Electronic Research Archive

https://www.aimspress.com/journal/era

ERA, 33(3): 1693–1762.
DOI: 10.3934/era.2025079
Received: 18 December 2024
Revised: 19 February 2025
Accepted: 10 March 2025
Published: 25 March 2025

Research article

An improved dung beetle optimizer based on Padé approximation strategy for global optimization and feature selection

Tianbao Liu*, **Lingling Yang, Yue Li and Xiwen Qin**

School of Mathematics and Statistics, Changchun University of Technology, Changchun 130012, China

* **Correspondence:** Email: liutianbao@ccut.edu.cn.

Abstract: Feature selection is a crucial data processing method used to reduce dataset dimensionality while preserving key information. In this paper, we proposed a multi-strategy enhanced dung beetle optimization algorithm (mDBO) that integrates multiple strategies to effectively address the feature selection problem. First, a novel population initialization strategy based on a hybrid tent-sine map and random opposition-based learning was proposed to generate initial population. This strategy yielded a more uniform distribution of the initial population, significantly improving the quality of the population distribution within the search space. Second, a new differential evolution mutation strategy with a periodic retrospective adaptive mutation factor was proposed. This strategy effectively improved the algorithm's ability to jump out of the local optimal and explore potential candidate solutions. Third, based on Padé approximation technology and the novel adaptive evolutionary boundary constraint method, an innovative approximation strategy was proposed. The strategy was integrated into the framework of the dung beetle optimizer, significantly improving the solution accuracy and population quality of the algorithm. Finally, the binary version of the mDBO algorithm (bmDBO) was applied to feature selection tasks. Experiments entailing CEC2017 benchmark functions and 17 datasets showed that both mDBO and bmDBO outperformed other algorithms. The mDBO method outperformed other algorithms in 11 of the 29 benchmark functions, ranked second in 8 functions, and achieved an average rank of 1.62 in the Friedman ranking, securing the overall first place; the bmDBO method outperformed in 12 of 17 datasets, achieving an average ranking of 1.35 in the Friedman ranking, securing the first position.

Keywords: feature selection; dung beetle optimization algorithm; differential evolution; Padé approximation; adaptive evolutionary boundary constraint handling

1. Introduction

Optimization technology has proven to be of immense practical value in the daily operations of both industry and the scientific community. Over the past years, a plethora of innovative numerical and computational methods have been developed to tackle optimization problems. The effectiveness of an optimizer is largely determined by the search capability of its algorithm, which directly influences its ability to accurately identify the optimal solution from numerous possibilities [1]. However, practical optimization problems often present challenges such as high dimensionality, nonlinearity, and intricate parametric constraints. Consequently, traditional methods, such as Newton's method and integer programming, while theoretically capable of solving optimal solutions for small-scale problems, become impractical for complex real-world scenarios due to their considerable time and space requirements.

Optimization fundamentally involves identifying the optimal value of an objective function under various constraints [2] and is extensively applied across fields such as mechanical design [3], path planning [4], image segmentation [5], and deep learning [6]. For example, Li et al. [7] improved the artificial rabbit optimization algorithm and applied it to multichannel signal denoising. Xiao et al. [8] enhanced the snow melt optimization algorithm with multiple strategies and applied it to both global optimization and practical engineering applications. Fei et al. [9] proposed a novelprojective model (PM) basedsparse MEKLM (PM-SMEKLM) algorithm tolearna cross-domain transformation by PM inway of theparameter-based TL, and then apply it to the neuroimaging-based CAD for brain diseases. Ma and Li [10] introduced a U-Net algorithm with a supervised attention mechanism for retinal vessel segmentation. Xiao and Guo [11] proposed an improved hybrid optimizer combining AO and AVOA to overcome the limitations of individual algorithms, providing higher-quality solutions to global optimization problems. These practical optimization problems frequently exhibit characteristics, including complexity, high dimensionality, non-convexity, and nonlinearity [12]. Additionally, with the rapid advancements in data collection and storage technologies, vast amounts of data have been accumulated across fields. However, these datasets often contain significant amounts of redundant and irrelevant information. Thus, effectively removing noise from these large datasets and selecting optimal feature combinations has emerged as a crucial research topic in the era of big data. Given the multitude of features present in datasets, it is neither feasible nor time-efficient to explore all possible feature combinations exhaustively. As a result, feature selection algorithms are widely utilized to address this NP-hard problem [13].

Feature selection (FS) is a process aimed at identifying and retaining key features from an original dataset while eliminating unnecessary, redundant, or noisy attributes based on predefined evaluation criteria. This approach facilitates dimensionality reduction without compromising the integrity of the original features, distinguishing it from feature extraction, which constructs new combinations of features. FS is widely utilized in various fields, including text mining. The techniques for FS are primarily categorized into three groups: filter, wrapper, and embedded methods.

The filter method assesses the significance of features based on their correlation with the target variable. While this approach is straightforward and easy to implement, it may sometimes lack in terms of classification accuracy. Common filter techniques include variance analysis, information gain calculation, and correlation analysis [14]. In contrast, the wrapper method employs machine learning models to iteratively evaluate subsets of features, seeking an optimal solution or meeting specific stopping criteria. This technique can enhance classification accuracy but typically incurs a higher computational cost, especially when applied in conjunction with models such as K-nearest neighbors (KNN) [15], support vector machines (SVM) [16], neural networks, or decision trees [17]. The embedded method integrates the feature selection process directly into the model training phase, thereby optimizing time efficiency, although it may increase the computational burden on the model. It is important to recognize that the effectiveness of various FS methods can vary across different datasets and classification tasks. Therefore, a flexible application of multiple FS techniques can maximize their respective advantages and significantly improve classification performance.

The rapid advancement of meta-heuristic algorithms [18] has significantly expanded the application scope of intelligent optimization techniques in the field of FS. These algorithms, inspired by natural phenomena and biological behavior patterns, have demonstrated remarkable effectiveness across various optimization domains due to their straightforward model construction, ease of implementation, and robust performance. However, the no free lunch (NFL) theorem [19] serves as a reminder that no single algorithm is universally applicable to all optimization problems. This insight drives researchers to continuously pursue algorithmic innovations to enhance the applicability of these methods. In the FS domain, researchers have actively sought enhancements to address the limitations of traditional algorithms, such as particle swarm optimization (PSO), particularly when dealing with high-dimensional datasets. For instance, Song et al. [20] ingeniously integrated the principle of mutual information into their bare-bones particle swarm optimization (BBPSO), significantly improving classification performance when utilized alongside the KNN classifier. Similarly, Faris et al. [21] developed the binary salp swarm algorithm (SSA), which enhances exploration capabilities by introducing crossover operators and transfer functions. Emary et al. [22] presented a novel binary grey wolf optimizer (GWO) method, achieving the binarization of continuous algorithms through the application of an S-shaped transfer function, which was validated on multiple datasets, thereby revitalizing the GWO approach. Zhao et al. [23] proposed a binary dandelion algorithm with improved seeds and a chaotic population. Experimental results indicate that, in most cases, the SBDA selects fewer features while achieving higher classification accuracy. Mafarja et al. [24] made strides in this area by proposing two improved whale optimization algorithms (WOAs). Their validation on UCI datasets, coupled with comparisons to other algorithms, demonstrated superior classification performance. Researchers have also developed a plethora of hybrid optimization algorithms. For example, Adamu et al. [25] introduced the enhanced chaotic crow search and particle swarm optimization-based hybrid binary algorithm, which effectively integrates particle swarm optimization with the crow search algorithm, utilizing opposition-based learning techniques. This approach significantly expands the search space and enhances optimization efficiency through various chaotic initialization methods, offering more diverse options for the FS field.

These innovative algorithms not only address the unique challenges of FS but also enhance classification accuracy and optimization speed by incorporating novel strategies and technologies. Notably, intelligent optimization algorithms have proven adept at solving combinatorial optimization problems [26]. For instance, Alic and Subasi [27] successfully utilized genetic algorithms (GAs) to extract key informative features for breast cancer diagnosis in 2017, leading to substantial improvements in classification accuracy through the rotation forest model based on GAs. Despite challenges such as insufficient accuracy and premature convergence associated with traditional optimization algorithms, intelligent optimization methods employing enhanced strategies have emerged as a dominant force in the FS domain. Furthermore, significant progress has been made in multi-objective feature selection methods, albeit often at higher computational costs. Overall, FS methodologies based on evolutionary computation (EC) have garnered considerable attention [28] due to their superior optimization capabilities. Various EC techniques, including particle swarm optimization, binary arithmetic optimization [29], ant colony optimization [30], binary grasshopper optimization [31], binary waterwheel plant optimization [32], and genetic programming [33], have been extensively applied in FS and are closely integrated with filter, wrapper, and hybrid methods, collectively driving the innovation and development of FS techniques.

Existing optimization algorithms also have certain limitations in the feature selection problem. For instance, some algorithms may encounter high computational complexity when handling large-scale datasets, leading to low efficiency. Additionally, many optimization algorithms are prone to getting stuck in local optima, preventing them from finding the global optimum. During the feature selection process, some algorithms may tend to select too many features, resulting in good performance on the training set but overfitting on the test set. Furthermore, certain algorithms may have slow convergence rates on high-dimensional datasets, making it difficult to find a satisfactory solution quickly.

The dung beetle optimization (DBO) [34] algorithm is an emerging meta-heuristic strategy that has demonstrated significant optimization potential when applied to benchmark functions. However, it faces several challenges, including local convergence and an imbalance between exploration and exploitation capabilities. To address these issues, we propose an enhanced DBO framework that integrates multiple strategies to improve its performance in both global optimization problems and feature selection tasks. First, a novel population initialization strategy is introduced, combining tent-sine mapping with random opposition-based learning to achieve a more uniform distribution of the initial population. Second, a new differential evolution mutation strategy is proposed, incorporating a periodic retrospective adaptive mutation factor to enhance the exploration of potential solutions. Furthermore, an innovative approximation strategy is presented, leveraging Padé approximation technology along with a novel adaptive evolutionary boundary constraint method, aimed at optimizing population position updates and improving the accuracy of the algorithm's solutions. Finally, the binary version of the enhanced mDBO algorithm is applied to feature selection tasks. The major contributions of this study are as follows:

(1) A novel population initialization strategy utilizing a hybrid tent-sine mapping combined with Random Opposition-Based Learning (ROBL) is proposed to generate the initial population. This approach promotes a more uniform distribution of the initial population, thereby enhancing the overall quality of the population distribution within the search space.

(2) A novel differential evolution mutation strategy is proposed, which integrates a periodic retrospective adaptive mutation factor. This strategy enhances the traditional foraging update process by incorporating a dynamic differential evolution mechanism, which effectively improves the algorithm's ability to escape local optima and broadens the search space for potential candidate solutions.

(3) An innovative approximation strategy is proposed that incorporates Padé approximation techniques along with an adaptive evolutionary boundary constraint method. This strategy is integrated into the framework of the dung beetle optimizer, leading to enhancements in solution accuracy and population quality.

(4) A binary version of the mDBO (bmDBO) is introduced, which utilizes an S-shaped transfer

function to transform the continuous solution space into discrete binary results.

(5) A comprehensive experimental design was employed, covering 29 standard mathematical test functions and 17 classical datasets, to evaluate the robustness and efficiency of the proposed algorithm in solving real-world problems.

The structure of this paper is organized as follows: In Section 2, we present the DBO model. In Section 3, we introduce a hybrid version of DBO, named mDBO, and details its population initialization strategy based on a hybrid tent-sine map and random opposition-based learning, the periodic retrospective adaptive evolution strategy, and the Padé approximation strategy with an adaptive evolutionary boundary constraint. In Section 4, we present numerical experiments and analyze the results of mDBO in comparison with other optimization algorithms. In Section 5, mDBO is applied to address the feature selection problem, demonstrating its competitive performance. Finally, in Section 6, we conclude the paper.

2. Dung beetle optimizer

The DBO algorithm is a novel swarm intelligence optimization algorithm proposed by Xue and Shen [34], inspierd by the natural behavior of dung beetles. The DBO algorithm seeks the optimal solution by simulating the habits of dung beetles, such as rolling balls, dancing, reproducing, foraging, and stealing. The basic dung beetle algorithm updates the population positions by simulating four behaviors of dung beetles in nature (rolling, spawning, foraging, and stealing).

2.1. Rolling dung beetle

The rolling behavior of dung beetles is categorized into two modes: Obstacle-free and obstacle mode. When a dung beetle advances without encountering any obstacles, it relies on celestial cues to maintain a straight-line trajectory for the dung ball during its rolling. In this mode, when the dung beetle rolls the ball, its position is updated in the manner described by Eq (2.1):

$$X_i^{t+1} = X_i^t + \lambda \cdot k \cdot X_i^{t-1} + b \cdot |X_i^t - X_{worst}^t|, \tag{2.1}$$

t represents the current iteration number, X_i^t represent the position of the dung beetle at the tth iteration, $k \in (0, 0.2]$ represents a constant value which indicates the deflection coefficient, b is a constant value from the interval $[0, 1]$, λ indicates whether the dung beetle has deviated from its original direction, assigned as either -1 or 1, where 1 indicates no deviation, -1 indicates deviation from the original direction, and X_{worst}^t represents the worst in the current population, and $|X_i^t - X_{worst}^t|$ is used to simulate the changes in light intensity.

In the natural world, when a dung beetle encounters an obstacle on its path, it will obtain a new rolling direction through the behavior of dancing. To simulate this situation, the algorithm uses a tangent function to simulate the dung beetle's dance behavior to obtain a new direction of advance. It should be noted that we only need to consider the values of the tangent function on the interval $[0,\pi]$. Once the dung beetle successfully determines a new direction, it should continue to roll the ball backward. The update of the dung beetle's position becomes Eq (2.2).

$$X_i^{t+1} = \begin{cases} X_i^t + \tan(\theta) \cdot |X_i^t - X_i^{t-1}| & ,\theta \neq 0,\pi,\frac{\pi}{2}, \\ X_i^t & ,\theta = 0,\pi,\frac{\pi}{2}. \end{cases} \tag{2.2}$$

2.2. Spawning dung beetle

In nature, female dung beetles consider the living environment for their larvae and roll their dung balls to a safe and suitable place for the larvae to survive. They also reproduce offspring through the egg balls laid. The activity range of dung beetles laying eggs is strictly limited to the safe area to achieve local development of this area. Therefore, DBO has proposed a boundary restriction strategy to imitate the area where female dung beetles lay eggs, which is defned by:

$$\begin{aligned} Lb^* &= max(X^* \cdot (1 - R), Lb), \\ Ub^* &= min(X^* \cdot (1 + R), Ub), \end{aligned} \tag{2.3}$$

Lb^* and Ub^* represent the lower and upper bounds of the egg-laying area, respectively. X^* is the current local optimum, and $R = 1 - t/T_{max}$, where T_{max} denotes the maximum number of iterations. Lb and Ub represent the lower and upper bounds of the optimization problem. Once the dung beetle has determined the optimal spawning area, it selects the dung balls to lay eggs within this area. In the original text, each spawning by a dung beetle is an update to the beetle's position. The boundary range of the spawning area is dynamically changing, mainly determined by the R value. This ensures the search in the area where the current optimum is located while preventing falling into a local optimum area. The position update of the spawning dung beetle is given by Eq (2.4).

$$X_i^{t+1} = X^* + b_1 \cdot (X_i^t - Lb^*) + b_2 \cdot (X_i^t - Ub^*), \tag{2.4}$$

where X_i^{t+1} represents the position of the ith larva at the tth iteration. b_1 and b_2 denote two independent random vectors, which are $1 \times D$ random variables, where D signifies the dimension of the optimization problem.

2.3. Foraging dung beetle

The foraging behavior of dung beetles primarily originates from the larvae. As the dung beetle larvae develop into adults, they emerge from the ground to search for food. The foraging area for the larvae is also dynamically updated, which can be mathematically represented as follows:

$$\begin{aligned} Lb^b &= max(X^b \cdot (1 - R), Lb), \\ Ub^b &= min(X^b \cdot (1 + R), Ub), \end{aligned} \tag{2.5}$$

where X^b represents the global best position of the population. Lb^b and Ub^b define the lower and upper bounds of the optimal foraging area for the dung beetle larvae, respectively. Therefore, the position update for the dung beetle larvae can be described as follows:

$$X_i^{t+1} = X_i^t + C_1 \cdot (X_i^t - Lb^b) + C_2 \cdot (X_i^t - Ub^b), \tag{2.6}$$

C_1 is a random number that follows a normal distribution, and C_2 is a random vector that belongs to the interval (0,1).

2.4. Stealing dung beetle

In the natural world, within dung beetle populations, some dung beetles engage in the behavior of stealing dung balls from others. To simulate this behavior, the original text uses the optimal global position X^b as the location of the contested dung ball. During the iterative process, the position information of the stealing dung beetle is updated, with the position update illustrated by Eq (2.7).

$$X_i^{t+1} = X^b + Q \cdot g \cdot (|X_i^t - X^*| + |X_i^t - X^b|), \tag{2.7}$$

g represents a random vector of size $1 \times D$, which is normally distributed, Q is a constant. Algorithm1 gives the pseudo code of DBO.

Algorithm 1 The framework of the DBO algorithm

```
Require: The maximum iteration $T_{max}$, the size of the particle's population $N$
Ensure: Optimal position $X_b$ and its fitness value $f_b$
 1: Initialize the particle's population $i \leftarrow 1, 2, \cdots, N$ and define its relevant parameters
 2: while $t \leq T_{max}$ do
 3:     for $i = 1$ to $N$ of Rolling dung beetle do
 4:         $\eta$=rand(1)
 5:         if $\alpha \leq 0.9$ then
 6:             Update the position of the Rolling dung beetle using Eq (2.1)
 7:         else
 8:             Update the position of the Rolling dung beetle using Eq (2.2)
 9:         end if
10:     end for
11:     The value of the nonlinear convergence factor is calculated by $R = 1 - t/T_{max}$
12:     for $i = 1$ to $N$ of Spawning dung beetle do
13:         Update the position of the Spawning dung beetle using Eq (2.3) and Eq (2.4)
14:     end for
15:     for $i = 1$ to $N$ of Foraging dung beetle do
16:         Update the position of the Spawning dung beetle using Eq (2.5) and Eq (2.6)
17:     end for
18:     for $i = 1$ to $N$ of Stealing dung beetle do
19:         Update the position of the Rolling dung beetle using Eq (2.7)
20:     end for
21: end while
22: Return $X_b$ and its fitness value $f_b$
```

3. The proposed mDBO algorithm

Although the DBO algorithm demonstrates rapid convergence speed and wide application in solving optimization problems, surpassing many other intelligent optimization algorithms, it is prone to challenges, including uneven distribution of the initial population, limited global search capacity, and a tendency to become trapped in local optima. To better address these challenges, we propose a series of improvement strategies.

3.1. Population initialization strategy based on hybrid tent-sine map and random opposition-based learning

In swarm intelligence algorithms, population initialization plays a critical role in determining both the convergence rate and the quality of the final solution. In the absence of prior knowledge about potential solutions, random initialization is commonly used to generate the initial population. However, random initialization often results in population agents becoming trapped in local optima. Chaotic maps, which generate chaotic sequences, offer distinct advantages over traditional random initialization due to their excellent ergodicity and high randomness, facilitating the generation of more diverse populations. Researchers [35, 36] have demonstrated that effective population initialization can accelerate convergence and improve system performance. In the experimental Section 4.2.1, population initialization plots based on common chaotic mappings, along with related analyses, are provided. In some chaotic maps, the tent map has gained widespread attention due to its rich chaotic properties and has been effectively applied to the population initialization stage of algorithms to enhance their performance [37–39]. The definition of the tent map is given by Eq (3.1).

$$P_{i+1} = \begin{cases} B_k P_i & , P_i < 0.5, \\ B_k(1 - P_i) & , P_i \geq 0.5, \end{cases} \tag{3.1}$$

where P_{i+1} represents the $i + 1$th mapped value, and P_i represents the ith mapped value. B_k is a parameter, the range (0,4] [40]. However, bifurcation analysis and Lyapunov exponent analysis show that the tent map has a limited chaotic range and nonuniform distribution of the variant density function [41, 42]. In terms of chaotic characteristics, the sine map also exhibits similar limitations to the tent map. The sine map is defined by Eq (3.2):

$$P_{i+1} = B_k \frac{\sin(\pi P_i)}{4}. \tag{3.2}$$

To address the aforementioned issues, Zhou et al. [41] integrated two one-dimensional chaotic maps, the tent map and the sine map, to generate a new chaotic map, the tent-sine map (TSM), as defined by Eq (3.3), as follows:

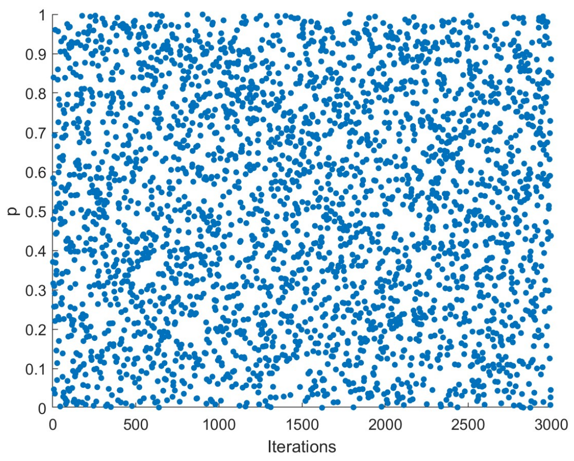
$$P_{i+1} = \begin{cases} \left(B_k \frac{P_i}{2} + (4 - B_k)\frac{\sin(\pi P_i)}{4}\right) mod\ 1 & , P_i < 0.5, \\ \left(B_k \frac{(1 - P_i)}{2} + (4 - B_k)\frac{\sin(\pi P_i)}{4}\right) mod\ 1 & , P_i \geq 0.5, \end{cases} \tag{3.3}$$

where B_k is a parameter within the range (0,4], we follow Bisht et al.'s work [40] in which they set the value of B_k as 3.999. Analysis shows that TSM exhibits excellent chaotic properties, including a wide range of parameter settings and a uniformly distributed varying density function [40–42]. We conducted 3000 iterations. The scatter plots of the three maps and random distribution are shown in Figure 1. The results indicate that TSM yields a more uniform distribution. The TSM, in contrast to the sine map, results in fewer individuals being positioned at the boundaries. Furthermore, when compared to the internal distribution of the tent map, the TSM shows reduced overlap among individuals. The initial distribution generated by the TSM leads to a greater degree of dispersion. This enhances population diversity and minimizes the likelihood of convergence to local optima. In the

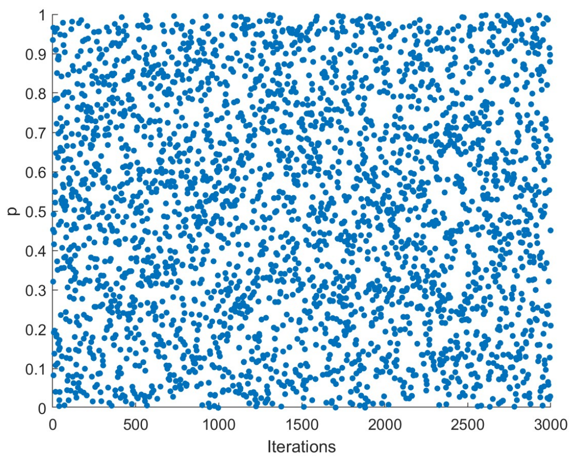
DBO algorithm, replacing the original random initialization method with the tent-sine chaotic map helps to distribute the population more evenly and enhances its diversity. The initialization method is shown in Eq (3.4).

$$X_i^t = LB_i^t + (UB_i^t - LB_i^t) \cdot P_i^t, \tag{3.4}$$

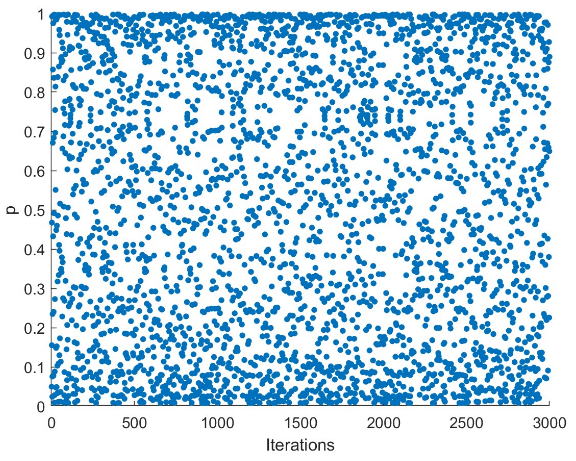
LB_i^t and UB_i^t represent the lower and upper bounds for the ith particle. The mDBO employs a hybrid chaotic mapping optimization strategy for population initialization, replacing the simple random initialization method used in the original algorithm, as shown in Eq (3.4). This innovative initialization method significantly enhances the quality of the initial population distribution in the search space, strengthening the exploration capabilities of the DBO algorithm during the global search process, while markedly improving its convergence speed and accuracy in finding global optimal solutions.



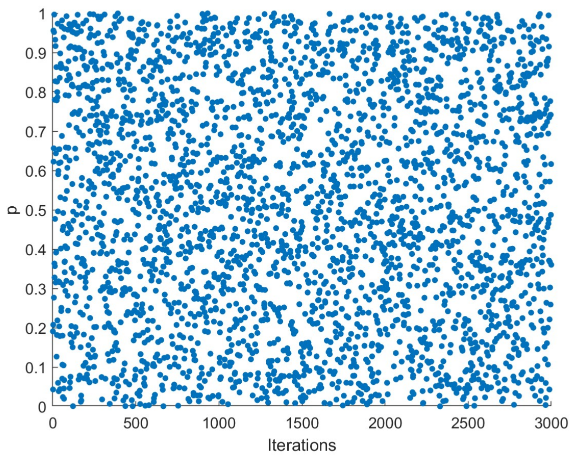
(a) Randomly distributed scatter plot



(b) Tent mapping scatter plot

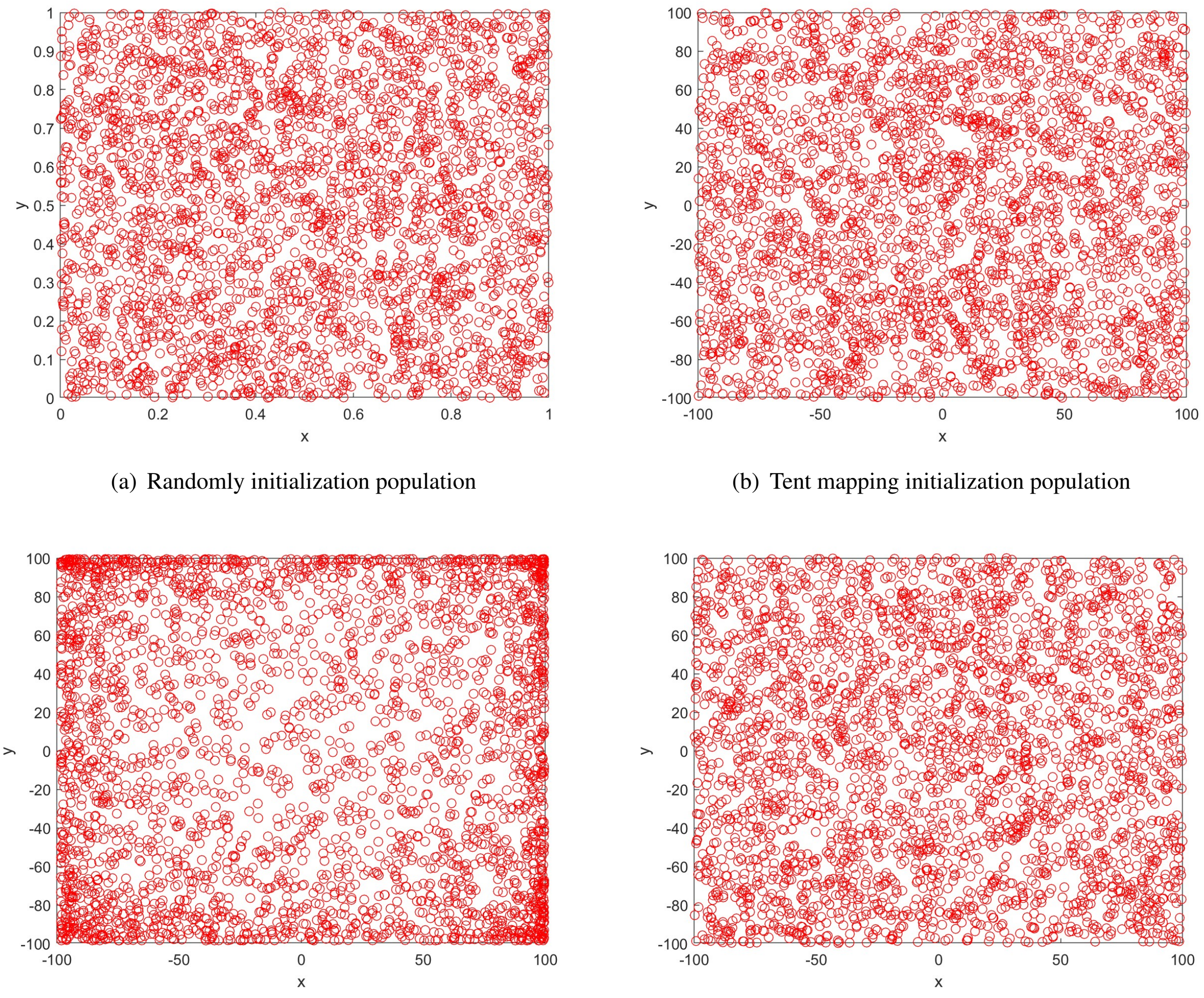


(c) Sine mapping scatter plot



(d) Tent-sine mapping scatter plot

Figure 1. Comparison of scatter plot.



(a) Randomly initialization population
(b) Tent mapping initialization population
(c) Sine mapping initialization population
(d) Tent-sine mapping initialization population

Figure 2. Population initialization plot.

Additionally, opposition-based learning (OL) is a powerful optimization tool proposed by Tizhoosh [43], and the concept has been successfully applied in various metaheuristic algorithms [44–46] to enhance convergence speed. Unlike traditional opposition-based learning strategies, a new OL strategy, called random OL (ROBL), was proposed in [47]. In contrast to conventional OL strategies, the opposition solutions described by random OL exhibit more randomness in exploration. Therefore, random OL can effectively enhance population diversity and help it escape local optima. ROBL [47] is a randomization process based on reverse learning, characterized by adaptability and diversity. In swarm intelligence algorithms, particularly during the initialization phase, ROBL can break the locality of the population through random perturbations, thereby increasing the coverage of the search space. The experimental section in Section 4.2.2 presents relevant analysis based on common opposition-based learning. Therefore, considering the aforementioned advantages of ROBL, this paper combines ROBL with TSM, enabling the population initialization to not only cover a broader solution space with better distribution characteristics but also accelerate convergence speed. ROBL is applied to refine the chaotic population by determining the opposite direction of each solution according to Eq (3.5). The fitness function for each solution in both the chaotic population X and its opposite population OX is then computed. From the union of these two populations, the best N solutions with the lowest fitness values are selected to form the initial population.

$$OX_i^t = LB_i^t + UB_i^t - rand \cdot X_i^t, \tag{3.5}$$

where LB_i^t and UB_i^t represent the lower and upper bounds for the ith particle, rand is an arbitrary value within the interval [0,1]; OX_i^t denotes the opposite solution, which compared to Eq (3.4), offers sufficient random exploration. Therefore, Eq (3.5) can successfully enhance the diversity of the population, help it avoid local optima, make the search method more flexible, and also expand the search space of the DBO, thereby improving the quality of the initial population solutions.

Algorithm 2 Population initialization strategy based on hybrid tent-sine map and random opposition-based learning

```
Require: $N,LB,UB$
Ensure: $X_i^t$
 1: Generate a random $P$
 2: for $i = 1$ to $N$ do
 3:     if $P_i < 0.5$ then
 4:         $P_{i+1} = \left(B_k \frac{P_i}{2} + (4 - B_k)\frac{\sin(\pi P_i)}{4}\right) mod\ 1$
 5:     else
 6:         $P_{i+1} = \left(B_k \frac{(1 - P_i)}{2} + (4 - B_k)\frac{\sin(\pi P_i)}{4}\right) mod\ 1$
 7:     end if
 8: end for
 9: for $i = 1$ to $N$ do
10:     $X_i^t = LB_i^t + (UB_i^t - LB_i^t)P_i^t$
11:     $OX_i^t = LB_i^t + UB_i^t - randX_i^t$
12:     if $rand < z$ then
13:         $X_i^t = OX_i^t$
14:     else
15:         $X_i^t = X_i^t$
16:     end if
17: end for
18: Return $X_i^t$
```

3.2. *Differential evolution mutation strategy with periodic retrospective adaptive mutation factor*

In 1995, Storn et al. [48] introduced an innovative algorithm known as differential evolution (DE), which revitalized the optimization field with its unique core processes of mutation, crossover, and selection. As an efficient stochastic search method, DE excels in solving a wide range of optimization problems across various practical applications. In the DE algorithm, each search agent is referred to as a target vector. During the initialization phase, DE randomly generates all vectors within the search domain. Subsequently, individuals are updated through mutation, crossover, and selection operations until the termination conditions are met. The mutation operation perturbs a set of specified target vectors to create new mutant vectors. The crossover operation combines these mutant vectors with their corresponding target vectors to form new trial vectors. Finally, the selection operation evaluates the objective function values of these trial and target vectors to determine which is superior, retaining the better solution for the next generation. DE is frequently integrated with genetic variation methods, effectively enhancing population diversity, mitigating premature convergence, and improving both exploratory capabilities and convergence accuracy. We integrate the periodic retrospective adaptive differential evolution (PRADE) process into the core framework of DBO, with the objective of enhancing optimization performance. This integration enables a more extensive exploration of the solution space, while improving the effectiveness and accuracy of the optimization results.

