122E+10	2.705E+10	3.003E+10	4.954E+13	7.036E+13	6.836E+11	6.673E+11	-5.846E+03	2.560E+10
	APO2	7.049E+11	2.824E+14	2.895E+14	2.595E+11	2.585E+11	2.824E+11	2.313E+14	3.265E+14	1.069E+14	9.093E+13	-5.936E+03	7.697E+08
	HBO	4.627E+12	3.639E+15	3.102E+15	6.087E+12	5.316E+12	6.071E+12	3.549E+15	3.297E+15	2.000E+15	1.346E+15	-5.456E+03	3.138E+10
Std	PDPSO	4.312E+10	1.107E+14	1.007E+14	6.496E+10	8.255E+10	6.527E+10	5.372E+13	9.701E+13	4.267E+13	4.350E+13	1.582E+02	1.221E+08
	XPSO	2.669E+11	8.939E+13	6.581E+13	2.274E+11	1.262E+11	1.490E+11	1.801E+14	4.240E+13	1.723E+14	2.004E+14	1.965E+02	1.498E+09
	EAPSO	8.491E+10	1.760E+14	1.486E+14	1.161E+11	2.833E+11	1.621E+11	2.219E+14	1.166E+14	1.415E+14	9.354E+13	1.152E+02	7.073E+09
	SRPSO	1.246E+11	9.375E+13	8.374E+13	1.101E+11	7.705E+10	7.176E+10	4.684E+13	7.168E+13	7.277E+13	9.797E+13	3.171E+02	2.133E+10
	GLS_MPA	2.523E+11	3.429E+13	4.881E+13	4.194E+10	6.105E+10	5.518E+10	4.659E+13	3.320E+13	3.719E+11	4.117E+11	9.341E+01	1.173E+09
	AROA	4.896E+10	4.030E+13	2.543E+13	2.633E+10	1.775E+10	2.000E+10	3.742E+13	1.936E+13	1.866E+13	1.794E+13	1.588E+02	5.565E+09
	EO	8.628E+10	1.841E+13	1.341E+13	1.188E+10	1.150E+10	1.018E+10	2.504E+13	2.719E+13	1.955E+11	1.929E+11	1.290E+02	8.422E+10
	APO2	2.134E+11	9.922E+13	1.043E+14	8.614E+10	9.820E+10	1.087E+11	7.017E+13	2.012E+14	6.612E+13	6.777E+13	7.054E+01	1.612E+09
	HBO	3.103E+12	3.105E+15	2.858E+15	3.857E+12	3.722E+12	4.237E+12	2.947E+15	2.929E+15	1.735E+15	1.379E+15	5.280E+01	5.848E+10
Best	PDPSO	1.116E+11	1.303E+14	1.596E+14	1.420E+11	1.564E+11	1.390E+11	7.728E+13	1.538E+14	3.318E+13	2.616E+13	-6.003E+03	2.470E-01
	XPSO	2.890E+11	3.564E+13	7.306E+13	6.083E+10	8.095E+10	5.908E+10	2.377E+13	7.033E+13	1.151E+14	8.659E+13	-5.607E+03	4.804E-01
	EAPSO	1.183E+11	1.399E+14	1.318E+14	2.061E+11	1.849E+11	1.934E+11	1.314E+14	1.371E+14	1.123E+14	9.468E+13	-5.999E+03	3.054E-01
	SRPSO	1.571E+11	1.128E+14	1.289E+14	1.272E+11	1.814E+11	1.873E+11	1.412E+14	1.324E+14	1.327E+14	9.537E+13	-5.924E+03	5.211E-01
	GLS_MPA	2.961E+11	5.530E+13	4.385E+13	5.942E+10	7.530E+10	8.877E+10	8.650E+13	9.986E+13	1.033E+12	1.309E+12	-6.018E+03	1.113E+00
	AROA	6.758E+10	1.874E+13	1.002E+13	2.533E+10	2.167E+10	2.530E+10	4.923E+13	2.602E+13	7.147E+11	1.128E+12	-6.187E+03	4.143E-01
	EO	4.620E+10	1.071E+13	1.029E+13	1.240E+10	1.173E+10	1.480E+10	2.109E+13	1.174E+13	3.658E+11	4.223E+11	-6.137E+03	1.075E+00
	APO2	3.627E+11	6.126E+13	1.070E+14	1.305E+11	1.323E+11	1.454E+11	1.325E+14	1.221E+14	9.307E+12	1.833E+13	-6.064E+03	7.280E-01
	HBO	6.741E+11	4.247E+14	6.427E+14	5.707E+11	7.287E+11	8.708E+11	5.369E+14	4.482E+14	5.420E+14	2.576E+14	-5.593E+03	3.278E-01
Median	PDPSO	2.088E+11	2.500E+14	2.377E+14	2.253E+11	2.606E+11	2.721E+11	1.657E+14	2.554E+14	9.108E+13	1.057E+14	-5.665E+03	7.551E-01
	XPSO	5.319E+11	1.300E+14	1.395E+14	1.832E+11	2.263E+11	1.804E+11	7.659E+13	1.062E+14	3.114E+14	2.874E+14	-5.320E+03	9.885E-01
	EAPSO	2.855E+11	3.463E+14	3.652E+14	4.118E+11	3.787E+11	3.842E+11	3.656E+14	3.039E+14	2.662E+14	2.102E+14	-5.730E+03	1.091E+00
	SRPSO	2.946E+11	2.276E+14	2.590E+14	2.982E+11	2.697E+11	2.714E+11	2.239E+14	2.369E+14	2.498E+14	2.540E+14	-5.629E+03	2.442E+08
	GLS_MPA	8.995E+11	1.207E+14	1.202E+14	1.313E+11	1.311E+11	1.407E+11	1.647E+14	1.477E+14	1.705E+12	1.604E+12	-5.804E+03	1.157E+00
	AROA	1.208E+11	5.119E+13	3.628E+13	4.290E+10	5.241E+10	4.373E+10	9.313E+13	9.629E+13	2.514E+12	2.157E+12	-5.947E+03	7.408E+07
	EO	7.681E+10	2.862E+13	2.041E+13	2.840E+10	2.478E+10	2.914E+10	4.684E+13	7.283E+13	6.491E+11	6.078E+11	-5.844E+03	1.423E+06
	APO2	6.909E+11	2.673E+14	2.771E+14	2.283E+11	2.367E+11	2.423E+11	2.231E+14	3.103E+14	1.027E+14	7.575E+13	-5.933E+03	1.167E+00
	HBO	3.439E+12	1.948E+15	2.059E+15	5.472E+12	4.501E+12	4.242E+12	2.819E+15	1.847E+15	1.384E+15	8.392E+14	-5.441E+03	7.666E+09
Worst	PDPSO	3.326E+11	5.465E+14	5.577E+14	3.775E+11	4.316E+11	3.845E+11	2.760E+14	5.499E+14	2.361E+14	1.954E+14	-5.231E+03	6.625E+08
	XPSO	1.264E+12	4.434E+14	3.154E+14	8.463E+11	4.811E+11	6.695E+11	9.423E+14	2.321E+14	8.511E+14	9.481E+14	-4.867E+03	7.107E+09
	EAPSO	5.691E+11	7.072E+14	7.180E+14	7.164E+11	1.511E+12	9.082E+11	9.547E+14	5.246E+14	6.474E+14	4.240E+14	-5.508E+03	3.817E+10
	SRPSO	7.028E+11	5.064E+14	5.005E+14	6.130E+11	4.198E+11	4.472E+11	3.404E+14	4.238E+14	5.253E+14	5.101E+14	-4.422E+03	8.544E+10
	GLS_MPA	1.381E+12	1.924E+14	2.252E+14	2.245E+11	3.131E+11	3.203E+11	2.815E+14	2.223E+14	2.545E+12	3.016E+12	-5.603E+03	4.624E+09
	AROA	2.455E+11	1.791E+14	1.129E+14	1.585E+11	9.835E+10	1.252E+11	2.282E+14	1.196E+14	6.997E+13	7.897E+13	-5.539E+03	2.455E+10
	EO	3.409E+11	8.357E+13	5.741E+13	6.347E+10	5.708E+10	4.683E+10	1.316E+14	1.091E+14	1.101E+12	1.087E+12	-5.584E+03	4.183E+11
	APO2	1.250E+12	5.170E+14	5.986E+14	4.178E+11	5.207E+11	5.677E+11	4.016E+14	1.247E+15	3.018E+14	3.077E+14	-5.801E+03	5.000E+09
	HBO	1.192E+13	1.073E+16	1.257E+16	1.327E+13	1.322E+13	1.445E+13	1.148E+16	1.107E+16	6.511E+15	6.687E+15	-5.379E+03	2.809E+11

		RC46	RC47	RC48	RC49	RC50	RC51	RC52	RC53	RC54	RC55	RC56	RC57
Mean	PDPSO	4.738E+07	2.357E+07	1.098E+10	3.103E+09	5.123E+09	1.262E+07	1.462E+04	1.016E+07	4.391E+06	2.142E+07	6.542E+07	8.272E+06
	XPSO	4.050E+08	4.325E+09	3.427E+10	6.706E+09	2.014E+10	7.107E+08	3.575E+07	2.611E+09	4.143E+08	4.491E+08	7.579E+10	6.919E+09

(*continued on next page*)

Table 12 (*continued*)

		RC46	RC47	RC48	RC49	RC50	RC51	RC52	RC53	RC54	RC55	RC56	RC57
	EAPSO	6.040E+09	4.616E+09	3.887E+10	1.861E+10	2.593E+10	4.523E+08	7.984E+07	2.598E+08	4.957E+08	9.720E+08	5.058E+09	2.201E+08
	SRPSO	7.129E+09	1.115E+10	5.543E+10	4.205E+10	3.565E+10	6.702E+09	2.783E+08	4.580E+09	6.020E+09	2.807E+10	2.776E+10	3.555E+10
	GLS_MPA	9.652E+08	2.075E+09	3.008E+10	1.112E+09	1.272E+10	6.661E+09	4.241E+07	9.126E+09	3.802E+09	1.582E+10	1.187E+11	1.821E+10
	AROA	2.137E+09	1.844E+09	1.969E+10	1.241E+10	1.323E+10	1.771E+08	3.261E+04	3.683E+07	1.253E+08	3.056E+08	6.848E+07	8.611E+06
	EO	1.716E+10	1.402E+10	7.770E+10	5.623E+10	3.955E+10	1.515E+09	3.629E+05	1.452E+09	9.347E+08	4.236E+08	1.549E+09	3.414E+07
	APO2	1.260E+09	1.042E+10	5.213E+10	6.756E+09	2.787E+10	3.586E+09	1.912E+07	5.595E+09	3.160E+09	7.894E+09	5.780E+10	1.456E+10
	HBO	2.819E+10	3.064E+10	6.665E+10	8.902E+10	1.108E+11	3.344E+08	2.205E+06	4.067E+08	1.528E+08	3.318E+07	7.513E+08	1.668E+07
Std	PDPSO	1.589E+08	9.341E+07	1.329E+10	7.992E+09	6.700E+09	1.617E+07	4.985E+04	1.703E+07	6.498E+06	1.450E+07	2.155E+07	4.020E+06
	XPSO	1.135E+09	6.160E+09	2.482E+10	1.467E+10	1.923E+10	7.311E+08	1.706E+08	4.351E+09	5.110E+08	1.844E+09	8.942E+10	1.325E+10
	EAPSO	2.084E+10	1.165E+10	5.832E+10	3.617E+10	3.690E+10	3.607E+08	4.047E+08	4.330E+08	1.032E+09	2.716E+09	1.465E+10	5.184E+08
	SRPSO	1.708E+10	2.484E+10	5.333E+10	4.840E+10	4.153E+10	5.711E+09	2.972E+08	3.928E+09	4.300E+09	1.361E+10	1.572E+10	2.539E+10
	GLS_MPA	5.630E+08	7.058E+09	9.895E+09	2.113E-02	5.769E+09	4.654E+09	3.243E+07	6.222E+09	2.912E+09	1.038E+10	6.548E+10	1.263E+10
	AROA	5.754E+09	5.923E+09	1.584E+10	1.322E+10	1.366E+10	1.725E+08	1.494E+05	4.454E+07	2.289E+08	1.586E+09	4.281E+07	6.513E+06
	EO	4.267E+10	2.638E+10	1.546E+11	6.518E+10	7.114E+10	2.753E+09	1.880E+06	3.270E+09	1.744E+09	2.163E+09	4.823E+09	1.339E+08
	APO2	1.050E+09	1.290E+10	2.802E+10	1.091E+10	1.949E+10	3.220E+09	2.187E+07	6.027E+09	3.705E+09	1.090E+10	6.798E+10	1.688E+10
	HBO	4.149E+10	7.105E+10	8.725E+10	1.203E+11	1.397E+11	4.526E+08	5.264E+06	1.981E+09	1.180E+08	1.759E+07	3.313E+09	5.933E+06
Best	PDPSO	9.735E-02	8.411E-02	1.831E-01	2.364E-01	1.349E+08	1.702E+06	4.770E+03	5.136E+05	2.151E+05	3.288E+06	1.263E+07	1.406E+06
	XPSO	1.400E-01	2.557E-01	3.257E+09	3.465E-01	3.000E+08	2.922E+07	5.083E+03	2.287E+07	3.738E+06	3.650E+06	7.195E+07	5.313E+06
	EAPSO	1.192E-01	8.723E-02	7.107E-01	2.496E-01	1.017E+08	1.615E+08	4.927E+03	2.318E+07	1.365E+08	1.267E+07	5.643E+07	6.355E+06
	SRPSO	2.322E-01	1.523E-01	2.093E+09	3.140E-01	1.919E+09	1.810E+09	1.122E+07	8.439E+08	1.028E+09	5.047E+09	5.623E+09	7.507E+09
	GLS_MPA	3.858E-01	5.013E-01	8.311E+01	1.112E+09	1.166E+10	4.142E+08	4.992E+03	9.722E+07	3.135E+08	2.275E+09	1.279E+10	2.219E+09
	AROA	1.600E-01	3.375E-01	5.328E-01	6.299E-01	3.288E-01	5.495E+06	4.294E+03	6.320E+05	4.500E+06	1.369E+06	1.741E+07	1.558E+06
	EO	4.121E-01	2.842E-01	1.095E+03	1.112E+09	8.074E+09	2.102E+06	5.094E+03	2.131E+06	1.064E+06	1.876E+06	1.698E+07	3.025E+06
	APO2	4.866E-01	3.091E-01	1.814E+04	2.922E+08	3.354E-01	2.830E+08	4.964E+03	3.902E+07	1.132E+08	2.573E+07	1.315E+09	4.468E+07
	HBO	3.845E-01	2.583E-01	9.480E+08	1.966E+08	2.575E+09	3.986E+07	5.373E+03	1.415E+07	3.138E+07	1.167E+07	5.806E+07	6.385E+06
Median	PDPSO	7.199E-01	4.221E-01	6.044E+09	1.794E+08	1.789E+09	9.781E+06	5.584E+03	4.803E+06	2.116E+06	1.530E+07	6.607E+07	7.038E+06
	XPSO	2.914E+07	1.308E+09	2.903E+10	5.087E+08	1.409E+10	4.426E+08	2.935E+06	2.496E+08	2.272E+08	3.743E+07	5.140E+10	5.459E+07
	EAPSO	1.945E+08	8.334E-01	1.318E+10	3.507E+09	7.893E+09	3.833E+08	6.132E+03	1.186E+08	2.804E+08	2.300E+08	2.398E+08	3.495E+07
	SRPSO	1.570E+08	1.798E+07	3.210E+10	2.481E+10	1.735E+10	4.530E+09	1.751E+08	3.262E+09	5.207E+09	2.646E+10	2.247E+10	2.760E+10
	GLS_MPA	1.296E+09	7.677E-01	2.958E+10	1.112E+09	1.166E+10	6.282E+09	4.205E+07	1.113E+10	3.251E+09	1.273E+10	1.281E+11	1.339E+10
	AROA	1.716E+08	6.385E+07	1.536E+10	4.905E+09	9.679E+09	1.544E+08	5.314E+03	2.266E+07	5.811E+07	1.001E+07	6.290E+07	7.148E+06
	EO	1.296E+09	6.217E+05	2.958E+10	5.274E+10	1.166E+10	7.832E+07	5.693E+03	4.623E+07	3.742E+07	3.020E+07	7.427E+07	8.608E+06
	APO2	1.296E+09	5.111E+09	5.633E+10	1.112E+09	3.179E+10	1.892E+09	6.073E+06	2.793E+09	9.901E+08	2.592E+09	3.280E+10	7.722E+09
	HBO	9.212E+09	9.171E+09	3.887E+10	3.911E+10	5.751E+10	2.127E+08	2.851E+05	4.804E+07	1.286E+08	2.774E+07	1.355E+08	1.554E+07
Worst	PDPSO	8.471E+08	5.047E+08	5.038E+10	4.286E+10	2.676E+10	8.969E+07	2.786E+05	7.132E+07	2.596E+07	5.326E+07	1.071E+08	1.879E+07
	XPSO	6.014E+09	2.674E+10	1.069E+11	6.660E+10	7.030E+10	2.934E+09	9.383E+08	1.415E+10	2.365E+09	9.950E+09	4.188E+11	4.319E+10
	EAPSO	1.128E+11	4.282E+10	2.131E+11	1.707E+11	1.583E+11	2.199E+09	2.220E+09	2.101E+09	5.924E+09	1.480E+10	7.490E+10	2.549E+09
	SRPSO	7.613E+10	1.249E+11	2.285E+11	2.022E+11	1.383E+11	2.304E+10	1.264E+09	1.748E+10	1.832E+10	7.034E+10	5.725E+10	9.958E+10
	GLS_MPA	1.296E+09	3.686E+10	7.398E+10	1.112E+09	4.326E+10	1.606E+10	1.397E+08	1.907E+10	8.838E+09	4.791E+10	2.377E+11	4.855E+10
	AROA	3.053E+10	3.193E+10	6.407E+10	4.322E+10	6.971E+10	7.439E+08	8.237E+05	2.115E+08	1.100E+09	8.701E+09	2.284E+08	2.946E+07
	EO	2.243E+11	8.429E+10	8.254E+11	2.391E+11	3.252E+11	8.033E+09	1.031E+07	1.048E+10	5.866E+09	1.188E+10	2.247E+10	7.423E+08
	APO2	6.106E+09	3.680E+10	1.098E+11	4.858E+10	7.103E+10	1.022E+10	6.829E+07	1.471E+10	9.465E+09	3.742E+10	2.551E+11	6.420E+10
	HBO	1.326E+11	3.790E+11	4.282E+11	5.709E+11	5.714E+11	2.490E+09	2.475E+07	1.090E+10	5.151E+08	7.074E+07	1.829E+10	3.275E+07