(1) Mutation process

The differential evolution mechanism enables the algorithm to generate mutated candidate solutions through the application of mutation operators. For each target candidate solution, the mutation operator is employed to create the corresponding mutated candidate solution. In this work, an adaptive mutation operator with Periodic Retrospective is proposed, specifically designed for the *i*th generation, as illustrated in Eq (3.6):

$$V_i^{t+1} = X_i^t + \underbrace{F(X_{r1}^t - X_{r2}^t)}_{part1} + \underbrace{(1-F)(X_{best}^t - X_i^t)}_{part2}, \tag{3.6}$$

where V_i^{t+1} denotes the mutated individual, X_{r1}^t and X_{r2}^t are randomly selected individuals, with the condition that $r1 \neq r2$, and X_{best}^t indicates the best individual. F represents the periodic retrospective adaptive mutation factor, and based on the inertia weight in Eq (3.7) [49], we have innovatively constructed the periodic retrospective mutation factor Eq (3.8). Through the comparative experiments in Section 4.2.3, the periodic retrospective mutation factor Eq (3.8) shows better performance compared to the inertia weight. The expression for F is as follows:

$$\tilde{F} = F_{min} + (F_{max} - F_{min})\left(1 - \frac{t}{T_{max}}\right), \tag{3.7}$$

$$F = \left[F_{min} + (F_{max} - F_{min})\left(1 - \frac{t\%\dfrac{T_{max}}{T_t}}{\dfrac{T_{max}}{T_t}}\right)H\right]\left(1 - \frac{t}{T_{max}}\right), \tag{3.8}$$

t represents the current function evaluation, and T_{max} denotes the maximum number of function evaluations. The symbol % is used to indicate either the remainder or the modulo operation. T_t is a variable used to determine the number of cycles, and extensive experiments were conducted over a wide range of the parameter T_t, with the experimental results shown in Table 1. The table indicates that when the parameter T_t is set to 10, most of the optimal values for the CEC2017 test functions are achieved. F_{max} and F_{min} are the maximum and minimum mutation factors, respectively. f_{mean} is the average value of the fitness function, which is calculated from f_{min} and f_{max}. H is the adaptive factor, with its value ranging between [0,1]. The expression for H is as follows:

$$H = 1 - \frac{f_{mean} - f_{min}}{f_{max} - f_{min} + \epsilon}. \tag{3.9}$$

Here, ϵ is a very small number introduced to prevent the denominator from becoming zero. In this study, ϵ is taken as *realmin*.

It is widely recognized that a robust global search capability is crucial at the onset of iterative optimization to explore a broad spectrum of possible solutions. At the end of iterative optimization, a strong local search capability is needed to improve accuracy. In the early iterations of the algorithm, a larger mutation scaling factor F leads to an increased contribution of the first part (part1) in Eq (3.6), thereby exerting a significant influence on the algorithm's behavior. Therefore, a larger mutation scaling factor F leads to stronger global exploration. At this stage, in the proposed Eq (3.8), the decrease in the mutation scaling factor F exhibits a dynamic periodicity, which can retrospectively enhance the influence of the mutation scaling factor F. This method can further enhance the algorithm's global exploration capability during this stage, thereby effectively reducing the probability of the algorithm missing effective potential candidate solutions. For later iterations, a smaller mutation scaling factor F allows the algorithm to focus on the role of the second part (part2) of the Eq (3.6), leading to stronger local exploitation capabilities. However, during later iterations, a smaller mutation scaling factor F can result in poorer population diversity, causing the algorithm to easily fall into local optima. Fortunately, at this stage, the mutation scaling F in Eq (3.8) of this work has a dynamic periodicity, which can retrospectively and adaptively increase the value of the mutation scaling factor F, thereby appropriately enhancing the role of the first part of the formula. This method significantly improves the population's ability to escape local optima. The values of F may vary slightly with different parameter values, but the general trend and behavior are as shown in Figure 3.

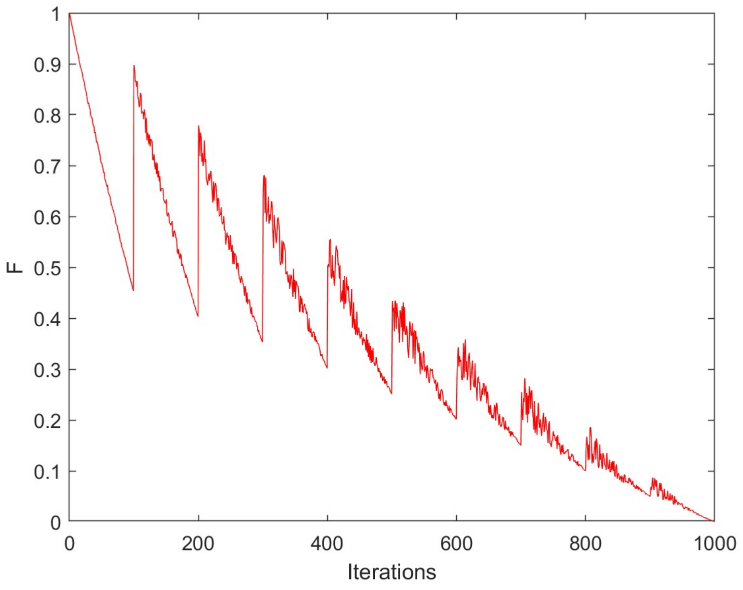


Figure 3. Trend of F-value changes.

(2) Crossover process

After the mutation process concludes, the crossover individuals U_i^{t+1} are obtained when the previous individuals are replaced with mutated individuals with a specific crossover probability (CR).

The crossover process is shown in Eq (3.10):

$$U_i^{t+1} = \begin{cases} V_i^{t+1} & , rand(0,1) \le CR \text{ or } j = j_{rand}, \\ X_i^{t+1} & , \text{otherwise}, \end{cases} \tag{3.10}$$

where $CR \in [0,1]$ denotes the crossover probability. In this study, $CR = 0.8$, and how to reach this value will be demonstrated later. X_i represents the contribution of the ith search agent. j_{rand} is a random integer within the range $[1, 2, ..., Dim]$, which ensures that at least one individual undergoes mutation.

(3) Selection process

After the mutation and crossover processes are completed, a greedy criterion is applied to compare the cross-seed U_i^{t+1} with the original target seed X_i^{t+1} to select the seed with the better fitness value to be carried into the next iteration. The model of the greedy criterion is as shown in Eq (3.11):

$$X_i^{t+1} = \begin{cases} U_i^{t+1} & , \text{if } f(U_i^{t+1}) \le f(X_i^{t+1}), \\ X_i^{t+1} & , \text{otherwise}. \end{cases} \tag{3.11}$$

The PRADE strategy enhances information exchange among individuals through mutation and crossover operations, thereby expanding the search range and diversity of the population. This effectively prevents the algorithm from stagnating prematurely and reduces the likelihood of converging to local optima, thus improving convergence accuracy. This strategy plays a crucial role in optimization algorithms by dynamically adjusting the search process, increasing the global search capability and exploration efficiency of the algorithm, making it more efficient and accurate in finding global optimal solutions. As demonstrated in the ablation experiments presented later in this paper, this strategy contributes to a notable improvement in the performance of the original algorithm. The specific process is shown in Algorithm 3.

3.3. Padé approximation strategy with adaptive evolutionary boundary constraint

3.3.1. Padé approximation strategy

Rational function approximation is essential in the field of nonlinear approximation due to its favorable properties. These functions not only demonstrate superior approximation performance in the vicinity of poles, but they also exhibit consistent convergence to a specified value as the independent variable approaches infinity. This characteristic underscores the advantages and reliability of rational function approximation in addressing problems that involve asymptotic behavior.

Table 1. Adjusting the parameters T_t.

Function	Metric	$T_t = 7$	$T_t = 8$	$T_t = 9$	$T_t = 10$	$T_t = 11$	$T_t = 12$
F1	mean	8.1122×10^{3}	5.8756×10^{3}	5.3932×10^{3}	4.5202×10^{3}	5.3976×10^{3}	6.9359×10^{3}
	std	2.1706×10^{7}	1.3613×10^{7}	1.8285×10^{7}	1.2579×10^{7}	1.9904×10^{7}	2.6727×10^{7}
F3	mean	1.8721×10^{3}	5.7857×10^{2}	6.3211×10^{2}	3.2275×10^{2}	1.5326×10^{3}	4.8257×10^{2}
	std	1.1319×10^{7}	3.2602×10^{5}	6.1607×10^{5}	1.7162×10^{3}	1.4683×10^{7}	2.4419×10^{5}
F4	mean	4.0648×10^{2}	4.1314×10^{2}	4.1695×10^{2}	4.0564×10^{2}	4.1085×10^{2}	4.1088×10^{2}
	std	1.1220×10^{1}	5.3637×10^{2}	7.4073×10^{2}	8.4362×10^{0}	2.4994×10^{2}	5.3349×10^{2}
F5	mean	5.3420×10^{2}	5.3113×10^{2}	5.3482×10^{2}	5.3243×10^{2}	5.2723×10^{2}	5.3562×10^{2}
	std	1.5674×10^{2}	1.1167×10^{2}	1.2814×10^{2}	1.7155×10^{2}	1.0460×10^{2}	5.4658×10^{1}
F6	mean	6.0698×10^{2}	6.0448×10^{2}	6.0502×10^{2}	6.0300×10^{2}	6.0639×10^{2}	6.0441×10^{2}
	std	2.6418×10^{1}	1.9135×10^{1}	1.5327×10^{1}	6.9462×10^{0}	3.4406×10^{1}	7.1919×10^{0}
F7	mean	7.4720×10^{2}	7.4229×10^{2}	7.4307×10^{2}	7.4029×10^{2}	7.5053×10^{2}	7.4132×10^{2}
	std	2.3223×10^{2}	4.0574×10^{2}	2.4143×10^{2}	4.9388×10^{1}	2.4722×10^{2}	9.9491×10^{1}
F8	mean	8.2432×10^{2}	8.3784×10^{2}	8.2361×10^{2}	8.2070×10^{2}	8.3362×10^{2}	8.2527×10^{2}
	std	1.0317×10^{2}	6.6917×10^{1}	5.5423×10^{1}	7.7831×10^{1}	1.7264×10^{2}	1.0256×10^{2}
F9	mean	9.2849×10^{2}	9.4981×10^{2}	9.5679×10^{2}	9.3779×10^{2}	9.9337×10^{2}	9.5579×10^{2}
	std	1.1026×10^{3}	1.3831×10^{4}	4.7782×10^{3}	2.5580×10^{3}	1.4509×10^{4}	3.2160×10^{3}
F10	mean	1.9452×10^{3}	1.8614×10^{3}	1.9615×10^{3}	1.8561×10^{3}	1.9830×10^{3}	1.9757×10^{3}
	std	1.6901×10^{5}	1.2264×10^{5}	5.4902×10^{4}	1.1535×10^{5}	1.0999×10^{5}	9.0506×10^{4}
F11	mean	1.1282×10^{3}	1.1537×10^{3}	1.1524×10^{3}	1.1231×10^{3}	1.1534×10^{3}	1.1939×10^{3}
	std	3.8222×10^{2}	4.6372×10^{3}	1.3911×10^{3}	5.0901×10^{2}	1.4892×10^{3}	1.1976×10^{4}
F12	mean	3.3812×10^{4}	2.2143×10^{4}	1.2329×10^{4}	1.9758×10^{4}	2.9559×10^{4}	1.3213×10^{4}
	std	5.9414×10^{8}	3.2021×10^{8}	4.2051×10^{7}	2.9496×10^{8}	3.3627×10^{8}	1.7277×10^{8}
F13	mean	1.2474×10^{4}	1.3313×10^{4}	1.4572×10^{4}	8.8748×10^{3}	1.4134×10^{4}	1.2353×10^{4}
	std	8.8279×10^{7}	1.2523×10^{8}	1.0816×10^{8}	8.2545×10^{7}	1.5836×10^{8}	9.3165×10^{7}
F14	mean	1.8822×10^{3}	1.5532×10^{3}	1.7199×10^{3}	1.7424×10^{3}	1.6285×10^{3}	2.1784×10^{3}
	std	3.9884×10^{5}	1.6019×10^{4}	6.5933×10^{4}	1.7866×10^{5}	2.6420×10^{4}	7.8149×10^{5}
F15	mean	3.8809×10^{3}	2.4504×10^{3}	3.0488×10^{3}	3.1537×10^{3}	4.3804×10^{3}	4.1290×10^{3}
	std	4.6433×10^{6}	5.6653×10^{5}	1.5017×10^{6}	2.9455×10^{6}	4.2111×10^{7}	3.9146×10^{6}
F16	mean	1.7740×10^{3}	1.7845×10^{3}	1.7999×10^{3}	1.7159×10^{3}	1.8204×10^{3}	1.7424×10^{3}
	std	9.7586×10^{3}	1.3216×10^{4}	1.5667×10^{4}	1.3644×10^{4}	1.0792×10^{4}	1.2300×10^{4}
F17	mean	1.7675×10^{3}	1.7610×10^{3}	1.7668×10^{3}	1.7505×10^{3}	1.7735×10^{3}	1.7761×10^{3}
	std	1.4619×10^{3}	4.5498×10^{2}	1.2293×10^{3}	2.7499×10^{2}	1.1710×10^{3}	2.7988×10^{3}
F18	mean	1.8055×10^{4}	1.9538×10^{4}	2.2831×10^{4}	1.3235×10^{4}	1.6057×10^{4}	1.7092×10^{4}
	std	2.7112×10^{8}	1.6530×10^{8}	2.1217×10^{8}	1.3104×10^{8}	2.1120×10^{8}	1.2895×10^{8}
F19	mean	4.6859×10^{3}	4.7406×10^{3}	3.9579×10^{3}	3.4620×10^{3}	5.5067×10^{3}	4.6813×10^{3}
	std	5.2842×10^{7}	1.6886×10^{7}	1.4383×10^{7}	1.0515×10^{6}	1.9581×10^{7}	8.5551×10^{6}
F20	mean	2.0842×10^{3}	2.1061×10^{3}	2.0645×10^{3}	2.1042×10^{3}	2.0766×10^{3}	2.1000×10^{3}
	std	3.1397×10^{3}	7.7265×10^{3}	2.0963×10^{3}	3.0038×10^{3}	4.5279×10^{3}	4.9507×10^{3}
F21	mean	2.3254×10^{3}	2.2784×10^{3}	2.2970×10^{3}	2.2601×10^{3}	2.2808×10^{3}	2.2773×10^{3}
	std	1.9762×10^{3}	4.1407×10^{3}	3.5777×10^{3}	5.1221×10^{3}	4.3997×10^{3}	4.0555×10^{3}
F22	mean	2.3103×10^{3}	2.3080×10^{3}	2.2985×10^{3}	2.3032×10^{3}	2.3053×10^{3}	2.3069×10^{3}
	std	4.1776×10^{2}	1.2884×10^{2}	3.8443×10^{2}	9.6182×10^{0}	4.9593×10^{1}	1.6449×10^{1}
F23	mean	2.6314×10^{3}	2.6360×10^{3}	2.6407×10^{3}	2.6307×10^{3}	2.6374×10^{3}	2.6362×10^{3}
	std	2.4638×10^{2}	9.0650×10^{1}	2.5108×10^{2}	1.3905×10^{2}	4.6866×10^{2}	2.8697×10^{2}
F24	mean	2.7747×10^{3}	2.7648×10^{3}	2.7470×10^{3}	2.7363×10^{3}	2.7759×10^{3}	2.7531×10^{3}
	std	8.5927×10^{1}	2.4186×10^{2}	1.5691×10^{3}	5.8645×10^{3}	1.9120×10^{2}	3.0901×10^{3}
F25	mean	2.8991×10^{3}	2.8991×10^{3}	2.9381×10^{3}	2.9460×10^{3}	2.9409×10^{3}	2.9478×10^{3}
	std	1.1781×10^{4}	1.8308×10^{3}	8.1615×10^{2}	3.3640×10^{2}	6.7308×10^{2}	3.4512×10^{2}
F26	mean	3.0564×10^{3}	3.0358×10^{3}	3.0566×10^{3}	2.9906×10^{3}	3.2910×10^{3}	3.1296×10^{3}
	std	1.6563×10^{4}	2.9422×10^{4}	1.7513×10^{4}	7.8682×10^{3}	2.5443×10^{5}	1.1462×10^{5}
F27	mean	3.1019×10^{3}	3.1040×10^{3}	3.1019×10^{3}	3.1007×10^{3}	3.1023×10^{3}	3.1027×10^{3}
	std	5.3855×10^{1}	2.3199×10^{2}	5.8394×10^{1}	4.7009×10^{1}	4.1858×10^{1}	4.2291×10^{1}
F28	mean	3.3443×10^{3}	3.3811×10^{3}	3.3267×10^{3}	3.2878×10^{3}	3.3505×10^{3}	3.3254×10^{3}
	std	6.6638×10^{3}	1.0542×10^{4}	9.2670×10^{3}	7.3558×10^{3}	1.5448×10^{4}	1.0975×10^{4}
F29	mean	3.2671×10^{3}	3.2578×10^{3}	3.2479×10^{3}	3.2231×10^{3}	3.2543×10^{3}	3.2441×10^{3}
	std	2.3999×10^{3}	4.0466×10^{3}	2.6586×10^{3}	1.9508×10^{3}	3.9782×10^{3}	2.4723×10^{3}
F30	mean	8.8219×10^{5}	4.7871×10^{5}	4.4451×10^{5}	4.0954×10^{5}	3.5397×10^{5}	8.3754×10^{5}
	std	6.9906×10^{11}	2.1046×10^{11}	3.7081×10^{11}	1.6362×10^{11}	1.1999×10^{11}	5.6646×10^{11}

Algorithm 3 Differential evolution mutation strategy with periodic retrospective adaptive mutation factor

```
Require: $X_i^t$,$T_{max}$,$T_t$,$F_{max}$,$F_{min}$,$Dim$
Ensure: $X_i^{t+1}$
 1: for $i$ = pNum+bNum+1 to pNum+bNum+sNum do
 2:     Calculate $X_i^{t+1}$ using Eq (2.6) and then calculate the fitness value $fit(X_i^{t+1})$
 3: end for
 4: for $i$=pNum+bNum+1 to pNum+bNum+sNum do
 5:     $F = \left[F_{min} + (F_{max} - F_{min})\left(1 - \frac{t\%\frac{T_{max}}{T_t}}{\frac{T_{max}}{T_t}}\right)H\right]\left(1 - \frac{t}{T_{max}}\right)$
        $H = 1 - \frac{f_{mean} - f_{min}}{f_{max} - f_{min} + \epsilon}$
 6:     $V_i^{t+1} = X_i^t + F(X_{r1}^t - X_{r2}^t) + (1 - F)(X_{best}^t - X_i^t)$
 7:     for $j$ = 1 to $Dim$ do
 8:         if $rand(0, 1) \leq CR$ or $j = j_{rand}$ then
 9:             $U_i^{t+1} = V_i^{t+1}$
10:         else
11:             $U_i^{t+1} = X_i^{t+1}$
12:         end if
13:     end for
14:     if $f(U_i^{t+1})\leq f(X_i^{t+1})$ then
15:         $X_i^{t+1} = U_i^{t+1}$
16:     else
17:         $X_i^{t+1} = X_i^{t+1}$
18:     end if
19: end for
20: Return $X_i^{t+1}$
```

Padé approximation is a method of rational function approximation that is widely used in numerical analysis and is theoretically applicable for approximating any function with a finite order of continuous derivatives. Compared to truncated Taylor series, Padé approximations are often more accurate, and even when Taylor series fail to converge, Padé approximations can be valid [50]. In our approach, the application of Padé approximation effectively improves the distribution quality of the solutions and enhances the accuracy of the algorithm. The reason for choosing Padé approximation is its favorable convergence properties and relatively low computational complexity, especially when handling nonlinear and complex problems, where it can provide more precise approximations. The core advantage of Padé approximation lies in its strong adaptability to initial conditions and perturbations. Even in cases with uncertain or complex boundary conditions, Padé approximation maintains good numerical stability. Vazquez-Leal and his colleagues [51] proposed a method that directly employs the Padé approximation to derive approximate solutions for nonlinear differential equations. This method demonstrates both high precision and convenience, underscoring its considerable potential for addressing a range of complex problems. Andrianov and Shatrov [52] investigated the application of the Padé approximation in boundary layer aerodynamics and heat transfer, focusing on the transformation of asymptotic expansions into rational or fractional-order functions. They utilized the Padé approximation to examine the external asymptotics of viscous gas boundary layer issues. Additionally, Xu and his team [53] developed a novel concentration overpotential parameterization model within the P2D framework, based on a simplified partial differential equation (PDE) for liquid phase diffusion. This model achieves a new analytical solution for the liquid phase diffusion PDE under enhanced boundary conditions, grounded in the principle of mass conservation as applied through the Padé approximation.

Padé approximation is a versatile and highly effective rational function approximation method that starts from the power series. Suppose

$$T(z) = \sum_{i=0}^{\infty} c_i z^i. \tag{3.12}$$

It is a given formal power series, suppose

$$\begin{aligned} \mathbb{H}_m &= \{P | P(z) = \sum_{i=0}^{m} c_i z^i, z, c_i \in \mathbb{C}^1\}, \\ \mathbb{R}_{m,n} &= \{R = P/Q | P \in \mathbb{H}_m, Q \in \mathbb{H}_n \backslash \{0\}\}. \end{aligned} \tag{3.13}$$

If the rational function $P/Q \in \mathbb{R}_{m,n}$, satisfies

$$\begin{aligned} T(z) - \frac{P(z)}{Q(z)} &= O(z^{m+n+1}), \\ Q(0) &= 1. \end{aligned} \tag{3.14}$$

Then, P/Q is referred to as the Padé approximant of $T(z)$ in $\mathbb{R}_{m,n}$, denoted as $[m/n]$.

$$\begin{aligned} P(z) &= p_0 + p_1 x + \cdots + p_m x^m, \\ Q(z) &= q_0 + q_1 x + \cdots + q_n x^n. \end{aligned} \tag{3.15}$$

Padé approximation utilizes rational functions to approximate a target function, frequently yielding more accurate results than truncated Taylor series [50]. When the degrees of the numerator and denominator are equal or closely matched, Padé approximation exhibits optimal convergence properties, indicating that the error decreases rapidly as the degree of approximation increases. The computational process of Padé approximation is straightforward, as it does not require complex numerical integration or differentiation, thus enabling efficient handling of large-scale data. These characteristics make rational functions an effective alternative to traditional polynomials. In this study, we propose a novel Padé approximation strategy to enhance the search performance of the DBO algorithm. In optimization problems, the Padé approximation strategy can be employed to leverage information from the current solution to approximate and assess the behavior of the objective function. This approach predicts the function's shape within the region of interest, providing valuable insights to the algorithm and helping to uncover more promising candidate solutions, thus reducing unnecessary searches. During the search process, three adjacent solutions are selected, and based on these points, a rational approximation function is constructed that passes through them. By solving a system of linear equations, the coefficients of the function are determined, yielding the expression of the approximation function. Subsequently, the derivative of the resulting rational function is computed to identify its stationary points, which may correspond to potential extreme values of the function. Suppose two extreme points are found, where one is the minimum point and the other is the maximum point. The minimum point is considered the most likely candidate for the optimal solution. However, the maximum point should not be disregarded, as it could assist in escaping local optima. Both points are potentially valuable solutions. Under the constraints of the boundary conditions, these two new solutions are compared, and the better solution is then compared with the best solution from the initial set of three solutions to update the optimal solution. This innovative Padé approximation strategy is integrated into the DBO algorithm, significantly improving the population quality and solution accuracy, especially in solving complex optimization problems. We adopt the following form of Padé approximation:

$$T(X) = \frac{a_1 + a_2 X}{1 + a_3 X^2}, \tag{3.16}$$

where a_1, a_2, and a_3 are real number parameters. Let $T(X)$ be the Padé approximation function of $f(X)$ at the three adjacent points X_i, X_{i+1}, X_{i+2}, and the three equations are as follows:

$$\begin{cases} T(X_i) = \dfrac{a_1 + a_2 X_i}{1 + a_3 X_i^2} = f(X_i), \\ T(X_{i+1}) = \dfrac{a_1 + a_2 X_{i+1}}{1 + a_3 X_{i+1}^2} = f(X_{i+1}), \\ T(X_{i+2}) = \dfrac{a_1 + a_2 X_{i+2}}{1 + a_3 X_{i+2}^2} = f(X_{i+2}). \end{cases} \tag{3.17}$$

The three corresponding coefficients are determined by solving the linear system of Eq (3.17), resulting in:

$$a_1 = \frac{(X_i^2 - X_{i+2}^2)X_{i+1}f(X_i)f(X_{i+2}) + (X_{i+1}^2 - X_i^2)X_{i+2}f(X_i)f(X_{i+1}) + (X_{i+2}^2 - X_{i+1}^2)X_i f(X_{i+1})f(X_{i+2})}{(X_{i+1} - X_{i+2})X_i^2 f(X_i) + (X_{i+2} - X_i)X_{i+1}^2(X_{i+1}) + (X_i - X_{i+1})X_{i+2}^2 f(X_{i+2})}, \tag{3.18}$$

$$a_2 = \frac{(X_i^2 - X_{i+1}^2)f(X_i)f(X_{i+1}) + (X_{i+2}^2 - X_i^2)f(X_i)f(X_{i+2}) + (X_{i+1}^2 - X_{i+2}^2)f(X_{i+1})f(X_{i+2})}{(X_{i+1} - X_{i+2})X_i^2 f(X_i) + (X_{i+2} - X_i)X_{i+1}^2 f(X_{i+1}) + (X_i - X_{i+1})X_{i+2}^2 f(X_{i+2})}, \tag{3.19}$$

$$a_3 = \frac{(X_{i+1} - X_i)f(X_{i+2}) + (X_{i+2} - X_{i+1})f(X_i) + (X_i - x_{i+2})f(X_{i+1})}{(X_{i+1} - X_{i+2})X_i^2 f(X_i) + (X_{i+2} - X_i)X_{i+1}^2 f(X_{i+1}) + (X_i - X_{i+1})X_{i+2}^2 f(X_{i+2})}. \tag{3.20}$$

By combining Eqs (3.18)–(3.20) and considering the stationary points of the rational function $T(X)$, the potential extreme points of the approximation function $T(X)$ are obtained as:

$$X^* = -\frac{a_1}{a_2} \pm \sqrt{\frac{a_1^2}{a_2^2} + \frac{1}{a_3}}. \tag{3.21}$$

To ensure the validity of Eq (3.21), it is necessary to determine the sign of $\frac{a_1^2}{a_2^2} + \frac{1}{a_3}$. Under the condition that $\frac{a_1^2}{a_2^2} + \frac{1}{a_3}$ is non-negative, the value of X^* is obtained. In this case, a greedy approach is used to compare the fitness values of these two X^* with the fitness value of the current best dung beetle, and the best one is selected to enter the next generation. When $\frac{a_1^2}{a_2^2} + \frac{1}{a_3}$ is negative, the current best dung beetle remains as the optimal solution. The Padé approximation strategy assists the DBO algorithm in making fine adjustments within the search space during the optimization process, thereby improving both the population quality and search accuracy. During the execution of the Padé approximation strategy, some potential solutions may exceed the defined problem boundaries, which is an unacceptable outcome. To address this issue, we propose a novel adaptive evolutionary boundary constraint handling method. The details of this method are discussed in Section 3.3.2.

3.3.2. Adaptive evolutionary boundary constraint

In 2012, Gandomi and Yang [54] introduced an innovative strategy called evolutionary boundary constraint handling (EBCH), which focuses on reintroducing particles that exceed the defined boundaries back into the feasible search space. This method has garnered significant attention due to its ease of integration with swarm intelligence and evolution-based optimization algorithms, thereby enhancing their performance. The application of EBCH across various optimization algorithms is documented in the works of Gandomi and Kashani [55] and Gandomi et al. [56, 57]. In conclusion, when a variable exceeds a defined boundary, the fitness function assigns a specific value to that solution, ensuring it cannot surpass its predecessors and preventing its selection for the subsequent generation. In minimization problems, an infinite value is typically employed to accomplish this, ensuring that solutions that violate constraints are deprioritized. This approach not only upholds the integrity of the optimization process but also facilitates progress towards identifying feasible and improved solutions.

In the evolutionary algorithm proposed by Gandomi and Yang [54], boundary constraint handling is achieved through a specific strategy: when a variable exceeds its predefined limits, it is replaced by a randomly generated value that lies between the current limits and the corresponding variable value of the best solution found thus far. This approach leverages the most recent solutions and continuously updated information from the search space to generate new candidate solutions. Building on prior research and acknowledging the need to enhance efficiency at various stages of the algorithm, it is essential to establish a more effective balance between exploration and exploitation, this paper presents a novel boundary method that focuses on exploration during the early stages and shifts towards exploitation in the later stages. Specifically, this strategy prioritizes global exploration during the initial phase, facilitating the identification of more promising regions. In this study, we define the update process for managing evolutionary boundary constraints as follows:

$$f(X_i \to X_i^*) = \begin{cases} P(t,k) \cdot LB_i + (1 - P(t,k)) \cdot X_i^b & ,X_i < LB_i, \\ P(t,k) \cdot UB_i + (1 - P(t,k)) \cdot X_i^b & ,X_i > UB_i, \end{cases} \tag{3.22}$$

where X_i^b is the relevant component of the global best solution, UB_i and LB_i are the upper and lower bounds of the search space, respectively, and $P(t,k)$ is a nonlinear function with its function value ranging between 0 and 1. The definition of $P(t,k)$ is shown in Eq (3.23).

$$P(t,k) = \left[1 - sin\left(\frac{\pi}{2} \cdot \left(\frac{t}{T_{max}}\right)^k\right)\right], \tag{3.23}$$

where t denotes the current iteration count, while T_{max} indicates the maximum number of iterations. Additionally, k is defined as a parameter.

In this study, a dynamic nonlinear function is employed to define $P(t, k)$, with its variation trend illustrated in Figure 4. In the early stages of the search, the focus is on exploration, resulting in a gradual decrease of the function curve. This approach facilitates the expansion of the search range and enhances population diversity during initial iterations. As the search progresses to the middle and later stages, the focus shifts to exploitation, leading to a more rapid decrease in the curve. This transition enables the algorithm to efficiently guide individuals toward the global optimal solution, thereby enhancing both the global search capability and convergence speed. This strategy establishes a pattern in which $P(t, k)$ decreases slowly at first during the iterations, followed by a rapid decrease, and ultimately stabilizes. This method enhances the balance between local exploration and global search throughout the optimization process, thereby reducing the risk of converging to a local optimum.

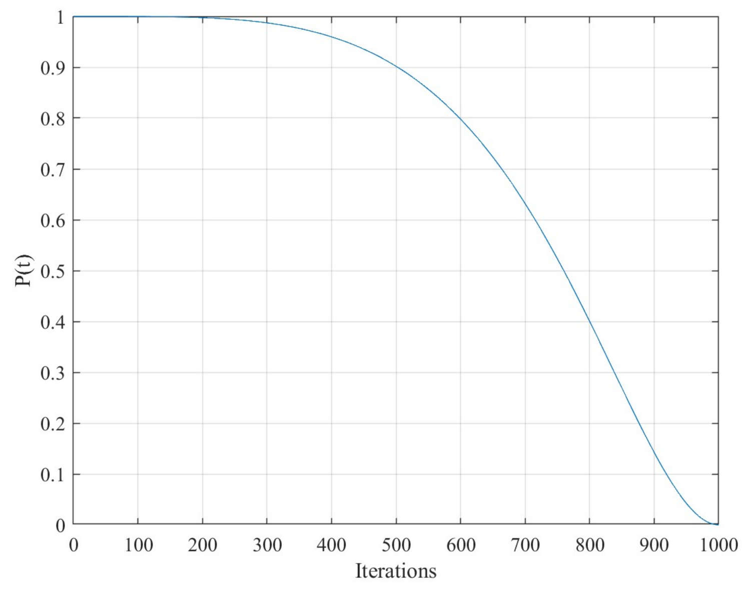


Figure 4. The plot of P value variation with k = 4.

The Padé approximation strategy plays a critical role in enhancing algorithm performance. This strategy leads to notable improvements in solution accuracy and population quality within the algorithm. In addition, the greedy strategy intelligently filters and retains individuals located between the current individual and the potential optimal solution X_i in each iteration. This process equips the algorithm with enhanced search and optimization capabilities. This is further illustrated in Eq (3.24):

$$\begin{cases} X_i^{t+1} = X_i^t & , f(X_i^t) \leq f(X^*), \\ X_i^{t+1} = X^* & , f(X_i^t) > f(X^*), \end{cases} \tag{3.24}$$

where t denotes the tth generation of the dung beetle population. After applying the Padé approximation strategy to DBO, the overall quality of the population is significantly enhanced. This not only improves the adaptability of each individual but also endows the population as a whole with greater survival and reproductive capabilities when facing environmental challenges. Based on the Padé approximation technique and a new adaptive evolutionary boundary constraint method, we propose an innovative approximation strategy. When integrated into the DBO framework, this strategy not only significantly improves the solution accuracy and population quality but also provides effective assistance in balancing exploration and exploitation. The corresponding pseudocode is provided in Algorithm 4.

Algorithm 4 Padé approximation strategy with adaptive evolutionary boundary constraint

```
Require: $X_i^t$
Ensure: $X_i^{t+1}$
1: for $i$=pNum+bNum+sNum+1 to $N$ do
2:     Calculate $X_{i,j}^{t+1}$ using Eq (2.7) and then calculate the fitness value $fit(X_{i,j}^{t+1})$
3: end for
4: for $i$=pNum+bNum+sNum+1 to $N$ do
5:     Select two agents $X_{i+1}$ and $X_{i+2}$, which surround $X_i$
6:     Obtain $a_1$, $a_2$, and $a_3$ according to Eqs (3.18)–(3.20)
7:     if $\frac{a_1^2}{a_2^2} + \frac{1}{a_3} \geq 0$ then
8:         Obtained from Eq (3.21), $G(i) = -\frac{a_1}{a_2} + \sqrt{\frac{a_1^2}{a_2^2} + \frac{1}{a_3}}$, $g(i) = -\frac{a_1}{a_2} - \sqrt{\frac{a_1^2}{a_2^2} + \frac{1}{a_3}}$
9:         Determine the boundaries based on Eqs (3.22) and (3.23), and obtain the final $G(i)$ and $g(i)$
10:        $Tag1 = fit(G(i)), Tag2 = fit(g(i))$
11:        if $Tag1 < Tag2$ then
12:            $Tag = Tag1, G(i) = G(i)$
13:        else
14:            $Tag = Tag2, G(i) = g(i)$
15:        end if
16:        if $Tag < fit(i)$ then
17:            $X_{i,j}^{t+1} = G(i), fit(i) = Tag$
18:        end if
19:    else
20:        $X_{i,j}^{t+1} = X_{i,j}^{t+1}$
21:    end if
22: end for
23: Return $X_{i,j}^{t+1}$
```

3.4. The framework of mDBO

In this study, the enhancements to the DBO algorithm are primarily presented in three aspects: First, a novel population initialization strategy is proposed, which combines the hybrid tent-sine map with ROBL to generate the initial population. This approach replaces the random value generation process used in the original algorithm and results in a more uniform distribution of the initial population, thereby improving the quality of the population distribution within the search space. Second, a new differential evolution mutation strategy is introduced, incorporating a periodic retrospective adaptive mutation factor. This strategy effectively mitigates the risk of premature convergence, reduces the likelihood of getting trapped in local minima, and enhances the exploration of the solution space. Finally, a strategy based on Padé approximation techniques and an adaptive evolutionary boundary constraint method is proposed to improve the quality and accuracy of the solutions across the population. By integrating these three strategies, we developed the mDBO. The implementation process of the mDBO algorithm is outlined in Algorithm 5 and illustrated in Figure 5.

Algorithm 5 The framework of the mDBO algorithm

```
Require: The maximum iteration $T_{max}$,the size of the population $N$, Crossover Probability $CR$, Global
    optimal solution $X_{best}$
Ensure: Optimal position $X_b$ and its fitness value $f_b$
 1: Initialize the population by Algorithm 2, and define its relevant parameters
 2: while $t \leq T_{max}$ do
 3:     for $i = 1$ to $N$ do
 4:         Update the position of the Dung beetle by Algorithm 1
 5:     end for
 6:     for $i$=pNum+bNum+1 to pNum+bNum+sNum do
 7:         Update the position of the Dung beetle by Algorithm 3
 8:     end for
 9:     for $i$=pNum+bNum+sNum+1 to $N$ do
10:         Update the position of the Dung beetle by Algorithm 4
11:     end for
12:     Generate optimal individuals $X_b$ and calculate fitness values $f_b$
13:     if $f(X_b)<f(X_{best})$ then
14:         Output the optimal position $X_b$
15:     else
16:         Output the optimal position $X_{best}$
17:     end if
18:     $t = t + 1$
19: end while
20: Return the best position and the best fitness
```

3.5. Binary mDBO for feature selection

In this section, we propose a binary adaptation of the technique that employs classifiers to assess the objective function and utilizes transfer functions to convert continuous outcomes into binary results (0,1). The effectiveness of the proposed approach will be validated through its application to real-world feature selection tasks, with evaluations based on publicly accessible datasets.

As discussed, FS is a preprocessing technique designed to eliminate unnecessary, noisy, and redundant variables from datasets. The primary objective of FS is to identify and select an optimal subset of features that streamlines data processing, reduces computational costs, and improves the classification performance of models. In this context, the bmDBO serves as an FS wrapper technique grounded in meta-heuristic methods. It utilizes classifiers to assess its predefined objective fitness function.

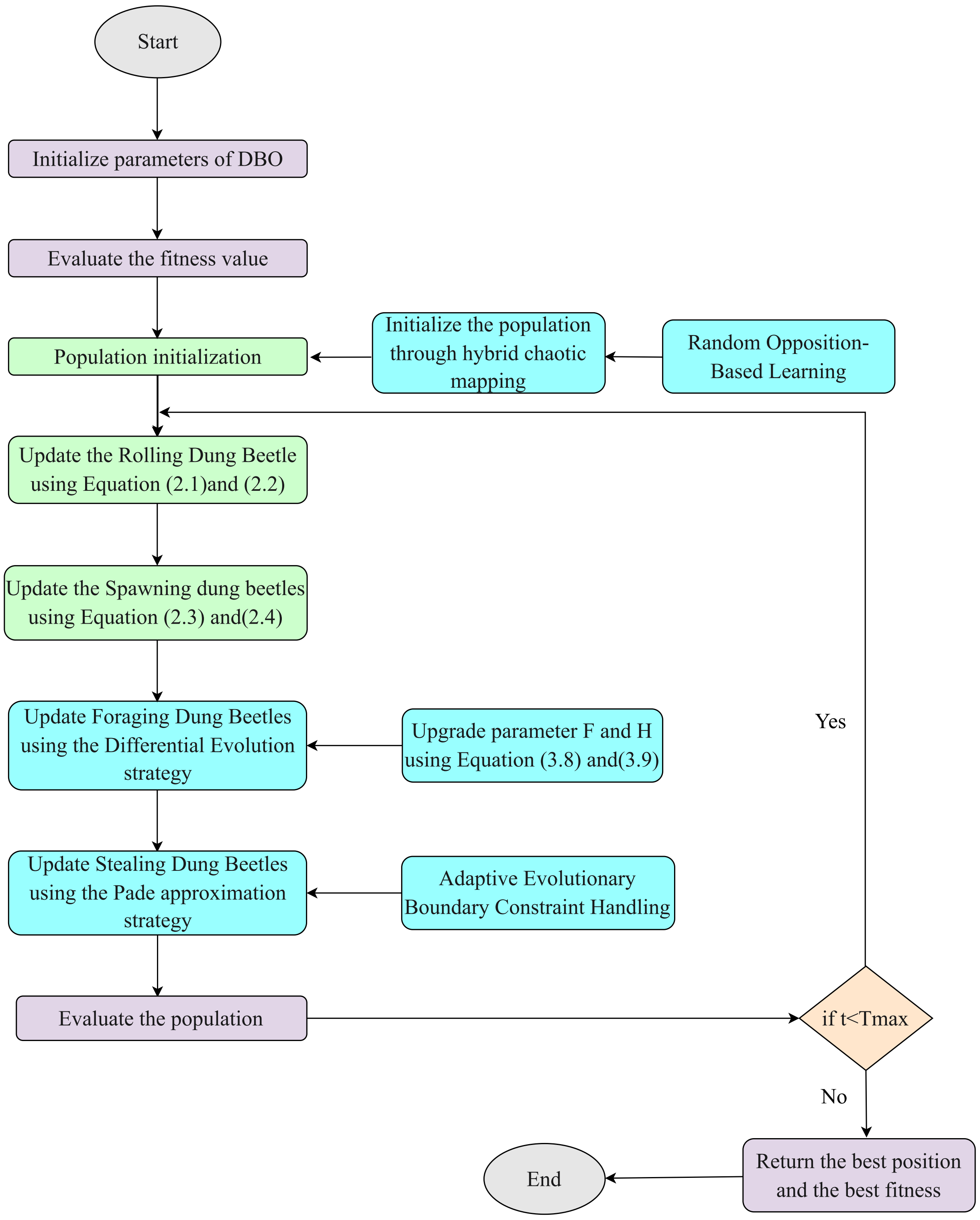

Start
Initialize parameters of DBO
Evaluate the fitness value
Population initialization
Initialize the population through hybrid chaotic mapping
Random Opposition-Based Learning
Update the Rolling Dung Beetle using Equation (2.1)and (2.2)
Update the Spawning dung beetles using Equation (2.3) and(2.4)
Update Foraging Dung Beetles using the Differential Evolution strategy
Upgrade parameter F and H using Equation (3.8) and(3.9)
Update Stealing Dung Beetles using the Pade approximation strategy
Adaptive Evolutionary Boundary Constraint Handling
Evaluate the population
if t<Tmax
Yes
No
Return the best position and the best fitness
End

Figure 5. Flowchart of the proposed mDBO algorithm.

3.5.1. Fitness function

The first step in FS involves defining an objective function, often referred to in the literature as a fitness function or cost function. FS aims to achieve a dual objective: Maximizing classification accuracy while minimizing the number of selected features. Given that FS tasks are typically framed as minimization problems, we utilize the classification error rate in the objective function instead of accuracy, as outlined by Cinar [58]. Accordingly, the objective function for this optimization model is expressed as follows:

$$Fitness = \alpha \cdot error + (1 - \alpha) \cdot \frac{N}{NP}, \tag{3.25}$$

where, we consider a weighting parameter $\alpha \in (0, 1)$, where α and $(1 - \alpha)$ denote their respective contributions. The term error refers to the error rate, defined as the ratio of misclassified instances to the total number of instances. This rate is evaluated using a learning classifier, such as KNN, SVM, or naive bayes. For our wrapper-based FS method, we employ the KNNs algorithm as the classification tool and utilize k-fold cross-validation to partition the dataset into training and testing sets. Let N represent the number of selected features, while NP denotes the total number of features in the original dataset. In this paper, α is 0.99.

3.5.2. KNNs

The KNN algorithm is a straightforward and efficient machine learning method widely utilized for classification tasks. It has demonstrated remarkable success across various fields, including text analysis, image recognition, audio analysis, and video processing. Due to its intuitive nature and ease of implementation, KNN is often the preferred choice for numerous data science projects. The algorithm's core mechanism involves searching the training dataset for the k closest data points to a new data point and classifying the new point based on the dominant class among these k neighbors. The performance of the KNN algorithm is influenced by several factors, particularly the choice of distance metric and the selection of the k value. Euclidean distance is the most commonly used metric and is calculated using a specific formula; however, other metrics can also be applied. The k value serves as a hyperparameter that determines the number of neighbors to consider when classifying new data points. The KNN classifier is frequently employed in the literature for classification tasks due to its simplicity, efficiency, and robustness to noisy data [59–61]. In practical applications, Euclidean distance and $k = 5$ are often used as default settings for KNN due to their relatively low computational complexity and ease of use.

$$d = \sqrt{\sum_{k=1}^{n} (y_j - y_{i,j})^2}, \tag{3.26}$$

d in Eq (3.26) represents the Euclidean distance.

3.5.3. Transfer function

The feature selection problem is fundamentally a discrete (binary) optimization task, which can be effectively addressed using discrete element algorithms to identify the optimal subset of features. In the context of intelligent algorithms, a transfer function is necessary to convert continuous solution values into binary representations. Two commonly employed transfer functions for binary mapping in the literature are the S-shaped and V-shaped transfer functions. In this study, we utilize the S-shaped transfer function proposed in [62], which aims to strike an effective balance between minimizing error rates and achieving rapid convergence of solutions. This transfer function is specifically defined in Eq (3.27).

$$Sigmoid(x_i^{t+1}) = \frac{1}{1 + exp(-3(x_i^{t+1} - 0.5))}, \tag{3.27}$$

x_i^{t+1} represents the position vector of the ith dung beetle at the $t + 1$th iteration. Consequently, the update of the position within the binary space is expressed as follows:

$$x_i^{t+1} = \begin{cases} 0 & , p < Sigmoid(x_i^{t+1}), \\ 1 & , otherwise, \end{cases} \tag{3.28}$$

where p represents the transition probability, which is defined within the interval (0,1).

3.6. Complexity analysis

3.6.1. Time complexity

The time complexity is a crucial metric for assessing algorithm performance and computational cost. The time complexity of DBO is $(O(f(D) \times D))$, where $f(D)$ represents the time required for evaluating the fitness function and D denotes the data dimension. The time complexity analysis for mDBO is as follows:

(1) Let N represent the number of dung beetle individuals, t_1 the time for initializing population parameters, and t_2 the time for generating chaotic values for each dimension using hybrid chaotic mapping. The time complexity O_1 is $O_1 = O(t_1 + N \times (f(D) + t_2 \times D))$.

(2) Let t_3 represent the time required to generate opposite solutions for each dimension. The time complexity O_2 for the random opposition-based learning method is $O_2 = O(N \times t_3 \times D)$.

(3) The position update formula for spawning dung beetles in the improved algorithm remains unchanged, so the time complexity O_3 is the same as that of the original algorithm.

(4) Let N_f represent the proportion of foraging dung beetles, and t_4 the time required to update the position for each dimension. The time complexity O_4 for the foraging dung beetle update method, incorporating the periodic retrospective adaptive differential evolution strategy, is $O_4 = O(N_f \times N \times t_4 \times D)$.

(5) Let N_t represent the proportion of Stealing dung beetles, and t_5 the time required to update the position for each dimension. The time complexity O_5 for the Stealing dung beetle update method, which utilizes the Padé approximation strategy and adaptive evolutionary boundary constraint strategy, is $O_5 = O(N_t \times N \times t_5 \times D)$.

Therefore, the total time complexity O' for mDBO is $O' = O_1 + \text{itermax} \times (O_2 + O_3 + O_4 + O_5) = O(f(D) \times D)$.

This shows that both mDBO and DBO share the same time complexity, indicating that the performance improvements achieved by mDBO do not incur additional computational cost in terms of time complexity.

3.6.2. Space complexity

Regarding space complexity, the memory requirements for mDBO depend on the dimension size and the number of search agents, which are determined during the initialization phase. Therefore, the space complexity of mDBO is $O(N \times D)$, where N is the population size and D is the dimensionality of the problem.

4. Experimental I: Solving benchmark functions

4.1. Benchmark functions

In this study, the performance of the mDBO algorithm is assessed by comparing it with eight benchmark algorithms, through several test functions for evaluation. These test functions are sourced from CEC2017 [63], with detailed descriptions provided in Table 2. The comparative algorithms include nine well-known metaheuristic algorithms: Whale optimizer (WOA) [64], sine cosine algorithm (SCA) [65], particle swarm optimizer (PSO) [66], GWO [67], Harris Hawks optimization (HHO) [68], arithmetic optimization algorithm (AOA) [69], butterfly optimization algorithm (BOA) [70], DE [48], DBO, LSHADE [71], and LSHADE_SPACMA [72]. Figure 6 presents the two-dimensional representations of several test functions employed in this study to evaluate the performance of the DBO algorithm. All experimental designs, including data preprocessing and analysis, are conducted using the 64-bit version of MATLAB 2023a. The experiments are performed on a system with an AMD Ryzen 5 processor, a CPU clock speed of 2.10 GHz, 8.00 GB of RAM, a 256 GB solid-state drive, and the Windows 10 Education operating system.

4.2. Strategy comparison

4.2.1. Chaos map comparison

To analyze the impact of different chaos maps on population distribution during the initialization phase, we employ several common chaos map methods [36], including the logistic map, the sine map, the tent map, the Singer map, the Chebyshev map, and the cubic map. A comparative analysis is conducted using scatter plots and population distribution maps. By examining the population distributions generated by each map, we assess their diversity and uniformity within the global search space.

Figures 7 and 8 illustrate the scatter plots and population distribution maps generated by each chaos map method, respectively. From the figures, it can be observed that different chaos maps have distinct effects on the population initialization phase. Both the sine and logistic maps show strong clustering in certain areas, which may result in the formation of concentrated regions during initialization, thereby reducing the diversity of the search space. This phenomenon could affect the early-stage search performance of optimization algorithms. The population distributions generated by the Singer, Chebyshev, and cubic maps appear more random with uneven density, showing concentrated or sparse areas in some regions. The tent map exhibits a relatively uniform distribution, effectively expanding across the search space and avoiding excessive concentration in any specific region. Relatively speaking, the tent map outperforms the others. Additionally, as discussed in Section 3.1, tent-sine offers advantages over both tent and sine in terms of population distribution and diversity.

Table 2. Benchmark test functions.

Type	No.	Function	F_{min}
Unimodal Functions	F1	Shifted and Rotated Bent Cigar Function	100
	F3	Shifted and Rotated Zakharov Function	300
Simple Multimodal Functions	F4	Shifted and Rotated Rosenbrock's Function	400
	F5	Shifted and Rotated Rastrigin's Function	500
	F6	Shifted and Rotated Expanded Scaffer's F6 Function	600
	F7	Shifted and Rotated Lunacek Bi_Rastrigin Function	700
	F8	Shifted and Rotated Non-Continuous Rastrigin's Function	800
	F9	Shifted and Rotated Levy Function	900
	F10	Shifted and Rotated Schwefel's Function	1000
Hybrid Functions	F11	Hybrid Function 1 (N=3)	1100
	F12	Hybrid Function 2 (N = 3)	1200
	F13	Hybrid Function 3 (N = 3)	1300
	F14	Hybrid Function 4 (N = 4)	1400
	F15	Hybrid Function 5 (N = 4)	1500
	F16	Hybrid Function 6 (N = 4)	1600
	F17	Hybrid Function 7 (N = 5)	1700
	F18	Hybrid Function 8 (N = 5)	1800
	F19	Hybrid Function 9 (N = 5)	1900
	F20	Hybrid Function 10 (N=6)	2000
Composition Functions	F21	Composition Function 1 (N=3)	2100
	F22	Composition Function 2 (N = 3)	2200
	F23	Composition Function 3 (N = 4)	2300
	F24	Composition Function 4 (N = 4)	2400
	F25	Composition Function 5 (N = 5)	2500
	F26	Composition Function 6 (N = 5)	2600
	F27	Composition Function 7 (N = 6)	2700
	F87	Composition Function 8 (N = 6)	2800
	F29	Composition Function 9 (N = 3)	2900
	F30	Composition Function 10 (N = 3)	3000
Search range: $[-100, 100]^D$			

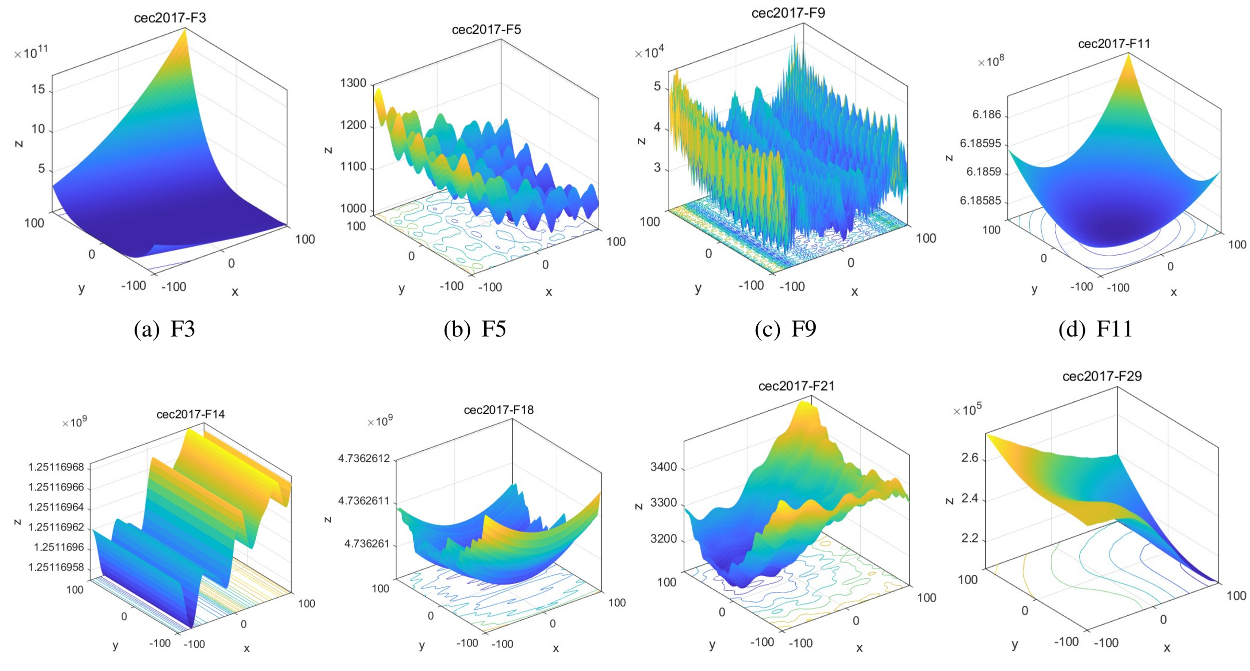
4.2.2. A comparison of ROBL, QRBL, and DOL

To validate the effectiveness of ROBL, we conduct comparative experiments between ROBL and otherOBL strategies, such as quasi-oppositional learning (QRBL) [73] and dynamic opposite learning (DOL) [74]. The experimental results are shown in Tables 3 and 4 and Figure 9.

As shown in Tables 3 and 4, and Figure 9, the ROBL-based algorithm outperforms other algorithms for most of the 29 benchmark functions and achieves the overall first rank, further validating the performance advantages of ROBL.

4.2.3. Comparison of differential mutation factors

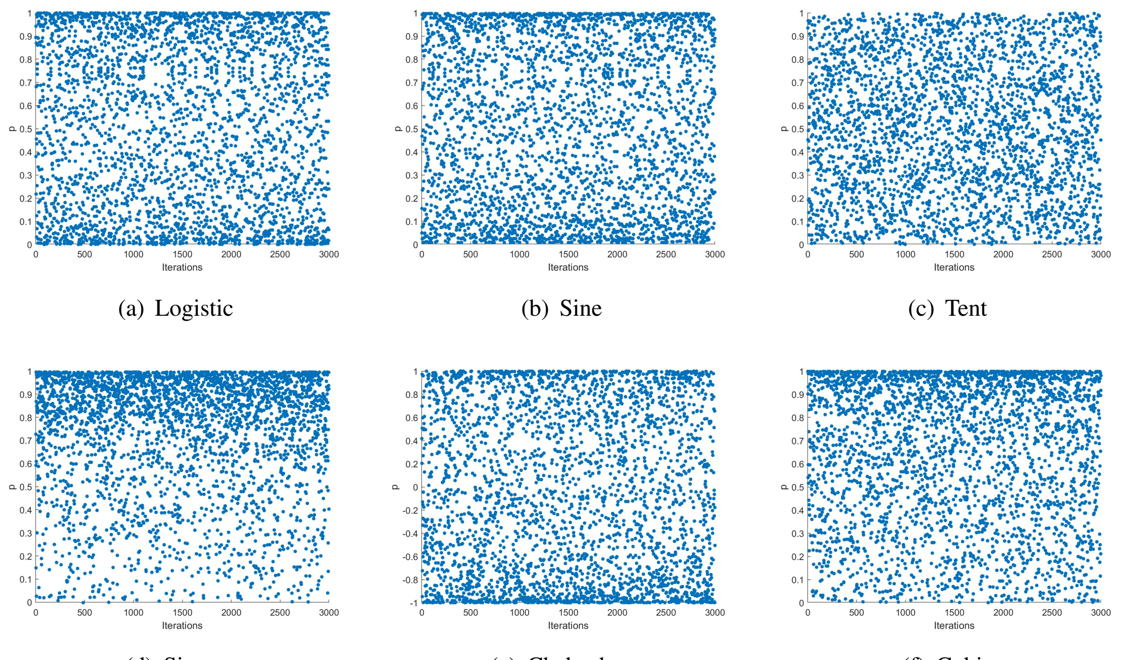
Inspired by the inertia weight in Eq (3.7) [49], two improvements were proposed based on it: one is the adaptive parameter H in Eq (3.9), and the other is periodic retrospective factor in Eq (3.8). A periodic retrospective adaptive factor was constructed, and its effectiveness was verified through experimental comparisons.



(a) F3 (b) F5 (c) F9 (d) F11

(e) F14 (f) F18 (g) F21 (h) F29

Figure 6. 2-D Versions of benchmark test functions.



(a) Logistic (b) Sine (c) Tent

(d) Singer (e) Chebyshev (f) Cubic

Figure 7. Chaos mapping scatter plot.

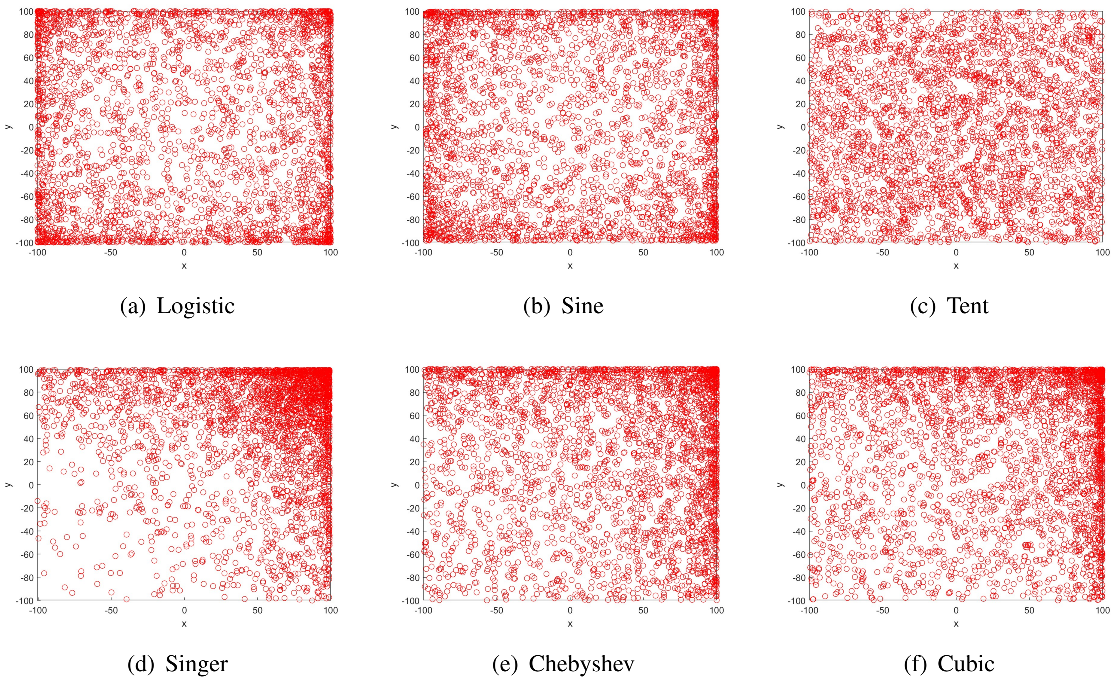


Figure 8. Initial population distribution chart.

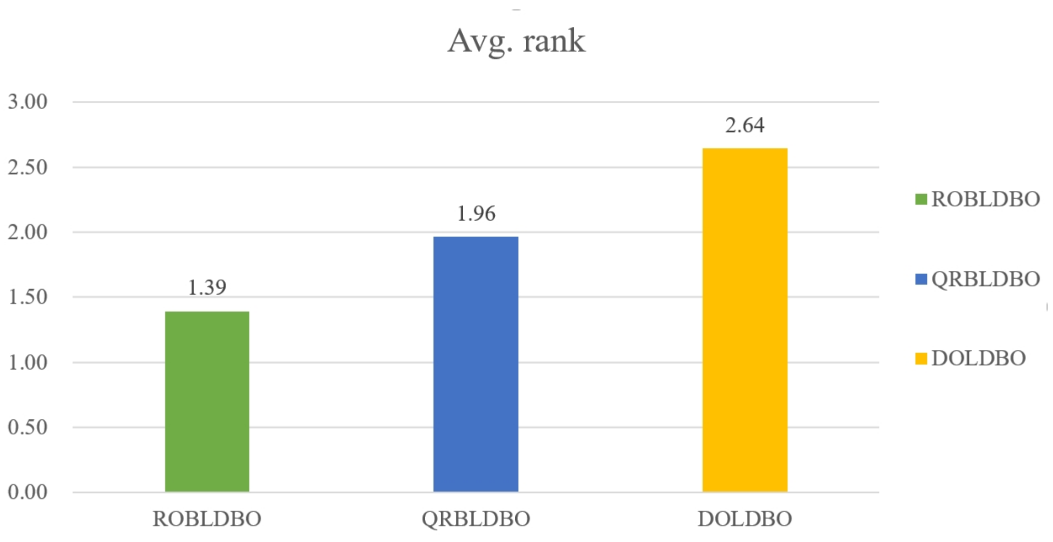


Figure 9. Comparison chart of different opposition learning.

As shown in Table 5 and Figure 10, PRADBO represents the algorithm using the periodic retrospective adaptive factor from Eq (3.8), while ADBO refers to the algorithm using the factor from Eq (3.7). the algorithm using the periodic retrospective adaptive strategy outperforms most functions across the 29 benchmark problems, ranking first overall. This further validates the performance advantage of the periodic retrospective adaptive strategy and demonstrates that the periodic retrospective adaptive factor F is more effective than the factor in Eq (3.7).

Table 3. Comparison of different opposition learning.

Function	Metric	ROBLDBO	QRBLDBO	DOLDBO
F1	mean	1.5924×10^{4}	9.9494×10^{5}	4.0538×10^{6}
	std	1.6743×10^{9}	2.7665×10^{13}	2.3891×10^{14}
		1	2	3
F3	mean	3.2499×10^{2}	4.1929×10^{2}	9.2207×10^{2}
	std	2.7121×10^{3}	1.1578×10^{5}	9.9985×10^{5}
		1	2	3
F4	mean	4.3317×10^{2}	4.2280×10^{2}	4.4067×10^{2}
	std	2.8018×10^{3}	1.1873×10^{3}	2.5965×10^{3}
		2	1	3
F5	mean	5.3655×10^{2}	5.3533×10^{2}	5.4405×10^{2}
	std	2.2982×10^{2}	1.4090×10^{2}	1.1116×10^{2}
		2	1	3
F6	mean	6.0770×10^{2}	6.1055×10^{2}	6.1192×10^{2}
	std	3.4715×10^{1}	5.4999×10^{1}	9.1900×10^{1}
		1	2	3
F7	mean	7.4543×10^{2}	7.5055×10^{2}	7.4811×10^{2}
	std	2.9523×10^{2}	2.8009×10^{2}	3.2386×10^{2}
		1	3	2
F8	mean	8.2588×10^{2}	8.2712×10^{2}	8.3338×10^{2}
	std	6.5180×10^{1}	8.7109×10^{1}	1.9092×10^{2}
		1	2	3
F9	mean	1.0156×10^{3}	9.7525×10^{2}	9.8535×10^{2}
	std	2.1721×10^{4}	1.1966×10^{4}	1.9094×10^{4}
		3	1	2
F10	mean	1.8227×10^{3}	2.0425×10^{3}	1.9484×10^{3}
	std	9.0637×10^{4}	1.2731×10^{5}	1.0769×10^{5}
		1	3	2
F11	mean	1.1799×10^{3}	1.2052×10^{3}	1.2699×10^{3}
	std	6.8788×10^{3}	8.5462×10^{3}	2.3557×10^{4}
		1	2	3
F12	mean	1.3000×10^{6}	2.0038×10^{6}	4.3396×10^{6}
	std	1.1759×10^{13}	3.5220×10^{13}	3.6814×10^{13}
		1	2	3
F13	mean	1.8648×10^{4}	1.1046×10^{4}	1.8912×10^{4}
	std	3.6610×10^{8}	7.4937×10^{7}	1.8283×10^{8}
		2	1	3
F14	mean	1.9709×10^{3}	2.2999×10^{3}	2.2619×10^{3}
	std	4.2683×10^{5}	1.5684×10^{6}	7.6642×10^{5}
		1	3	2
F15	mean	2.9825×10^{3}	3.4918×10^{3}	9.3039×10^{3}
	std	5.3144×10^{6}	4.2138×10^{6}	1.1433×10^{8}
		1	2	3
F16	mean	1.7870×10^{3}	1.8370×10^{3}	1.7796×10^{3}
	std	2.3363×10^{4}	1.6778×10^{4}	2.5137×10^{4}
		2	3	1
F17	mean	1.7710×10^{3}	1.7644×10^{3}	1.7760×10^{3}
	std	9.4105×10^{2}	1.1195×10^{3}	1.5521×10^{3}
		2	1	3

Table 4. Comparison of different opposition learning.

Function	Metric	ROBLDBO	QRBLDBO	DOLDBO
F18	mean	1.8646×10^4	2.1905×10^4	3.2793×10^4
	std	1.8612×10^8	2.7939×10^8	3.3732×10^8
		1	2	3
F19	mean	9.5766×10^3	1.1649×10^4	3.0471×10^4
	std	2.4385×10^8	3.0868×10^8	4.7091×10^9
		1	2	3
F20	mean	2.1018×10^3	2.1262×10^3	2.1255×10^3
	std	3.1826×10^3	4.2548×10^3	5.4996×10^3
		1	3	2
F21	mean	2.2256×10^3	2.2422×10^3	2.2340×10^3
	std	2.2740×10^3	3.1389×10^3	2.5703×10^3
		1	3	2
F22	mean	2.3116×10^3	2.3087×10^3	2.3052×10^3
	std	6.4436×10^1	2.7428×10^1	4.1417×10^2
		3	2	1
F23	mean	2.6492×10^3	2.6512×10^3	2.6523×10^3
	std	3.9880×10^2	3.2423×10^2	5.3914×10^2
		1	2	3
F24	mean	2.7086×10^3	2.6688×10^3	2.7200×10^3
	std	1.6068×10^4	1.5899×10^4	9.2810×10^3
		2	1	3
F25	mean	2.9308×10^3	2.9389×10^3	2.9440×10^3
	std	5.9283×10^2	3.4060×10^2	6.7638×10^2
		1	2	3
F26	mean	2.9904×10^3	3.0686×10^3	3.1638×10^3
	std	2.8680×10^4	9.1005×10^4	8.7369×10^4
		1	2	3
F27	mean	3.1144×10^3	3.1105×10^3	3.1163×10^3
	std	3.5914×10^2	5.2002×10^2	7.3854×10^2
		2	1	3
F28	mean	3.3250×10^3	3.3343×10^3	3.4048×10^3
	std	9.1794×10^3	1.0625×10^4	2.1199×10^4
		1	2	3
F29	mean	3.2446×10^3	3.2617×10^3	3.2715×10^3
	std	3.9858×10^3	3.4351×10^3	7.8141×10^3
		1	2	3
F30	mean	5.3148×10^5	6.1626×10^5	7.1576×10^5
	std	6.7176×10^{11}	4.3819×10^{11}	7.3114×10^{11}
		1	2	3
Avg. rank		1.39	1.96	2.64
Overall rank		1	2	3

Table 5. Comparison of differential mutation factors.