The constraints are defined as:

Subject to:

$$\begin{cases} g_i(\overline{\boldsymbol{x}}) \le 0, i = 1, \cdots, n \\ h_i(\overline{\boldsymbol{x}}) = 0, i = n+1, \cdots, n+m \end{cases} \tag{21}$$

where $g_i(\overline{\boldsymbol{x}})$ represents an inequality constraint, while $h_i(\overline{\boldsymbol{x}})$ stands for an equality constraint, n, and m is the number of the two types of constraints, respectively.

In this paper, we use an adaptive penalty function method to transform a constraint-containing problem into an unconstrained single-objective optimization problem. The adopted form of an optimization problem is defined as:

Minimize:

$$f(\overline{\boldsymbol{x}})^{'} = f(\overline{\boldsymbol{x}}) + P_g + P_h, \tag{22}$$

where P_g is a value used to punish the solution that violates the inequality constraints, P_h is a value used to punish the solution that violates the equality constraints.

According to the minimum penalty rule, the penalty should be kept as low as possible, while slightly above the infeasible solutions, at the same time, the penalty should contain relevant information about the boundaries and constraints [122].

Therefore, with reference to the range of theoretical optimal values of the optimization problem, the P_g is defined as:

$$P_g = \sum_{i=1}^{n} \left(g_i(\overline{\boldsymbol{x}}) \ge 0\right) \cdot |g_i(\overline{\boldsymbol{x}})| \times 10^{10}, \tag{23}$$

where $\left(g_i(\overline{\boldsymbol{x}}) \ge 0\right)$ is a logical value, if the conditions in parentheses are satisfied then 1, otherwise 0, 10^{10} denotes a coefficient that controls the penalty.

The P_h is defined as:

$$P_h = \sum_{i=1}^{n} \left(|h_i(\overline{\boldsymbol{x}})| \ge 10^{-4}\right) \cdot |h_i(\overline{\boldsymbol{x}})| \times 10^{10} \tag{24}$$

where $\left(|h_i(\overline{\boldsymbol{x}})| \ge 10^{-4}\right)$ is a logical value, if the conditions in parentheses are satisfied then 1, otherwise 0, 10^{-4} is a threshold given by Kumar et al [123], 10^{10} is a coefficient that controls the penalty.

4.4.1. 24 optimization problems set from CEC2020-RW

The CEC2020-RW test set serves as a benchmark for evaluating evolutionary algorithms in real-world constrained optimization, encompassing 57 problems in areas like chemical process design and mechanical structure optimization [123]. For testing the PDPSO algorithm, we selected 24 high-dimensional, complex problems from RC34-RC57, chosen for their multiple constraints and optimization challenges. Notably, RC34-RC43 include more equational constraints, whereas RC44-RC57 have fewer. Table 12 presents the optimization results across all algorithms, highlighting the optimal values in bold.

In Table 12, on problems RC34-RC43, the PDPSO algorithm's average rankings of the five fundamental indicators among all algorithms are 4.8, 4.6, 5.8, 5.2, 4.6, ranking in the middle of all algorithms; the EO algorithm shows the most substantial competitiveness among all algorithms, and its average rankings of the five fundamental indicators are 1.0, 1.4, 1.1, 1.0, and 1.2, ranking among all algorithms. On problems RC44-RC57, the PDPSO algorithm outperforms the other compared algorithms with an average ranking of 1.4, 1.6, 1.7, 1.6, 1.5 for the five fundamental indicators across all algorithms; the second-ranked algorithm is AROA, with an average ranking of 2.8, 3.3, 2.9, 3.1, 2.9 for the five fundamental indicators across all algorithms; EO's average rankings on these problems were 6.0, 6.4, 4.8, 4.1, 6.0. Summarizing, the PDPSO algorithm is more suitable for high dimensional problems with multiple inequality constraints, as it exhibits high optimization accuracy and stability in these problems.

4.4.2. Heat exchanger network design

Heat exchanger networks are vital in the chemical and refining industries, efficiently transferring heat between different processes. The goal of this case is to design a network of heat exchangers that minimizes total costs by optimizing structural parameters. This involves multi-stage heat transfer between hot and cold fluids, requiring a balance between energy recovery and equipment investment, as detailed in CEC2020-RW [123].

The statistics results of all algorithms for the five fundamental indicators of the optimization results are shown in Table 13, where the optimal value of each indicator is bolded. The theoretical optimum solution to this problem is about 7.0490369540E+03. The Best and Median values of PDPSO and EAPSO on this optimization problem are close to the theoretical optimum, but the Std value of PDPSO is smaller and more stable. The error value of the Mean of PDPSO is only 12.74% of the suboptimal algorithm EAPSO, and the Std value is only 9.0% of that of EAPSO, which shows excellent optimization ability and stability.

4.4.3. Industrial refrigeration system design

The design of industrial refrigeration systems involves complex engineering optimization that balances thermodynamic efficiency, equipment cost, and operational reliability. This problem features 14 design variables and 15 nonlinear constraints, reflecting

Table 13
The comparison results of the heat exchanger network design problem.

Algorithm	Mean	Std	Best	Median	Worst
PDPSO	7.06765810E+03	7.17514089E+01	7.04903695E+03	7.04903695E+03	7.37116705E+03
XPSO	1.81100700E+20	1.60838865E+20	2.16405890E+19	1.37960833E+20	5.65494690E+20
EAPSO	7.19509172E+03	7.98606937E+02	7.04903695E+03	7.04903695E+03	1.14234313E+04
SRPSO	9.34172282E+10	1.72131920E+11	1.19141147E+08	2.82244138E+10	7.60382400E+11
GLS_MPA	8.40634846E+15	1.67035504E+16	1.44171556E+12	8.36455386E+14	5.71784051E+16
AROA	6.20231120E+21	2.78873838E+21	8.27890036E+18	7.42844010E+21	7.42844010E+21
EO	8.49252851E+15	4.65143122E+16	2.53320399E+05	1.10998039E+09	2.54769596E+17
APO2	1.06259969E+12	2.13979518E+12	2.55936620E+08	1.25691859E+11	1.01570093E+13
HBO	2.11338090E+15	4.39555165E+15	7.83701058E+03	4.92898125E+13	1.65988834E+16

Table 14
The comparison results of the industrial refrigeration system design problem.

Algorithm	Mean	Std	Best	Median	Worst
PDPSO	9.61950475E-02	3.02427611E-02	4.52801837E-02	9.58243324E-02	1.74990063E-01
XPSO	7.20949314E+09	1.02509664E+10	1.62942916E+04	4.05000748E+09	4.02237587E+10
EAPSO	1.56031079E+09	3.54860567E+09	3.22961272E-02	6.02815023E-02	9.36186474E+09
SRPSO	1.24824874E+09	3.23682666E+09	5.18691188E-02	1.46684951E-01	9.36186595E+09
GLS_MPA	5.51027935E-01	2.01014575E+00	3.63249934E-02	1.23162263E-01	1.11359148E+01
AROA	4.05681227E+09	4.71844098E+09	3.26871059E-02	9.16966119E+03	9.36186474E+09
EO	1.56031079E+09	3.54860567E+09	3.22277801E-02	5.71937844E-02	9.36186474E+09
APO2	2.80855942E+09	4.36348651E+09	3.22518331E-02	5.62457094E-02	9.36186474E+09
HBO	1.25620792E-01	3.71521647E-02	6.85062211E-02	1.20603607E-01	2.11519725E-01

Table 15
The comparison results of the four-stage gear box design problem.

Algorithm	Mean	Std	Best	Median	Worst
PDPSO	3.31833821E+11	1.11824266E+12	4.44423277E+01	2.31771840E+10	6.14262572E+12
XPSO	1.06117807E+11	1.67138571E+11	5.44383716E+01	3.97201692E+10	7.07002783E+11
EAPSO	2.96455645E+12	7.32077836E+12	8.70292941E+01	1.46084543E+11	3.21939421E+13
SRPSO	6.60242981E+10	2.78192229E+11	5.25472175E+01	8.55480473E+01	1.52659483E+12
GLS_MPA	2.01109187E+12	1.69353433E+12	1.08252593E+10	1.76428028E+12	5.92899012E+12
AROA	2.44622945E+11	7.96790661E+11	4.47481437E+01	1.13325250E+10	4.06524416E+12
EO	3.18758316E+12	2.57285561E+12	5.63078416E+08	2.88876158E+12	9.23085893E+12
APO2	6.13552418E+11	8.33131842E+11	6.31368424E+01	2.70674672E+11	3.62381468E+12
HBO	5.96500693E+12	4.85250388E+12	2.54890297E+11	4.37262781E+12	2.17801011E+13

interactions among key components in the refrigeration cycle. It is characterized by high dimensionality, nonlinearity, and tightly coupled constraints. Specific definitions of the optimization objective and constraints can be found in CEC2020-RW [123].

Table 14 shows the statistical results of all algorithms for the five key optimization indicators, with the optimal values highlighted. The theoretical optimal solution of this case is approximately 3.2213000814E-02. The Mean value of PDPSO is closest to this optimal value, with an error of only 68.50% compared to the suboptimal HBO algorithm. Regarding the Std and Worst values, PDPSO shows the best stability among all the algorithms. In this problem, the PDPSO algorithm has an average result on the Best and Median values but remains a feasible solution.

4.4.4. Four-stage gear box design

Four-stage gearboxes are essential in heavy machinery, such as wind turbines and ship propulsion systems, allowing for the conversion of high torque and low speed to low torque and high speed. The problem involves 22 design variables and 86 nonlinear constraints aimed at minimizing the gearbox's weight while ensuring it meets strength, lifetime, and geometric compatibility requirements. It is a mixed-integer nonlinear problem, with details on the optimization objectives and constraints available in CEC2020-RW [123].

The statistics results of all algorithms for the five fundamental indicators of the optimization results are shown in Table 15, where the optimal value of each indicator is bolded. The theoretical optimal solution of the problem is about 3.5359231973E+01, on which all the algorithms were unsuccessful in finding a feasible solution. Hence, the Mean, Std, and Worst values are not analyzed further. For the Best value, the PDPSO algorithm performs better than the other compared algorithms, with a 3.26% decrease in the error value compared to the suboptimal algorithm AROA. Specifically, only the Median value of SRPSO remains a feasible solution, indicating that SRPSO outperforms the other algorithms in finding a feasible solution to this problem.

In this problem, while PDPSO, XPSO, EAPSO, SRPSO, AROA, and APO2 found feasible solutions, only the median value from the SRPSO algorithm was viable. This suggests that despite the challenge, feasible designs can emerge from repeated experiments.

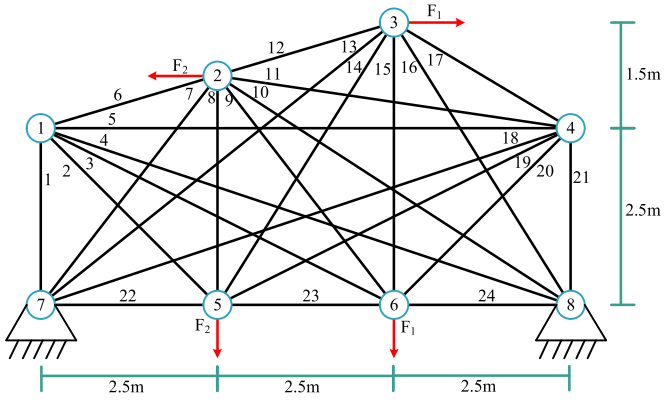


Fig. 13. Schematic of the 24-truss optimization problem.