Function	Metric	PRADBO	ADBO	Function	Metric	PRADBO	ADBO
F1	mean	4.2589×10^3	5.8979×10^3	F17	mean	1.7884×10^3	1.7715×10^3
	std	1.7527×10^7	1.7585×10^7		std	2.2323×10^3	1.4012×10^3
		1	2			2	1
F3	mean	7.3854×10^2	1.1037×10^3	F18	mean	1.5027×10^4	1.5917×10^4
	std	3.4461×10^6	6.5719×10^6		std	2.8351×10^8	1.6650×10^8
		1	2			1	2
F4	mean	4.1973×10^2	4.2138×10^2	F19	mean	3.6553×10^3	4.7179×10^3
	std	1.3111×10^3	9.3775×10^2		std	1.8244×10^7	1.9666×10^7
		1	2			1	2
F5	mean	5.3305×10^2	5.3433×10^2	F20	mean	2.0968×10^3	2.0778×10^3
	std	1.7758×10^2	1.8578×10^2		std	5.1168×10^3	2.6363×10^3
		1	2			2	1
F6	mean	6.0682×10^2	6.0810×10^2	F21	mean	2.2703×10^3	2.2804×10^3
	std	3.4087×10^1	7.4083×10^1		std	3.9652×10^3	4.6421×10^3
		1	2			1	2
F7	mean	7.4371×10^2	7.4584×10^2	F22	mean	2.2998×10^3	2.3090×10^3
	std	3.2103×10^2	1.8591×10^2		std	3.9630×10^2	1.7782×10^2
		1	2			1	2
F8	mean	8.2915×10^2	8.2851×10^2	F23	mean	2.6334×10^3	2.6342×10^3
	std	1.4096×10^2	1.7518×10^2		std	1.3139×10^2	1.5788×10^2
		2	1			1	2
F9	mean	9.4522×10^2	9.9008×10^2	F24	mean	2.7123×10^3	2.7570×10^3
	std	2.4964×10^3	1.9269×10^4		std	1.0128×10^4	3.3880×10^3
		1	2			1	2
F10	mean	1.9590×10^3	1.9414×10^3	F25	mean	2.9395×10^3	2.9396×10^3
	std	8.9112×10^4	9.9167×10^4		std	1.1497×10^3	6.7220×10^2
		2	1			1	2
F11	mean	1.1391×10^3	1.1732×10^3	F26	mean	3.1210×10^3	3.1328×10^3
	std	1.9566×10^3	1.0478×10^4		std	7.7317×10^4	1.2599×10^5
		1	2			1	2
F12	mean	2.0206×10^4	2.9191×10^5	F27	mean	3.1019×10^3	3.0997×10^3
	std	3.3909×10^8	2.2240×10^{12}		std	6.4453×10^1	5.0040×10^1
		1	2			2	1
F13	mean	9.5628×10^3	1.5008×10^4	F28	mean	3.3113×10^3	3.3242×10^3
	std	1.0601×10^8	1.6210×10^8		std	9.1137×10^3	7.6950×10^3
		1	2			1	2
F14	mean	1.7370×10^3	1.7657×10^3	F29	mean	3.2339×10^3	3.2557×10^3
	std	2.5852×10^5	2.6291×10^5		std	4.1540×10^3	4.0233×10^3
		1	2			1	2
F15	mean	3.6298×10^3	3.3469×10^3	F30	mean	4.4287×10^5	3.9121×10^5
	std	9.5706×10^6	6.0534×10^6		std	3.1382×10^{11}	2.8234×10^{11}
		2	1			2	1
F16	mean	1.7476×10^3	1.7380×10^3	Avg. rank		1.28	1.72
	std	1.0704×10^4	7.0219×10^3	Overall rank		1	2
		2	1				

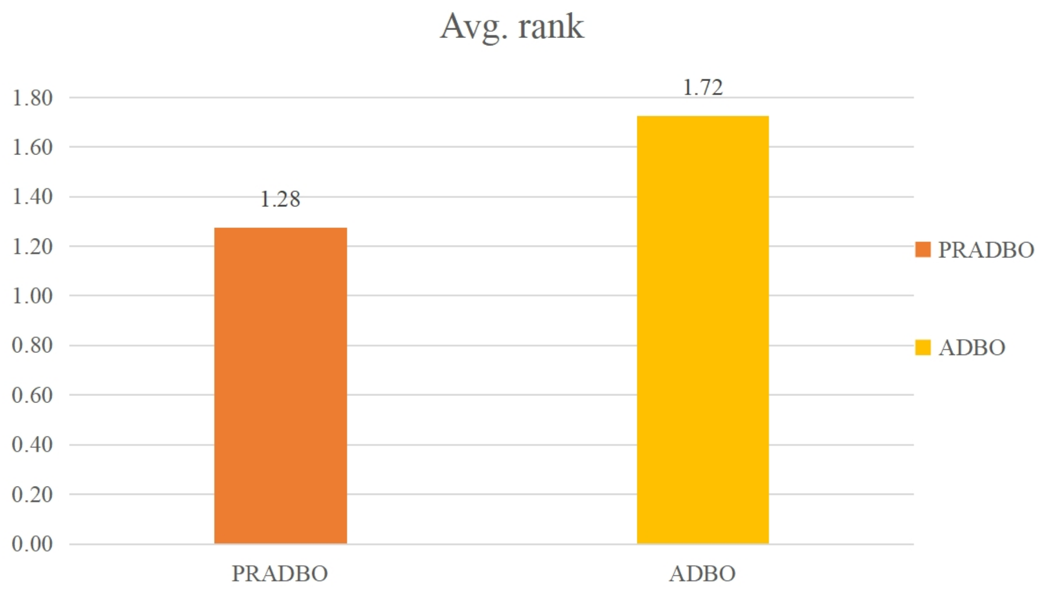


Figure 10. The comparison of parameter F.

4.3. Parameter settings

The parameter settings for the comparative algorithms are presented in Table 6. To ensure fairness in the experiments, an initial population size of 30 and a maximum of 500 iterations were used for all algorithms. To mitigate any experimental bias, the evaluation criteria include the average and standard deviation of the solution results, with each of the eight comparative algorithms and the mDBO algorithm being independently executed 30 times on each test function.

Table 6. Parameter values of algorithms.

Algorithm	Parameters values
WOA	$a_2 \in (0, 2), r_3 \in [0, 1], l \in [-1, 1]$
SCA	$\alpha = 2$
PSO	$V = (-6, 6), C_1 = C_2 = 2, w = (0.2, 0.9)$
GWO	α is decreasing linearly from 2 to 0
HHO	$q \in [0, 1], r_4.r_5, r_6, r_7 \in [0, 1], E_0 \in [-1, 1], u_1, v_1[0, 1]$
DBO	$k = 0.1, \lambda = 0.1, b = 0.3, S = 0.5$
AOA	$\alpha = 5, \mu = 0.499$
BOA	$c = 0.01, p = 0.8$
DE	$F = 0.5, CR = 0.8$
LSHADE	$NP_{init} = 18, D, NP_{min} = 4, P = 0.11, \|A\| = 2.6, H = 6$
LSHADE_SPACMA	$NP_{init} = 18, D, NP_{min} = 4, P = 0.11, \|A\| = 1.4, H = 5, \delta = 0.5, L\_rate = 0.8$

The proposed mDBO algorithm features three key control parameters that must be optimally tuned to achieve peak performance: F_{min}, CR, and k. Parameter F_{min} ensures that the mutation factor F maintains a minimum value, thereby preventing the algorithm from becoming overly conservative during the search process and avoiding weak mutation operations that could lead to premature convergence on local optima. The crossover probability CR in DE ranges from 0 to 1 and is essential for enhancing the search accuracy of mDBO. The scaling factor k is critical for obtaining an optimal P that adjusts the boundaries, thus enabling a broader set of solutions. We performed extensive experiments across ranges for F_{min}, CR, and k. The results of these experiments, conducted over 30 independent runs of the mDBO algorithm on 30 ten-dimensional CEC2017 functions, are presented in Tables 7–12. The CEC2017 suite comprises 29 single-objective test functions, classified as follows: F1 and F3 are unimodal functions; F4 to F10 are simple multimodal functions; F11 to F20 are mixed-type functions; and F21 to F30 are composite functions. Due to uncontrollable factors encountered during the experiments, we were unable to obtain experimental data for function F2, and consequently, no experimental research was conducted for this function.

Table 7. Adjusting the parameters F_{min} , CR, and k of mDBO.

F	Type	(CR, k, F_{min})					
		(0.8,3,0)	(0.8,3,0.1)	(0.8,3,0.2)	(0.8,4,0)	(0.8,4,0.1)	(0.8,4,0.2)
F1	mean	4.9010×10^3	4.1102×10^3	5.1503×10^3	4.8420×10^3	5.3186×10^3	3.5626×10^3
	std	1.5142×10^7	1.8438×10^7	2.2259×10^7	1.3613×10^7	1.7863×10^7	9.9009×10^6
		8	4	9	6	10	2
F3	mean	4.0209×10^2	3.2982×10^2	6.6577×10^2	3.8963×10^2	3.1670×10^2	8.6298×10^2
	std	1.7302×10^5	1.0178×10^4	3.7558×10^6	1.5934×10^5	3.0988×10^3	6.7839×10^6
		6	2	10	5	1	11
F4	mean	4.1373×10^2	4.1244×10^2	4.1310×10^2	4.0769×10^2	4.1326×10^2	4.1151×10^2
	std	5.3282×10^2	4.3227×10^2	5.1582×10^2	1.9435×10^2	7.4818×10^2	4.2163×10^2
		10	7	8	1	9	4
F5	mean	5.2894×10^2	5.3031×10^2	5.3054×10^2	5.3033×10^2	5.3346×10^2	5.2703×10^2
	std	1.2902×10^2	1.3010×10^2	1.4780×10^2	2.4325×10^2	1.6903×10^2	1.6419×10^2
		4	6	8	7	12	2
F6	mean	6.0284×10^2	6.0355×10^2	6.0325×10^2	6.0429×10^2	6.0235×10^2	6.0378×10^2
	std	1.6401×10^1	1.4362×10^1	1.6775×10^1	2.9547×10^1	1.1610×10^1	2.1280×10^1
		5	7	6	11	2	8
F7	mean	7.3796×10^2	7.3905×10^2	7.3434×10^2	7.4027×10^2	7.3694×10^2	7.4219×10^2
	std	1.2700×10^2	1.3549×10^2	1.4201×10^2	1.0922×10^2	1.1659×10^2	1.0035×10^2
		7	8	1	10	5	12
F8	mean	8.2607×10^2	8.2505×10^2	8.2363×10^2	8.2331×10^2	8.2540×10^2	8.2614×10^2
	std	3.8558×10^1	4.8858×10^1	7.5256×10^1	6.2295×10^1	6.5463×10^1	1.0150×10^2
		11	9	6	5	10	12
F9	mean	9.6687×10^2	9.3606×10^2	9.4504×10^2	9.4908×10^2	9.4750×10^2	9.6682×10^2
	std	1.1345×10^4	4.4185×10^3	6.4358×10^3	6.9472×10^3	4.7380×10^3	8.5638×10^3
		11	2	5	8	7	10
F10	mean	1.7389×10^3	1.9758×10^3	1.8798×10^3	1.9683×10^3	1.9484×10^3	1.8741×10^3
	std	7.2374×10^4	6.5120×10^4	9.4515×10^4	1.0553×10^5	8.0215×10^4	1.1089×10^5
		1	12	3	11	9	2
F11	mean	1.1260×10^3	1.1289×10^3	1.1511×10^3	1.1477×10^3	1.1408×10^3	1.1533×10^3
	std	5.4787×10^2	3.6480×10^2	7.0745×10^3	6.5061×10^3	4.1637×10^3	7.8085×10^3
		1	2	9	7	4	11
F12	mean	2.9188×10^5	5.8531×10^5	4.5276×10^5	1.3670×10^4	1.3601×10^5	3.4648×10^5
	std	2.2313×10^{12}	4.2809×10^{12}	2.9498×10^{12}	6.4613×10^7	4.0679×10^{11}	1.5830×10^{12}
		6	12	9	1	3	7
F13	mean	1.2472×10^4	1.1221×10^4	1.2327×10^4	1.1920×10^4	1.0586×10^4	1.0606×10^4
	std	1.1973×10^8	6.2646×10^7	1.3813×10^8	1.0670×10^8	7.0414×10^7	9.0204×10^7
		9	5	8	7	3	4
F14	mean	1.6701×10^3	1.7099×10^3	1.6306×10^3	1.6806×10^3	1.7091×10^3	1.8318×10^3
	std	1.4177×10^5	1.9805×10^5	2.9736×10^4	1.4740×10^5	1.8908×10^5	3.7225×10^5
		4	8	1	5	7	10

The experimental results are shown in Tables 7–12. From these tables, it is evident that when the hyperparameters F_{min} are set to 0, CR is set to 0.8, and k is set to 4, the mDBO algorithm performs excellently across several CEC2017 benchmark functions, particularly achieving optimal mean values on functions F4, F12, F20, F21, F23, F24, F25, F27, and F28. These results demonstrate that, under

Table 8. Adjusting the parameters F_{min} , CR, and k of mDBO.

F	Type	(CR, k, F_{min})					
		(0.8,3,0)	(0.8,3,0.1)	(0.8,3,0.2)	(0.8,4,0)	(0.8,4,0.1)	(0.8,4,0.2)
F15	mean	2.8206×10^3	3.6982×10^3	2.1200×10^3	3.0343×10^3	2.5797×10^3	2.7264×10^3
	std	3.4755×10^6	3.1989×10^7	2.3318×10^5	7.3910×10^6	1.2672×10^6	4.9043×10^6
		10	12	1	11	4	8
F16	mean	1.7345×10^3	1.7468×10^3	1.7655×10^3	1.7616×10^3	1.7816×10^3	1.7896×10^3
	std	1.4273×10^4	1.1863×10^4	1.9078×10^4	1.7870×10^4	1.8373×10^4	1.8536×10^4
		1	3	7	6	10	11
F17	mean	1.7622×10^3	1.7578×10^3	1.7488×10^3	1.7596×10^3	1.7631×10^3	1.7595×10^3
	std	1.1455×10^3	1.2316×10^3	4.1359×10^2	1.2448×10^3	9.4164×10^2	1.1788×10^3
		11	5	1	9	12	8
F18	mean	1.6053×10^4	1.6845×10^4	1.9842×10^4	1.3271×10^4	1.5547×10^4	1.3046×10^4
	std	1.8348×10^8	1.9570×10^8	2.5641×10^8	1.1385×10^8	2.0925×10^8	1.6785×10^8
		7	9	12	2	6	1
F19	mean	4.3128×10^3	6.6362×10^3	4.2931×10^3	5.4589×10^3	4.9344×10^3	5.5921×10^3
	std	1.2468×10^7	5.2119×10^7	6.1751×10^6	2.4042×10^7	2.3095×10^7	3.5445×10^7
		5	12	4	9	8	10
F20	mean	2.0987×10^3	2.0824×10^3	2.0725×10^3	2.0660×10^3	2.0881×10^3	2.0923×10^3
	std	4.6594×10^3	3.3644×10^3	3.4166×10^3	2.2940×10^3	4.7326×10^3	4.8410×10^3
		11	7	4	1	8	10
F21	mean	2.2911×10^3	2.2789×10^3	2.2848×10^3	2.2582×10^3	2.2696×10^3	2.2621×10^3
	std	3.8594×10^3	3.8584×10^3	4.0228×10^3	3.8567×10^3	4.1166×10^3	4.7402×10^3
		11	6	8	1	3	2
F22	mean	2.3052×10^3	2.3057×10^3	2.3052×10^3	2.3080×10^3	2.3067×10^3	2.3069×10^3
	std	1.3638×10^1	1.1900×10^2	5.5049×10^1	8.7597×10^1	8.1196×10^1	2.7246×10^1
		4	7	5	12	9	10
F23	mean	2.6370×10^3	2.6413×10^3	2.6362×10^3	2.6356×10^3	2.6384×10^3	2.6405×10^3
	std	1.4575×10^2	2.9394×10^2	1.5844×10^2	1.5494×10^2	1.7241×10^2	2.3786×10^2
		3	12	2	1	6	10
F24	mean	2.7399×10^3	2.7290×10^3	2.7329×10^3	2.6935×10^3	2.7560×10^3	2.7142×10^3
	std	7.2952×10^3	9.6582×10^3	6.7682×10^3	1.3558×10^4	3.5176×10^3	1.0560×10^4
		10	6	7	1	12	2
F25	mean	2.9381×10^3	2.9318×10^3	2.9219×10^3	2.9100×10^3	2.9354×10^3	2.9383×10^3
	std	5.7007×10^2	8.5526×10^2	4.1908×10^3	7.5092×10^3	4.4811×10^2	7.6467×10^2
		9	5	3	1	8	10
F26	mean	3.0195×10^3	3.0533×10^3	3.0041×10^3	3.0036×10^3	2.9931×10^3	3.0236×10^3
	std	6.2962×10^4	2.2488×10^4	1.9532×10^4	1.6141×10^4	1.1247×10^4	2.3375×10^4
		10	12	6	5	3	11
F27	mean	3.1013×10^3	3.1004×10^3	3.1024×10^3	3.0989×10^3	3.1012×10^3	3.1020×10^3
	std	8.9415×10^1	4.0949×10^1	1.7357×10^2	3.8199×10^1	5.9927×10^1	4.4907×10^1
		7	5	10	1	6	9

Table 9. Adjusting the parameters F_{min} , CR, and k of mDBO.

F	Type	(CR, k, F_{min})					
		(0.8,3,0)	(0.8,3,0.1)	(0.8,3,0.2)	(0.8,4,0)	(0.8,4,0.1)	(0.8,4,0.2)
F28	mean	3.2943×10^3	3.3165×10^3	3.2900×10^3	3.2849×10^3	3.3099×10^3	3.3241×10^3
	std	1.5826×10^4	9.2487×10^3	9.5967×10^3	8.7752×10^3	1.0568×10^4	1.4911×10^4
		5	9	4	1	8	11
F29	mean	3.2358×10^3	3.2719×10^3	3.2690×10^3	3.2304×10^3	3.2404×10^3	3.2141×10^3
	std	4.2795×10^3	4.6953×10^3	5.3692×10^3	3.1722×10^3	3.8359×10^3	2.5371×10^3
		5	12	11	3	6	1
F30	mean	5.5342×10^5	3.5688×10^5	5.5578×10^5	5.4300×10^5	7.0202×10^5	4.8740×10^5
	std	1.2234×10^{12}	2.5686×10^{11}	2.6883×10^{11}	3.8727×10^{11}	1.3000×10^{12}	7.0985×10^{11}
		9	2	10	7	12	5

Table 10. Adjusting the parameters F_{min}, CR, and k of mDBO.

F	Type	(CR, k, F_{min})					
		(0.9,3,0)	(0.9,3,0.1)	(0.9,3,0.2)	(0.9,4,0)	(0.9,4,0.1)	(0.9,4,0.2)
F1	mean	3.4989×10^3	4.1101×10^3	4.5607×10^3	5.7720×10^3	5.4229×10^3	4.8554×10^3
	std	1.3572×10^7	1.6736×10^7	1.5480×10^7	1.7715×10^7	2.1267×10^7	2.2801×10^7
		1	3	5	12	11	7
F3	mean	3.7994×10^2	9.9641×10^2	6.1714×10^2	4.3509×10^2	3.6824×10^2	5.9219×10^2
	std	1.6674×10^5	1.2894×10^7	2.4640×10^6	4.0009×10^5	6.3014×10^4	2.3777×10^6
		4	12	9	7	3	8
F4	mean	4.1103×10^2	4.1198×10^2	4.1210×10^2	4.1392×10^2	4.0780×10^2	4.1643×10^2
	std	3.6379×10^2	3.7241×10^2	3.5862×10^2	4.9825×10^2	1.0013×10^2	7.5598×10^2
		3	5	6	11	2	12
F5	mean	5.3075×10^2	5.2525×10^2	5.3118×10^2	5.2837×10^2	5.3190×10^2	5.2993×10^2
	std	1.6232×10^2	1.3597×10^2	1.9518×10^2	1.5200×10^2	2.6645×10^2	1.5253×10^2
		9	1	10	3	11	5
F6	mean	6.0206×10^2	6.0253×10^2	6.0400×10^2	6.0399×10^2	6.0454×10^2	6.0276×10^2
	std	1.2052×10^1	1.2469×10^1	3.6164×10^1	4.9816×10^1	3.3860×10^1	2.3116×10^1
		1	3	10	9	12	4
F7	mean	7.3607×10^2	7.3634×10^2	7.3943×10^2	7.3768×10^2	7.3648×10^2	7.4143×10^2
	std	1.8249×10^2	1.1958×10^2	1.7408×10^2	9.0678×10^1	1.0949×10^2	1.4135×10^2
		2	3	9	6	4	11
F8	mean	8.2262×10^2	8.2408×10^2	8.2409×10^2	8.2224×10^2	8.2124×10^2	8.2175×10^2
	std	9.5029×10^1	9.6003×10^1	6.5796×10^1	9.2580×10^1	5.1979×10^1	7.4384×10^1
		4	7	8	3	1	2
F9	mean	9.4512×10^2	9.3661×10^2	9.2880×10^2	9.6728×10^2	9.3758×10^2	9.4992×10^2
	std	4.5627×10^3	2.6105×10^3	1.5545×10^3	1.1691×10^4	3.7155×10^3	3.6086×10^3
		6	3	1	12	4	9
F10	mean	1.9001×10^3	1.9079×10^3	1.9024×10^3	1.8839×10^3	1.9551×10^3	1.9139×10^3
	std	8.8106×10^4	1.3627×10^5	6.4491×10^4	9.2317×10^4	1.0434×10^5	1.0628×10^5
		5	7	6	4	10	8
F11	mean	1.1512×10^3	1.1455×10^3	1.1371×10^3	1.1549×10^3	1.1480×10^3	1.1461×10^3
	std	9.1894×10^3	5.4094×10^3	3.1273×10^3	8.5569×10^3	5.0966×10^3	4.7523×10^3
		10	5	3	12	8	6
F12	mean	1.7937×10^5	4.5228×10^5	5.6505×10^5	1.6459×10^4	2.8660×10^5	5.6291×10^5
	std	8.2720×10^{11}	2.9466×10^{12}	4.3237×10^{12}	4.0577×10^8	2.2290×10^{12}	4.3257×10^{12}
		4	8	11	2	5	10
F13	mean	1.0253×10^4	1.6473×10^4	1.4550×10^4	9.0526×10^3	1.1572×10^4	1.3416×10^4
	std	9.3341×10^7	1.3159×10^8	1.2072×10^8	6.9535×10^7	1.0518×10^8	1.1915×10^8
		2	12	11	1	6	10
F14	mean	1.6826×10^3	1.8362×10^3	1.7284×10^3	1.6626×10^3	1.8409×10^3	1.6431×10^3
	std	1.6708×10^5	2.6464×10^5	1.3914×10^5	5.0269×10^4	3.4653×10^5	1.3818×10^5
		6	11	9	3	12	2

Table 11. Adjusting the parameters F_{min}, CR, and k of mDBO.

F	Type	(CR, k, F_{min})					
		(0.9,3,0)	(0.9,3,0.1)	(0.9,3,0.2)	(0.9,4,0)	(0.9,4,0.1)	(0.9,4,0.2)
F15	mean	2.8049×10^{3}	2.7162×10^{3}	2.3915×10^{3}	2.7150×10^{3}	2.4522×10^{3}	2.6008×10^{3}
	std	1.5816×10^{6}	2.8779×10^{6}	1.9089×10^{6}	1.6134×10^{6}	1.0041×10^{6}	1.5656×10^{6}
		9	7	2	6	3	5
F16	mean	1.7702×10^{3}	1.7446×10^{3}	1.7978×10^{3}	1.7601×10^{3}	1.7805×10^{3}	1.7469×10^{3}
	std	1.9485×10^{4}	1.5957×10^{4}	1.4430×10^{4}	1.2963×10^{4}	1.9354×10^{4}	1.4892×10^{4}
		8	2	12	5	9	4
F17	mean	1.7602×10^{3}	1.7590×10^{3}	1.7548×10^{3}	1.7593×10^{3}	1.7569×10^{3}	1.7490×10^{3}
	std	1.2502×10^{3}	1.3058×10^{3}	9.9834×10^{2}	8.6165×10^{2}	7.4687×10^{2}	7.4711×10^{2}
		10	6	3	7	4	2
F18	mean	1.4326×10^{4}	1.7434×10^{4}	1.4171×10^{4}	1.7387×10^{4}	1.6363×10^{4}	1.3956×10^{4}
	std	2.4093×10^{8}	1.7412×10^{8}	1.4572×10^{8}	2.4286×10^{8}	1.9160×10^{8}	1.2474×10^{8}
		5	11	4	10	8	3
F19	mean	3.5826×10^{3}	6.1194×10^{3}	4.4673×10^{3}	4.8179×10^{3}	3.9334×10^{3}	4.0505×10^{3}
	std	6.5887×10^{6}	4.6357×10^{7}	1.1556×10^{7}	2.3329×10^{7}	1.1613×10^{7}	1.7045×10^{7}
		1	11	6	7	2	3
F20	mean	2.0691×10^{3}	2.0691×10^{3}	2.0903×10^{3}	2.0742×10^{3}	2.1006×10^{3}	2.0756×10^{3}
	std	3.4469×10^{3}	2.7723×10^{3}	3.3432×10^{3}	4.0442×10^{3}	5.5333×10^{3}	2.5197×10^{3}
		2	3	9	5	12	6
F21	mean	2.2776×10^{3}	2.2701×10^{3}	2.2811×10^{3}	2.2848×10^{3}	2.2929×10^{3}	2.2855×10^{3}
	std	4.2217×10^{3}	4.5235×10^{3}	3.8789×10^{3}	3.8412×10^{3}	3.6058×10^{3}	4.2232×10^{3}
		5	4	7	9	12	10
F22	mean	2.3047×10^{3}	2.3030×10^{3}	2.3053×10^{3}	2.2970×10^{3}	2.3058×10^{3}	2.3069×10^{3}
	std	5.3247×10^{1}	1.3414×10^{2}	1.6501×10^{1}	5.8811×10^{2}	1.8973×10^{1}	9.2737×10^{1}
		3	2	6	1	8	11
F23	mean	2.6412×10^{3}	2.6401×10^{3}	2.6394×10^{3}	2.6395×10^{3}	2.6384×10^{3}	2.6373×10^{3}
	std	3.0875×10^{2}	2.4657×10^{2}	1.3591×10^{2}	2.5869×10^{2}	3.3210×10^{2}	1.6076×10^{2}
		11	9	7	8	5	4
F24	mean	2.7213×10^{3}	2.7482×10^{3}	2.7233×10^{3}	2.7282×10^{3}	2.7361×10^{3}	2.7358×10^{3}
	std	9.4670×10^{3}	6.1684×10^{3}	1.0491×10^{4}	9.6473×10^{3}	8.0617×10^{3}	7.5765×10^{3}
		3	11	4	5	9	8
F25	mean	2.9434×10^{3}	2.9324×10^{3}	2.9322×10^{3}	2.9225×10^{3}	2.9147×10^{3}	2.9391×10^{3}
	std	5.7937×10^{2}	6.1779×10^{2}	5.3206×10^{2}	4.1430×10^{3}	4.1599×10^{3}	6.9200×10^{2}
		12	7	6	4	2	11
F26	mean	3.0131×10^{3}	3.0003×10^{3}	3.0041×10^{3}	2.9725×10^{3}	2.9831×10^{3}	3.0155×10^{3}
	std	1.9661×10^{4}	1.8508×10^{4}	1.9059×10^{4}	1.6773×10^{4}	2.7434×10^{4}	2.8042×10^{4}
		8	4	7	1	2	9
F27	mean	3.1014×10^{3}	3.1002×10^{3}	3.1000×10^{3}	3.1081×10^{3}	3.1034×10^{3}	3.0992×10^{3}
	std	1.2188×10^{2}	7.8473×10^{1}	2.5213×10^{1}	5.3786×10^{2}	1.8389×10^{2}	2.2062×10^{1}
		8	4	3	12	11	2

Table 12. Adjusting the parameters F_{min}, CR, and k of mDBO.

F	Type	(CR, k, F_{min})					
		(0.9,3,0)	(0.9,3,0.1)	(0.9,3,0.2)	(0.9,4,0)	(0.9,4,0.1)	(0.9,4,0.2)
F28	mean	3.3076×10^{3}	3.3383×10^{3}	3.3023×10^{3}	3.3208×10^{3}	3.2878×10^{3}	3.2887×10^{3}
	std	1.4063×10^{4}	9.4229×10^{3}	1.3444×10^{4}	1.1854×10^{4}	1.0362×10^{4}	1.2737×10^{4}
		7	12	6	10	2	3
F29	mean	3.2636×10^{3}	3.2612×10^{3}	3.2449×10^{3}	3.2355×10^{3}	3.2574×10^{3}	3.2287×10^{3}
	std	5.9821×10^{3}	4.7098×10^{3}	5.0638×10^{3}	5.5056×10^{3}	4.8487×10^{3}	3.4887×10^{3}
		10	9	7	4	8	2
F30	mean	4.5316×10^{5}	5.7595×10^{5}	3.4976×10^{5}	4.8910×10^{5}	3.5884×10^{5}	5.4746×10^{5}
	std	2.7814×10^{11}	3.7877×10^{11}	2.8156×10^{11}	2.1879×10^{11}	3.7710×10^{11}	3.1246×10^{11}
		4	11	1	6	3	8

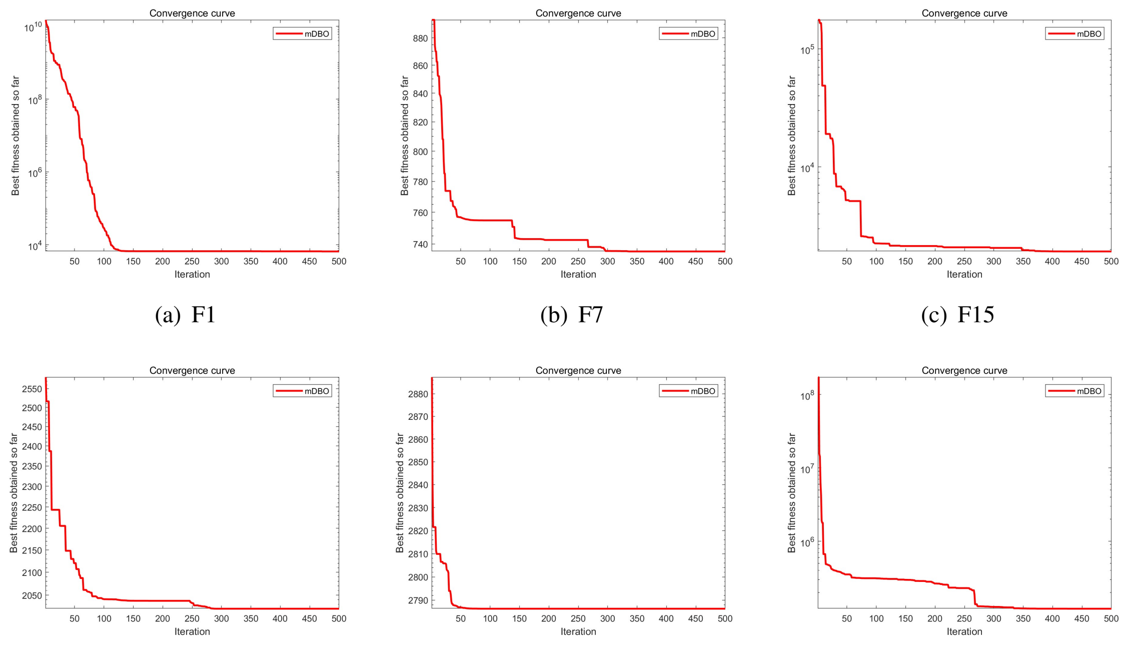
this parameter configuration, mDBO effectively solves the optimization problem and yields improved results, showcasing strong global optimization capabilities. Specifically, F_{min} set to 0 indicates that the algorithm uses a lower mutation strength during the search process, while CR set to 0.8 provides a higher crossover probability, thereby facilitating the generation of diverse solutions. Additionally, k = 4 suggests that the selected neighborhood size strikes a good balance between global and local search in the selection process. Given the favorable performance of these parameters on the aforementioned test functions, we opt to retain this fixed parameter setting as $F_{min} = 0$, $CR = 0.8$, and $k = 4$ in subsequent experiments to ensure stability and reproducibility of the results.

4.4. *Qualitative evaluation*

In this section, a quantitative evaluation of the proposed mDBO is conducted through three experiments, analyzing its convergence behavior, population diversity, and the balance between exploration and exploitation.

4.4.1. Behavior analysis

To evaluate the performance of mDBO in solving different test functions, the convergence curves of mDBO are obtained. The results are shown in Figure 11. The convergence curves illustrate the accelerated degradation of fitness values across all test functions. This behavior indicates that the proposed mDBO algorithm can improve candidate solutions and find promising solutions before half of the iterations.



(a) F1 (b) F7 (c) F15

(d) F20 (e) F24 (f) F30

Figure 11. The mDBO convergence curve.

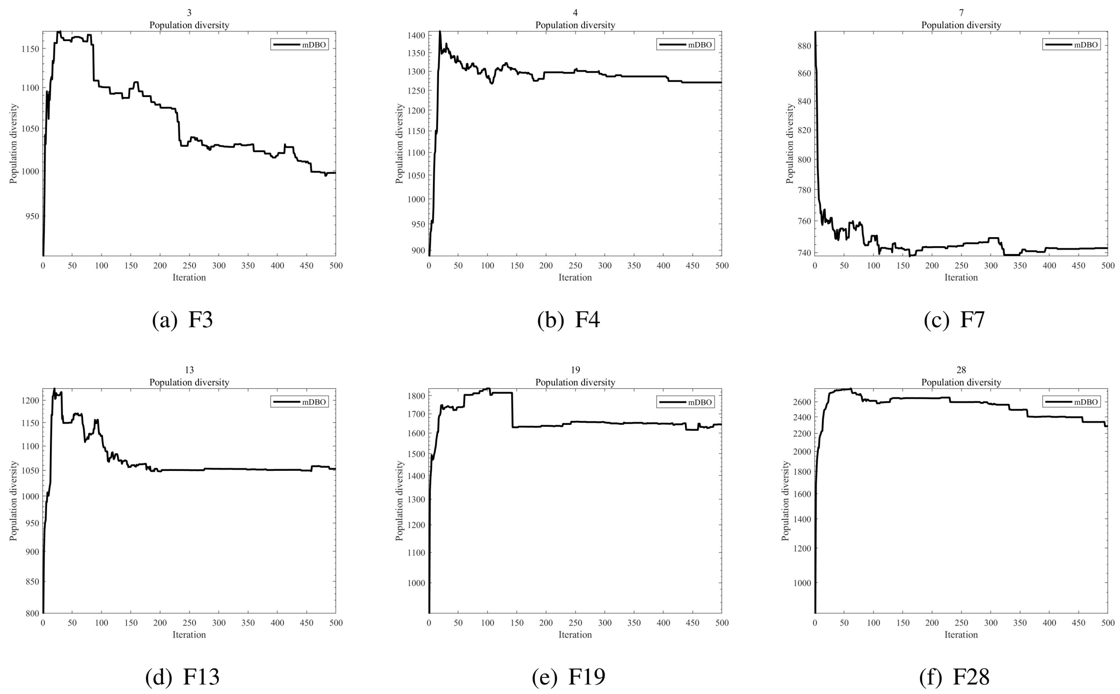
4.4.2. Population diversity analysis

The proposed strategy enables the mDBO algorithm to escape the trap of local optima and reduces the risk of stagnation. To analyze the impact of the proposed strategy on the performance of the mDBO algorithm, the population diversity is calculated using the moment of inertia I_c, as shown in Eq (4.1) [75]. The Ic represents the dispersion of the population relative to its mass center c at each iteration, and is computed using Eq (4.2), where the parameter X_{id}^t denotes the value of the d-th dimension of the i-th individual at iteration t.

$$I_c(t) = \sqrt{\sum_{i=1}^{N} \sum_{d=1}^{D} (X_{id}^t - C_d^t)^2}, \tag{4.1}$$

$$c_d(t) = \frac{1}{D} \sum_{i=1}^{N} X_{id}^t. \tag{4.2}$$

In Figure 12, the population diversity of the mDBO algorithm is tracked using Eq (4.1), with test functions including unimodal, multimodal, hybrid, and composite functions. From the curve, it can be observed that the diversity measure I_c fluctuates significantly from the initial value. In the early stages of evolution, the population diversity is typically high due to the large differences between individuals. As iterations progress, the algorithm gradually selects superior individuals, leading to a concentration of genes or traits within the population, resulting in a decrease in diversity. The plotted results show a diversity maintenance phase, where the proposed strategy proves effective in maintaining individual diversity. Therefore, mDBO is capable of preserving population diversity in the mature phase of the evolutionary search process, preventing premature convergence, exploring new regions of the search space, and thus demonstrating great potential in finding the optimal solution.



(a) F3 (b) F4 (c) F7

(d) F13 (e) F19 (f) F28

Figure 12. Population diversity of the mDBO algorithm.

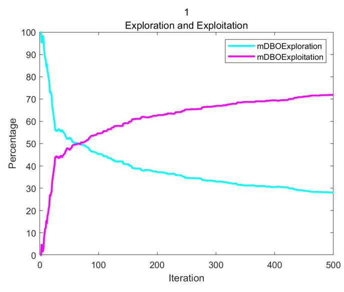
4.4.3. Exploration and exploitation analysis

The strong effect of exploration increases the distance between individuals, while the strong effect of exploitation decreases the distance among the population during the search process. Therefore, the balance between exploration and exploitation ensures the avoidance of premature convergence and the loss of diversity. The percentages of exploration and exploitation are calculated using Eqs (4.3) and (4.4) [76]. In these equations, parameter Div(t) represents the dimension-wise diversity measure [76, 77], which is computed using Eq (4.5) to indicate the increase or decrease of distance between individuals during the search process. Div_{max} refers to the maximum diversity across all iterations, and parameter X_{id} represents the position of the i-th individual in dimension d. The effects of exploration and exploitation are evaluated in unimodal, multimodal, hybrid, and composition test functions. The resulting data are plotted in Figure 13, where the percentage indicates the level of exploration and exploitation for the population during the evolution process.

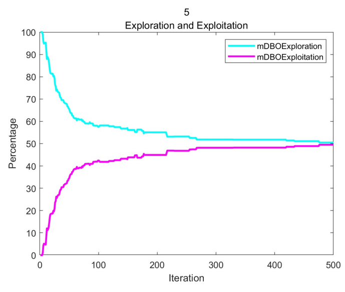
$$Exploration(t) = \frac{Div(t)}{DIV_{max}} \times 100, \tag{4.3}$$

$$Exploitation(t) = \frac{|Div(t) - Div_{max}|}{Div_{max}} \times 100, \tag{4.4}$$

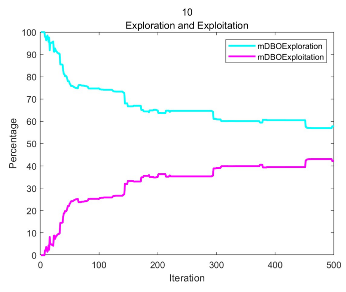
$$Div(t) = \frac{1}{D}\sum_{d=1}^{D}\frac{1}{N}\sum_{i=1}^{N}\left|median(X_d(t)) - X_{id}(t)\right|. \tag{4.5}$$



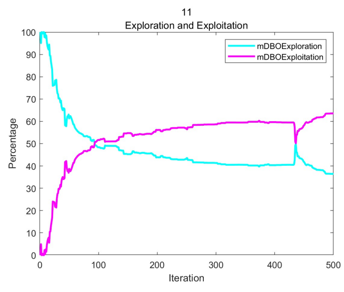
(a) F1



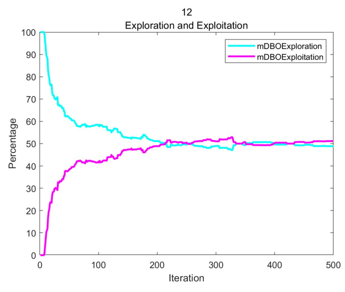
(b) F5



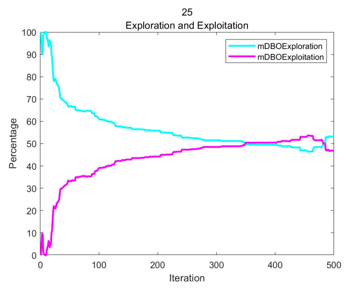
(c) F10



(d) F11



(e) F12



(f) F25

Figure 13. Exploration and exploitation behavior analysis of the mDBO algorithm.

In the process of algorithm optimization, the balance between exploration and exploitation is crucial. As shown in Figure 13, during the early stages, the algorithm explores the solution space with a higher exploration rate to discover diverse solutions. As iterations progress, the exploration rate gradually decreases, while the exploitation rate increases, and the algorithm begins to focus more on refining the solutions found.

4.5. Ablation experiment

The mDBO framework incorporates three distinct strategies into the foundational DBO algorithm. Three improved strategies are introduced in DBO, namely: Chaos-based opposition learning strategy (CRO), dynamic retrospective adaptive differential mutation strategy (PRADE), and Padé approximation and adaptive evolutionary boundary constraint strategy (PAAEBC). Table 13 presents the variants of DBO in the ablation experiment, where 1 indicates that the strategy has been added, and 0 indicates that the strategy has not been added. To assess the effectiveness of this approach, ablation experiments are conducted for each of the three strategies. The results of these experiments are detailed in Tables 14 and 15 and Figure 14.

Table 13. Variants of DBO in ablation experiments.

Algorithms	CRO	PRADE	PAAEBC
DBO	0	0	0
CDBO	1	0	0
DDBO	0	1	0
PDBO	0	0	1
mDBO	1	1	1

The results presented in Tables 14 and 15 indicate that mDBO outperforms CDBO for 20 benchmark functions, exceeds DDBO for 21 functions, and surpasses PDBO for 24 functions. Conversely, CDBO outperforms DBO for 24 benchmark functions, DDBO exceeds DBO for 26 functions, and PDBO surpasses DBO for 24 functions. In the context of unimodal functions (F1 and F3), CDBO and DDBO demonstrate superior performance compared to PDBO. For multimodal functions (F4 to F10), PDBO shows the highest effectiveness, suggesting that the pade approximation strategy is beneficial for this class of functions. In the case of hybrid functions (F11 to F20), CDBO exhibits the most pronounced effect, indicating the efficacy of the chaos-based opposition-based learning strategy. For composition functions (F21 to F30), DDBO demonstrates the strongest impact, highlighting the advantages of the dynamic retrospective adaptive differential mutation strategy in this context. Overall, the experimental results show that all DBO variants, incorporating individual strategies, exhibit significant performance improvements over the standard DBO, with mDBO emerging as the most effective variant.

4.6. Statistical results analysis

In this section, we present a comprehensive quantitative analysis of the optimization capabilities of the proposed mDBO algorithm. We conduct 30 independent runs for each of the ten-dimensional test functions (F1 to F30) using the mDBO algorithm, alongside nine established intelligent algorithms: WOA, SCA, PSO, GWO, HHO, AOA, BOA, DE, DBO, LSHADE, and LSHADE_SPACMA. The statistical data collected from these experiments are meticulously organized in Tables 16–19 and Figure 15. The last three rows of the tables feature symbols “+”, “=”, and “−”, which indicate that the proposed method outperforms, is comparable to, or underperforms relative to the other algorithms, respectively. Furthermore, the tables present the average and final rankings for all algorithms included

Table 14. Comparison results of different DBO variants.

F	Type	mDBO	CDBO	DDBO	PDBO	DBO
F1	mean	4.0151×10^3	5.6590×10^3	5.0656×10^3	4.9190×10^3	7.2274×10^5
	std	1.5028×10^7	1.3131×10^7	1.6576×10^7	1.4059×10^7	9.8979×10^{12}
		1	4	3	2	5
F3	mean	3.0402×10^2	3.6949×10^2	5.4869×10^2	6.5734×10^2	5.8088×10^2
	std	7.8624×10^1	7.2916×10^4	1.9646×10^5	9.9017×10^5	1.5816×10^5
		1	2	3	5	4
F4	mean	4.0822×10^2	4.1053×10^2	4.1506×10^2	4.2123×10^2	4.3732×10^2
	std	2.9596×10^2	4.2037×10^2	4.8862×10^2	1.1659×10^3	1.9228×10^3
		1	2	3	4	5
F5	mean	5.2809×10^2	5.3403×10^2	5.3385×10^2	5.3132×10^2	5.4101×10^2
	std	1.4399×10^2	1.6499×10^2	1.9844×10^2	1.6660×10^2	1.8678×10^2
		1	4	3	2	5
F6	mean	6.0412×10^2	6.0834×10^2	6.0723×10^2	6.0414×10^2	6.1193×10^2
	std	1.7939×10^1	6.3318×10^1	4.6160×10^1	2.5885×10^1	6.5646×10^1
		1	4	3	2	5
F7	mean	7.3903×10^2	7.4980×10^2	7.4278×10^2	7.4005×10^2	7.4693×10^2
	std	2.0406×10^2	2.2915×10^2	3.2510×10^2	1.4149×10^2	4.3974×10^2
		1	5	3	2	4
F8	mean	8.2532×10^2	8.2687×10^2	8.2834×10^2	8.2887×10^2	8.3592×10^2
	std	9.7939×10^1	7.5604×10^1	1.7489×10^2	1.7033×10^2	1.6376×10^2
		1	2	3	4	5
F9	mean	9.3782×10^2	9.6100×10^2	9.7430×10^2	9.6843×10^2	9.8164×10^2
	std	3.1985×10^3	6.0940×10^3	1.2375×10^4	1.6415×10^4	8.9017×10^3
		1	2	4	3	5
F10	mean	2.0288×10^3	1.9817×10^3	1.8870×10^3	1.8574×10^3	2.1127×10^3
	std	9.3686×10^4	1.2561×10^5	1.4586×10^5	9.6889×10^4	7.5340×10^4
		4	3	2	1	5
F11	mean	1.1492×10^3	1.1566×10^3	1.1473×10^3	1.1477×10^3	1.2520×10^3
	std	3.9519×10^3	4.8361×10^3	2.8706×10^3	2.6669×10^3	1.7806×10^4
		3	4	1	2	5
F12	mean	1.6133×10^4	4.5646×10^5	5.6317×10^5	5.6674×10^5	2.2173×10^6
	std	2.0858×10^8	2.9696×10^{12}	4.3245×10^{12}	4.3188×10^{12}	1.3802×10^{13}
		1	2	3	4	5
F13	mean	1.1479×10^4	1.0253×10^4	1.3566×10^4	1.3759×10^4	1.4771×10^4
	std	6.9548×10^7	1.0764×10^8	1.1918×10^8	1.3479×10^8	1.3512×10^8
		2	1	3	4	5
F14	mean	1.6698×10^3	1.6803×10^3	1.8371×10^3	1.8354×10^3	1.9694×10^3
	std	4.3283×10^4	1.6086×10^5	3.7208×10^5	2.0932×10^5	5.2379×10^5
		1	2	4	3	5
F15	mean	2.9685×10^3	2.3226×10^3	3.1348×10^3	3.2995×10^3	4.6039×10^3
	std	1.4048×10^6	1.2053×10^6	3.0868×10^6	3.9377×10^6	3.2058×10^7
		2	1	3	4	5
F16	mean	1.7453×10^3	1.7337×10^3	1.7223×10^3	1.7650×10^3	1.8064×10^3
	std	1.7604×10^4	1.8927×10^4	9.3227×10^3	2.0006×10^4	1.3478×10^4
		3	2	1	4	5
F17	mean	1.7589×10^3	1.7736×10^3	1.7635×10^3	1.7869×10^3	1.7910×10^3
	std	6.8539×10^2	1.3468×10^3	1.2513×10^3	2.3545×10^3	2.3285×10^3
		1	3	2	4	5

Table 15. Comparison results of different DBO variants.

F	Type	mDBO	CDBO	DDBO	PDBO	DBO
F18	mean	1.5044×10^4	1.4947×10^4	1.3933×10^4	1.6328×10^4	2.4579×10^4
	std	1.7380×10^8	1.9474×10^8	1.4152×10^8	2.3849×10^8	2.5829×10^8
		3	2	1	4	5
F19	mean	5.4835×10^3	5.3592×10^3	5.1333×10^3	4.2590×10^3	9.4305×10^3
	std	4.9692×10^7	3.9768×10^7	4.3710×10^7	3.1635×10^7	1.3207×10^8
		4	3	2	1	5
F20	mean	2.0695×10^3	2.0780×10^3	2.0883×10^3	2.0797×10^3	2.0998×10^3
	std	3.3404×10^3	5.8792×10^3	3.7676×10^3	2.8449×10^3	5.0822×10^3
		1	2	4	3	5
F21	mean	2.2781×10^3	2.2789×10^3	2.2868×10^3	2.2977×10^3	2.2208×10^3
	std	4.1732×10^3	4.4939×10^3	4.1807×10^3	3.9820×10^3	1.2751×10^3
		2	3	4	5	1
F22	mean	2.3060×10^3	2.3053×10^3	2.3023×10^3	2.3015×10^3	2.3091×10^3
	std	2.2743×10^2	1.6838×10^1	2.6147×10^2	1.5947×10^2	3.3667×10^1
		4	3	2	1	5
F23	mean	2.6363×10^3	2.6351×10^3	2.6321×10^3	2.6376×10^3	2.6448×10^3
	std	2.1618×10^2	1.6293×10^2	1.6775×10^2	1.7771×10^2	1.9188×10^2
		3	2	1	4	5
F24	mean	2.7040×10^3	2.7202×10^3	2.7621×10^3	2.7407×10^3	2.7344×10^3
	std	1.2200×10^4	9.9252×10^3	2.6250×10^3	6.4097×10^3	6.6527×10^3
		1	2	5	4	3
F25	mean	2.9333×10^3	2.9363×10^3	2.9411×10^3	2.9284×10^3	2.9342×10^3
	std	5.4438×10^2	6.3833×10^2	8.4327×10^2	6.8985×10^2	4.0947×10^3
		2	4	5	1	3
F26	mean	3.0164×10^3	3.0810×10^3	3.0512×10^3	3.0337×10^3	3.1028×10^3
	std	5.1650×10^4	5.1252×10^4	7.1225×10^4	1.8342×10^4	3.3958×10^4
		1	4	3	2	5
F27	mean	3.1013×10^3	3.1061×10^3	3.1028×10^3	3.1045×10^3	3.1043×10^3
	std	4.3282×10^1	4.7293×10^2	1.5881×10^2	2.0371×10^2	1.3389×10^2
		1	5	2	4	3
F28	mean	3.2780×10^3	3.3233×10^3	3.3353×10^3	3.3003×10^3	3.3441×10^3
	std	1.1377×10^4	1.5049×10^4	1.6816×10^4	1.3849×10^4	1.1018×10^4
		1	3	4	2	5
F29	mean	3.2380×10^3	3.2877×10^3	3.2488×10^3	3.2554×10^3	3.2527×10^3
	std	4.8295×10^3	6.7508×10^3	4.6284×10^3	5.5444×10^3	4.3175×10^3
		1	5	2	4	3
F30	mean	4.6651×10^5	2.2673×10^5	4.5411×10^5	7.6241×10^5	1.1384×10^6
	std	7.7540×10^{11}	1.1362×10^{11}	2.1224×10^{11}	9.4741×10^{11}	1.4732×10^{12}
		3	1	2	4	5
Avg rank		1.79	2.83	2.79	3.07	4.52
Overal rank		1	3	2	4	5

in this analysis.

After evaluating the unimodal functions F1 and F3, we found that compared to traditional DBO, mDBO achieves significant improvements in both average fitness and standard deviation. mDBO not only successfully identifies the global optimal solutions for these two test functions but also outperforms all other algorithms in the comparison. Notably, the PSO algorithm exhibits excellent performance on the F3 function, slightly surpassing mDBO in terms of average fitness. When applied to simple multimodal functions (F4 to F10), mDBO generally outperforms other algorithms in terms of both average fitness and standard deviation. However, for the F5, and F8 functions, LSHADE_SPACMA performed best among all algorithms. This may be attributed to the adoption of a quadratic interpolation strategy, which effectively balances the relationship between exploration and exploitation. In tests involving hybrid benchmark functions (F11 to F20), mDBO excels in 5 of the 10 test functions. The remarkable global search capability of mDBO can be attributed to its hybrid chaotic mapping strategy, which ensures a high-quality initial population, thereby conferring a distinct advantage in the early iterations of the algorithm. When evaluating composite functions (F21 to F30), mDBO performs best for 6 of the 10 test functions. Although it does not achieve the highest ranking on the remaining four functions, it consistently places second or third, further demonstrating mDBO's robust search capabilities in tackling complex optimization problems. This may be primarily due to the enhanced performance of the DE strategy in avoiding premature convergence to local optima. Furthermore, this method achieves a Friedman average ranking of 1.62, securing the top position, closely followed by the LSHADE_SPACMA algorithm. Table 20 presents the average runtime of different algorithms. Compared to other algorithms, the proposed algorithm shows a slight increase in runtime; however, it achieves a significant improvement in accuracy, resulting in better performance. Therefore, despite the additional computational overhead, the enhanced accuracy makes this cost acceptable. Overall, the mDBO demonstrates a significantly stronger competitive advantage for the CEC2017 benchmark suite.

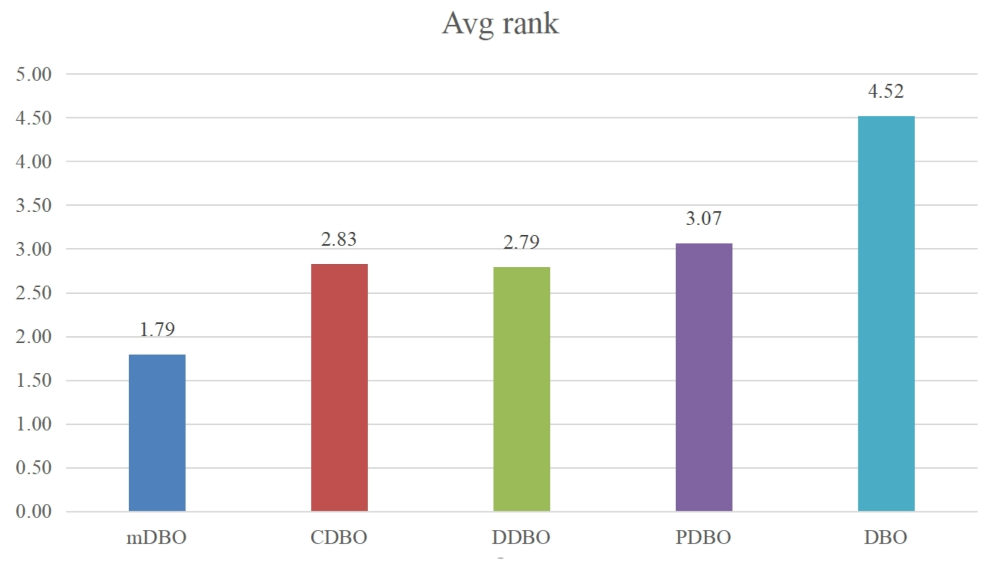


Figure 14. Comparison results of different DBO variants Avg rank.

To evaluate the statistical significance of the simulation results for the proposed mDBO method

Table 16. A comparison of different algorithms in solving benchmark functions.

Function	Metric	mDBO	WOA	SCA	PSO	GWO	HHO
F1	mean	5.9830×10^3	6.0680×10^7	9.3749×10^8	1.0260×10^8	6.4487×10^7	1.4935×10^6
	std	1.9664×10^7	7.3331×10^{15}	1.0054×10^{17}	1.6371×10^{17}	1.7801×10^{16}	1.4342×10^{12}
		2	6	9	8	7	4
F3	mean	4.7105×10^2	6.2869×10^3	2.4981×10^3	3.0065×10^2	3.3085×10^3	7.1312×10^2
	std	4.2893×10^5	3.5071×10^7	1.4270×10^6	9.4727×10^{-1}	7.7841×10^6	7.7209×10^4
		2	10	6	1	7	3
F4	mean	4.0000×10^2	4.7636×10^2	4.5769×10^2	4.0018×10^2	4.0780×10^2	4.9382×10^2
	std	4.4185×10^2	2.8773×10^3	4.8958×10^2	4.3018×10^2	2.9294×10^2	1.6587×10^3
		1	7	6	2	5	8
F5	mean	5.1492×10^2	5.8300×10^2	5.5847×10^2	5.3582×10^2	5.1622×10^2	5.7876×10^2
	std	1.0929×10^2	6.3636×10^2	1.2763×10^1	1.2643×10^2	3.4993×10^1	5.5369×10^2
		2	11	7	5	4	10
F6	mean	6.0000×10^2	6.4541×10^2	6.1774×10^2	6.0000×10^2	6.0143×10^2	6.2976×10^2
	std	8.2097×10^1	1.6445×10^2	3.9719×10^1	2.4873×10^{-7}	1.2922	3.7760×10^2
		1	11	7	1	5	9
F7	mean	7.1453×10^2	7.6719×10^2	7.7828×10^2	7.2873×10^2	7.4118×10^2	7.8524×10^2
	std	9.8852×10^1	5.3015×10^2	1.3864×10^2	1.9533×10^1	5.2822×10^1	3.7965×10^2
		1	7	8	3	5	10
F8	mean	8.0696×10^2	8.4606×10^2	8.4545×10^2	8.1492×10^2	8.2516×10^2	8.1699×10^2
	std	1.0210×10^2	2.5278×10^2	1.1091×10^2	5.5419×10^1	3.5753×10^1	6.1848×10^1
		2	9	8	3	6	4
F9	mean	9.0000×10^2	1.1749×10^3	1.0422×10^3	9.0000×10^2	9.0117×10^2	1.1329×10^3
	std	1.1021×10^4	1.8815×10^5	2.5956×10^3	1.6496×10^{-13}	1.5091×10^2	4.6552×10^4
		1	9	7	1	5	8
F10	mean	1.1287×10^3	1.9809×10^3	2.6108×10^3	1.9928×10^3	1.6545×10^3	1.8648×10^3
	std	1.6392×10^5	5.9407×10^4	5.6942×10^3	7.4046×10^4	1.1364×10^5	1.8030×10^5
		1	6	10	7	4	5
F11	mean	1.1117×10^3	1.1572×10^3	1.2388×10^3	1.1135×10^3	1.1699×10^3	1.1929×10^3
	std	3.4828×10^2	3.2314×10^3	5.3175×10^2	2.0315×10^2	3.0651×10^3	2.7237×10^2
		2	5	9	3	6	7
F12	mean	6.1019×10^3	2.3647×10^5	1.3416×10^7	1.3233×10^4	7.1902×10^3	3.7228×10^6
	std	1.4289×10^8	3.7592×10^{13}	3.2679×10^{13}	7.2045×10^7	6.9672×10^{11}	1.0055×10^{13}
		1	6	10	4	2	7
F13	mean	9.5429×10^3	1.8746×10^4	7.3534×10^4	1.0962×10^4	1.4659×10^4	2.3247×10^4
	std	8.9888×10^7	2.0509×10^8	8.2053×10^9	1.2400×10^8	2.3708×10^7	5.9259×10^8
		3	10	12	4	9	11
F14	mean	1.4936×10^3	2.2571×10^3	2.5745×10^3	1.5550×10^3	1.5464×10^3	2.0563×10^3
	std	2.7722×10^4	3.2324×10^6	1.5715×10^5	1.3967×10^6	2.3460×10^6	4.9974×10^4
		3	10	12	6	5	9
F15	mean	2.2228×10^3	8.1827×10^6	5.6390×10^7	5.9624×10^3	5.1072×10^5	2.1888×10^5
	std	9.9682×10^8	8.3300×10^{13}	7.8407×10^{15}	1.2840×10^8	1.8130×10^{12}	4.3987×10^9
		1	8	9	4	6	5
F16	mean	1.6019×10^3	1.9217×10^3	1.8322×10^3	1.9595×10^3	1.8804×10^3	1.7562×10^3
	std	9.5137×10^3	3.1810×10^4	1.3157×10^4	1.8875×10^4	9.7285×10^3	2.5851×10^4
		1	10	6	11	7	5
F17	mean	1.7060×10^3	1.7864×10^3	1.8101×10^3	1.7546×10^3	1.7526×10^3	1.8055×10^3
	std	1.5321×10^3	3.4911×10^3	2.4389×10^2	2.5119×10^2	2.7267×10^2	1.6243×10^3
		1	7	9	5	4	8

Table 17. A comparison of different algorithms in solving benchmark functions.

Function	Metric	mDBO	WOA	SCA	PSO	GWO	HHO
F18	mean	2.5288×10^{3}	1.2687×10^{4}	4.7095×10^{5}	8.1018×10^{3}	4.2539×10^{4}	3.7524×10^{3}
	std	1.9006×10^{8}	1.4603×10^{8}	2.7838×10^{10}	2.8407×10^{8}	1.9150×10^{8}	9.1356×10^{7}
		3	7	11	6	8	5
F19	mean	2.0442×10^{3}	9.0014×10^{4}	1.4781×10^{4}	5.8505×10^{3}	2.1983×10^{3}	1.0012×10^{4}
	std	2.9953×10^{7}	1.1696×10^{9}	1.3748×10^{8}	6.1337×10^{6}	1.8266×10^{7}	3.6725×10^{7}
		3	12	9	6	4	7
F20	mean	2.0016×10^{3}	2.1334×10^{3}	2.1530×10^{3}	2.1297×10^{3}	2.0521×10^{3}	2.3096×10^{3}
	std	6.0599×10^{2}	1.4069×10^{4}	3.1264×10^{3}	2.1846×10^{3}	1.6232×10^{4}	8.5374×10^{3}
		1	7	9	6	4	12
F21	mean	2.2063×10^{3}	2.3564×10^{3}	2.2253×10^{3}	2.3225×10^{3}	2.3071×10^{3}	2.3850×10^{3}
	std	3.4761×10^{3}	5.3132×10^{3}	5.3471×10^{3}	2.9730×10^{3}	8.6179	9.0067×10^{3}
		1	10	4	8	7	12
F22	mean	2.3008×10^{3}	2.3241×10^{3}	2.3548×10^{3}	2.3487×10^{3}	2.3091×10^{3}	2.3100×10^{3}
	std	1.5367×10^{1}	5.0379×10^{5}	2.6023×10^{3}	1.6782×10^{5}	7.0941×10^{1}	2.4169×10^{1}
		1	7	9	8	4	5
F23	mean	2.6223×10^{3}	2.6648×10^{3}	2.6568×10^{3}	2.6431×10^{3}	2.6229×10^{3}	2.6646×10^{3}
	std	7.0070×10^{1}	5.3789×10^{2}	4.0340×10^{1}	5.4175×10^{2}	1.2285×10^{2}	4.6013×10^{2}
		2	10	7	5	3	9
F24	mean	2.6820×10^{3}	2.7769×10^{3}	2.7888×10^{3}	2.6538×10^{3}	2.7510×10^{3}	2.8284×10^{3}
	std	1.9056×10^{4}	1.0759×10^{2}	8.0283×10^{1}	1.9711×10^{4}	6.5472×10^{1}	3.0033×10^{3}
		2	8	9	1	6	11
F25	mean	2.9190×10^{3}	2.9596×10^{3}	2.9800×10^{3}	2.9269×10^{3}	2.9319×10^{3}	2.9193×10^{3}
	std	7.4391×10^{2}	5.4819×10^{2}	3.3701×10^{1}	5.7310×10^{2}	2.4926×10^{2}	1.0317×10^{4}
		1	8	9	4	6	2
F26	mean	2.8770×10^{3}	3.5214×10^{3}	3.1170×10^{3}	3.2580×10^{3}	3.5114×10^{3}	3.6097×10^{3}
	std	3.2225×10^{4}	9.3681×10^{5}	1.7274×10^{3}	3.2049×10^{5}	3.0960×10^{5}	4.4288×10^{5}
		1	10	5	7	9	11
F27	mean	3.0923×10^{3}	3.2184×10^{3}	3.1025×10^{3}	3.0976×10^{3}	3.1056×10^{3}	3.1081×10^{3}
	std	4.8263×10^{1}	3.4885×10^{3}	9.0634	8.3882×10^{2}	2.5841×10^{1}	2.4612×10^{3}
		2	11	5	4	6	7
F28	mean	3.1883×10^{3}	3.1901×10^{3}	3.3295×10^{3}	3.3838×10^{3}	3.4029×10^{3}	3.5267×10^{3}
	std	4.9551×10^{4}	2.2896×10^{4}	5.5565×10^{3}	7.6759×10^{3}	4.5755×10^{1}	2.6088×10^{4}
		1	2	6	7	8	12
F29	mean	3.1415×10^{3}	3.2618×10^{3}	3.3225×10^{3}	3.1886×10^{3}	3.1527×10^{3}	3.4866×10^{3}
	std	8.3463×10^{3}	2.4339×10^{3}	4.2984×10^{3}	9.5113×10^{2}	1.1813×10^{3}	2.3560×10^{4}
		1	6	8	4	2	11
F30	mean	7.8147×10^{3}	9.0791×10^{5}	3.0734×10^{6}	2.7188×10^{4}	3.7176×10^{4}	2.3251×10^{6}
	std	1.6203×10^{11}	9.8223×10^{11}	1.4798×10^{12}	2.6606×10^{12}	3.0456×10^{11}	3.6345×10^{11}
		3	7	10	4	5	9
Paired rank +\=\−		–	29/0/0	29/0/0	25/2/2	29/0/0	29/0/0
Avg. rank		1.62	8.17	8.14	4.76	5.48	7.79
Overall rank		1	9	8	4	5	7

Table 18. A comparison of different algorithms in solving benchmark functions.