Table 16
The comparison results of the 24-bar truss design problem.

Algorithm	Mean	Std	Best	Median	Worst
PDPSO	1.2570884354E+02	7.0664556109E+00	1.2007977372E+02	1.2444526957E+02	1.4240767940E+02
XPSO	1.6155700681E+02	5.1563627640E+01	1.2625177278E+02	1.4131542061E+02	2.9072159814E+02
EAPSO	1.6224217512E+02	2.1621473658E+01	1.2625177278E+02	1.6558811592E+02	1.9111291848E+02
SRPSO	1.4332617147E+02	2.0109384579E+01	1.2176003017E+02	1.3454373194E+02	1.7895587488E+02
GLS_MPA	1.3976032173E+02	1.1061302735E+01	1.2625177278E+02	1.3819987144E+02	1.6468217362E+02
AROA	1.3647434810E+02	9.8501112571E+00	1.2625177278E+02	1.3519180744E+02	1.6123395098E+02
EO	1.4063153466E+02	1.8850403291E+01	1.2071074623E+02	1.3859798346E+02	1.7249282938E+02
APO2	1.4058407463E+02	9.0023774639E+00	1.3329725267E+02	1.3765384929E+02	1.6219183690E+02
HBO	1.3394499946E+02	7.1535241968E+00	1.2333493526E+02	1.3366589997E+02	1.4547054397E+02

However, the gap between these feasible and optimal solutions is significant, so it's advisable to increase the population size to enhance the diversity of candidate design solutions or enhance the number of iterations to further optimize those feasible solutions.

4.4.5. 24-bar truss design

Truss design is a key challenge in structural engineering focused on minimizing weight by optimizing the cross-sectional area or material properties of the bars while meeting stress, displacement, and frequency constraints. This addresses the truss topology optimization problem from Hu et al. [9], featuring 24 design variables, with the schematic illustrated in Fig. 13.

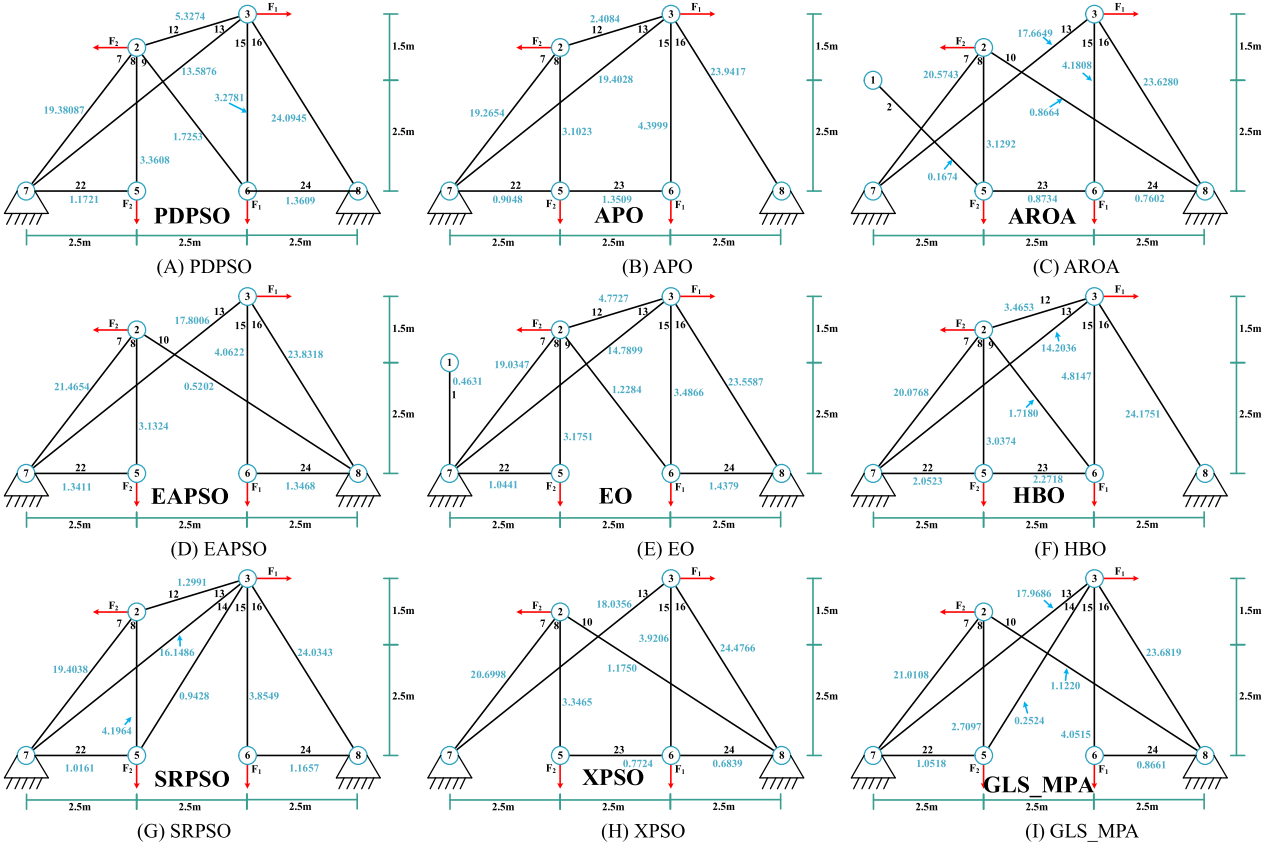
The statistics results of all algorithms for the five fundamental indicators of the optimization results are shown in Table 16, where the optimal value of each indicator is bolded. Fig. 14 shows a schematic diagram of the Truss structure for the best solution of the 9 algorithms. According to Fig. 14 and Table 16, PDPSO outperforms the other compared algorithms in five fundamental indicators when optimizing this problem.

The algorithm (PDPSO) with the smallest weight in the optimal structure uses 9 truss structures. SRPSO, EO, and HBO use a 9-truss structure and also obtain a smaller structural weight. Other algorithms have more weight, although the number of trusses is mostly 8, proving that these algorithms fall into locally optimal structures when handling the 24-bar truss design problem. With respect to node selection, EO and AROA choose to keep node 1 and abandon node 4. The optimal structure of EO is significantly lighter than the optimal structure of AROA, and the optimal structure of EO is ranked second among all the algorithms, which is only 0.53% heavier than the optimal structure of the PDPSO algorithm, indicating that the selection of node 1 even does not have a significant impact on the weight of the optimal structure. All algorithms abandon node 4, indicating that the structure using this node will substantially increase the structure weight. Taken together, both the PDPSO and EO algorithms show strong competitiveness in this problem since they use more trusses and obtain a smaller structural weight in this problem. The PDPSO algorithm achieves the best performance in all five fundamental performance indicators on this problem and shows the best optimization accuracy and stability.

4.4.6. Rolling element bearing design

Rolling bearings are crucial for reducing friction and supporting rotating parts in mechanical systems, impacting equipment life, efficiency, and reliability. This optimization problem seeks to enhance bearing load capacity with 10 variables and 9 nonlinear design constraints, involving both discrete and continuous variables. A schematic representation can be found in Fig. 15, with detailed objective and constraint definitions in CEC2020-RW [123].

The statistics results of all algorithms for the five fundamental indicators of the optimization results are shown in Table 17, where the optimal value of each indicator is bolded. The theoretical optimal solution of this problem optimization problem is about 1.4614135715E+04. PDPSO, EAPSO, and HBO have the same accuracy except for the Std value. The Std value of the PDPSO algorithm



(A) PDPSO (B) APO (C) AROA

(D) EAPSO (E) EO (F) HBO

(G) SRPSO (H) XPSO (I) GLS_MPA

Fig. 14. Structure diagram corresponding to the optimal values of the 9 algorithms.

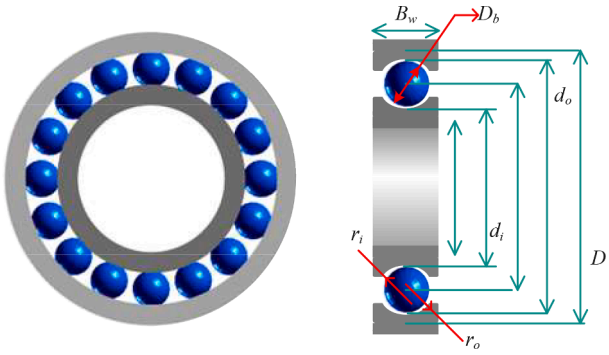


Fig. 15. Schematic of the rolling element bearing.

Table 17
The comparison results of the rolling element bearing design problem.

Algorithm	Mean	Std	Best	Median	Worst
PDPSO	1.69582022E+04	9.06353087E-12	1.69582022E+04	1.69582022E+04	1.69582022E+04
XPSO	1.75997157E+04	3.09677661E+02	1.72059516E+04	1.74962795E+04	1.84325567E+04
EAPSO	1.69582022E+04	5.14487897E-12	1.69582022E+04	1.69582022E+04	1.69582022E+04
SRPSO	1.69582050E+04	4.57056762E-03	1.69582022E+04	1.69582032E+04	1.69582227E+04
GLS_MPA	1.69582688E+04	4.14711747E-02	1.69582105E+04	1.69582678E+04	1.69583709E+04
AROA	1.69582022E+04	2.66766452E-05	1.69582022E+04	1.69582022E+04	1.69582023E+04
EO	1.69616340E+04	1.23223324E+01	1.69582022E+04	1.69582022E+04	1.70210805E+04
APO2	1.69582026E+04	1.79156001E-03	1.69582022E+04	1.69582022E+04	1.69582120E+04
HBO	1.69582022E+04	8.54517885E-12	1.69582022E+04	1.69582022E+04	1.69582022E+04

Table 18
The comparison results of the topology optimization problem.

Algorithm	Mean	Std	Best	Median	Worst
PDPSO	2.6393464970E+00	3.4967619351E-15	2.6393464970E+00	2.6393464970E+00	2.6393464970E+00
XPSO	3.1413006834E+00	1.2083960730E-01	2.9241340617E+00	3.1386079969E+00	3.5151642035E+00
EAPSO	2.6393476945E+00	1.2068361980E-06	2.6393466843E+00	2.6393472419E+00	2.6393511846E+00
SRPSO	2.6412612632E+00	2.2828697250E-03	2.6396442685E+00	2.6403400112E+00	2.6495299475E+00
GLS_MPA	2.6393464970E+00	0.0000000000E+00	2.6393464970E+00	2.6393464970E+00	2.6393464970E+00
AROA	2.6393464970E+00	1.9996771556E-15	2.6393464970E+00	2.6393464970E+00	2.6393464970E+00
EO	2.6408667548E+00	3.8391595632E-03	2.6393464970E+00	2.6393464970E+00	2.6579955885E+00
APO2	2.6393464970E+00	4.9479177602E-16	2.6393464970E+00	2.6393464970E+00	2.6393464970E+00
HBO	2.6393464970E+00	1.7159916505E-15	2.6393464970E+00	2.6393464970E+00	2.6393464970E+00

possesses the same smaller order of magnitude as the EAPSO with the minimum value of Std. Thus PDPSO possesses better optimization ability and stability on this problem.

4.4.7. Topology optimization problem

Topology Optimization is a design methodology that optimizes material distribution in a given domain to achieve objectives like minimum weight or maximum stiffness while meeting performance constraints. It is commonly used in mechanical, aerospace, automotive, and architectural fields, and involves algorithms that create innovative lightweight structures. The problem, based on the power-law approach, involves 30 variables and 30 inequality constraints, with specific definitions available in CEC2020-RW [123].

The statistics results of all algorithms for the five fundamental indicators of the optimization results are shown in Table 18, where the optimal value of each indicator is bolded. The theoretical optimal solution of the problem is about 2.6393464970E+00. where PDPSO, GLS_MPA, AROA, APO2, and HBO obtained the same accuracy except for the Std value, and the optimal solution was found in every repetition of the experiments, so PDPSO possesses better optimization ability and stability in this problem.

4.4.8. Traveling salesman problem

The Traveling Salesman Problem (TSP) is a classic combinatorial optimization challenge that seeks the shortest closed circuit among a set of cities, where each city is visited once before returning to the starting point. For this problem, we randomly selected a city cluster with 30 cities, as shown in Fig. 16. The optimization involves 30 design variables. It aims to minimize the Hamiltonian circuit distance with no constraints. Given TSP's low computational demand, we set a maximum of 2000 iterations for all algorithms while maintaining consistent parameter settings.Fig. 17 displays the average convergence curve for all algorithms used.

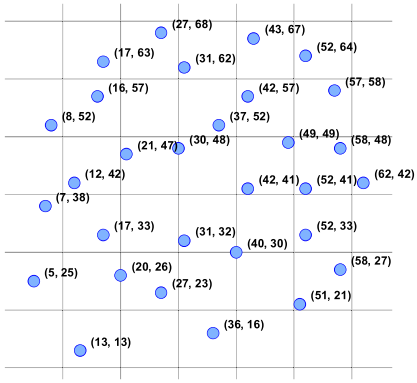


Fig. 16. Location Information Map.

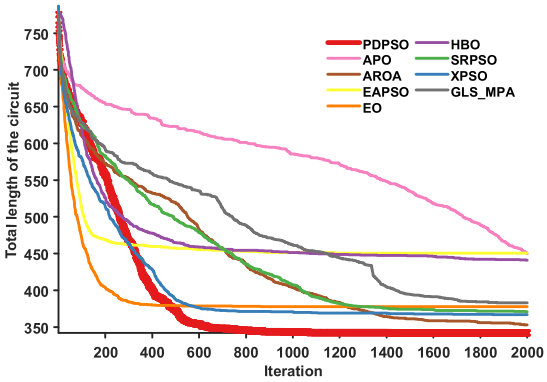


Fig. 17. Convergence curves for all algorithms.

Table 19
The comparison results of the TSP problem.

Algorithm	Mean	Std	Best	Median	Worst
PDPSO	341.93	31.45	309.89	336.43	413.24
XPSO	366.99	30.12	313.94	362.49	442.94
EAPSO	450.44	47.73	384.64	441.89	602.21
SRPSO	370.85	37.85	311.72	368.33	485.60
GLS_MPA	383.04	30.38	332.71	381.25	452.35
AROA	352.91	24.60	313.79	353.84	425.24
EO	377.80	41.36	322.93	381.69	451.77
APO2	449.61	61.50	318.33	451.85	585.66
HBO	441.08	30.68	365.79	448.63	491.09

The TSP is challenging to optimize due to its combinatorial explosion and computational complexity. Many algorithms risk missing the global optimum by focusing too soon on shorter paths, making them prone to local traps. As shown in Fig. 17, EAPSO and EO ceased updating after 300 iterations, whereas XPSO and HBO did so after 600 iterations. Although PDPSO converged more slowly in the first 200 iterations, it continued to update effectively and found a near-optimal solution after 700 iterations. Table 19 presents statistics for all algorithms, highlighting that PDPSO outperformed others in most path length indicators, except for the Std value.

Fig. 18 shows the path circuits of the best solutions from the 9 algorithms. EAPSO and HBO exhibit noticeable loop structures, indicating their high susceptibility to local optima in the TSP problem.

4.4.9. Robot swarm command generation problem

The path planning problem for robot swarms involves coordinating multiple robots to complete tasks efficiently while navigating complex environments with dynamic and static obstacles. It is characterized by the need for collaboration and competition for resources, as well as the challenges presented by environmental dynamics and uncertainty.

Referring to the multi-robot path planning model proposed by Akay et al. [124], we construct a path instruction generation

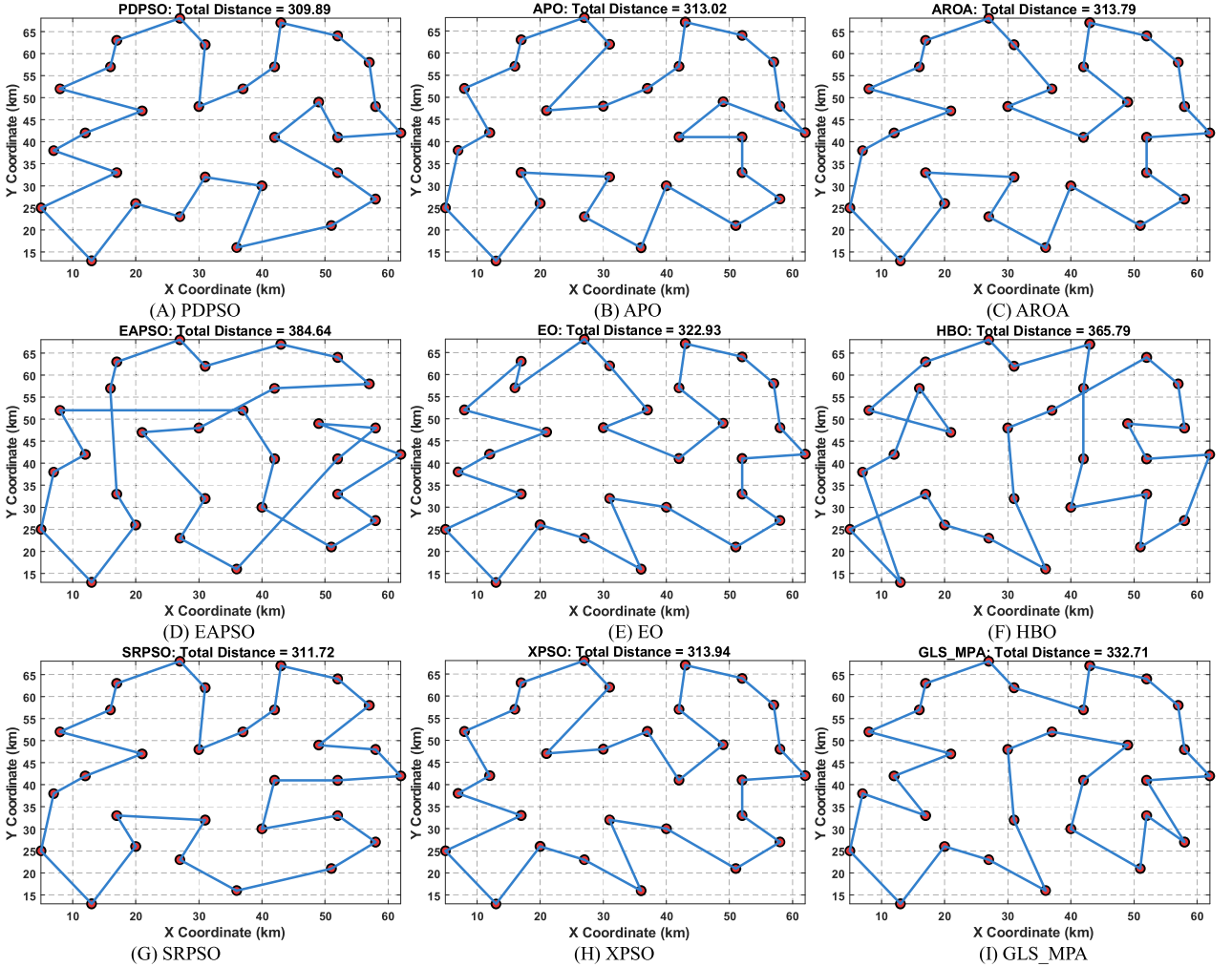


Fig. 18. Circuit plan corresponding to the optimal values of the 9 algorithms.

Table 20
The comparison results of the robot swarm command generation problem.

Robot ID	PDPSO	APO	AROA	EAPSO	EO	HBO	SRPSO	XPSO	GLS_MPA
1	95.65%	96.48%	96.15%	86.43%	89.08%	96.23%	95.62%	95.73%	96.43%
2	96.47%	95.89%	96.52%	90.82%	90.64%	96.66%	95.75%	96.48%	95.63%
3	98.33%	98.36%	98.71%	94.89%	93.57%	96.36%	91.28%	92.37%	97.96%
4	94.82%	95.94%	95.96%	88.39%	90.80%	95.49%	90.75%	92.74%	95.38%
5	96.37%	93.40%	96.15%	91.20%	90.31%	93.05%	94.55%	95.91%	94.17%
6	96.56%	97.84%	98.26%	88.58%	87.03%	97.58%	94.95%	97.25%	96.30%
7	95.47%	95.79%	95.91%	89.09%	86.53%	95.19%	88.01%	94.80%	92.31%
8	96.30%	95.89%	96.97%	90.45%	85.43%	95.21%	95.80%	96.63%	96.54%
9	93.92%	95.81%	95.61%	88.33%	90.53%	95.48%	90.93%	94.75%	93.46%
10	95.14%	97.82%	97.77%	95.49%	86.48%	98.03%	96.02%	95.59%	97.08%
Average	95.90%	96.32%	96.80%	90.37%	89.04%	95.93%	93.37%	95.22%	95.53%

problem for a large-scale robot cluster. Here, the number of robots is 10, and the number of obstacles is 60. The problem requires successive calls to the algorithm to simultaneously optimize the turning angles and moving distances of 10 robots, so the optimization problem contains 60 design variables and many constraints. The problem requires combining multiple conflicting objectives, such as path length, safe distance, execution time, and energy consumption, which are further complicated by inter-robot collision avoidance constraints. The experiment has the experimental property of heavily repeated experiments since each robot must repeatedly invoke the algorithm to generate local paths in real-time.

Path efficiency can be used to measure the movement efficiency of individuals in a robot swarm, and this value is based on a straight-line distance as a metric, calculated as:

$$R_i = \frac{L_i}{L0_i}, \tag{25}$$

where L_i denotes the distance traveled by the ith robot; and $L0_i$ denotes the straight line distance of the ith robot.

Table 20 summarizes the path efficiency of each algorithm on this problem. The average path efficiency of AROA on this problem is 96.80%, which is better than the other compared algorithms. The path efficiency of PDPSO is 95.90%, which is only 0.9% less than AROA.

Fig. 19 displays the moving trajectories of the robot swarms under the control of different algorithms, where the robot swarms move from above the map to below, and the starting point is from left to right from the individual robots with IDs 1 to ID 10 in order. In Fig. 19, the EAPSO, EO, and SRPSO algorithms have significant shaking in their moving trajectories, while the other algorithms maintain good stability. According to Fig. 19, the global paths of robots 4, 9, and 10 of the PDPSO algorithm chose a different way of bypassing the obstacles from the other comparative algorithms, which is related to the environmental differences and the local path planner's limitation, and has little relatedness to the optimization algorithm's solving accuracy. This limitation means that the algorithm can make the optimal instruction only for the current time node due to the insufficient information input to the path planner about the global environment. In this problem, PDPSO, APO, AROA, HBO, and XPSO obtained smoother and more stable path instruction results in this problem, this conclusion is not based merely on Table 20 as a reference.

4.4.10. Large-scale sensor deployment problem

Sensor deployment optimization in wireless sensor networks aims to maximize coverage and reduce blind spots by strategically placing resource-constrained nodes [125]. In this case, a balance between sensing probability and redundancy from overlapping areas is crucial, as the sensed environment is influenced by various complex factors, posing challenges for algorithm robustness.

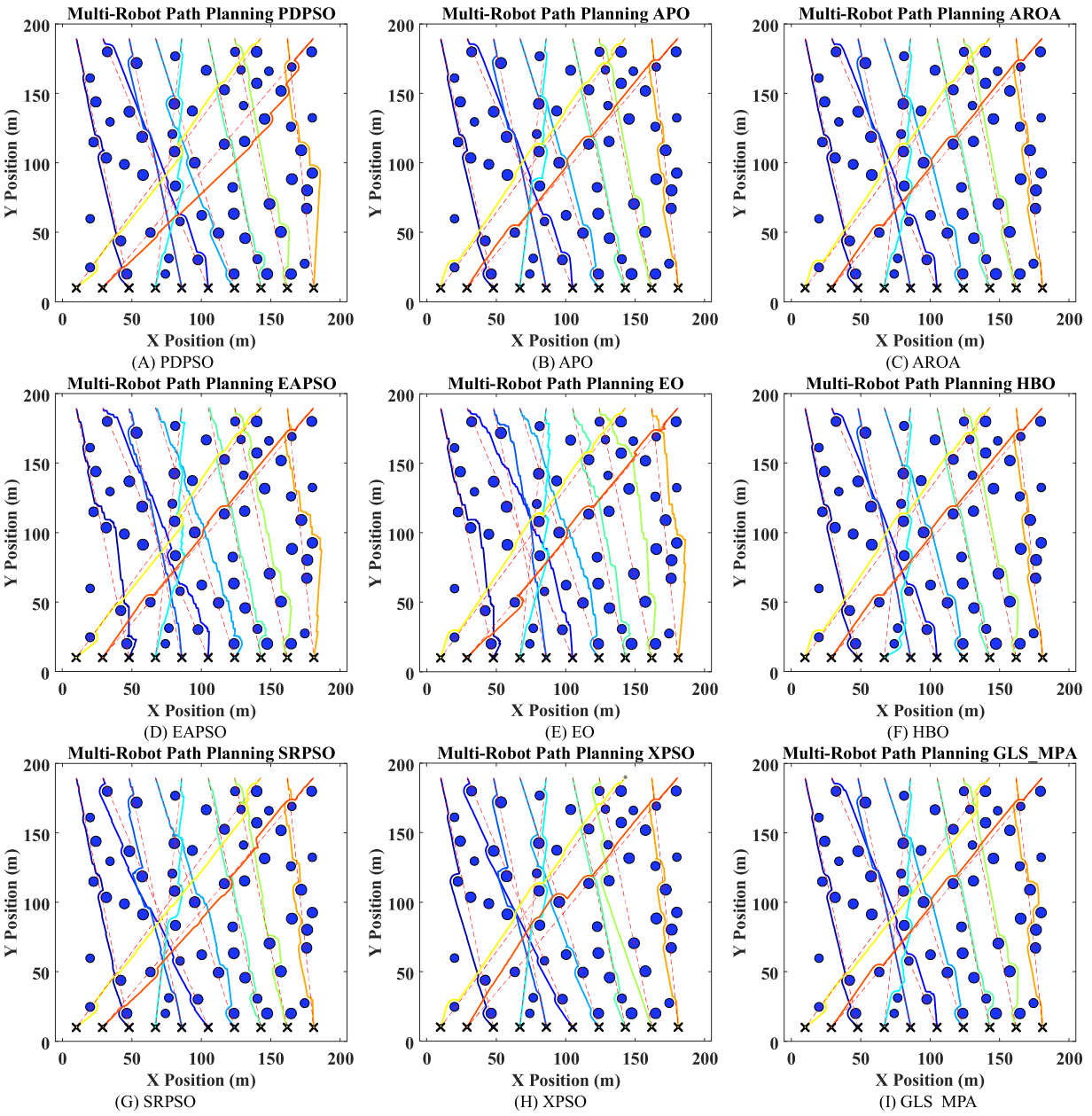
The probabilistic perception model of the sensor is schematically illustrated in Fig. 20. The perception probability $P_{S,Q}$ of the sensor node S to the monitoring point Q can be expressed as:

$$P_{S,Q} = \begin{cases} 1 & d(S,Q) \le r - r_e \\ e^{-\alpha_1 \frac{(r_e - r + d(S,Q))^{\beta_1}}{(r_e + r - d(S,Q))^{\beta_2}} + \alpha_2} & r - r_e \le d(S,Q) \le r + r_e \\ 0 & d(S,Q) \ge r + r_e \end{cases}, \tag{26}$$

where r denotes the nominal sensing radius, r_e denotes the sensing radius fluctuation value, $d(S,Q)$ denotes the Euclidean distance between the sensor node S and the monitoring point Q, and α_1, α_2, β_1, β_2 denotes some environment-related coefficients.

In this case, the sensor coverage environment is defined as a square with an edge length 40 *km* and the number of sensors is 30, $r = 4$ *km*, $r_e = 2.5$ *km*, $\alpha_1 = 1$, $\alpha_2 = 0$, $\beta_1 = 1$, $\beta_2 = 1.5$ [126]. The probability of sensing in the overlapping region of different sensors is calculated as the probability of not being sensed by both sensors simultaneously. The presence of uniform signal interference in the environment makes the sensing probability of the sensors attenuated, and the degree of signal attenuation is shown in Fig. 21.

There were 60 design variables for this problem, the statistics results of all algorithms for the five fundamental indicators of the optimization results are shown in Table 21, where the optimal value of each indicator is bolded. According to Table 21, PDPSO obtains



(A) PDPSO (B) APO (C) AROA
(D) EAPSO (E) EO (F) HBO
(G) SRPSO (H) XPSO (I) GLS MPA

Fig. 19. Paths corresponding to the 9 algorithms.

the best value in the sensor coverage statistics and ranks second in the Mean, Median, and Worst values, with a reduction of 0.01%, 0.04%, and 0.13% relative to the XPSO algorithm coverage, respectively. Overall, both PDPSO and XPSO perform better in solving this problem.

Fig. 22 shows a schematic representation of the sensor coverage of the solution corresponding to the Best values of the 9 algorithms. There are apparent uncovered regions for HBO and obvious gaps in the boundaries of the coverage regions for APO, GLS_MPA and SRPSO, which proves that these two algorithms have the problem of searching insufficiently in this problem.