Function	Metric	AOA	BOA	DE	DBO	LSHADE	LSHADE_SPACMA
F1	mean	7.0685×10^{9}	6.7460×10^{9}	2.4561×10^{9}	2.3211×10^{6}	2.7844×10^{4}	1.0000×10^{2}
	std	7.1302×10^{18}	9.3630×10^{18}	1.1356×10^{18}	8.1983×10^{13}	4.9720×10^{9}	6.6771×10^{-8}
		12	11	10	5	3	1
F3	mean	5.4842×10^{3}	9.9005×10^{3}	3.9036×10^{4}	9.7402×10^{2}	4.7442×10^{3}	1.5556×10^{3}
	std	3.2265×10^{6}	6.0830×10^{6}	1.4895×10^{8}	2.2598×10^{6}	2.1950×10^{7}	1.1565×10^{7}
		9	11	12	4	8	5
F4	mean	8.3545×10^{2}	1.9510×10^{3}	5.0949×10^{2}	5.1260×10^{2}	4.0358×10^{2}	4.0024×10^{2}
	std	3.8490×10^{4}	3.3001×10^{5}	6.9764×10^{3}	2.8143×10^{3}	2.2020	1.8397×10^{-2}
		11	12	9	10	4	3
F5	mean	5.7058×10^{2}	5.8961×10^{2}	5.7617×10^{2}	5.5224×10^{2}	5.1542×10^{2}	5.0327×10^{2}
	std	1.8773×10^{1}	8.5899×10^{1}	5.2894×10^{1}	1.6111×10^{2}	1.0757×10^{2}	1.5438
		8	12	9	6	3	1
F6	mean	6.1889×10^{2}	6.4798×10^{2}	6.4135×10^{2}	6.1264×10^{2}	6.0001×10^{2}	6.0008×10^{2}
	std	6.3491×10^{1}	4.7401×10^{1}	2.8158×10^{1}	2.4556×10^{1}	1.1155×10^{-4}	1.0914×10^{-3}
		8	12	10	6	3	4
F7	mean	7.8299×10^{2}	8.0521×10^{2}	8.5162×10^{2}	7.3765×10^{2}	7.4466×10^{2}	7.1466×10^{2}
	std	2.2834×10^{2}	6.6414×10^{1}	2.0078×10^{2}	5.8889×10^{1}	9.2006×10^{1}	1.1255×10^{1}
		9	11	12	4	6	2
F8	mean	8.5093×10^{2}	8.4289×10^{2}	8.8154×10^{2}	8.5894×10^{2}	8.2292×10^{2}	8.0630×10^{2}
	std	8.0949×10^{1}	1.8840×10^{1}	1.6222×10^{2}	1.6272×10^{2}	1.5607×10^{1}	1.1391×10^{1}
		10	7	12	11	5	1
F9	mean	1.2093×10^{3}	1.5195×10^{3}	2.0063×10^{3}	9.0509×10^{2}	9.0000×10^{2}	9.0000×10^{2}
	std	3.4668×10^{4}	3.6381×10^{4}	5.9519×10^{5}	3.9915×10^{4}	2.7536×10^{-5}	1.6031×10^{-3}
		10	11	12	6	1	1
F10	mean	2.2416×10^{3}	2.4785×10^{3}	2.7927×10^{3}	1.5871×10^{3}	2.8391×10^{3}	1.1503×10^{3}
	std	3.4085×10^{4}	3.9971×10^{4}	5.4974×10^{4}	7.8338×10^{4}	1.2848×10^{5}	4.0489×10^{4}
		8	9	11	3	12	2
F11	mean	1.5253×10^{3}	1.6659×10^{3}	1.6138×10^{3}	1.2194×10^{3}	1.1197×10^{3}	1.1020×10^{3}
	std	2.7897×10^{5}	2.6490×10^{4}	1.1005×10^{5}	5.7012×10^{3}	4.2563×10^{1}	1.9337
		10	12	11	8	4	1
F12	mean	3.9273×10^{6}	7.6091×10^{7}	1.5098×10^{8}	4.9633×10^{6}	1.7551×10^{4}	9.6215×10^{3}
	std	7.3636×10^{14}	5.3368×10^{15}	1.0509×10^{16}	1.2391×10^{13}	1.5596×10^{10}	1.2996×10^{7}
		8	11	12	9	5	3
F13	mean	1.4378×10^{4}	1.3423×10^{5}	1.3081×10^{6}	1.1291×10^{4}	1.4358×10^{3}	1.3217×10^{3}
	std	2.1282×10^{7}	1.6068×10^{10}	1.3093×10^{12}	1.6890×10^{8}	5.7221×10^{4}	3.6856×10^{2}
		8	7	6	5	2	1
F14	mean	1.5432×10^{3}	2.3131×10^{3}	1.6894×10^{4}	1.7796×10^{3}	1.4257×10^{3}	1.4220×10^{3}
	std	5.8954×10^{3}	3.5040×10^{5}	3.7145×10^{7}	4.4141×10^{4}	1.1240×10^{1}	8.5869×10^{1}
		4	11	7	8	2	1
F15	mean	9.9404×10^{7}	7.3127×10^{7}	2.8179×10^{8}	5.4715×10^{3}	8.8211×10^{5}	4.2803×10^{3}
	std	7.8709×10^{16}	3.8759×10^{16}	1.4387×10^{17}	1.0824×10^{13}	1.6010×10^{11}	1.0848×10^{6}
		11	10	12	3	7	2
F16	mean	1.6538×10^{3}	1.9050×10^{3}	1.9645×10^{3}	1.9175×10^{3}	1.6134×10^{3}	1.6032×10^{3}
	std	1.8874×10^{4}	5.6388×10^{3}	4.3812×10^{4}	2.0797×10^{4}	4.6773×10^{2}	1.6650
		4	8	12	9	3	2
F17	mean	1.8622×10^{3}	1.8241×10^{3}	1.8466×10^{3}	1.7832×10^{3}	1.7322×10^{3}	1.7111×10^{3}
	std	9.7869×10^{2}	5.4547×10^{2}	2.3069×10^{3}	2.8934×10^{3}	3.1865×10^{1}	5.1725×10^{1}
		12	10	11	6	3	2

Table 19. A comparison of different algorithms in solving benchmark functions.

Function	Metric	AOA	BOA	DE	DBO	LSHADE	LSHADE_SPACMA
F18	mean	1.1879×10^{5}	2.3989×10^{6}	9.3960×10^{5}	2.9466×10^{3}	1.8370×10^{3}	1.9244×10^{3}
	std	1.8019×10^{14}	1.4310×10^{12}	7.1484×10^{12}	2.1176×10^{8}	3.3961×10^{1}	1.5445×10^{3}
		9	10	12	4	1	2
F19	mean	1.2649×10^{4}	6.0892×10^{4}	3.9509×10^{4}	5.6610×10^{3}	1.9016×10^{3}	1.9011×10^{3}
	std	2.0266×10^{8}	7.6105×10^{10}	1.1141×10^{10}	1.1908×10^{7}	2.2162×10^{0}	1.2741×10^{3}
		8	11	10	5	2	1
F20	mean	2.1447×10^{3}	2.1168×10^{3}	2.2327×10^{3}	2.1823×10^{3}	2.0211×10^{3}	2.0035×10^{3}
	std	4.5367×10^{2}	2.6284×10^{3}	6.0275×10^{2}	5.0733×10^{3}	2.2428×10^{-1}	6.6985×10^{1}
		8	5	11	10	3	2
F21	mean	2.2751×10^{3}	2.2123×10^{3}	2.3847×10^{3}	2.2154×10^{3}	2.3272×10^{3}	2.3064×10^{3}
	std	3.5537×10^{3}	1.5235×10^{2}	5.0672×10^{1}	2.1209×10^{2}	8.0671×10^{1}	2.1668×10^{3}
		5	2	11	3	9	6
F22	mean	2.8376×10^{3}	2.3811×10^{3}	2.5990×10^{3}	2.3120×10^{3}	2.3021×10^{3}	2.3012×10^{3}
	std	3.3572×10^{4}	3.7237×10^{3}	1.3465×10^{4}	3.9777×10^{1}	7.3032×10^{-1}	2.5032×10^{-1}
		12	10	11	6	3	2
F23	mean	2.7203×10^{3}	2.6618×10^{3}	2.6830×10^{3}	2.6433×10^{3}	2.6362×10^{3}	2.6070×10^{3}
	std	1.1179×10^{2}	3.2718×10^{2}	6.1070×10^{1}	4.8438×10^{1}	6.9366×10^{1}	7.3553×10^{-1}
		12	8	11	6	4	1
F24	mean	2.8460×10^{3}	2.6962×10^{3}	2.7983×10^{3}	2.7057×10^{3}	2.7564×10^{3}	2.7382×10^{3}
	std	1.3690×10^{3}	8.4326×10^{3}	7.9795×10^{1}	6.5491×10^{3}	1.8418×10^{2}	4.3491
		12	3	10	4	7	5
F25	mean	3.1454×10^{3}	3.5777×10^{3}	3.1220×10^{3}	2.9265×10^{3}	2.9298×10^{3}	2.9357×10^{3}
	std	1.0907×10^{4}	5.0451×10^{4}	6.6673×10^{3}	7.7418×10^{2}	4.7535×10^{2}	4.0833×10^{2}
		11	12	10	3	5	7
F26	mean	3.4452×10^{3}	3.2354×10^{3}	3.6720×10^{3}	3.1023×10^{3}	2.9352×10^{3}	2.9186×10^{3}
	std	1.1306×10^{5}	5.2268×10^{4}	6.5760×10^{4}	8.9596×10^{3}	2.0497×10^{3}	1.7293×10^{3}
		8	6	12	4	3	2
F27	mean	3.2212×10^{3}	3.1188×10^{3}	3.1365×10^{3}	3.1162×10^{3}	3.0739×10^{3}	3.0972×10^{3}
	std	2.7274×10^{2}	6.2799×10^{2}	1.8776×10^{2}	7.1363×10^{1}	6.1082×10^{1}	8.5967
		12	9	10	8	1	3
F28	mean	3.3000×10^{3}	3.4268×10^{3}	3.4991×10^{3}	3.4119×10^{3}	3.2725×10^{3}	3.2185×10^{3}
	std	6.5701×10^{4}	3.9019×10^{4}	3.9135×10^{3}	1.5440×10^{4}	1.5312×10^{1}	1.8561×10^{4}
		5	10	11	9	4	3
F29	mean	3.3471×10^{3}	3.3572×10^{3}	3.4962×10^{3}	3.3028×10^{3}	3.2498×10^{3}	3.1836×10^{3}
	std	1.5572×10^{3}	1.2516×10^{3}	9.9096×10^{3}	7.6504×10^{3}	1.3627×10^{3}	3.1805×10^{2}
		9	10	12	7	5	3
F30	mean	1.2257×10^{6}	9.5690×10^{6}	6.5465×10^{6}	2.7014×10^{5}	3.3118×10^{3}	4.4055×10^{3}
	std	2.3923×10^{12}	2.5390×10^{12}	2.1162×10^{13}	1.4269×10^{11}	1.6210×10^{5}	1.8826×10^{5}
		8	12	11	6	1	2
Paired rank +\=\−		29/0/0	29/0/0	29/0/0	29/0/0	22/1/6	18/1/10
Avg. rank		9.00	9.41	10.69	6.14	4.10	2.45
Overall rank		10	11	12	6	3	2

Table 20. The average runtime for the CEC2017 benchmark functions.

F	mDBO	WOA	SCA	PSO	GWO	HHO	AOA	BOA	DE	DBO	LSHADE	LSHADE_SPACMA
F1	2.1180×10^{-1}	4.1688×10^{-2}	6.6277×10^{-2}	4.3867×10^{-2}	6.4856×10^{-2}	1.1639×10^{-1}	5.6916×10^{-2}	8.2459×10^{-2}	3.3507×10^{-3}	9.1394×10^{-2}	2.1726×10^{-1}	1.7602×10^{-1}
F3	2.4247×10^{-1}	4.6121×10^{-2}	6.2362×10^{-2}	4.7833×10^{-2}	7.1436×10^{-2}	1.2137×10^{-1}	6.3162×10^{-2}	8.8714×10^{-2}	3.6576×10^{-3}	1.0563×10^{-1}	2.3962×10^{-1}	2.1647×10^{-1}
F4	2.4422×10^{-1}	5.3028×10^{-2}	6.5788×10^{-2}	4.9338×10^{-2}	7.2637×10^{-2}	1.1044×10^{-1}	5.7342×10^{-2}	7.9301×10^{-2}	3.8378×10^{-3}	9.8605×10^{-2}	2.1860×10^{-1}	2.2917×10^{-1}
F5	2.9549×10^{-1}	8.9907×10^{-2}	1.4224×10^{-1}	1.0226×10^{-1}	1.7282×10^{-1}	2.3361×10^{-1}	1.2446×10^{-1}	1.6723×10^{-1}	6.2394×10^{-3}	1.4093×10^{-1}	2.4790×10^{-1}	5.0101×10^{-1}
F6	3.0326×10^{-1}	8.5746×10^{-2}	1.0131×10^{-1}	8.9405×10^{-2}	1.1205×10^{-1}	2.3228×10^{-1}	9.9250×10^{-2}	1.6437×10^{-1}	5.4102×10^{-3}	1.5106×10^{-1}	2.4151×10^{-1}	1.8166×10^{-1}
F7	2.8038×10^{-1}	5.6048×10^{-2}	7.0933×10^{-2}	5.3779×10^{-2}	8.0320×10^{-2}	1.4458×10^{-1}	6.8370×10^{-2}	1.0520×10^{-1}	4.9642×10^{-3}	1.1249×10^{-1}	2.1417×10^{-1}	2.8411×10^{-1}
F8	2.5153×10^{-1}	5.0026×10^{-2}	6.6519×10^{-2}	5.2670×10^{-2}	7.3156×10^{-2}	1.4154×10^{-1}	7.2705×10^{-2}	1.0133×10^{-1}	3.5256×10^{-3}	1.0299×10^{-1}	2.0585×10^{-1}	2.2076×10^{-1}
F9	2.6699×10^{-1}	5.4766×10^{-2}	7.0502×10^{-2}	5.4075×10^{-2}	7.6263×10^{-2}	1.5711×10^{-1}	7.0028×10^{-2}	1.0341×10^{-1}	4.4157×10^{-3}	1.0333×10^{-1}	2.0912×10^{-1}	1.6490×10^{-1}
F10	2.5734×10^{-1}	5.9962×10^{-2}	7.9521×10^{-2}	5.9418×10^{-2}	8.3367×10^{-2}	1.6348×10^{-1}	7.6781×10^{-2}	1.1665×10^{-1}	4.4201×10^{-3}	1.1958×10^{-1}	2.1664×10^{-1}	2.6649×10^{-1}
F11	2.5240×10^{-1}	4.7148×10^{-2}	6.2636×10^{-2}	4.8595×10^{-2}	7.0849×10^{-2}	1.2473×10^{-1}	6.4505×10^{-2}	9.0408×10^{-2}	3.7916×10^{-3}	9.8024×10^{-2}	2.0438×10^{-1}	2.2191×10^{-1}
F12	2.5113×10^{-1}	5.3640×10^{-2}	7.1467×10^{-2}	5.4787×10^{-2}	7.2891×10^{-2}	1.4993×10^{-1}	7.3660×10^{-2}	9.4518×10^{-2}	4.4171×10^{-3}	9.9383×10^{-2}	2.3001×10^{-1}	2.0587×10^{-1}
F13	2.5181×10^{-1}	4.8252×10^{-2}	6.2828×10^{-2}	5.0044×10^{-2}	7.2644×10^{-2}	1.3686×10^{-1}	6.4033×10^{-2}	9.4540×10^{-2}	3.7771×10^{-3}	9.7476×10^{-2}	2.1026×10^{-1}	2.6249×10^{-1}
F14	3.0046×10^{-1}	6.5634×10^{-2}	8.6676×10^{-2}	6.4520×10^{-2}	8.7072×10^{-2}	1.7582×10^{-1}	8.3779×10^{-2}	1.3619×10^{-1}	1.2739×10^{-1}	1.3586×10^{-1}	2.7729×10^{-1}	3.9984×10^{-1}
F15	2.3825×10^{-1}	4.7997×10^{-2}	6.6082×10^{-2}	5.7006×10^{-2}	7.4893×10^{-2}	1.3583×10^{-1}	6.2141×10^{-2}	9.0521×10^{-2}	3.7208×10^{-3}	1.0427×10^{-1}	2.1758×10^{-1}	1.6928×10^{-1}
F16	2.4595×10^{-1}	5.9401×10^{-2}	7.1952×10^{-2}	5.7880×10^{-2}	8.4343×10^{-2}	1.6052×10^{-1}	7.2220×10^{-2}	1.1074×10^{-1}	5.0307×10^{-3}	1.0980×10^{-1}	2.1981×10^{-1}	2.8480×10^{-1}
F17	3.2423×10^{-1}	7.6138×10^{-2}	9.3148×10^{-2}	7.6856×10^{-2}	1.0003×10^{-1}	2.0749×10^{-1}	9.3835×10^{-2}	1.4920×10^{-1}	5.0013×10^{-3}	1.2954×10^{-1}	2.4186×10^{-1}	3.0310×10^{-1}
F18	2.4721×10^{-1}	5.9932×10^{-2}	7.3303×10^{-2}	5.1936×10^{-2}	7.6203×10^{-2}	1.4965×10^{-1}	7.4522×10^{-2}	1.0532×10^{-1}	4.3501×10^{-3}	1.0963×10^{-1}	2.2962×10^{-1}	2.1203×10^{-1}
F19	6.0785×10^{-1}	2.2009×10^{-1}	2.3823×10^{-1}	2.2066×10^{-1}	2.4661×10^{-1}	5.5174×10^{-1}	2.3903×10^{-1}	4.3405×10^{-1}	9.7064×10^{-3}	2.7785×10^{-1}	3.2122×10^{-1}	3.5907×10^{-1}
F20	3.0456×10^{-1}	8.4240×10^{-2}	9.9306×10^{-2}	8.3128×10^{-2}	1.0261×10^{-1}	2.2068×10^{-1}	9.3638×10^{-2}	1.5665×10^{-1}	5.5297×10^{-3}	1.3886×10^{-1}	2.3527×10^{-1}	3.8265×10^{-1}
F21	3.0995×10^{-1}	7.5315×10^{-2}	9.8639×10^{-2}	8.0568×10^{-2}	1.0269×10^{-1}	2.1353×10^{-1}	9.6302×10^{-2}	1.5017×10^{-1}	4.6890×10^{-3}	1.3562×10^{-1}	2.4325×10^{-1}	3.2403×10^{-1}
F22	3.5357×10^{-1}	9.2046×10^{-2}	1.1357×10^{-1}	9.4427×10^{-2}	1.1716×10^{-1}	2.3589×10^{-1}	1.0750×10^{-1}	1.7919×10^{-1}	4.9642×10^{-3}	1.4163×10^{-1}	2.3413×10^{-1}	2.7493×10^{-1}
F23	3.2751×10^{-1}	9.5525×10^{-2}	1.1845×10^{-1}	1.0019×10^{-1}	1.2113×10^{-1}	2.5390×10^{-1}	1.1998×10^{-1}	1.8835×10^{-1}	5.3920×10^{-3}	1.4806×10^{-1}	2.4855×10^{-1}	2.8860×10^{-1}
F24	3.7016×10^{-1}	1.0925×10^{-1}	1.2447×10^{-1}	1.0661×10^{-1}	1.3406×10^{-1}	2.7229×10^{-1}	1.2138×10^{-1}	2.0861×10^{-1}	6.2103×10^{-3}	1.5728×10^{-1}	2.5970×10^{-1}	3.2168×10^{-1}
F25	1.6648×10^{-1}	4.0168×10^{-2}	4.7568×10^{-2}	4.0209×10^{-2}	5.1051×10^{-2}	1.0197×10^{-1}	4.7570×10^{-2}	7.7072×10^{-2}	2.3003×10^{-3}	6.3753×10^{-2}	1.1400×10^{-1}	8.3496×10^{-2}
F26	3.7372×10^{-1}	1.0682×10^{-1}	1.2856×10^{-1}	1.0787×10^{-1}	1.3154×10^{-1}	2.8313×10^{-1}	1.2265×10^{-1}	2.1352×10^{-1}	5.6298×10^{-3}	1.7072×10^{-1}	2.4512×10^{-1}	1.9468×10^{-1}
F27	3.8605×10^{-1}	1.1417×10^{-1}	1.3471×10^{-1}	1.1534×10^{-1}	1.3953×10^{-1}	2.9670×10^{-1}	1.2846×10^{-1}	2.2895×10^{-1}	5.6472×10^{-3}	1.6660×10^{-1}	2.5172×10^{-1}	2.0656×10^{-1}
F28	3.5782×10^{-1}	1.0215×10^{-1}	1.1787×10^{-1}	1.0296×10^{-1}	1.2367×10^{-1}	2.5290×10^{-1}	1.1926×10^{-1}	1.9321×10^{-1}	5.1529×10^{-3}	1.5854×10^{-1}	2.5019×10^{-1}	1.9515×10^{-1}
F29	3.4389×10^{-1}	1.0155×10^{-1}	1.1721×10^{-1}	1.0091×10^{-1}	1.2502×10^{-1}	2.6561×10^{-1}	1.2032×10^{-1}	2.0483×10^{-1}	6.0818×10^{-3}	1.5468×10^{-1}	2.4912×10^{-1}	3.2627×10^{-1}
F30	6.6985×10^{-1}	2.4780×10^{-1}	2.6368×10^{-1}	2.5438×10^{-1}	2.7229×10^{-1}	6.0958×10^{-1}	2.7421×10^{-1}	4.8998×10^{-1}	1.1740×10^{-2}	3.0076×10^{-1}	3.5806×10^{-1}	3.3879×10^{-1}

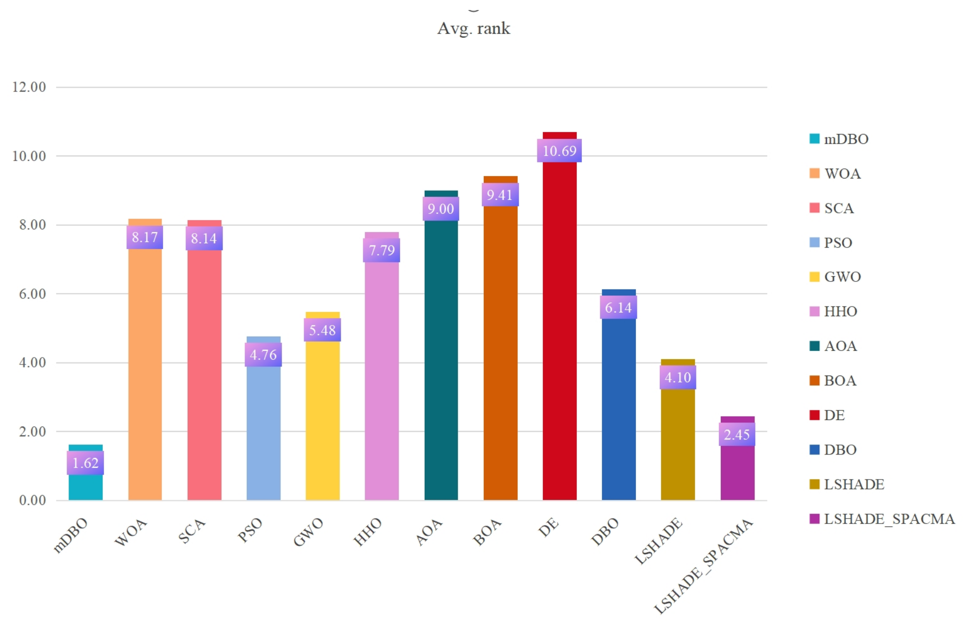


Figure 15. A comparison of algorithms in solving benchmark functions Avg rank.

relative to other algorithms, a non-parametric paired Wilcoxon rank-sum test is conducted. This test is used to assess whether there are significant differences between the two algorithms. The p-values obtained from the Wilcoxon rank-sum test, performed at the 5% significance level, are provided in Table 21. A p-value less than 0.05 indicates a statistically significant difference between the two algorithms. The results presented in the table show that almost most of the p-values are below 0.05, suggesting that the proposed mDBO method statistically outperforms the other algorithms in comparison. Overall, the performance of mDBO in the comparative tests supports its effectiveness in terms of development and exploration, confirming the utility of the three mechanisms introduced in this study.

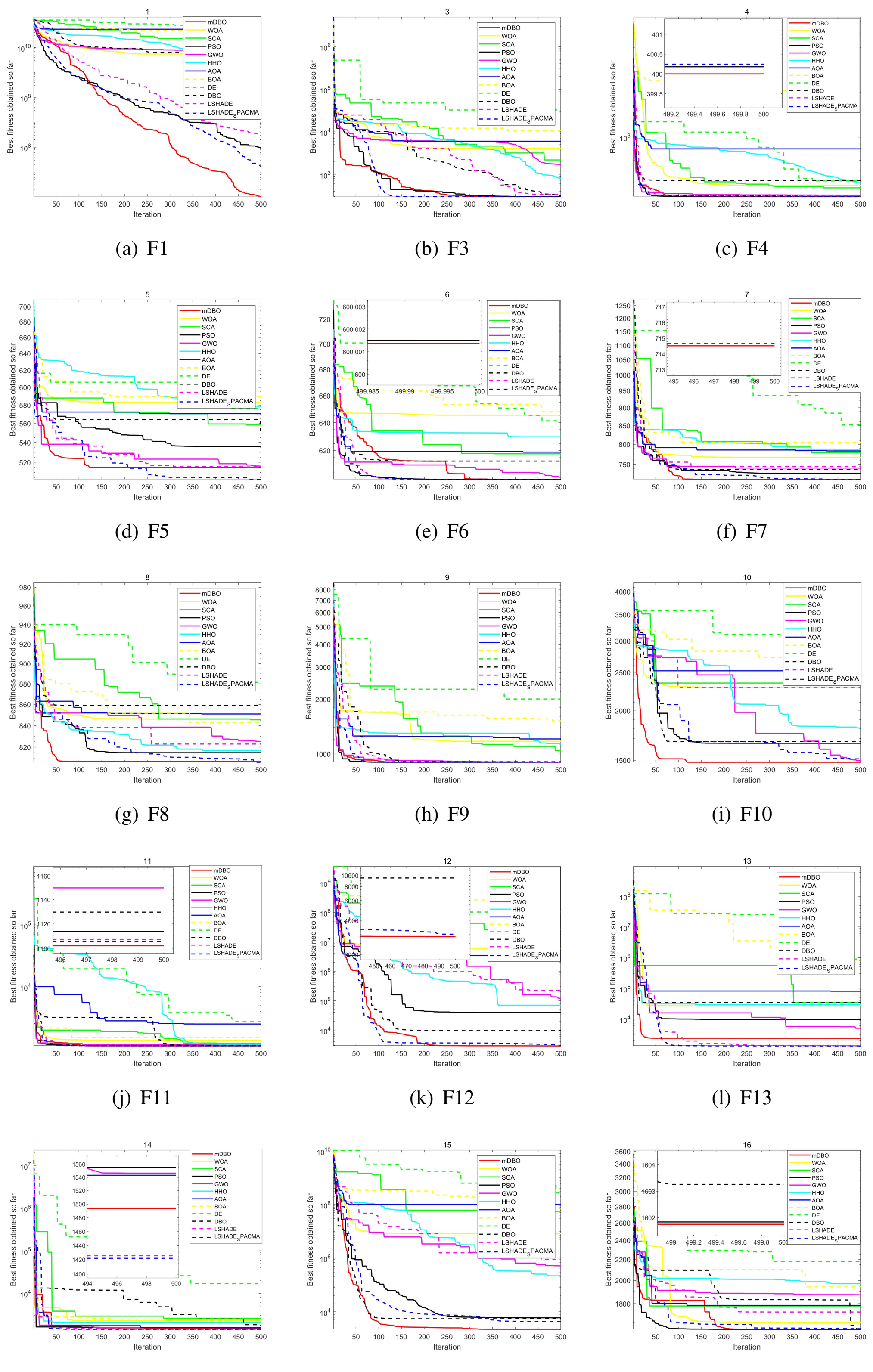
4.7. Convergence behavior analysis

In the CEC2017 benchmark tests, the convergence speed and accuracy of algorithms such as mDBO, WOA, SCA, PSO, GWO, HHO, AOA, BOA, DE, DBO, LSHADE, and LSHADE_SPACMA are illustrated in Figures 16 and 17. Analysis of these curves reveals that mDBO outperforms the other algorithms in terms of convergence speed, displaying minimal fluctuations and high stability. This performance indicates not only a faster approach to optimal solutions but also a significant improvement in efficiency and robustness when addressing complex problems. Moreover, mDBO demonstrates a rapid convergence trend from the beginning of most test problems and continually enhances its search capability as iterations progress. This accelerating convergence trend suggests that mDBO optimizes its search strategy throughout the iterative process, enabling it to quickly hone in on higher-quality solutions. This progressively improving search capability ensures that mDBO not only converges rapidly but also consistently enhances the quality of its solutions in tackling complex optimization challenges.

Analysis of Figures 16 and 17 reveals that mDBO consistently converges to the optimal solution

Table 21. The result of the Wilcoxon rank-sum test for the CEC2017 benchmark functions.

F	WOA	SCA	PSO	GWO	HHO	AOA	BOA	DE	DBO	LSHADE	LSHADE_SPACMA
F1	3.0180×10^{-11}	3.0180×10^{-11}	8.7700×10^{-2}	4.9721×10^{-11}	3.0180×10^{-11}	3.0180×10^{-11}	3.0180×10^{-11}	3.0180×10^{-11}	2.3400×10^{-1}	1.7000×10^{-3}	2.3620×10^{-12}
F3	5.0723×10^{-10}	3.0199×10^{-11}	4.6430×10^{-1}	1.4643×10^{-10}	7.0881×10^{-8}	3.0199×10^{-11}	3.0199×10^{-11}	3.0199×10^{-11}	2.3168×10^{-6}	1.6690×10^{-1}	4.0772×10^{-11}
F4	2.8773×10^{-11}	2.8773×10^{-11}	8.5884×10^{-11}	2.8773×10^{-11}	2.8773×10^{-11}	2.8773×10^{-11}	2.8773×10^{-11}	2.8773×10^{-11}	4.0235×10^{-10}	2.8773×10^{-11}	3.6129×10^{-7}
F5	3.8053×10^{-7}	5.0723×10^{-10}	1.5400×10^{-2}	1.8731×10^{-7}	1.8000×10^{-3}	3.3384×10^{-11}	3.0199×10^{-11}	4.6159×10^{-10}	3.4678×10^{-7}	1.5846×10^{-4}	5.1436×10^{-12}
F6	5.8737×10^{-4}	3.3520×10^{-8}	3.7661×10^{-4}	4.6390×10^{-5}	4.1127×10^{-7}	5.4941×10^{-11}	3.6897×10^{-11}	3.8249×10^{-9}	1.5000×10^{-2}	9.8329×10^{-8}	3.0199×10^{-11}
F7	3.3384×10^{-11}	3.0199×10^{-11}	6.0104×10^{-8}	5.1060×10^{-1}	3.0199×10^{-11}	3.0199×10^{-11}	3.0199×10^{-11}	3.0199×10^{-11}	1.6947×10^{-9}	8.9345×10^{-11}	6.0053×10^{-8}
F8	1.7769×10^{-10}	8.9934×10^{-11}	9.2603×10^{-9}	4.2900×10^{-1}	9.9186×10^{-11}	2.3715×10^{-10}	3.3384×10^{-11}	4.0772×10^{-11}	5.7500×10^{-2}	1.8608×10^{-6}	3.3384×10^{-11}
F9	3.5161×10^{-7}	3.8192×10^{-9}	4.2082×10^{-5}	1.1066×10^{-6}	9.5000×10^{-3}	6.0028×10^{-8}	8.9773×10^{-11}	3.3321×10^{-11}	3.6000×10^{-3}	2.9567×10^{-5}	3.0142×10^{-11}
F10	7.3757×10^{-11}	2.9171×10^{-9}	3.6403×10^{-10}	1.4940×10^{-1}	5.4840×10^{-11}	2.6054×10^{-10}	1.9534×10^{-10}	1.0917×10^{-10}	3.7683×10^{-4}	1.4501×10^{-7}	4.2781×10^{-9}
F11	1.2477×10^{-4}	5.0922×10^{-8}	1.6062×10^{-6}	7.1988×10^{-5}	1.7150×10^{-1}	4.3106×10^{-8}	4.0772×10^{-11}	3.0199×10^{-11}	9.8230×10^{-1}	9.3520×10^{-1}	3.6897×10^{-11}
F12	9.2603×10^{-9}	2.6695×10^{-9}	8.8830×10^{-1}	4.9000×10^{-3}	1.4918×10^{-6}	7.3891×10^{-11}	3.1589×10^{-10}	4.1997×10^{-10}	2.0283×10^{-7}	4.3106×10^{-8}	1.1032×10^{-6}
F13	2.6099×10^{-10}	3.0199×10^{-11}	9.1170×10^{-1}	5.5727×10^{-10}	4.0772×10^{-11}	3.0199×10^{-11}	3.0199×10^{-11}	3.0199×10^{-11}	3.8307×10^{-5}	7.4800×10^{-2}	5.4941×10^{-11}
F14	4.0600×10^{-2}	5.1857×10^{-7}	6.1500×10^{-2}	9.3520×10^{-1}	2.2820×10^{-1}	1.3350×10^{-1}	9.7555×10^{-10}	5.4617×10^{-9}	2.5810×10^{-1}	3.0199×10^{-11}	3.3384×10^{-11}
F15	7.5991×10^{-7}	4.7445×10^{-6}	1.0800×10^{-2}	3.3900×10^{-2}	1.6000×10^{-3}	4.5100×10^{-2}	1.2870×10^{-9}	1.4067×10^{-4}	2.2400×10^{-2}	3.0199×10^{-11}	3.0199×10^{-11}
F16	5.5329×10^{-8}	4.0840×10^{-5}	8.6500×10^{-1}	5.5611×10^{-4}	2.1959×10^{-7}	6.7362×10^{-6}	1.5465×10^{-9}	1.0188×10^{-5}	2.3200×10^{-2}	3.0199×10^{-11}	3.3384×10^{-11}
F17	5.8587×10^{-6}	1.6300×10^{-2}	2.3885×10^{-4}	2.9000×10^{-3}	2.1959×10^{-7}	1.1567×10^{-7}	9.5332×10^{-7}	6.5183×10^{-9}	2.2820×10^{-1}	1.8567×10^{-9}	2.6099×10^{-10}
F18	1.5292×10^{-5}	1.8608×10^{-6}	7.2830×10^{-1}	3.7110×10^{-1}	1.5000×10^{-2}	1.8608×10^{-6}	1.2023×10^{-8}	1.2057×10^{-10}	1.3000×10^{-3}	3.4742×10^{-10}	1.3289×10^{-10}
F19	2.9050×10^{-1}	3.0199×10^{-11}	6.4000×10^{-3}	5.8737×10^{-4}	1.3730×10^{-1}	2.2260×10^{-1}	3.8202×10^{-10}	1.0937×10^{-10}	8.8830×10^{-1}	3.3384×10^{-11}	3.6897×10^{-11}
F20	6.1210×10^{-10}	1.8916×10^{-4}	5.2980×10^{-1}	5.9400×10^{-2}	3.0811×10^{-8}	3.1967×10^{-9}	8.1527×10^{-11}	8.8829×10^{-6}	2.1500×10^{-2}	3.0199×10^{-11}	8.9934×10^{-11}
F21	7.5991×10^{-7}	3.3681×10^{-5}	2.6430×10^{-1}	8.7700×10^{-2}	3.6459×10^{-8}	5.1857×10^{-7}	2.3768×10^{-7}	2.1540×10^{-6}	1.8400×10^{-2}	4.5043×10^{-11}	3.0123×10^{-11}
F22	4.1178×10^{-6}	5.3000×10^{-3}	1.3300×10^{-2}	5.6000×10^{-3}	6.5486×10^{-4}	1.5964×10^{-7}	1.8090×10^{-1}	3.1967×10^{-9}	2.3980×10^{-1}	1.0870×10^{-1}	6.5200×10^{-1}
F23	6.1210×10^{-10}	9.9186×10^{-11}	9.3520×10^{-1}	3.2700×10^{-2}	7.1186×10^{-9}	3.0199×10^{-11}	3.0199×10^{-11}	3.0199×10^{-11}	5.8737×10^{-4}	1.6351×10^{-5}	7.7578×10^{-9}
F24	1.7479×10^{-5}	5.5329×10^{-8}	3.6320×10^{-1}	7.2951×10^{-4}	4.1825×10^{-9}	7.3891×10^{-11}	2.1959×10^{-7}	2.8314×10^{-8}	2.8389×10^{-4}	4.7138×10^{-4}	5.4941×10^{-11}
F25	4.7373×10^{-6}	7.0734×10^{-8}	3.3647×10^{-4}	1.9070×10^{-1}	8.0796×10^{-10}	7.0305×10^{-7}	3.1500×10^{-2}	1.9512×10^{-10}	4.2100×10^{-2}	9.9410×10^{-1}	3.6400×10^{-2}
F26	3.5201×10^{-7}	3.4971×10^{-9}	4.2900×10^{-1}	1.1200×10^{-2}	4.2100×10^{-2}	3.0199×10^{-11}	3.0199×10^{-11}	3.0199×10^{-11}	1.8000×10^{-3}	8.8830×10^{-1}	9.9200×10^{-2}
F27	1.5271×10^{-5}	4.4359×10^{-7}	1.9580×10^{-1}	6.1000×10^{-1}	1.6328×10^{-5}	3.0104×10^{-11}	1.1734×10^{-4}	1.4600×10^{-10}	5.5562×10^{-4}	1.5790×10^{-1}	9.0000×10^{-3}
F28	4.1178×10^{-6}	1.5846×10^{-4}	3.5618×10^{-4}	4.3600×10^{-2}	4.1825×10^{-9}	1.0937×10^{-10}	4.5726×10^{-9}	2.7726×10^{-5}	1.2200×10^{-2}	3.0180×10^{-11}	2.6470×10^{-4}
F29	4.9782×10^{-4}	5.1900×10^{-2}	6.6000×10^{-3}	9.1000×10^{-3}	1.2230×10^{-1}	2.7100×10^{-2}	1.5939×10^{-7}	3.0756×10^{-8}	9.2340×10^{-1}	1.8350×10^{-1}	1.0260×10^{-1}
F30	1.8608×10^{-6}	2.6430×10^{-1}	8.6500×10^{-1}	4.7138×10^{-4}	1.9963×10^{-5}	8.1975×10^{-7}	2.1540×10^{-6}	2.2273×10^{-9}	5.5920×10^{-1}	7.0430×10^{-7}	4.1825×10^{-9}