4.4.11. Ultra-large-scale wind farm layout optimization problem

Wind energy is the fastest-growing renewable energy source worldwide, and the establishment of wind farms allows for the conversion of wind into a renewable, clean energy source. The spatial arrangement of the turbines highly influences the efficiency of a wind farm's energy conversion. This layout must consider both the power generation efficiency of each turbine and the optimal utilization of natural resources. When wind flows through a turbine rotor, some of its kinetic energy is transformed into electrical

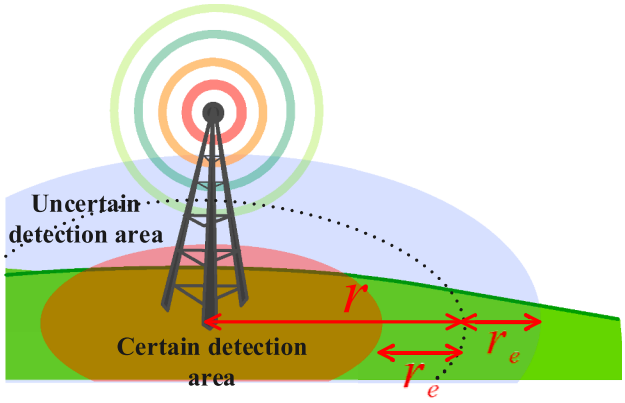


Fig. 20. Schematic of probabilistic perception model.

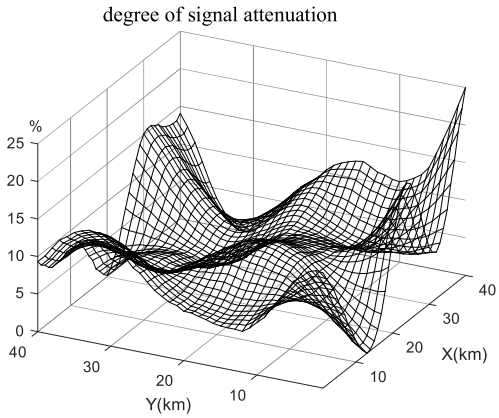


Fig. 21. Percentage of signal attenuation.

Table 21
The comparison results of the large-scale sensor deployment problem.

Algorithm	Mean	Std	Best	Median	Worst
PDPSO	76.48%	1.4386E-01	76.68%	76.48%	76.18%
XPSO	75.11%	1.1424E+00	76.24%	75.46%	71.13%
EAPSO	76.49%	8.5791E-02	76.59%	76.52%	76.31%
SRPSO	76.13%	2.6368E-01	76.41%	76.18%	75.24%
GLS_MPA	75.30%	4.3937E-01	76.30%	75.31%	74.47%
AROA	75.95%	2.1678E-01	76.32%	75.97%	75.60%
EO	75.48%	7.0650E-01	76.14%	75.61%	73.15%
APO2	66.62%	1.4304E+00	69.54%	66.28%	65.10%
HBO	67.06%	5.3591E-01	68.13%	67.13%	65.86%

energy. However, the remaining airflow creates a low-speed turbulence zone downstream of the turbine, which gradually disperses and generates a wake pattern similar to that produced by a ship. Fig. 23 illustrates this process, with the blue and green grids representing isopycnic surfaces at different wind speeds. Suppose a downstream turbine is situated in this wake area. In that case, its ability to capture wind will be significantly diminished, leading to a considerable reduction in the overall power output of the wind farm. Therefore, it is essential to maximize the total power generation capacity by strategically placing turbines within a limited area. This case must focus on minimizing the wake effect and construction costs to optimize wind farm layouts effectively.

Referring to the wind farm layout problem model proposed by Wang et al. [127], we set a test field as a square site with the side length of 3 *km*, the number of wind turbines as 100, and the wake diffusion coefficient as 0.075 to adapt the simulation environment to the large land area scenario, while other parameters remain unchanged from the wind farm layout problem model proposed by Wang et al. The problem contains 200 design variables and many constraints. It is very susceptible to other turbines and environmental factors during the layout process, making it a complex optimization problem with multiple constraints, high dimensionality, and

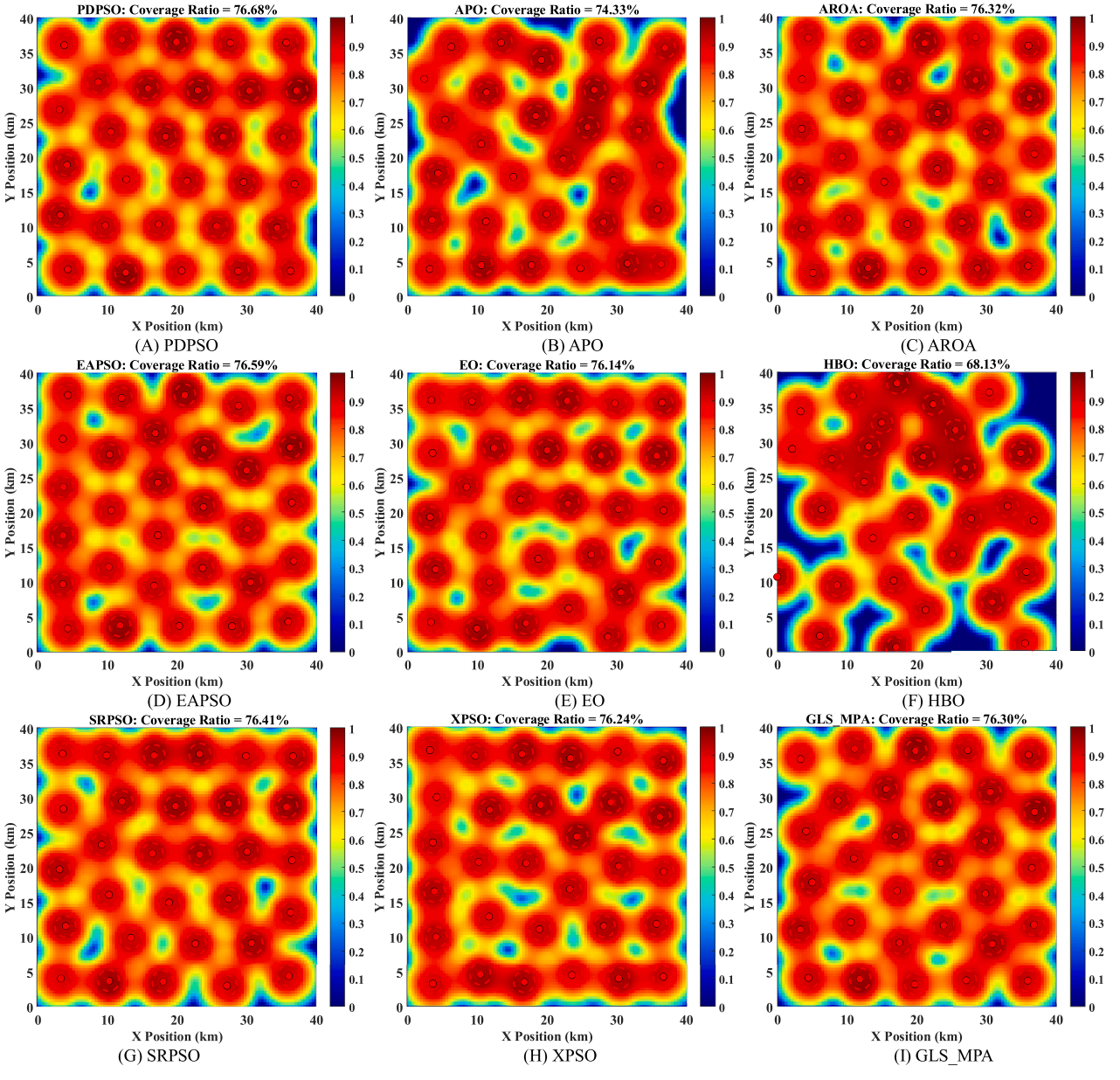


Fig. 22. Coverage program corresponding to the optimal values of the 9 algorithms.

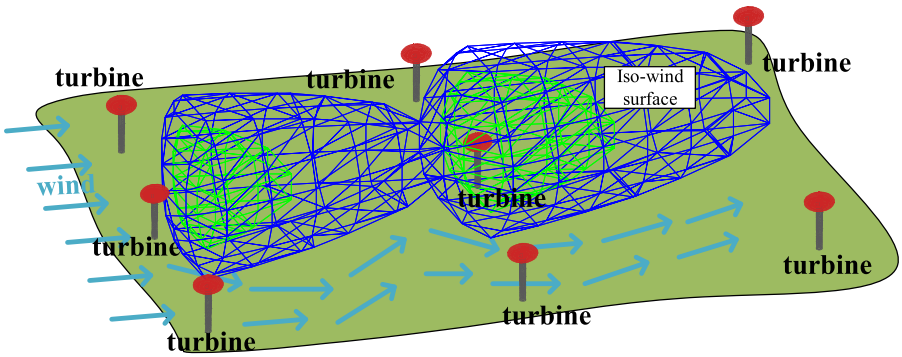


Fig. 23. Schematic diagram of wake effect.

Table 22
The comparison results of the ultra-large-scale wind farm layout optimization problem.

Algorithm	Mean	Std	Best	Median	Worst
PDPSO	2.6898106573E+04	5.5994610531E+02	2.7832743213E+04	2.6857174703E+04	2.5888626616E+04
XPSO	9.5942697793E+03	7.1723372618E+03	2.1014003971E+04	8.3182423606E+03	-3.0658890267E+03
EAPSO	2.6868301489E+04	5.0606588588E+02	2.7709156730E+04	2.6990779750E+04	2.5411979932E+04
SRPSO	1.4204020637E+04	8.6286596740E+03	2.4182066695E+04	1.5970841678E+04	-3.1666437297E+03
GLS_MPA	2.7578452853E+04	3.2842703662E+02	2.8207497973E+04	2.7600194850E+04	2.6973857264E+04
AROA	2.5848626122E+04	3.1718416567E+03	2.7749949063E+04	2.6615738778E+04	9.9032542442E+03
EO	2.4335453079E+04	1.8777829777E+03	2.6636106893E+04	2.5022639822E+04	1.8416766858E+04
APO2	6.8490797306E+03	1.3210454492E+03	9.8567572937E+03	6.7418681325E+03	4.8282045109E+03
HBO	1.6470217294E+04	1.1398089808E+03	1.8845802622E+04	1.6349427398E+04	1.4065695040E+04

nonlinearity.

The statistics results of all algorithms for the five fundamental indicators of the optimization results are shown in Table 22, where the optimal value of each indicator is bolded. Results show that the GLS_MPA algorithm produces the highest power generation for the layout scheme, the PDPSO algorithm ranks second in power generation for the corresponding layout scheme for the Mean, Best, and Worst values, and the power generation for the layout scheme of PDPSO is slightly lower than that of GLS_MPA by 2.47%,1.33%, and 4.02%, respectively. Particularly, the Worst values of the XPSO and SRPSO algorithms are negative, which is associated with the penalty function design methodology, suggesting that some of the layout schemes of these algorithms may have seriously violated the constraints. The optimal layouts of the algorithms and the schematic diagrams of the wake effects under the corresponding main wind direction are given in Fig. 24.

Based on Fig. 24, the PDPSO, EO, and GLS_MPA wind farm optimal layouts are relatively decentralized, the different turbines can maintain a spacing between them with a relatively low impact, and the turbines almost entirely utilize the entire layout space. The optimal layouts of the APO, AROA, HBO, SRPSO, and XPSO algorithms have large underutilized power generation areas, proving that these algorithms fail to adequately search for diversified layout scenarios when handling large-scale wind farm layout problems, and it is not easy to maintain the diversity of solutions. Furthermore, the optimal layout scheme of the APO algorithm has many regions with proximity, indicating that this algorithm may have trouble finding feasible solutions in some complex scenarios.

4.4.12. Ultra-large-scale flexible job shop scheduling problem

The Flexible Job Shop Scheduling Problem (FJSP) is a complex challenge in manufacturing, focusing on coordinating multiple workpieces across various workstations without fixed process routes. This flexibility increases decision-making complexity and requires optimizing three interconnected variables: scheduling process sequences, allocating machines, and coordinating between workstations [128]. Additionally, factors such as resource costs, equipment downtime, and workpiece delay penalties must be considered. Traditional scheduling methods struggle with these interdependencies, as the goal is to minimize maximum completion time, reduce average process time, and maintain equipment health, creating a high-dimensional Pareto optimization problem. By converting various cost elements into monetary units, the problem can be reframed into a single-objective high-dimensional engineering optimization model.

Referring to the model proposed by Zhao et al. and Guo et al. [129,130], we designed a flexible job shop scheduling problem

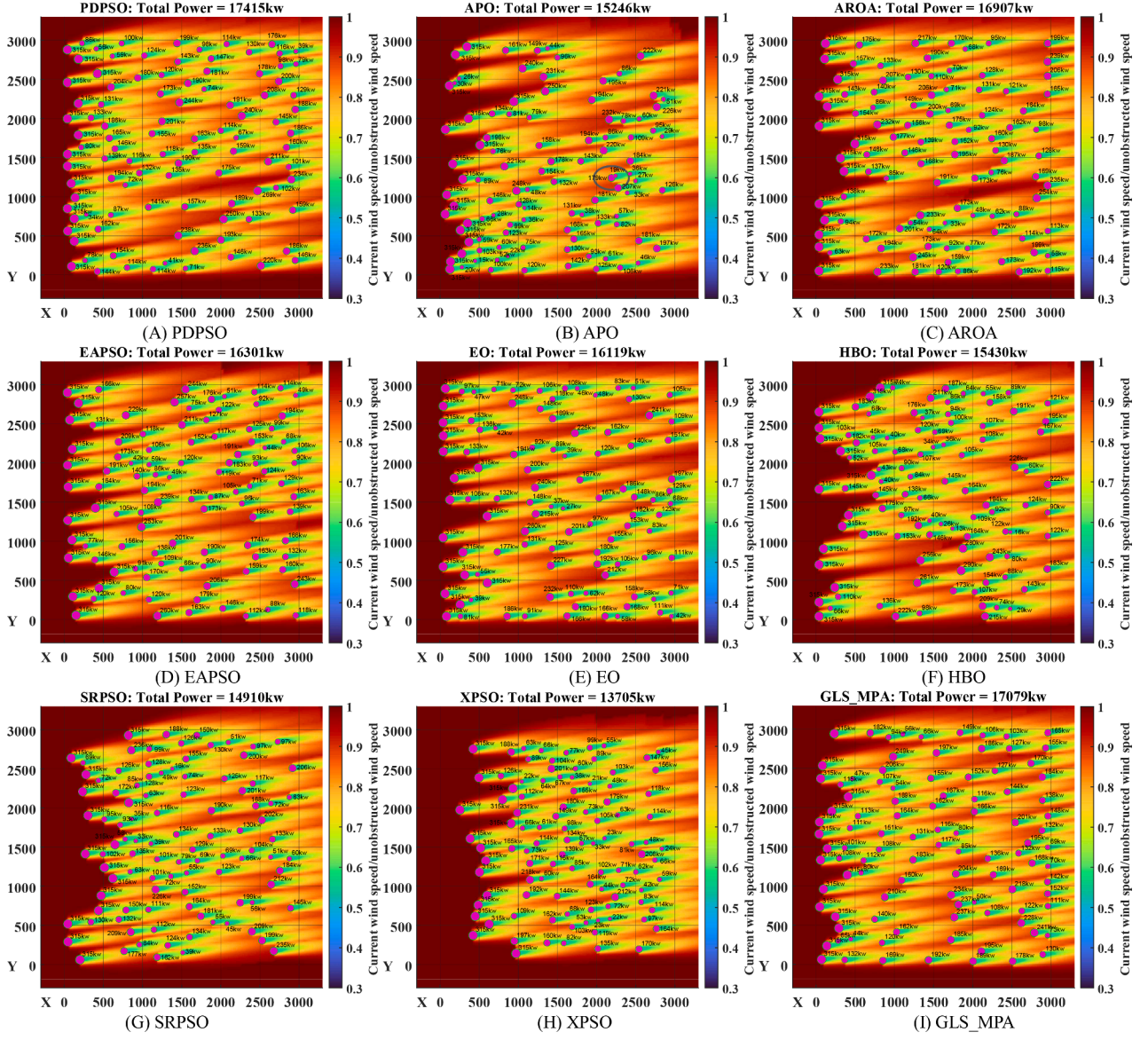


Fig. 24. Layout plan corresponding to the optimal values of the 9 algorithms.

Table 23
The comparison results of the ultra-large-scale flexible job shop scheduling problem.

Algorithm	Mean	Std	Best	Median	Worst
PDPSO	7.7766001440E+04	4.2498589306E+03	6.6704375400E+04	7.8058912200E+04	8.2181520000E+04
XPSO	7.9748243587E+04	5.0656058332E+03	7.1038699600E+04	7.9644577467E+04	8.8966450067E+04
EAPSO	8.4428159662E+04	3.6596516913E+03	7.9277866067E+04	8.4944565733E+04	9.0784232400E+04
SRPSO	8.0030700449E+04	5.3725951723E+03	7.3666149333E+04	7.9371533133E+04	8.9426257800E+04
GLS_MPA	7.7424734711E+04	6.1542462099E+03	6.9037482600E+04	7.6672154667E+04	9.1641220667E+04
AROA	8.1383204969E+04	5.6686990386E+03	6.8025426467E+04	8.1152830133E+04	8.9300588267E+04
EO	9.6374847271E+04	9.6500821986E+03	7.5935757667E+04	9.7043611267E+04	1.1339717527E+05
APO2	8.1844199191E+04	2.5063963647E+03	7.7371804000E+04	8.1567862600E+04	8.6978514333E+04
HBO	9.9540919302E+04	4.7427337916E+03	8.9469607467E+04	9.9484036533E+04	1.0650520407E+05

involving 22 machines of 4 processing operations, 83 workpieces of 4 types, and 2 maintenance methods. In this problem, the processing times of different workpieces vary across processing operations, requiring reasonable scheduling of processing sequences and preventive maintenance strategies to minimize costs. The problem has 996 dimensions and is classified as an ultra-high-dimensional engineering optimization problem. For the specific definition of the problem, please refer to the supplementary materials.

Table 23

The statistics results of all algorithms for the five fundamental indicators of the optimization results are shown in Table 22, where the optimal value of each indicator is bolded. In this problem, GLS_MPA achieved the best Mean performance, reducing the average cost by 0.3% compared to PDPSO. PDPSO performed best in terms of Best and Worst values, demonstrating the best robustness. GLS_MPA obtained a poor Worst value, proving that the algorithm is prone to instability when dealing with this problem.

The production Gantt charts corresponding to the Best values of each algorithm are shown in Fig. 25, where the horizontal axis represents time and the vertical axis corresponds to the machine and its associated production process, red and blue colors correspond to major and minor repairs. In Fig. 25, the PDPSO algorithm obtained the optimal production schedule with its intensive production planning and infrequent maintenance planning.

5. Conclusion

This paper introduces PDPSO, a competitive variant of traditional PSO that addresses issues like algorithmic instability. PDPSO features a priority-driven search strategy that improves stability and reduces time consumption, along with a dynamic candidate solution management strategy that offers diverse and effective learning targets. This approach enhances the algorithm's ability to convert computational resources into optimization power efficiently.

We conducted multiple experiments to verify the effectiveness of the PDPSO algorithm. A series of parameter combinations were generated using the grid method, and the best combination was selected via performance testing and cross-validation. Ablation experiments confirmed that the strategy combination in PDPSO is reasonable without significantly increasing time complexity. Comparison with other algorithms on high-dimensional test functions revealed that PDPSO achieves high optimization accuracy, strong stability, low time complexity, and is less affected by the dimensionality of the optimization problem.

This paper evaluates the performance of the PDPSO algorithm on idealized test functions and real-world high-dimensional optimization problems, encompassing various types (discrete, continuous, mixed), constraints (multi-equality, multi-inequality, mixed), and dimensions (10, 22, 30, 60, 200, 996). The results demonstrate that the PDPSO algorithm is robust, has strong generalization ability, and effectively searches solution spaces.

While the PDPSO algorithm demonstrates strong competitiveness, we have identified areas for improvement. Firstly, if the number of evaluations is low, it may waste resources as the priority table needs time to stabilize after initialization. Secondly, while PDPSO accelerates convergence in the mid-optimization stage, it shows only slight improvements in the later stages. To enhance its performance, we plan to focus on accelerating the stabilization of the priority table and integrating strategies that improve convergence accuracy in the late stages of optimization.

CRediT authorship contribution statement

Gang Hu: Writing – review & editing, Writing – original draft, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Peidong He:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation. **Heming Jia:** Writing – review & editing, Writing – original draft, Visualization, Supervision, Project administration, Methodology, Investigation, Data curation, Conceptualization. **Essam H. Houssein:** Writing – review & editing, Writing – original draft, Visualization, Validation,

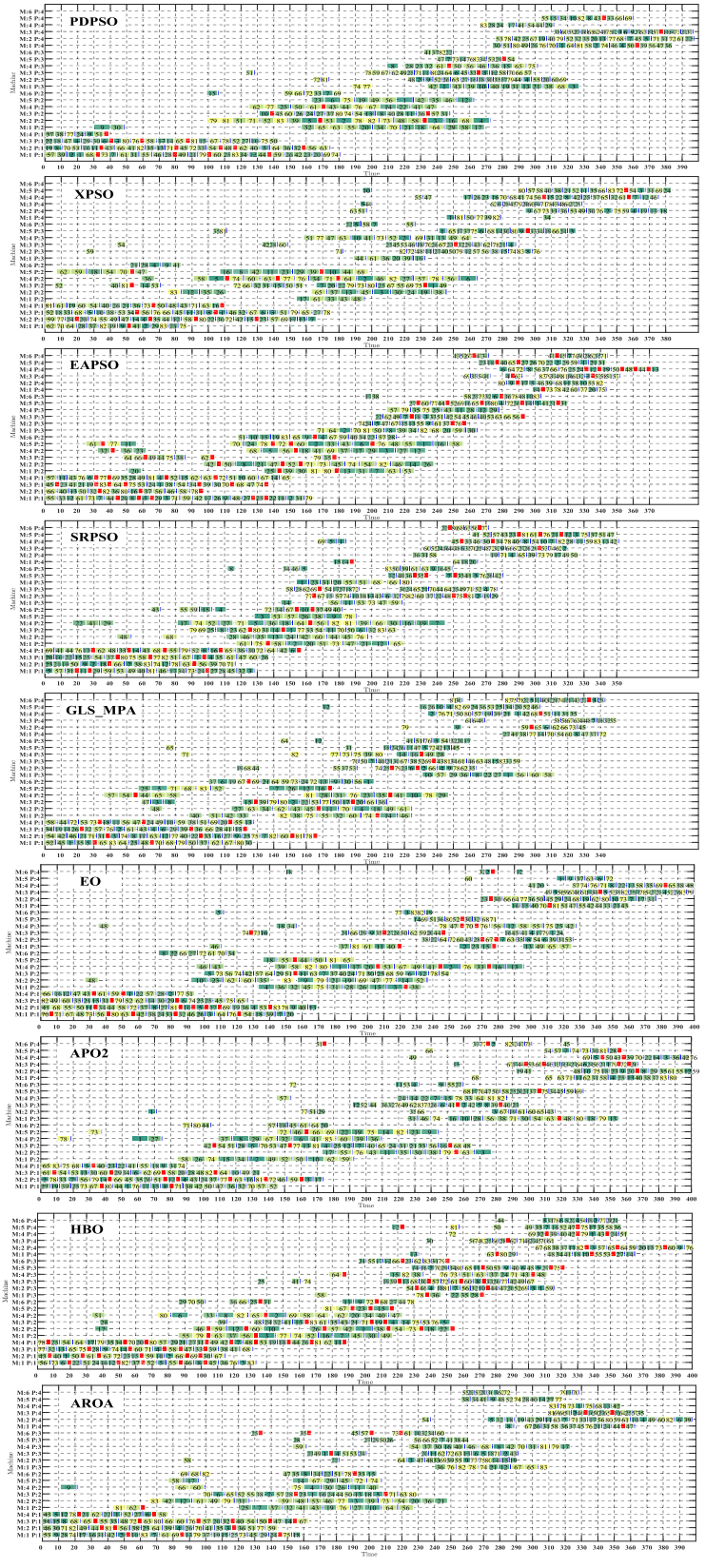


Fig. 25. Gantt charts corresponding to the optimal values of the 9 algorithms.

Resources, Methodology, Investigation, Formal analysis, Conceptualization. **Laith Abualigah:** Writing – review & editing, Writing – original draft, Supervision, Software, Project administration, Methodology, Investigation, Formal analysis, Data curation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grants No. 52375264).

Data availability

All data generated or analyzed during this study are included in this published article.

References

[1] S.N. Makhadmeh, M.A. Al-Betar, F. Al-Obeidat, O.A. Alomari, A.K. Abasi, M. Tubishat, Z. Elgamal, W. Alomoush, A multi-objective Grey Wolf optimizer for energy planning problem in smart home using renewable energy systems, Sustain. Oper. Comput. (2024), https://doi.org/10.1016/j.susoc.2024.04.001. S2666412724000059.
[2] U. Lee, A new adaptive Kriging-based optimization (AKBO) framework for constrained optimization problems: a case study on shared autonomous electric vehicle system design, Expert. Syst. Appl. 252 (2024) 124147, https://doi.org/10.1016/j.eswa.2024.124147.
[3] B. Toaza, D. Esztergár-Kiss, A review of metaheuristic algorithms for solving TSP-based scheduling optimization problems, Appl. Soft. Comput. 148 (2023) 110908, https://doi.org/10.1016/j.asoc.2023.110908.
[4] B.R.B. Sahu, A.K. Bhurjee, P. Kumar, Efficient solutions for vector optimization problem on an extended interval vector space and its application to portfolio optimization, Expert. Syst. Appl. 249 (2024) 123653, https://doi.org/10.1016/j.eswa.2024.123653.
[5] D. Wu, A. Lisser, Solving constrained pseudoconvex optimization problems with deep learning-based neurodynamic optimization, Math. Comput. Simul. 219 (2024) 424–434, https://doi.org/10.1016/j.matcom.2023.12.032.
[6] G. Hu, F. Huang, A. Seyyedabbasi, G. Wei, Enhanced multi-strategy bottlenose dolphin optimizer for UAVs path planning, Appl. Math. Model. 130 (2024) 243–271, https://doi.org/10.1016/j.apm.2024.03.001.
[7] G. Hu, B. Du, K. Chen, G. Wei, Super eagle optimization algorithm based three-dimensional ball security corridor planning method for fixed-wing UAVs, Adv. Eng. Informat. 59 (2024) 102354, https://doi.org/10.1016/j.aei.2024.102354.
[8] G. Hu, M. Cheng, G. Sheng, G. Wei, ACEPSO: a multiple adaptive co-evolved particle swarm optimization for solving engineering problems, Adv. Eng. Inf. 61 (2024) 102516, https://doi.org/10.1016/j.aei.2024.102516.
[9] G. Hu, F. Huang, K. Chen, G. Wei, MNEARO: A meta swarm intelligence optimization algorithm for engineering applications, Comput. Methods Appl. Mech. Eng. 419 (2024) 116664, https://doi.org/10.1016/j.cma.2023.116664.
[10] P.Sing Babu, A. Goswami, V.S. Pandit, Optimization of beam line parameters for space charge dominated multi-species beam using random search method, Phys. Lett. a 376 (2012) 3192–3198, https://doi.org/10.1016/j.physleta.2012.08.031.
[11] H. Duan, Z. Zhang, M.M. Rahman, X. Guo, X. Zhang, J. Cai, Insight into torrefaction of woody biomass: kinetic modeling using pattern search method, Energy 201 (2020) 117648, https://doi.org/10.1016/j.energy.2020.117648.
[12] Z. Wang, A survey on convex optimization for guidance and control of vehicular systems, Annu Rev. Control 57 (2024) 100957, https://doi.org/10.1016/j.arcontrol.2024.100957.
[13] A.H. Paras, E.G.R. Gacuan, E. Halim, A.A.N.P. Redi, J.D. German, Optimizing project scheduling using linear programming approach: a case study of heating ventilation & air conditioning mechanical installation, Procedia Comput. Sci. 234 (2024) 683–690, https://doi.org/10.1016/j.procs.2024.03.054.
[14] S. Cipolla, J. Gondzio, F. Zanetti, A regularized interior point method for sparse optimal transport on graphs, Eur. J. Oper. Res. (2023), https://doi.org/10.1016/j.ejor.2023.11.027. S0377221723008688.
[15] J. Wang, S. Qin, A recurrent neural network approach for nonconvex interval quadratic programming, Neurocomputing. 588 (2024) 127636, https://doi.org/10.1016/j.neucom.2024.127636.
[16] H.A. Pedrozo, G. Panagakos, L.T. Biegler, Including CFD rigorous models in the optimal design of carbon capture plants through trust-region methods, Chem. Eng. Sci. 286 (2024) 119646, https://doi.org/10.1016/j.ces.2023.119646.
[17] M. Aristizabal, J.L. Hernández-Estrada, M. Garcia, H. Millwater, Solution and sensitivity analysis of nonlinear equations using a hypercomplex-variable Newton-Raphson method, Appl. Math. Comput. 451 (2023) 127981, https://doi.org/10.1016/j.amc.2023.127981.
[18] S. Abbasbandy, A. Jafarian, Steepest descent method for solving fuzzy nonlinear equations, Appl. Math. Comput. 174 (2006) 669–675, https://doi.org/10.1016/j.amc.2005.04.092.
[19] Z. Zhang, X. Chen, A conjugate gradient method for distributed optimal control problems with nonhomogeneous Helmholtz equation, Appl. Math. Comput. 402 (2021) 126019, https://doi.org/10.1016/j.amc.2021.126019.
[20] M.H. Nadimi-Shahraki, S. Taghian, S. Mirjalili, An improved grey wolf optimizer for solving engineering problems, Expert. Syst. Appl. 166 (2021) 113917, https://doi.org/10.1016/j.eswa.2020.113917.
[21] K. Rajwar, K. Deep, S. Das, An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications, and open challenges, Artif. Intell. Rev. 56 (2023) 13187–13257, https://doi.org/10.1007/s10462-023-10470-y.
[22] K. Hussain, M.N. Mohd Salleh, S. Cheng, Y. Shi, Metaheuristic research: a comprehensive survey, Artif. Intell. Rev. 52 (2019) 2191–2233, https://doi.org/10.1007/s10462-017-9605-z.
[23] A.A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, H. Chen, Harris hawks optimization: algorithm and applications, Fut. Gener. Comp. Sy. 97 (2019) 849–872, https://doi.org/10.1016/j.future.2019.02.028.
[24] S. Liang, M. Yin, G. Sun, J. Li, H. Li, Q. Lang, An enhanced sparrow search swarm optimizer via multi-strategies for high-dimensional optimization problems, Swarm. Evol. Comput. 88 (2024) 101603, https://doi.org/10.1016/j.swevo.2024.101603.
[25] X. Pan, Y. Wang, Y. Lu, N. Sun, Improved artificial bee colony algorithm based on two-dimensional queue structure for complex optimization problems, Alexandr. Eng. J. 86 (2024) 669–679, https://doi.org/10.1016/j.aej.2023.12.011.
[26] B. Shu, G. Hu, M. Cheng, C. Zhang, MSFPSO: multi-algorithm integrated particle swarm optimization with novel strategies for solving complex engineering design problems, Comput. Methods Appl. Mech. Eng. 437 (2025) 117791, https://doi.org/10.1016/j.cma.2025.117791.
[27] M. Abdel-salam, S.A. Alomari, M.H. Almomani, G. Hu, S. Lee, K. Saleem, A. Smerat, L. Abualigah, Quadruple strategy-driven hiking optimization algorithm for low and high-dimensional feature selection and real-world skin cancer classification, Knowl. Based. Syst. 315 (2025) 113286, https://doi.org/10.1016/j.knosys.2025.113286.