(a) F1 (b) F3 (c) F4

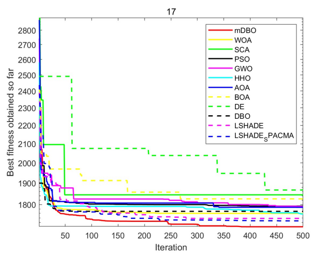
(d) F5 (e) F6 (f) F7

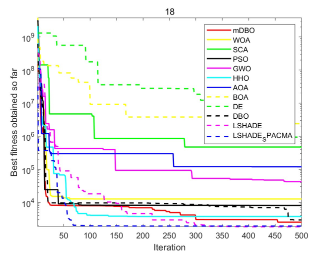
(g) F8 (h) F9 (i) F10

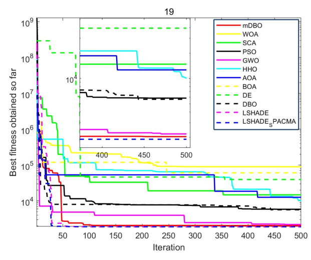
(j) F11 (k) F12 (l) F13

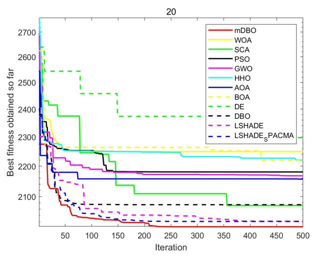
(m) F14 (n) F15 (o) F16

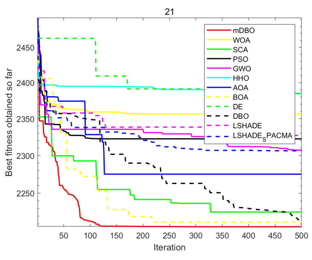
Figure 16. Comparison of convergence curves.

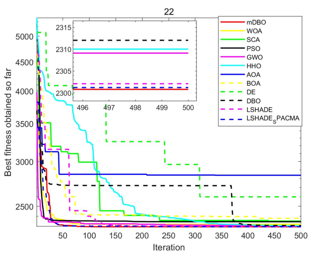

(a) F17

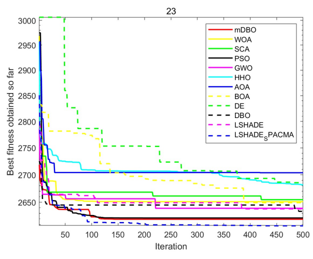

(b) F18

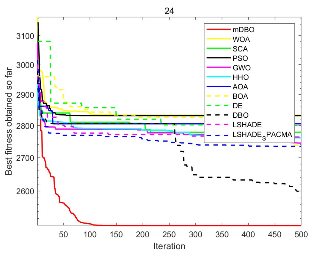

(c) F19

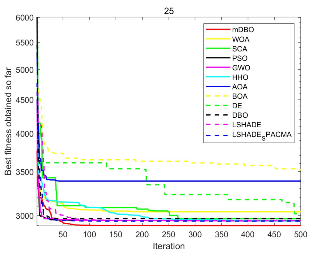

(d) F20

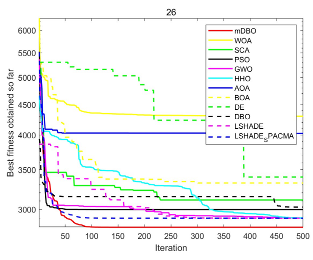

(e) F21

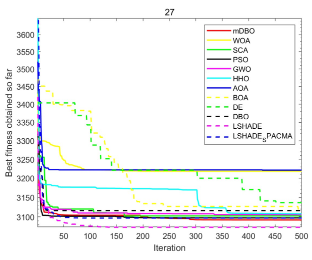

(f) F22

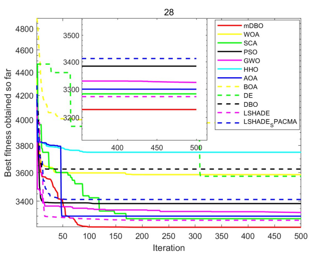

(g) F23

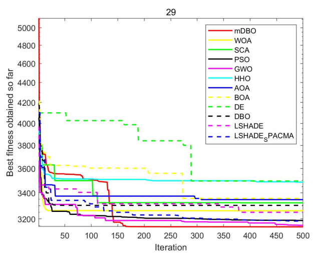

(h) F24


(i) F25


(j) F26


(k) F27


(l) F28


(m) F29

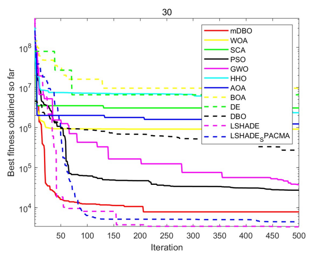

(n) F30

Figure 17. Comparison of convergence curves.

within 500 consecutive iterations across most test cases, exhibiting minimal stagnation. This performance underscores its exceptional exploration and exploitation capabilities. In the context of unimodal functions (F1 and F3), mDBO demonstrates significant improvements in both convergence accuracy and speed compared to DBO, outperforming other optimization algorithms in these respects. For multimodal functions (F4 to F10), mDBO maintains a remarkably high convergence rate and effectively demonstrates the ability to evade local optima. Moreover, among these seven functions, mDBO achieves the highest convergence precision in five cases. LSHADE_SPACMA displays competitive performance on F5. Regarding hybrid functions (F11 to F20), mDBO quickly converges to the optimal position during the initial iterations, highlighting its robust convergence performance. Among these ten functions, mDBO achieves the highest convergence precision, with the exceptions of F12, F14, F18, and F20. In terms of composition functions (F21 to F30), mDBO effectively transitions from the exploration phase to the exploitation phase, outperforming its competitors to varying degrees in both convergence accuracy and speed.

5. Experimental II: Feature selection

5.1. Data acquisition and preprocessing

We select several standard datasets from the UCI Machine Learning Repository. Tab 22 briefly describes these datasets, including the dataset name, number of samples, number of features, and number of category labels. We employ a newly proposed FS method aimed at identifying and selecting key features that contribute to improving classification accuracy. These datasets contain potential patterns that aid in feature selection and classification analysis. Each dataset consists of varying numbers of records (samples or observations) and attributes (features or variables), which are crucial for achieving the research objectives.

Table 22. Description of the dataset.

Dataset	Number of samples	Number of features	Number of category labels
Breastcancer	699	9	2
BreastEW	596	30	2
Congress	435	16	2
Dermatology	366	34	6
Diabets	768	8	2
Exactly	1000	13	2
Glass	214	9	7
HeartEW	269	14	2
ionosphere	351	34	2
Sonar	208	60	2
Tic-tac-toe	958	9	2
Vehicule	846	18	4
Vote	300	16	2
Vowel	901	10	2
WDBC	569	31	2
Wine	178	13	2
Zoo	101	16	7

The collected datasets are carefully examined for missing values and outliers, as these factors can adversely affect the performance of distance-based machine learning models, such as the KNN algorithm. A thorough analysis of the dataset structure is conducted to classify the variables as either categorical or numerical. In classification tasks, input variables typically need to be numerical, while the target variable is treated as a categorical factor. To address the presence of NaN values and non-numeric items, we employ various statistical strategies, including data imputation, exclusion, and variable transformation. Prior to inputting the data into the algorithm, we apply standardization techniques to datasets with large numerical ranges, ensuring that the algorithm can evaluate different features on a consistent scale. Additionally, we utilize a 10-fold cross-validation method for model training and testing, allocating 80% of the data for training and 20% for testing. Throughout this process, we fine-tune the parameters to minimize the loss function, using the test set to evaluate the model's accuracy and overall performance.

5.2. Statistical results and discussion

To assess the performance of the newly developed bmDBO in FS tasks, we conduct experiments on 17 benchmark datasets, with the results presented in Table 23 to Table 26 and Figure 19. To validate the effectiveness of bmDBO, we compare it against several wrapper-based feature selection algorithms, which serve as baseline methods. The comparison reveals that bmDBO exhibits substantial effectiveness in addressing feature selection challenges. In this study, we utilize nine state-of-the-art alternative algorithms for comparison, including: The binary particle swarm optimization (bPSO) [78], the binary genetic algorithm (bGA), the binary differential evolution (bDE) [79], the binary black chimpanzee optimization algorithm (bBOA) [70], the binary grey wolf optimizer (bGWO) [80], the binary whale optimization algorithm (bWOA) [24], the binary ant colony optimization (bACO) [81], the binary Harris Hawks optimization (bHHO) [82], and the original binary dung beetle optimizer (bDBO). Throughout the experiments, we adhere to the default parameter settings outlined in the relevant literature for these algorithms, ensuring a fair comparison.

Table 23 presents the statistical results of optimal fitness values, with the best outcomes under each evaluation criterion highlighted in bold. Over 30 independent runs, the bmDBO algorithm identifies the most effective feature subsets in 12 of 17 datasets, representing a significant proportion of 66.67%. This finding is compelling and underscores the bmDBO algorithm's robust capability to discover high-precision solutions amidst numerous combinations, aligning with its outstanding performance in benchmark tests. In comparison, the bPSO, bACO, and bHHO algorithms achieve effectiveness rankings of 58.82%, 64.71%, and 58.82%, respectively, across the same 17 datasets, indicating their competency in feature selection tasks. However, bmDBO's superior performance distinguishes it as a highly competitive method for addressing other feature selection challenges. In terms of average fitness rankings, bmDBO leads with a score of 1.41, followed closely by bPSO at 2.29. The remaining algorithms are ranked from third to tenth as follows: bACO (2.65), bHHO (2.82), bDE (4.18), bDBO (4.24), bGA (4.29), bWPA (7.71), bBOA (8.59), and bGWO (8.71).

Table 24 presents the average number of features selected across 30 computational runs for the bmDBO algorithm and several competing methods. The algorithms' performance in selecting the smallest feature subsets from standard datasets is evaluated and ranked for each dataset. The overall ranking of feature subset sizes is obtained by averaging the rankings across the candidate algorithms. Notably, all algorithms effectively reduce the dimensionality of the datasets. While the bGWO and bWOA algorithms demonstrate relatively lower classification accuracy, they perform well in terms of dimensionality reduction. The bmDBO algorithm ranks fifth in feature subset size but achieves the highest ranking for accuracy, highlighting the improved DBO algorithm's capability to balance classification accuracy and dimensionality reduction effectively.

Furthermore, we assess the classification accuracy and F1 score for the feature subsets selected by each algorithm to further validate their effectiveness in identifying key features. Tables 25 and 26 present detailed average accuracy and F1 score data for each algorithm, respectively, utilizing the KNN classification method with ($k = 5$) and the 10-fold cross-validation approach. It is noteworthy that higher numerical values reflect better performance for each algorithm, with the optimal values highlighted in bold. A comparative analysis of the data in these tables clearly demonstrates that our proposed method outperforms the others.

As shown in Table 25 and Figure 18, the bmDBO algorithm exhibits superior performance in terms of average accuracy across 15 of 17 datasets. In comparison, the bACO and bHHO algorithms outperform others in 12 datasets each, while the bPSO and bDE algorithms excel in 11 and 8 datasets, respectively. The bGA and original DBO algorithm (designated as bDBO) demonstrate good performance in 7 and 6 datasets, respectively. The bWOA algorithm achieves better results in 4 datasets, whereas the bGWO and bBOA algorithms excel in only one dataset each. Overall, the bmDBO algorithm consistently ranks highest for average accuracy compared to the other methods, illustrating its robust capability to identify key features that enhance model accuracy, even when its feature subset size is not the smallest among competing algorithms. For example, in the Congress dataset, the bWOA algorithm selects the smallest average feature subset size (4.6), while bmDBO has a size of 6.4; nevertheless, bmDBO achieves a higher average accuracy of 98.62%, compared to bWOA's 97.93%. In another instance, the Sonar dataset demonstrates that while bGWO selects an average of 10.4 features, bmDBO's count is 13.8, though their respective average accuracies are 91.22% and 98.05%, highlighting the superiority of the newly proposed bmDBO method. The observed trends are consistent across multiple datasets. Significantly, in all cases where bmDBO produces the smallest average feature subset size, it also achieves the highest average accuracy. This demonstrates that the proposed algorithm effectively strikes a balance between enhancing accuracy and reducing feature subset size.

Based on the analysis of data presented in Table 26, the bmDBO algorithm demonstrates superior performance compared to other algorithms. Evaluated using the F1 score, bmDBO outperforms in 12 of 17 datasets and overall achieves the highest F1 score. In comparison, the bPSO, bACO, and bHHO algorithms attain optimal F1 scores in 8, 10, and 10 datasets, respectively, while bGA and bDE excel in 7 datasets each. Additionally, the bDBO algorithm performs best in 6 datasets, whereas bWOA and bGWO achieve their highest scores in 4 and 2 datasets, respectively. The bBOA algorithm reaches the highest score in only 1 dataset. Considering the average performance rankings across all datasets, our optimized method, which effectively selects key features from large datasets to achieve high accuracy, emerges as the ideal choice. Specifically, bmDBO ranks first with an average F1 score of 1.35. The remaining algorithms are ranked in descending order: bPSO (2.18), bACO (2.18), bHHO (2.35), bGA (3.47), bDE (3.71), bDBO (5.06), bWOA (6.47), bGWO (7.47), and bBOA (8.41).

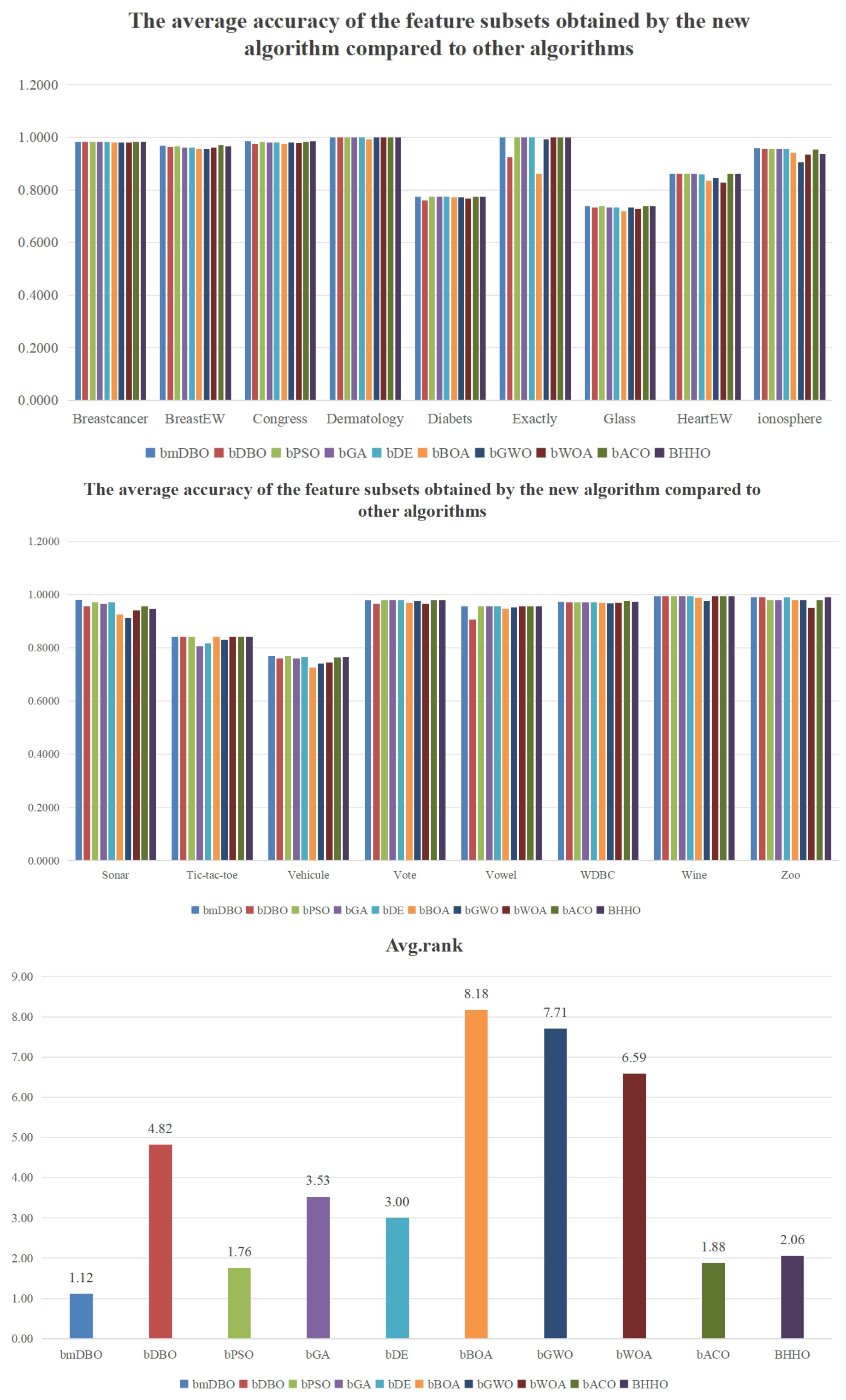


Figure 18. The average accuracy of the feature subsets obtained by the new algorithm compared to other algorithms.

Table 23. The optimal fitness of the new algorithm compared to other methods.

Dataset	Type	bmDBO	bDBO	bPSO	bGA	bDE	bBOA	bGWO	bWOA	bACO	bHHO
Breastcancer	Avg	2.1316×10^{-2}	2.1316×10^{-2}	2.1316×10^{-2}	2.1316×10^{-2}	2.1760×10^{-2}	2.2740×10^{-2}	2.4831×10^{-2}	2.2962×10^{-2}	2.1316×10^{-2}	2.1316×10^{-2}
	Rank	1	1	1	1	7	8	10	9	1	1
BreastEW	Avg	3.3406×10^{-2}	3.6778×10^{-2}	3.5025×10^{-2}	3.9882×10^{-2}	3.9882×10^{-2}	4.5605×10^{-2}	4.6053×10^{-2}	3.9949×10^{-2}	3.1588×10^{-2}	3.5559×10^{-2}
	Rank	2	5	3	6	6	9	10	8	1	4
Congress	Avg	1.7655×10^{-2}	1.9306×10^{-2}	1.9181×10^{-2}	2.1082×10^{-2}	2.1207×10^{-2}	2.6509×10^{-2}	2.2457×10^{-2}	2.3733×10^{-2}	1.9306×10^{-2}	1.7530×10^{-2}
	Rank	2	4	3	6	7	10	8	9	4	1
Dermatology	Avg	3.0588×10^{-3}	3.5882×10^{-3}	2.8824×10^{-3}	2.7647×10^{-3}	2.6471×10^{-3}	1.1961×10^{-2}	5.1176×10^{-3}	3.6471×10^{-3}	3.8824×10^{-3}	3.8235×10^{-3}
	Rank	4	5	3	2	1	10	9	6	8	7
Diabets	Avg	2.2863×10^{-1}	2.3197×10^{-1}	2.2863×10^{-1}	2.2888×10^{-1}	2.2888×10^{-1}	2.2968×10^{-1}	2.3093×10^{-1}	2.3385×10^{-1}	2.2863×10^{-1}	2.2863×10^{-1}
	Rank	1	9	1	5	5	7	8	10	1	1
Exactly	Avg	4.6154×10^{-3}	8.0096×10^{-2}	4.6154×10^{-3}	4.6154×10^{-3}	4.6154×10^{-3}	1.4163×10^{-1}	1.2997×10^{-2}	4.6154×10^{-3}	4.6154×10^{-3}	4.6154×10^{-3}
	Rank	1	9	1	1	1	10	8	1	1	1
Glass	Avg	2.6440×10^{-1}	2.6867×10^{-1}	2.6440×10^{-1}	2.6889×10^{-1}	2.6889×10^{-1}	2.8237×10^{-1}	2.7000×10^{-1}	2.7338×10^{-1}	2.6440×10^{-1}	2.6440×10^{-1}
	Rank	1	5	1	6	7	10	8	9	1	1
HeartEW	Avg	1.4040×10^{-1}	1.4040×10^{-1}	1.4040×10^{-1}	1.4040×10^{-1}	1.4397×10^{-1}	1.6771×10^{-1}	1.5777×10^{-1}	1.7392×10^{-1}	1.4040×10^{-1}	1.4040×10^{-1}
	Rank	1	1	1	1	7	9	8	10	1	1
ionosphere	Avg	4.0600×10^{-2}	4.3605×10^{-2}	4.4487×10^{-2}	4.4076×10^{-2}	4.4193×10^{-2}	5.7807×10^{-2}	9.8225×10^{-2}	6.6234×10^{-2}	4.6492×10^{-2}	6.4817×10^{-2}
	Rank	1	2	5	3	4	7	10	9	6	8
Sonar	Avg	2.1617×10^{-2}	4.6463×10^{-2}	3.2276×10^{-2}	3.5905×10^{-2}	3.1076×10^{-2}	7.4739×10^{-2}	9.2427×10^{-2}	5.9685×10^{-2}	4.5530×10^{-2}	5.6789×10^{-2}
	Rank	1	6	3	4	2	9	10	8	5	7
Tic-tac-toe	Avg	1.6653×10^{-1}	1.6653×10^{-1}	1.6653×10^{-1}	1.9911×10^{-1}	1.8771×10^{-1}	1.6653×10^{-1}	1.7705×10^{-1}	1.6653×10^{-1}	1.6653×10^{-1}	1.6653×10^{-1}
	Rank	1	1	1	10	9	1	8	1	1	1
Vehicule	Avg	2.3346×10^{-1}	2.4406×10^{-1}	2.3335×10^{-1}	2.4166×10^{-1}	2.3592×10^{-1}	2.7597×10^{-1}	2.6280×10^{-1}	2.5751×10^{-1}	2.3815×10^{-1}	2.3636×10^{-1}
	Rank	2	7	1	6	3	10	9	8	5	4
Vote	Avg	2.2675×10^{-2}	2.6975×10^{-2}	2.2675×10^{-2}	2.2675×10^{-2}	2.2800×10^{-2}	3.2450×10^{-2}	2.8100×10^{-2}	3.6000×10^{-2}	2.2675×10^{-2}	2.2800×10^{-2}
	Rank	1	7	1	1	5	9	8	10	1	5
Vowel	Avg	5.1300×10^{-2}	5.1300×10^{-2}	5.1300×10^{-2}	5.2200×10^{-2}	5.1300×10^{-2}	6.2000×10^{-2}	5.6300×10^{-2}	5.2200×10^{-2}	5.1300×10^{-2}	5.1300×10^{-2}
	Rank	1	1	1	7	1	10	9	7	1	1
WDBC	Avg	2.7444×10^{-2}	2.9519×10^{-2}	2.9261×10^{-2}	2.9197×10^{-2}	2.9068×10^{-2}	3.1852×10^{-2}	3.5088×10^{-2}	3.0820×10^{-2}	2.4650×10^{-2}	2.8993×10^{-2}
	Rank	2	7	6	5	4	9	10	8	1	3
Wine	Avg	8.1187×10^{-3}	8.1187×10^{-3}	8.1187×10^{-3}	8.1187×10^{-3}	8.1187×10^{-3}	1.3930×10^{-2}	2.6936×10^{-2}	8.2725×10^{-3}	8.1187×10^{-3}	8.1187×10^{-3}
	Rank	1	1	1	1	1	9	10	8	1	1
Zoo	Avg	1.2775×10^{-2}	1.2775×10^{-2}	2.2300×10^{-2}	2.2425×10^{-2}	1.2775×10^{-2}	2.3050×10^{-2}	2.2925×10^{-2}	5.1750×10^{-2}	2.2300×10^{-2}	1.2775×10^{-2}
	Rank	1	1	6	8	1	9	5	10	6	1
Avg.rank		1.41	4.24	2.29	4.29	4.18	8.59	8.71	7.71	2.65	2.82
Overall rank		1	6	2	7	5	9	10	8	3	4

Table 24. The average feature selection size of the new algorithm compared to other algorithms.

Dataset	Type	bmDBO	bDBO	bPSO	bGA	bDE	bBOA	bGWO	bWOA	bACO	bHHO
Breastcancer	Avg	3.8	3.8	3.8	4.2	3.8	4.4	4	3.8	3.8	3.8
	Rank	1	1	1	9	1	10	8	1	1	1
BreastEW	Avg	5.6	5.2	5.2	4	5.4	12	4.2	4	5.4	6.8
	Rank	8	4	4	1	6	10	3	1	6	9
Congress	Avg	6.4	5.6	5.2	4.8	6	6.8	5.2	4.6	5.4	6.2
	Rank	9	6	3	2	7	10	3	1	5	8
Dermatology	Avg	10.4	12.2	9.8	9	13	17.4	12.4	9.4	13.2	13
	Rank	4	5	2	1	7	10	6	2	9	7
Diabets	Avg	3.8	4.6	3.8	4	3.6	4.6	2.8	4	3.8	3.8
	Rank	3	9	3	7	2	9	1	7	3	3
Exactly	Avg	6	7.6	6	6	7.8	6.6	6	6	6	6
	Rank	1	9	1	1	10	8	1	1	1	1
Glass	Avg	4.6	4.2	4.6	4.4	3.8	5.4	4.2	4.4	4.6	4.6
	Rank	6	2	6	4	1	10	2	4	6	6
HeartEW	Avg	5	5	5	5.2	5	5.4	4.2	5	5	5
	Rank	2	2	2	9	2	10	1	2	2	2
ionosphere	Avg	3.4	4	7	6	4.2	16.6	4	5.6	4.2	8.8
	Rank	1	2	8	7	4	10	2	6	4	9
Sonar	Avg	13.8	18	19.8	12.6	13.8	33	10.4	12.6	12.4	22
	Rank	5	7	8	3	5	10	1	3	2	9
Tic-tac-toe	Avg	9	9	9	6.6	9	8.2	9	6.6	9	9
	Rank	3	3	3	1	3	2	3	1	3	3
Vehicule	Avg	9	11.2	8.8	9.2	9.6	11.2	8	9	9	10
	Rank	3	9	2	6	7	9	1	3	3	8
Vote	Avg	4.6	6	4.6	4.8	4.4	8	4.8	4.6	4.6	4.8
	Rank	2	9	2	6	1	10	6	2	2	6
Vowel	Avg	8.4	7.8	8.4	8.4	9.2	9	8.2	8.2	8.4	8.4
	Rank	4	1	4	4	10	9	2	2	4	4
WDBC	Avg	3.6	4.6	3.8	3.2	6.4	11	3.2	3.6	5.8	8.4
	Rank	3	6	5	1	8	10	1	3	7	9
Wine	Avg	3.2	3.2	3.2	3.2	3.4	5.6	3.4	3.2	3.2	3.2
	Rank	1	1	1	1	8	10	8	1	1	1
Zoo	Avg	4.6	4.6	4	4.4	5.2	5	3.6	4.2	4	4.6
	Rank	6	6	2	5	10	9	1	4	2	6
Avg.rank		3.65	4.82	3.35	4.00	5.41	9.18	2.94	2.59	3.59	5.41
Overall rank		5	7	3	6	9	10	2	1	4	8

Table 25. The average accuracy of the feature subsets obtained by the new algorithm compared to other algorithms.