[28] S. Chakraborty, S. Nama, A.K. Saha, An improved symbiotic organisms search algorithm for higher dimensional optimization problems, Knowl. Based. Syst. 236 (2022) 107779, https://doi.org/10.1016/j.knosys.2021.107779.
[29] Z. Hou, H. Peng, J. Li, A novel multi-strategy combined whale optimization algorithm for cascade reservoir operation of complex engineering optimization, Appl. Soft. Comput. 173 (2025) 112917, https://doi.org/10.1016/j.asoc.2025.112917.
[30] J. Zhang, M. Li, X. Yue, X. Wang, M. Shi, A hierarchical surrogate assisted optimization algorithm using teaching-learning-based optimization and differential evolution for high-dimensional expensive problems, Appl. Soft. Comput. 152 (2024) 111212, https://doi.org/10.1016/j.asoc.2023.111212.
[31] H. Geng, F. Song, J. Shen, J. Li, A classification and regression assisted optimization algorithm for high-dimensional expensive many-objective problems, Neurocomputing. 586 (2024) 127629, https://doi.org/10.1016/j.neucom.2024.127629.
[32] C. Ren, Q. Xu, Z. Meng, J.-S. Pan, Surrogate-assisted fully-informed particle swarm optimization for high-dimensional expensive optimization, Appl. Soft. Comput. 167 (2024) 112464, https://doi.org/10.1016/j.asoc.2024.112464.
[33] Z. Zhang, Y. Gao, Y. Liu, W. Zuo, A hybrid biogeography-based optimization algorithm to solve high-dimensional optimization problems and real-world engineering problems, Appl. Soft. Comput. 144 (2023) 110514, https://doi.org/10.1016/j.asoc.2023.110514.
[34] H. Wang, D. Zheng, W. Yan, ERPSSA-GBFT: a hybrid algorithm for multi-objective high-dimensional optimization problems in mixed train pantograph-catenary design, Swarm. Evol. Comput. 96 (2025) 101998, https://doi.org/10.1016/j.swevo.2025.101998.
[35] J. Zou, Q. Chang, X. Ou, J. Arinez, G. Xiao, Resilient adaptive control based on renewal particle swarm optimization to improve production system energy efficiency, J. Manuf. Syst. 50 (2019) 135–145, https://doi.org/10.1016/j.jmsy.2018.12.007.
[36] J.H. Holland, Genetic algorithms, Sci. Am. (1992).
[37] J. Kennedy', R. Eberhart, Particle swarm optimization, (n.d.).
[38] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Appl. Soft. Comput. 8 (2008) 687–697, https://doi.org/10.1016/j.asoc.2007.05.007.
[39] W. Zhao, L. Wang, S. Mirjalili, Artificial hummingbird algorithm: a new bio-inspired optimizer with its engineering applications, Comput. Methods Appl. Mech. Eng. 388 (2022) 114194, https://doi.org/10.1016/j.cma.2021.114194.
[40] S. Zhao, T. Zhang, L. Cai, R. Yang, Triangulation topology aggregation optimizer: a novel mathematics-based meta-heuristic algorithm for continuous optimization and engineering applications, Expert. Syst. Appl. 238 (2024) 121744, https://doi.org/10.1016/j.eswa.2023.121744.
[41] L. Wang, Q. Cao, Z. Zhang, S. Mirjalili, W. Zhao, Artificial rabbits optimization: a new bio-inspired meta-heuristic algorithm for solving engineering optimization problems, Eng. Appl. Artif. Intell. 114 (2022) 105082, https://doi.org/10.1016/j.engappai.2022.105082.
[42] K. Cymerys, M. Oszust, Attraction–repulsion optimization algorithm for global optimization problems, Swarm. Evol. Comput. 84 (2024) 101459, https://doi.org/10.1016/j.swevo.2023.101459.
[43] J. Xue, B. Shen, Dung beetle optimizer: a new meta-heuristic algorithm for global optimization, J. Supercomput. 79 (2023) 7305–7336, https://doi.org/10.1007/s11227-022-04959-6.
[44] A. Faramarzi, M. Heidarinejad, B. Stephens, S. Mirjalili, Equilibrium optimizer: a novel optimization algorithm, Knowl. Based. Syst. 191 (2020) 105190, https://doi.org/10.1016/j.knosys.2019.105190.
[45] J.O. Agushaka, A.E. Ezugwu, L. Abualigah, Dwarf mongoose optimization algorithm, Comput. Methods Appl. Mech. Eng. 391 (2022) 114570, https://doi.org/10.1016/j.cma.2022.114570.
[46] T.M. Luan, S. Khatir, M.T. Tran, B. De Baets, T. Cuong-Le, Exponential-trigonometric optimization algorithm for solving complicated engineering problems, Comput. Methods Appl. Mech. Eng. 432 (2024) 117411, https://doi.org/10.1016/j.cma.2024.117411.
[47] W. Zhao, L. Wang, Z. Zhang, H. Fan, J. Zhang, S. Mirjalili, N. Khodadadi, Q. Cao, Electric eel foraging optimization: a new bio-inspired optimizer for engineering applications, Expert. Syst. Appl. 238 (2024) 122200, https://doi.org/10.1016/j.eswa.2023.122200.
[48] A. Qi, D. Zhao, A.A. Heidari, L. Liu, Y. Chen, H. Chen, FATA: an efficient optimization method based on geophysics, Neurocomputing. 607 (2024) 128289, https://doi.org/10.1016/j.neucom.2024.128289.
[49] G. Hu, Y. Guo, G. Wei, L. Abualigah, Genghis Khan shark optimizer: a novel nature-inspired algorithm for engineering optimization, Adv. Eng. Inf. 58 (2023) 102210, https://doi.org/10.1016/j.aei.2023.102210.
[50] Z. Tian, M. Gai, Football team training algorithm: a novel sport-inspired meta-heuristic optimization algorithm for global optimization, Expert. Syst. Appl. 245 (2024) 123088, https://doi.org/10.1016/j.eswa.2023.123088.
[51] N. Chopra, M.Mohsin Ansari, Golden jackal optimization: a novel nature-inspired optimizer for engineering applications, Expert. Syst. Appl. 198 (2022) 116924, https://doi.org/10.1016/j.eswa.2022.116924.
[52] I. Ahmadianfar, O. Bozorg-Haddad, X. Chu, Gradient-based optimizer: a new metaheuristic optimization algorithm, Inf. Sci. (Ny) 540 (2020) 131–159, https://doi.org/10.1016/j.ins.2020.06.037.
[53] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey Wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61, https://doi.org/10.1016/j.advengsoft.2013.12.007.
[54] X. Ma, Z. Zhong, Y. Li, D. Li, Y. Qiao, A novel reinforcement learning based heap-based optimizer, Knowl. Based. Syst. 296 (2024) 111907, https://doi.org/10.1016/j.knosys.2024.111907.
[55] F.A. Hashim, E.H. Houssein, K. Hussain, M.S. Mabrouk, W. Al-Atabany, Honey Badger Algorithm: new metaheuristic algorithm for solving optimization problems, Math. Comput. Simul. 192 (2022) 84–110, https://doi.org/10.1016/j.matcom.2021.08.013.
[56] J. Ji, T. Wu, C. Yang, Neural population dynamics optimization algorithm: a novel brain-inspired meta-heuristic method, Knowl. Based. Syst. 300 (2024) 112194, https://doi.org/10.1016/j.knosys.2024.112194.
[57] M.-Y. Cheng, M.N. Sholeh, RETRACTED: optical microscope algorithm: a new metaheuristic inspired by microscope magnification for solving engineering optimization problems, Knowl. Based. Syst. 279 (2023) 110939, https://doi.org/10.1016/j.knosys.2023.110939.
[58] A. Faramarzi, Marine predators algorithm: a nature-inspired metaheuristic, Expert. Syst. Appl. (2020), https://doi.org/10.1016/j.eswa.2020.113377.
[59] J. Cheng, W. De Waele, Weighted average algorithm: a novel meta-heuristic optimization algorithm based on the weighted average position concept, Knowl. Based. Syst. 305 (2024) 112564, https://doi.org/10.1016/j.knosys.2024.112564.
[60] M. Abdel-Basset, R. Mohamed, M. Jameel, M. Abouhawwash, Nutcracker optimizer: a novel nature-inspired metaheuristic algorithm for global optimization and engineering design problems, Knowl. Based. Syst. 262 (2023) 110248, https://doi.org/10.1016/j.knosys.2022.110248.
[61] S. Mirjalili, A. Lewis, The whale optimization algorithm, Adv. Eng. Softw. 95 (2016) 51–67, https://doi.org/10.1016/j.advengsoft.2016.01.008.
[62] W. Wang, W. Tian, D. Xu, H. Zang, Arctic puffin optimization: a bio-inspired metaheuristic algorithm for solving engineering design optimization, Adv. Eng. Softw. 195 (2024) 103694, https://doi.org/10.1016/j.advengsoft.2024.103694.
[63] E. Fadakar, M. Ebrahimi, A new metaheuristic football game inspired algorithm, in: 2016 1st Conference on Swarm Intelligence and Evolutionary Computation (CSIEC), IEEE, Bam, Iran, 2016, pp. 6–11, https://doi.org/10.1109/CSIEC.2016.7482120.
[64] X. Wang, V. Snášel, S. Mirjalili, J.-S. Pan, L. Kong, H.A. Shehadeh, Artificial protozoa optimizer (APO): a novel bio-inspired metaheuristic algorithm for engineering optimization, Knowl. Based. Syst. 295 (2024) 111737, https://doi.org/10.1016/j.knosys.2024.111737.
[65] M. Kumar, A.J. Kulkarni, S.C. Satapathy, Socio evolution & learning optimization algorithm: a socio-inspired optimization methodology, Fut. Gener. Comp. Syst. 81 (2018) 252–272, https://doi.org/10.1016/j.future.2017.10.052.
[66] J. Bai, H. Nguyen-Xuan, E. Atroshchenko, G. Kosec, L. Wang, M.Abdel Wahab, Blood-sucking leech optimizer, Adv. Eng. Softw. 195 (2024) 103696, https://doi.org/10.1016/j.advengsoft.2024.103696.
[67] R.V. Rao, V.J. Savsani, D.P. Vakharia, Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems, Comput.-Aided Des. 43 (2011) 303–315, https://doi.org/10.1016/j.cad.2010.12.015.
[68] M. Abdel-Basset, R. Mohamed, M. Abouhawwash, Crested porcupine optimizer: a new nature-inspired metaheuristic, Knowl. Based. Syst. 284 (2024) 111257, https://doi.org/10.1016/j.knosys.2023.111257.
[69] Z. Chen, S. Li, A.T. Khan, S. Mirjalili, Competition of tribes and cooperation of members algorithm: an evolutionary computation approach for model free optimization, Expert. Syst. Appl. 265 (2025) 125908, https://doi.org/10.1016/j.eswa.2024.125908.

[70] M. Ghasemi, M. Zare, P. Trojovský, R.V. Rao, E. Trojovská, V. Kandasamy, Optimization based on the smart behavior of plants with its engineering applications: Ivy algorithm, Knowl. Based. Syst. 295 (2024) 111850, https://doi.org/10.1016/j.knosys.2024.111850.
[71] K. Ouyang, S. Fu, Y. Chen, Q. Cai, A.A. Heidari, H. Chen, Escape: an optimization method based on crowd evacuation behaviors, Artif. Intell. Rev. 58 (2024) 19, https://doi.org/10.1007/s10462-024-11008-6.
[72] C. Zhang, H. Li, S. Long, X. Yue, H. Ouyang, Z. Chen, S. Li, Piranha predation optimization algorithm (PPOA) for global optimization and engineering design problems, Appl. Soft. Comput. 165 (2024) 112085, https://doi.org/10.1016/j.asoc.2024.112085.
[73] S.O. Oladejo, S.O. Ekwe, S. Mirjalili, The hiking optimization algorithm: a novel human-based metaheuristic approach, Knowl. Based. Syst. 296 (2024) 111880, https://doi.org/10.1016/j.knosys.2024.111880.
[74] S. Fu, K. Li, H. Huang, C. Ma, Q. Fan, Y. Zhu, Red-billed blue magpie optimizer: a novel metaheuristic algorithm for 2D/3D UAV path planning and engineering design problems, Artif. Intell. Rev. 57 (2024) 134, https://doi.org/10.1007/s10462-024-10716-3.
[75] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Computat. 1 (1997) 67–82, https://doi.org/10.1109/4235.585893.
[76] C. Cheng, Q. Sha, B. He, G. Li, Path planning and obstacle avoidance for AUV: a review, Ocean Eng. 235 (2021) 109355, https://doi.org/10.1016/j.oceaneng.2021.109355.
[77] A. Refaat, A. Elbaz, A.-E. Khalifa, M.M. Elsakka, A. Kalas, M.H. Elfar, Performance evaluation of a novel self-tuning particle swarm optimization algorithm-based maximum power point tracker for porton exchange membrane fuel cells under different operating conditions, Energy Convers. Manag. 301 (2024) 118014, https://doi.org/10.1016/j.enconman.2023.118014.
[78] J. Lehman, K.O. Stanley, Abandoning objectives: evolution through the search for novelty alone, Evol. Comput. 19 (2011) 189–223, https://doi.org/10.1162/EVCO_a_00025.
[79] X. Xia, H. Song, Y. Zhang, L. Gui, X. Xu, K. Li, Y. Li, A particle swarm optimization with adaptive learning weights tuned by a multiple-input multiple-output fuzzy logic controller, IEEe Trans. Fuzzy. Syst. 31 (2023) 2464–2478, https://doi.org/10.1109/TFUZZ.2022.3227464.
[80] Q. Wu, R. Law, Cauchy mutation based on objective variable of Gaussian particle swarm optimization for parameters selection of SVM, Expert. Syst. Appl. 38 (2011) 6405–6411, https://doi.org/10.1016/j.eswa.2010.08.069.
[81] Q. Wu, Hybrid forecasting model based on support vector machine and particle swarm optimization with adaptive and cauchy mutation, Expert. Syst. Appl. 38 (2011) 9070–9075, https://doi.org/10.1016/j.eswa.2010.11.093.
[82] X. Zhang, X. Wang, Q. Kang, J. Cheng, Differential mutation and novel social learning particle swarm optimization algorithm, Inf. Sci. 480 (2019) 109–129, https://doi.org/10.1016/j.ins.2018.12.030.
[83] X. Long, W. Cai, L. Yang, H. Huang, Improved particle swarm optimization with reverse learning and neighbor adjustment for space surveillance network task scheduling, Swarm. Evol. Comput. 85 (2024) 101482, https://doi.org/10.1016/j.swevo.2024.101482.
[84] H. Haklı, H. Uğuz, A novel particle swarm optimization algorithm with Levy flight, Appl. Soft. Comput. 23 (2014) 333–345, https://doi.org/10.1016/j.asoc.2014.06.034.
[85] M. Song, H. Jia, L. Abualigah, Q. Liu, Z. Lin, D. Wu, M. Altalhi, Modified Harris Hawks optimization algorithm with exploration factor and random walk strategy, Comput. Intell. Neurosci. 2022 (2022) 1–23, https://doi.org/10.1155/2022/4673665.
[86] W. Li, J. Jing, Y. Chen, Y. Chen, A cooperative particle swarm optimization with difference learning, Inf. Sci. 643 (2023) 119238, https://doi.org/10.1016/j.ins.2023.119238.
[87] X. Yang, H. Li, Evolutionary-state-driven multi-swarm cooperation particle swarm optimization for complex optimization problem, Inf. Sci. 646 (2023) 119302, https://doi.org/10.1016/j.ins.2023.119302.
[88] C. Huang, X. Zhou, X. Ran, J. Wang, H. Chen, W. Deng, Adaptive cylinder vector particle swarm optimization with differential evolution for UAV path planning, Eng. Appl. Artif. Intell. 121 (2023) 105942, https://doi.org/10.1016/j.engappai.2023.105942.
[89] A. Madani, A. Engelbrecht, B. Ombuki-Berman, Cooperative coevolutionary multi-guide particle swarm optimization algorithm for large-scale multi-objective optimization problems, Swarm. Evol. Comput. 78 (2023) 101262, https://doi.org/10.1016/j.swevo.2023.101262.
[90] M.A. Elaziz, D. Yousri, A.O. Aseeri, L. Abualigah, M.A.A. Al-qaness, A.A. Ewees, Fractional-order modified heterogeneous comprehensive learning particle swarm optimizer for intelligent disease detection in IoMT environment, Swarm. Evol. Comput. 84 (2024) 101430, https://doi.org/10.1016/j.swevo.2023.101430.
[91] B. Han, B. Li, C. Qin, A novel hybrid particle swarm optimization with marine predators, Swarm. Evol. Comput. 83 (2023) 101375, https://doi.org/10.1016/j.swevo.2023.101375.
[92] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEe Trans. Evol. Comput. 10 (2006) 281–295, https://doi.org/10.1109/TEVC.2005.857610.
[93] Hao Wu, Junping Geng, Ronghong Jin, Jizheng Qiu, Wei Liu, Jing Chen, Suna Liu, An improved comprehensive learning particle swarm optimization and its application to the semiautomatic design of antennas, IEEe Trans. Antennas. Propag. 57 (2009) 3018–3028, https://doi.org/10.1109/TAP.2009.2028608.
[94] S. Xu, G. Xiong, A.W. Mohamed, H.R.E.H. Bouchekara, Forgetting velocity based improved comprehensive learning particle swarm optimization for non-convex economic dispatch problems with valve-point effects and multi-fuel options, Energy 256 (2022) 124511, https://doi.org/10.1016/j.energy.2022.124511.
[95] C. Hu, S. Zeng, C. Li, Scalable GP with hyperparameters sharing based on transfer learning for solving expensive optimization problems, Appl. Soft. Comput. 148 (2023) 110866, https://doi.org/10.1016/j.asoc.2023.110866.
[96] J. Shen, P. Wang, H. Dong, W. Wang, J. Li, Surrogate-assisted evolutionary algorithm with decomposition-based local learning for high-dimensional multi-objective optimization, Expert. Syst. Appl. 240 (2024) 122575, https://doi.org/10.1016/j.eswa.2023.122575.
[97] F. Li, Y. Li, X. Cai, L. Gao, A surrogate-assisted hybrid swarm optimization algorithm for high-dimensional computationally expensive problems, Swarm. Evol. Comput. 72 (2022) 101096, https://doi.org/10.1016/j.swevo.2022.101096.
[98] F. Yu, W. Gong, H. Zhen, A data-driven evolutionary algorithm with multi-evolutionary sampling strategy for expensive optimization, Knowl. Based. Syst. 242 (2022) 108436, https://doi.org/10.1016/j.knosys.2022.108436.
[99] X. Cai, H. Qiu, L. Gao, C. Jiang, X. Shao, An efficient surrogate-assisted particle swarm optimization algorithm for high-dimensional expensive problems, Knowl. Based. Syst. 184 (2019) 104901, https://doi.org/10.1016/j.knosys.2019.104901.
[100] S. Gupta, K. Deep, S. Mirjalili, J.H. Kim, A modified Sine Cosine algorithm with novel transition parameter and mutation operator for global optimization, Expert. Syst. Appl. 154 (2020) 113395, https://doi.org/10.1016/j.eswa.2020.113395.
[101] H. Jia, K. Sun, W. Zhang, X. Leng, An enhanced chimp optimization algorithm for continuous optimization domains, Complex Intell. Syst. 8 (2022) 65–82, https://doi.org/10.1007/s40747-021-00346-5.
[102] Yu Li, Y. Zhao, J. Liu, Dimension by dimension dynamic sine cosine algorithm for global optimization problems, Appl. Soft. Comput. 98 (2021) 106933, https://doi.org/10.1016/j.asoc.2020.106933.
[103] M. Han, Z. Du, H. Zhu, Y. Li, Q. Yuan, H. Zhu, Golden-sine dynamic marine predator algorithm for addressing engineering design optimization, Expert. Syst. Appl. 210 (2022) 118460, https://doi.org/10.1016/j.eswa.2022.118460.
[104] W. Long, J. Jiao, M. Xu, M. Tang, T. Wu, S. Cai, Lens-imaging learning Harris hawks optimizer for global optimization and its application to feature selection, Expert. Syst. Appl. 202 (2022) 117255, https://doi.org/10.1016/j.eswa.2022.117255.
[105] H. Chen, A.A. Heidari, H. Chen, M. Wang, Z. Pan, A.H. Gandomi, Multi-population differential evolution-assisted Harris hawks optimization: framework and case studies, Fut. Gener. Comput. Syst. 111 (2020) 175–198, https://doi.org/10.1016/j.future.2020.04.008.
[106] Y. Fan, P. Wang, A.A. Heidari, M. Wang, X. Zhao, H. Chen, C. Li, Boosted hunting-based fruit fly optimization and advances in real-world problems, Expert. Syst. Appl. 159 (2020) 113502, https://doi.org/10.1016/j.eswa.2020.113502.
[107] J. Xue, B. Shen, A novel swarm intelligence optimization approach: sparrow search algorithm, Syst. Sci. Control Eng. 8 (2020) 22–34, https://doi.org/10.1080/21642583.2019.1708830.

[108] A.W. Mohamed, A.A. Hadi, A.M. Fattouh, K.M. Jambi, LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems, in: 2017 IEEE Congress on Evolutionary Computation (CEC), Donostia, San Sebastián, Spain,, IEEE, 2017, pp. 145–152, https://doi.org/10.1109/CEC.2017.7969307.
[109] N.H. Awad, M.Z. Ali, P.N. Suganthan, Ensemble sinusoidal differential covariance matrix adaptation with euclidean neighborhood for solving CEC2017 benchmark problems, in: 2017 IEEE Congress on Evolutionary Computation (CEC), Donostia, San Sebastián, Spain, IEEE, 2017, pp. 372–379, https://doi.org/10.1109/CEC.2017.7969336.
[110] A.W. Mohamed, A.A. Hadi, A.K. Mohamed, N.H. Awad, Evaluating the performance of adaptive GainingSharing knowledge based algorithm on CEC 2020 benchmark problems, in: 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, United Kingdom, IEEE, 2020, pp. 1–8, https://doi.org/10.1109/CEC48606.2020.9185901.
[111] S. Biswas, D. Saha, S. De, A.D. Cobb, S. Das, B.A. Jalaian, Improving differential evolution through bayesian hyperparameter optimization, in: 2021 IEEE Congress on Evolutionary Computation (CEC), Kraków, Poland, IEEE, 2021, pp. 832–840, https://doi.org/10.1109/CEC45853.2021.9504792.
[112] X. Xia, L. Gui, G. He, B. Wei, Y. Zhang, F. Yu, H. Wu, Z.-H. Zhan, An expanded particle swarm optimization based on multi-exemplar and forgetting ability, Inf. Sci. 508 (2020) 105–120, https://doi.org/10.1016/j.ins.2019.08.065.
[113] X. Xia, L. Gui, F. Yu, H. Wu, B. Wei, Y.-L. Zhang, Z.-H. Zhan, Triple archives particle swarm optimization, IEEe Trans. Cybern. 50 (2020) 4862–4875, https://doi.org/10.1109/TCYB.2019.2943928.
[114] Y. Zhang, Elite archives-driven particle swarm optimization for large scale numerical optimization and its engineering applications, Swarm. Evol. Comput. 76 (2023) 101212, https://doi.org/10.1016/j.swevo.2022.101212.
[115] M.R. Tanweer, S. Suresh, N. Sundararajan, Self regulating particle swarm optimization algorithm, Inf. Sci. 294 (2015) 182–202, https://doi.org/10.1016/j.ins.2014.09.053.
[116] M. Qaraad, S. Amjad, N.K. Hussein, M.A. Farag, S. Mirjalili, M.A. Elhosseini, Quadratic interpolation and a new local search approach to improve particle swarm optimization: solar photovoltaic parameter estimation, Expert. Syst. Appl. 236 (2024) 121417, https://doi.org/10.1016/j.eswa.2023.121417.
[117] D.Chauhan Shivani, D. Rani, A feasibility restoration particle swarm optimizer with chaotic maps for two-stage fixed-charge transportation problems, Swarm. Evol. Comput. 91 (2024) 101776, https://doi.org/10.1016/j.swevo.2024.101776.
[118] H. Jia, C. Lu, Guided learning strategy: a novel update mechanism for metaheuristic algorithms design and improvement, Knowl. Based. Syst. 286 (2024) 111402, https://doi.org/10.1016/j.knosys.2024.111402.
[119] E.H. Houssein, N. Abdalkarim, N.A. Samee, M. Alabdulhafith, E. Mohamed, Improved kepler optimization algorithm for enhanced feature selection in liver disease classification, Knowl. Based. Syst. 297 (2024) 111960, https://doi.org/10.1016/j.knosys.2024.111960.
[120] R. Sowmya, M. Premkumar, P. Jangir, Newton-Raphson-based optimizer: a new population-based metaheuristic algorithm for continuous optimization problems, Eng. Appl. Artif. Intell. 128 (2024) 107532, https://doi.org/10.1016/j.engappai.2023.107532.
[121] N.H. Awad, M.Z. Ali, P.N. Suganthan, J.J. Liang, B.Y. Qu, Problem definitions and evaluation criteria for the CEC 2017 Special session and competition on single objective real-parameter numerical optimization, (n.d.).
[122] C.A. Coello Coello, Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art, Comput. Methods Appl. Mech. Eng. 191 (2002) 1245–1287, https://doi.org/10.1016/S0045-7825(01)00323-1.
[123] A. Kumar, G. Wu, M.Z. Ali, R. Mallipeddi, P.N. Suganthan, S. Das, A test-suite of non-convex constrained optimization problems from the real-world and some baseline results, Swarm. Evol. Comput. 56 (2020) 100693, https://doi.org/10.1016/j.swevo.2020.100693.
[124] R. Akay, M.Y. Yildirim, Multi-strategy and self-adaptive differential sine–cosine algorithm for multi-robot path planning, Expert. Syst. Appl. 232 (2023) 120849, https://doi.org/10.1016/j.eswa.2023.120849.
[125] Y. Li, L. Zhao, Y. Wang, Q. Wen, Improved sand cat swarm optimization algorithm for enhancing coverage of wireless sensor networks, Measurement 233 (2024) 114649, https://doi.org/10.1016/j.measurement.2024.114649.
[126] J. Wang, Y. Liu, S. Rao, X. Zhou, J. Hu, A novel self-adaptive multi-strategy artificial bee colony algorithm for coverage optimization in wireless sensor networks, Ad. Hoc. Netw. 150 (2023) 103284, https://doi.org/10.1016/j.adhoc.2023.103284.
[127] Y. Wang, H. Liu, H. Long, Z. Zhang, S. Yang, Differential evolution with a new encoding mechanism for optimizing wind farm layout, (n.d.).
[128] Z. Zhang, Y. Fu, K. Gao, Q. Pan, M. Huang, A learning-driven multi-objective cooperative artificial bee colony algorithm for distributed flexible job shop scheduling problems with preventive maintenance and transportation operations, Comput. Ind. Eng. 196 (2024) 110484, https://doi.org/10.1016/j.cie.2024.110484.
[129] L. Zhao, W. Cheng, L. Meng, C. Zhang, Y. Ren, B. Zhang, P. Duan, MILP modeling and optimization of flexible job shop scheduling problem with preventive maintenance, Comput. Ind. Eng. 201 (2025) 110861, https://doi.org/10.1016/j.cie.2025.110861.
[130] J. Guo, B. Peng, B. Du, K. Wang, Y. Li, Q-learning-based multi-objective spotted hyena algorithm for flexible open shop scheduling problem with consideration of preventive maintenance and travel/setup times, J. Manuf. Syst. 82 (2025) 224–238, https://doi.org/10.1016/j.jmsy.2025.06.001.