Dataset	Type	bmDBO	bDBO	bPSO	bGA	bDE	bBOA	bGWO	bWOA	bACO	bHHO
Breastcancer	Avg	0.9827	0.9827	0.9827	0.9827	0.9827	0.9813	0.9799	0.9813	0.9827	0.9827
	Rank	1	1	1	1	1	8	10	8	1	1
BreastEW	Avg	0.9681	0.9646	0.9664	0.9611	0.9611	0.9558	0.9575	0.9611	0.9699	0.9664
	Rank	2	5	3	6	6	10	9	6	1	3
Congress	Avg	0.9862	0.9747	0.9839	0.9816	0.9816	0.9770	0.9816	0.9793	0.9839	0.9862
	Rank	1	10	3	5	5	9	5	8	3	1
Dermatology	Avg	1	1	1	1	1	0.9918	1	1	1	1
	Rank	1	1	1	1	1	10	1	1	1	1
Diabets	Avg	0.7739	0.7595	0.7739	0.7739	0.7739	0.7725	0.7725	0.7673	0.7739	0.7739
	Rank	1	10	1	1	1	7	7	9	1	1
Exactly	Avg	1	0.925	1	1	1	0.863	0.992	1	1	1
	Rank	1	9	1	1	1	10	8	1	1	1
Glass	Avg	0.7381	0.7333	0.7381	0.7333	0.7333	0.7190	0.7333	0.7286	0.7381	0.7381
	Rank	1	5	1	5	5	10	5	9	1	1
HeartEW	Avg	0.8621	0.8621	0.8621	0.8621	0.8586	0.8345	0.8448	0.8276	0.8621	0.8621
	Rank	1	1	1	1	7	9	8	10	1	1
ionosphere	Avg	0.96	0.9571	0.9571	0.9571	0.9571	0.9429	0.9057	0.9343	0.9543	0.9371
	Rank	1	2	2	2	2	7	10	9	2	8
Sonar	Avg	0.9805	0.9561	0.9707	0.9659	0.9707	0.9268	0.9122	0.9415	0.9561	0.9463
	Rank	1	5	2	4	2	9	10	8	5	7
Tic-tac-toe	Avg	0.8419	0.8419	0.8419	0.8063	0.8178	0.8419	0.8304	0.8419	0.8419	0.8419
	Rank	1	1	1	10	9	1	8	1	1	1
Vehicule	Avg	0.7692	0.7598	0.7692	0.7609	0.7669	0.7266	0.7408	0.7444	0.7645	0.7669
	Rank	1	7	1	6	3	10	9	8	5	3
Vote	Avg	0.98	0.9667	0.98	0.98	0.98	0.97	0.9767	0.9667	0.98	0.98
	Rank	1	9	1	1	1	8	7	9	1	1
Vowel	Avg	0.9567	0.9078	0.9567	0.9556	0.9567	0.9467	0.9522	0.9556	0.9567	0.9567
	Rank	1	10	1	6	1	9	9	6	1	1
WDBC	Avg	0.9735	0.9717	0.9717	0.9717	0.9717	0.9699	0.9681	0.9699	0.9770	0.9735
	Rank	2	4	4	4	4	8	10	8	1	2
Wine	Avg	0.9943	0.9943	0.9943	0.9943	0.9943	0.9886	0.9771	0.9943	0.9943	0.9943
	Rank	1	1	1	1	1	9	10	1	1	1
Zoo	Avg	0.99	0.99	0.98	0.98	0.99	0.98	0.98	0.95	0.98	0.99
	Rank	1	1	5	5	1	5	5	10	5	1
Avg.rank		1.12	4.82	1.76	3.53	3.00	8.18	7.71	6.59	1.88	2.06
Overall rank		1	7	2	6	5	10	9	8	3	4

Table 26. The average F1 score of the features selected by the new algorithm compared to other algorithms

Dataset	Type	bmDBO	bDBO	bPSO	bGA	bDE	bBOA	bGWO	bWOA	bACO	bHHO
Breastcancer	Avg	0.9815	0.9815	0.9815	0.9815	0.9812	0.9799	0.9781	0.9797	0.9815	0.9815
	Rank	1	1	1	1	7	8	10	9	1	1
BreastEW	Avg	0.9659	0.9618	0.9638	0.9580	0.9580	0.9520	0.9541	0.9578	0.9676	0.9639
	Rank	2	5	4	6	7	10	9	8	1	3
Congress	Avg	0.9858	0.9740	0.9836	0.9814	0.9810	0.9767	0.9811	0.9789	0.9836	0.9858
	Rank	1	10	3	5	7	9	6	8	3	1
Dermatology	Avg	1	1	1	1	1	0.9913	1	1	1	1
	Rank	1	1	1	1	1	10	1	1	1	1
Diabets	Avg	0.7386	0.7214	0.7386	0.7391	0.7382	0.7366	0.7376	0.7313	0.7386	0.7386
	Rank	2	10	2	1	6	8	7	9	2	2
Exactly	Avg	1	0.9108	1	1	1	0.8300	0.9899	1	1	1
	Rank	1	9	1	1	1	10	8	1	1	1
Glass	Avg	0.7021	0.6770	0.7021	0.7021	0.7021	0.6714	0.7048	0.6765	0.7021	0.7021
	Rank	2	8	2	2	2	10	1	9	2	2
HeartEW	Avg	0.8527	0.8527	0.8527	0.8527	0.8491	0.8217	0.8329	0.8138	0.8527	0.8527
	Rank	1	1	1	1	7	9	8	10	1	1
ionosphere	Avg	0.9431	0.9431	0.9431	0.9558	0.9570	0.9294	0.8851	0.9413	0.9413	0.9129
	Rank	3	3	3	2	1	8	10	3	3	9
Sonar	Avg	0.9811	0.9582	0.9709	0.9664	0.9710	0.9280	0.9157	0.9422	0.9561	0.9487
	Rank	1	5	3	4	2	9	10	8	6	7
Tic-tac-toe	Avg	0.8348	0.8348	0.8348	0.7899	0.8025	0.8348	0.8191	0.8348	0.8348	0.8348
	Rank	1	1	1	10	9	1	8	1	1	1
Vehicule	Avg	0.7798	0.7691	0.7789	0.7720	0.7787	0.7403	0.7475	0.7570	0.7717	0.7762
	Rank	1	7	2	5	3	10	8	8	6	4
Vote	Avg	0.9786	0.9654	0.9786	0.9786	0.9786	0.9685	0.9754	0.9640	0.9786	0.9786
	Rank	1	9	1	1	1	8	7	10	1	1
Vowel	Avg	0.9561	0.9418	0.9561	0.9554	0.9561	0.9490	0.9512	0.9554	0.9561	0.9561
	Rank	1	10	1	6	1	9	8	6	1	1
WDBC	Avg	0.9721	0.9704	0.9704	0.9702	0.9702	0.9683	0.9666	0.9683	0.9759	0.9721
	Rank	2	4	4	6	6	9	10	8	1	3
Wine	Avg	0.9945	0.9945	0.9945	0.9945	0.9945	0.9872	0.9731	0.9945	0.9945	0.9945
	Rank	1	1	1	1	1	9	10	1	1	1
Zoo	Avg	0.9857	0.9857	0.9738	0.9738	0.9857	0.9738	0.9738	0.9438	0.9738	0.9857
	Rank	1	1	6	6	1	6	6	10	5	1
Avg.rank		1.35	5.06	2.18	3.47	3.71	8.41	7.47	6.47	2.18	2.35
Overall rank		1	7	2	5	6	10	9	8	2	4

Table 27. The result of the Wilcoxon rank-sum test by the new algorithm compared to other algorithms.

Dataset	bDBO	bPSO	bGA	bDE	bBOA	bGWO	bWOA	bACO	bHHO
Breastcancer	6.0658×10^{-11}	2.8290×10^{-8}	7.3803×10^{-10}	1.2541×10^{-7}	3.0199×10^{-11}	1.5581×10^{-8}	3.0103×10^{-7}	3.0199×10^{-11}	3.0199×10^{-11}
BreastEW	3.0199×10^{-11}	3.0199×10^{-11}	1.4294×10^{-8}	3.6897×10^{-11}	2.6806×10^{-4}	8.1465×10^{-5}	1.8608×10^{-6}	4.3764×10^{-1}	3.0199×10^{-11}
Congress	2.1482×10^{-10}	7.5390×10^{-11}	3.0059×10^{-4}	5.2640×10^{-4}	3.0199×10^{-11}	2.3885×10^{-4}	8.0727×10^{-1}	3.0199×10^{-11}	1.0277×10^{-6}
Dermatology	5.1060×10^{-1}	6.1210×10^{-10}	3.0103×10^{-7}	8.8829×10^{-6}	8.1014×10^{-10}	3.0103×10^{-7}	8.4848×10^{-9}	1.5465×10^{-9}	2.1544×10^{-10}
Diabets	1.4412×10^{-2}	4.9752×10^{-11}	6.5261×10^{-7}	4.6159×10^{-10}	2.8716×10^{-10}	2.8790×10^{-6}	5.9706×10^{-5}	8.8411×10^{-7}	3.5923×10^{-5}
Exactly	7.6973×10^{-4}	3.6441×10^{-11}	6.5204×10^{-1}	3.3520×10^{-8}	5.8587×10^{-6}	9.9186×10^{-11}	3.6322×10^{-1}	2.1506×10^{-2}	3.5923×10^{-5}
Glass	1.7290×10^{-6}	1.1364×10^{-11}	1.8577×10^{-1}	8.8910×10^{-10}	6.5277×10^{-8}	1.2023×10^{-8}	2.6015×10^{-8}	1.8608×10^{-6}	8.1014×10^{-10}
HeartEW	3.5547×10^{-1}	1.2870×10^{-9}	1.6955×10^{-2}	8.1465×10^{-5}	4.6390×10^{-5}	1.9883×10^{-2}	5.7460×10^{-2}	2.3897×10^{-8}	3.0418×10^{-1}
ionosphere	2.8378×10^{-1}	8.8829×10^{-6}	4.6756×10^{-2}	5.2978×10^{-1}	4.4440×10^{-7}	2.2257×10^{-1}	6.0971×10^{-3}	4.6427×10^{-1}	3.1466×10^{-2}
Sonar	1.3703×10^{-3}	2.2780×10^{-5}	3.3242×10^{-6}	2.0338×10^{-9}	6.6955×10^{-11}	1.5798×10^{-1}	1.0407×10^{-4}	9.4683×10^{-3}	6.5486×10^{-4}
Tic-tac-toe	5.4933×10^{-1}	3.1830×10^{-1}	1.6285×10^{-2}	6.5261×10^{-7}	1.0666×10^{-7}	4.2039×10^{-1}	1.4733×10^{-7}	1.1674×10^{-5}	3.6459×10^{-8}
Vehicule	1.9112×10^{-2}	2.2360×10^{-2}	3.1967×10^{-9}	1.6947×10^{-9}	1.5581×10^{-8}	6.3533×10^{-2}	7.2951×10^{-4}	7.7725×10^{-9}	3.3874×10^{-2}
Vote	6.9724×10^{-3}	1.1199×10^{-1}	8.4848×10^{-9}	3.0199×10^{-11}	3.0199×10^{-11}	2.9205×10^{-2}	1.1567×10^{-7}	3.0199×10^{-11}	3.6322×10^{-1}
Vowel	1.7145×10^{-1}	1.0547×10^{-1}	7.1988×10^{-5}	8.1527×10^{-11}	4.6390×10^{-5}	2.6077×10^{-2}	9.2603×10^{-9}	3.0199×10^{-11}	8.6634×10^{-5}
WDBC	7.5057×10^{-1}	8.5330×10^{-1}	5.4933×10^{-1}	1.0407×10^{-4}	8.8829×10^{-6}	1.1711×10^{-2}	9.4673×10^{-3}	2.1506×10^{-2}	2.2360×10^{-2}
Wine	8.0727×10^{-1}	8.5000×10^{-2}	7.7312×10^{-1}	2.8314×10^{-8}	3.7704×10^{-4}	5.5611×10^{-4}	2.4327×10^{-5}	2.2539×10^{-4}	1.1747×10^{-4}
Zoo	4.2896×10^{-1}	9.5207×10^{-4}	3.0922×10^{-4}	3.8202×10^{-10}	2.8790×10^{-6}	6.2040×10^{-1}	3.7782×10^{-2}	8.5641×10^{-4}	1.8090×10^{-1}

The findings from experiment two conclusively demonstrate that the newly introduced bmDBO algorithm is an effective optimizer specifically designed for real-world FS problems. The exceptional performance of bmDBO can be attributed to its ability to achieve an optimal balance between exploration and exploitation. This characteristic enables the algorithm to efficiently navigate through potential solution spaces in search of optimal subsets. Given bmDBO's impressive capabilities in feature selection, it is expected to deliver favorable results when applied to complex optimization models, including those related to path planning and parameter tuning.

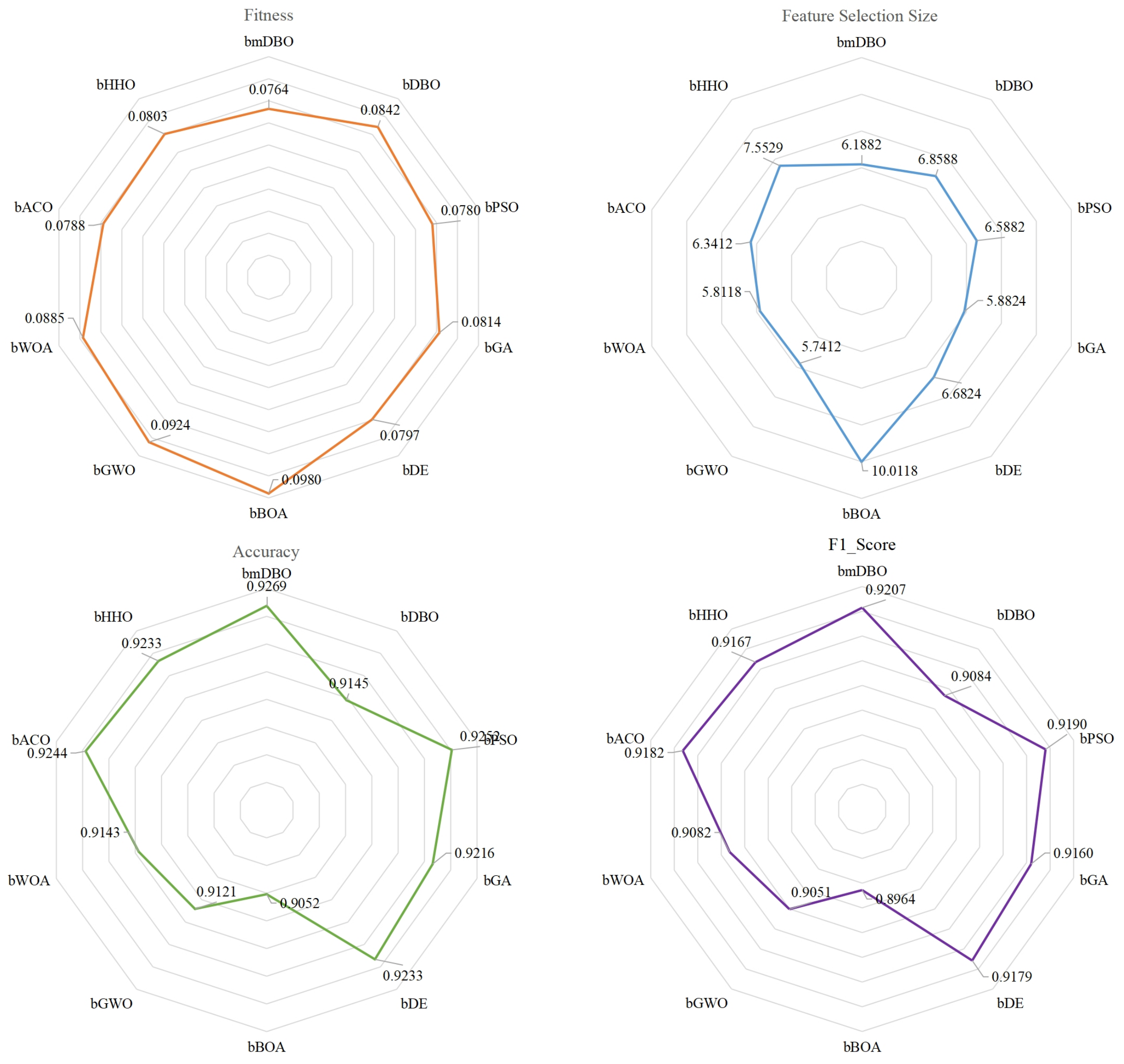


Figure 19. Radar chart comparing different algorithms on all datasets.

6. Conclusions and future work

In this paper, we introduce a multi-strategy integrated Dung Beetle Optimization algorithm, designated as mDBO, which enhances the original DBO algorithm. mDBO incorporates three key strategies. First, a novel population initialization method is introduced, which integrates a hybrid tent-sine mapping with ROBL to generate a more uniformly distributed initial population, thereby enhancing the quality of population distribution within the search space. Subsequently, a new differential evolution mutation strategy is proposed that incorporates a periodic retrospective adaptive mutation factor. This strategy improves the traditional foraging update process by employing a dynamic differential evolution mechanism, which effectively enhances the algorithm's ability to escape local optima and expands the search space for potential candidate solutions. Additionally, an innovative approximation strategy based on Padé approximation techniques and an adaptive evolutionary boundary constraint method is designed and integrated into the framework of the dung beetle optimizer, significantly improving the solution accuracy and overall quality of the population. Following this, a transformation function is employed to convert the optimized DBO algorithm into a binary format, resulting in the enhanced bmDBO algorithm. The bmDBO algorithm is then applied to feature selection tasks. In the experimental phase, in experiment 1, we conduct a comprehensive evaluation of the mDBO algorithm's performance in solving global optimization problems using the CEC2017 benchmark test functions. In experiment 2, we further investigate the effectiveness of the bmDBO algorithm in feature selection tasks across 17 well-known datasets. Comparisons with existing algorithms in the literature indicate that both mDBO and bmDBO algorithms exhibit significant performance advantages. The results from both experiments substantiate these findings. Although the Padé approximation strategy can significantly enhance the population quality and convergence accuracy of algorithms, it may incur additional computational costs. Therefore, the design and optimization of approximation strategies will be a key focus of future research. In this study, mDBO is primarily applied to single-objective optimization problems. While its structure and strategy offer certain extensibility, its application to multi-objective optimization problems requires further validation and research. In future work, we will further design and optimize improvement strategies for the algorithm, including the Padé approximation strategy, to enhance the overall performance of the algorithm. We will also explore the application of the mDBO algorithm to multi-objective optimization problems and attempt to apply it to fields such as control scheduling, image analysis, and industrial modeling.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work is supported by Natural Science Foundation of Jilin Province (YDZJ202501ZYTS664), Education Department of Jilin Province project (JJKH20220662KJ), National Natural Science Foundation of China (12026430) and Department of Science and Technology of Jilin Province project (20210101149JC, 20200403182SF).

Conflict of interest

The author declares that there is no conflict of interest.

References

1. M. Azizi, U. Aickelin, H. A. Khorshidi, M. Baghalzadeh-Shishehgarkhaneh, Energy valley optimizer: A novel metaheuristic algorithm for global and engineering optimization, *Sci. Rep.*, **13** (2023), 226. https://doi.org/10.1038/s41598-022-27344-y
2. 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
3. S. Gupta, H. Abderazek, B. S. Yıldız, A. R. Yildiz, S. Mirjalili, S. M. Sait, Comparison of metaheuristic optimization algorithms for solving constrained mechanical design optimization problems, *Exp. Syst. Appl.*, **183** (2021), 115351. https://doi.org/10.1016/j.eswa.2021.115351
4. S. Aslan, T. Erkin, An immune plasma algorithm based approach for ucav path planning, *J. King Saud Univ. Comput. Inf. Sci.*, **35** (2023), 56–69. https://doi.org/10.1016/j.jksuci.2022.06.004
5. L. Abualigah, K. H. Almotairi, M. A. Elaziz, Multilevel thresholding image segmentation using meta-heuristic optimization algorithms: Comparative analysis, open challenges and new trends, *Appl. Intell.*, **53** (2023), 11654–11704. https://doi.org/10.1007/s10489-022-04064-4
6. M. Chan-Ley, G. Olague, Categorization of digitized artworks by media with brain programming, *Appl. Opt.*, **59** (2020), 4437–4447, 2020. https://doi.org/10.1364/AO.385552
7. Y. Li, G. Tian, Y. Yi, Y. Yuan, Improved artificial rabbit optimization and its application in multichannel signal denoising, *IEEE Sensors J.*, **2024** (2024). https://doi.org/10.1109/JSEN.2024.3456290
8. Y. Xiao, H. Cui, A. G. Hussien, F. A. Hashim, MSAO: A multi-strategy boosted snow ablation optimizer for global optimization and real-world engineering applications, *Adv. Eng. Inf.*, **61** (2024), 102464. https://doi.org/10.1016/j.aei.2024.102464
9. X. Fei, J. Wang, S. Ying, Z. Hu, J. Shi, Projective parameter transfer based sparse multiple empirical kernel learning machine for diagnosis of brain disease, *Neurocomputing*, **413** (2020), 271–283. https://doi.org/10.1016/j.neucom.2020.07.008
10. Z. Ma, X. Li, An improved supervised and attention mechanism-based u-net algorithm for retinal vessel segmentation, *Comput. Biol. Med.*, **168** (2024), 107770. https://doi.org/10.1016/j.compbiomed.2023.107770
11. Y. Xiao, Y. Guo, H. Cui, Y. Wang, J. Li, Y. Zhang, IHAOAVOA: An improved hybrid aquila optimizer and African vultures optimization algorithm for global optimization problems, *Math. Biosci. Eng.*, **19** (2022), 10963–11017. https://doi.org/10.3934/mbe.2022512
12. J. Zhang, M. Xiao, L. Gao, Q. Pan, Queuing search algorithm: A novel metaheuristic algorithm for solving engineering optimization problems, *Appl. Math. Modell.*, **63** (2018), 464–490. https://doi.org/10.1016/j.apm.2018.06.036
13. U. Kamath, K. De Jong, A. Shehu, Effective automated feature construction and selection for classification of biological sequences, *PloS One*, **9** (2014), e99982. https://doi.org/10.1371/journal.pone.0099982
14. F. Thabtah, F. Kamalov, S. Hammoud, S. R. Shahamiri, Least loss: A simplified filter method for feature selection, *Inf. Sci.*, **534** (2020), 1–15. https://doi.org/10.1016/j.ins.2020.05.017

15. L. Sun, J. Zhang, W. Ding, J. Xu, Feature reduction for imbalanced data classification using similarity-based feature clustering with adaptive weighted k-nearest neighbors, *Inf. Sci.*, **593** (2022), 591–613. https://doi.org/10.1016/j.ins.2022.02.004

16. Z. Tao, L. Huiling, W. Wenwen, Y. Xia, GA-SVM based feature selection and parameter optimization in hospitalization expense modeling, *Appl. Soft Comput.*, **75** (2019), 323–332. https://doi.org/10.1016/j.asoc.2018.11.001

17. H. Liu, Z. Zhao, Manipulating data and dimension reduction methods: Feature selection, in *Computational Complexity: Theory, Techniques, and Applications*, Springer, (2012), 790–1800. https://doi.org/10.1007/978-1-4614-1800-9_115

18. M. Rostami, K. Berahmand, E. Nasiri, S. Forouzandeh, Review of swarm intelligence-based feature selection methods, *Eng. Appl. Artif. Intell.*, **100** (2021), 104210. https://doi.org/10.1016/j.engappai.2021.104210

19. D. H. Wolpert, W. G. Macready, No free lunch theorems for optimization, *IEEE Trans. Evol. Comput.*, **1** (1997), 67–82. https://doi.org/10.1109/4235.585893

20. X. Song, Y. Zhang, D. Gong, X. Sun, Feature selection using bare-bones particle swarm optimization with mutual information, *Pattern Recognit.*, **112** (2021), 107804. https://doi.org/10.1016/j.patcog.2020.107804

21. H. Faris, M. M. Mafarja, A. A. Heidari, I. Aljarah, A. M. Al-Zoubi, S. Mirjalili, et al., An efficient binary salp swarm algorithm with crossover scheme for feature selection problems, *Knowl. Based Syst.*, **154** (2018), 43–67. https://doi.org/10.1016/j.knosys.2018.05.009

22. E. Emary, H. M. Zawbaa, A. E. Hassanien, Binary grey wolf optimization approaches for feature selection, *Neurocomputing*, **172** (2016), 371–381. https://doi.org/10.1016/j.neucom.2015.06.083

23. Y. Zhao, J. Dong, X. Li, H. Chen, S. Li, A binary dandelion algorithm using seeding and chaos population strategies for feature selection, *Appl. Soft Comput.*, **125** (2022), 109166. https://doi.org/10.1016/j.asoc.2022.109166

24. M. Mafarja, S. Mirjalili, Whale optimization approaches for wrapper feature selection, *Appl. Soft Comput.*, **62** (2018), 441–453. https://doi.org/10.1016/j.asoc.2017.11.006

25. A. Adamu, M. Abdullahi, S. B. Junaidu, I. H. Hassan, An hybrid particle swarm optimization with crow search algorithm for feature selection, *Mach. Learn. Appl.*, **6** (2021), 100108. https://doi.org/10.1016/j.mlwa.2021.100108

26. Y. Xue, T. Tang, A. X. Liu, Large-scale feedforward neural network optimization by a self-adaptive strategy and parameter based particle swarm optimization, *IEEE Access*, **7** (2019), 52473–52483. https://doi.org/10.1109/ACCESS.2019.2911530

27. E. Aličković, A. Subasi, Breast cancer diagnosis using GA feature selection and rotation forest, *Neural Comput. Appl.*, **28** (2017), 753–763. https://doi.org/10.1007/s00521-015-2103-9

28. B. Xue, M. Zhang, W. N. Browne, X. Yao, A survey on evolutionary computation approaches to feature selection, *IEEE Trans. Evol. Comput.*, **20** (2015), 606–626. https://doi.org/10.1109/TEVC.2015.2504420

29. N. Khodadadi, E. Khodadadi, Q. Al-Tashi, E. S. M. El-Kenawy, L. Abualigah, S. J. Abdulkadir, et al., BAOA: Binary arithmetic optimization algorithm with K-

nearest neighbor classifier for feature selection, *IEEE Access*, **11** (2023), 94094–94115. https://doi.org/10.1109/ACCESS.2023.3310429

30. W. N. Chen, D. Z. Tan, Q. Yang, T. Gu, J. Zhang, Ant colony optimization for the control of pollutant spreading on social networks, *IEEE Trans. Cybern.*, **50** (2019), 4053–4065. https://doi.org/10.1109/TCYB.2019.2922266
31. H. Hichem, M. Elkamel, M. Rafik, M. T. Mesaaoud, C. Ouahiba, A new binary grasshopper optimization algorithm for feature selection problem, *J. King Saud University Comput. Inf. Sci.*, **34** (2022), 316–328. https://doi.org/10.1016/j.jksuci.2019.11.007
32. A. A. Alhussan, A. A. Abdelhamid, E. S. M. El-Kenawy, A. Ibrahim, M. M. Eid, D. S. Khafaga, et al., A binary waterwheel plant optimization algorithm for feature selection, *IEEE Access*, **11** (2023), 94227–94251. https://doi.org/10.1109/ACCESS.2023.3312022
33. K. Nag, N. R. Pal, A multiobjective genetic programming-based ensemble for simultaneous feature selection and classification, *IEEE Trans. Cybern.*, **46** (2015), 499–510. https://doi.org/10.1109/TCYB.2015.2404806
34. 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
35. X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, *IEEE Trans. Evol. Comput.*, **3** (1999), 82–102. https://doi.org/10.1109/4235.771163
36. T. Y. Wu, H. Li, S. C. Chu, CPPE: An improved phasmatodea population evolution algorithm with chaotic maps, *Mathematics*, **11** (2023), 1977. https://doi.org/10.3390/math11091977
37. P. Qu, Q. Yuan, F. Du, Q. Gao, An improved manta ray foraging optimization algorithm, *Sci. Rep.*, **14** (2024), 10301. https://doi.org/10.1038/s41598-024-59960-1
38. H. Yu, Y. Wang, H. Jia, L. Abualigah, Modified prairie dog optimization algorithm for global optimization and constrained engineering problems, *Math. Biosci. Eng*, **20** (2023), 19086–19132. https://doi.org/10.3934/mbe.2023844
39. C. Liu, L. Li, Y. Qiang, S. Zhang, Predicting construction accidents on sites: An improved atomic search optimization algorithm approach, *Eng. Rep.*, **6** (2024), e12773. https://doi.org/10.1002/eng2.12773
40. A. Bisht, M. Dua, S. Dua, A novel approach to encrypt multiple images using multiple chaotic maps and chaotic discrete fractional random transform, *J. Ambient Intell. Humanized Comput.*, **10** (2019), 3519–3531. https://doi.org/10.1007/s12652-018-1072-0
41. Y. Zhou, L. Bao, C. L. P. Chen, A new 1D chaotic system for image encryption, *Signal Process.*, **97** (2014), 172–182. https://doi.org/10.1016/j.sigpro.2013.10.034
42. F. Han, X. Liao, B. Yang, Y. Zhang, A hybrid scheme for self-adaptive double color-image encryption, *Multimedia Tools Appl.*, **77** (2018), 14285–14304. https://doi.org/10.1007/s11042-017-5029-7
43. H. R. Tizhoosh, Opposition-based learning: A new scheme for machine intelligence, in *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, **1** (2005), 695–701. https://doi.org/10.1109/CIMCA.2005.1631345

44. S. Rahnamayan, H. R. Tizhoosh, M. M. A. Salama, Opposition-based differential evolution, *IEEE Trans. Evol. Comput.*, **12** (2008), 64–79. https://doi.org/10.1109/TEVC.2007.894200

45. H. Wang, Z. Wu, S. Rahnamayan, Y. Liu, M. Ventresca, Enhancing particle swarm optimization using generalized opposition-based learning, *Inf. Sci.*, **181** (2011), 4699–4714. https://doi.org/10.1016/j.ins.2011.03.016

46. M. Abd Elaziz, D. Oliva, S. Xiong, An improved opposition-based sine cosine algorithm for global optimization, *Exp. Syst. Appl.*, **90** (2017), 484–500. https://doi.org/10.1016/j.eswa.2017.07.043

47. W. Long, J. Jiao, X. Liang, S. Cai, M. Xu, A random opposition-based learning grey wolf optimizer, *IEEE Access*, **7** (2019), 113810–113825. https://doi.org/10.1109/ACCESS.2019.2934994

48. R. Storn, K. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, *J. Global Optim.*, **11** (1997), 341–359. https://doi.org/10.1023/A:1008202821328

49. A. Nickabadi, M. M. Ebadzadeh, R. Safabakhsh, A novel particle swarm optimization algorithm with adaptive inertia weight, *Appl. Soft Comput.*, **11** (2011), 3658–3670. https://doi.org/10.1016/j.asoc.2011.01.037

50. Y. Honshuku, H. Isakari, A topology optimisation of acoustic devices based on the frequency response estimation with the Padé approximation, *Appl. Math. Modell.*, **110** (2022), 819–840. https://doi.org/10.1016/j.apm.2022.06.020

51. H. Vazquez-Leal, B. Benhammouda, U. Filobello-Nino, A. Sarmiento-Reyes, V. M. Jimenez-Fernandez, J. L. Garcia-Gervacio, et al., Direct application of Padé approximant for solving nonlinear differential equations, *SpringerPlus*, **3** (2014), 1–11. https://doi.org/10.1186/2193-1801-3-563

52. I. V. Andrianov, A. Shatrov, Padé approximants, their properties, and applications to hydrodynamic problems, *Symmetry*, **13** (2021),1869. https://doi.org/10.3390/sym13101869

53. J. Xu, T. Wang, L. Pei, S. Mao, C. Zhu, Parameter identification of electrolyte decomposition state in lithium-ion batteries based on a reduced pseudo two-dimensional model with Padé approximation, *J. Power Sources*, **460** (2020), 228093. https://doi.org/10.1016/j.jpowsour.2020.228093

54. A. H. Gandomi, X. S. Yang, Evolutionary boundary constraint handling scheme, *Neural Comput. Appl.*, **21** (2012), 1449–1462. https://doi.org/10.1007/s00521-012-1069-0

55. A. H. Gandomi, A. R. Kashani, Evolutionary bound constraint handling for particle swarm optimization, in *2016 4th International Symposium on Computational and Business Intelligence (ISCBI)*, (2016), 148–152. https://doi.org/10.1109/ISCBI.2016.7743274

56. A. H. Gandomi, A. R. Kashani, M. Mousavi, Boundary constraint handling affection on slope stability analysis, in *Engineering and Applied Sciences Optimization. Computational Methods in Applied Sciences* (eds. N. Lagaros and M. Papadrakakis), Springer, (2015), 341–358. https://doi.org/10.1007/978-3-319-18320-6_18

57. A. H. Gandomi, A. R. Kashani, F. Zeighami, Retaining wall optimization using interior search algorithm with different bound constraint handling, *Int. J. Numer. Anal. Methods Geomech.*, **41** (2017), 1304–1331. https://doi.org/10.1002/nag.2678

58. A. C. Cinar, A comprehensive comparison of accuracy-based fitness functions of metaheuristics for feature selection, *Soft Comput.*, **27** (2023), 8931–8958. https://doi.org/10.1007/s00500-023-08414-3

59. I. Al-Shourbaji, N. Helian, Y. Sun, S. Alshathri, M. Abd Elaziz, Boosting ant colony optimization with reptile search algorithm for churn prediction, *Mathematics*, **10** (2022), 1031. https://doi.org/10.3390/math10071031

60. E. S. M. El-Kenawy, S. Mirjalili, A. Ibrahim, M. Alrahmawy, M. El-Said, R. M. Zaki, et al., Advanced meta-heuristics, convolutional neural networks, and feature selectors for efficient COVID-19 X-ray chest image classification, *IEEE Access*, **9** (2021). 36019–36037. https://doi.org/10.1109/ACCESS.2021.3061058

61. T. Khosla, O. P. Verma, An adaptive hybrid particle swarm optimizer for constrained optimization problem, in *2021 International Conference in Advances in Power, Signal, and Information Technology (APSIT)*, IEEE, (2021), 1–7. https://doi.org/10.1109/APSIT52773.2021.9641410

62. B. D. Kwakye, Y. Li, H. H. Mohamed, E. Baidoo, T. Q. Asenso, Particle guided metaheuristic algorithm for global optimization and feature selection problem, *Exp. Syst. Appl.*, **248** (2024), 123362. https://doi.org/10.1016/j.eswa.2024.123362

63. G. Wu, R. Mallipeddi, P. N. Suganthan, Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization, *National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, Technical Report*, 2017.

64. S. Mirjalili, A. Lewis, The whale optimization algorithm, *Adv. Eng. Software*, **95** (2016), 51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008

65. S. Mirjalili, SCA: A sine cosine algorithm for solving optimization problems, *Knowl. Based Syst.*, **96** (2016), 120–133. https://doi.org/10.1016/j.knosys.2015.12.022

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

67. S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey wolf optimizer, *Adv. Eng. Software*, **69** (2014), 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007

68. A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, H. Chen, Harris Hawks optimization: Algorithm and applications, *Future Gener. Comput. Syst.*, **97** (2019), 849–872. https://doi.org/10.1016/j.future.2019.02.028

69. L. Abualigah, A. Diabat, S. Mirjalili, M Abd Elaziz, A. H. Gandomi, The arithmetic optimization algorithm, *Comput. Methods Appl. Mech. Eng.*, **376** (2021), 113609. https://doi.org/10.1016/j.cma.2020.113609

70. S. Arora, P. Anand, Binary butterfly optimization approaches for feature selection, *Exp. Syst. Appl.*, **116** (2019), 147–160. https://doi.org/10.1016/j.eswa.2018.08.051

71. R. Tanabe, A. S. Fukunaga, Improving the search performance of SHADE using linear population size reduction, in *2014 IEEE Congress on Evolutionary Computation (CEC)*, (2014), 1658–1665. https://doi.org/10.1109/CEC.2014.6900380

72. 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)*, (2017), 145–152. https://doi.org/10.1109/CEC.2017.7969307

73. S. Zhao, Y. Wu, S. Tan, J. Wu, Z. Cui, Y. G. Wang, QQLMPA: A quasi-opposition learning and Q-learning based marine predators algorithm, *Exp. Syst. Appl.*, **213** (2023), 119246. https://doi.org/10.1016/j.eswa.2022.119246

74. Y. Xu, Z. Yang, X. Li, H. Kang, X. Yang, Dynamic opposite learning enhanced teaching-learning-based optimization, *Knowl. Based Syst.*, **188** (2020), 104966. https://doi.org/10.1016/j.knosys.2019.104966

75. R. W. Morrison, *Designing Evolutionary Algorithms for Dynamic Environments*, Springer, 2004. https://doi.org/10.1007/978-3-662-06560-0

76. K. Hussain, M. N. M. Salleh, S. Cheng, Y. Shi, On the exploration and exploitation in popular swarm-based metaheuristic algorithms, *Neural Comput. Appl.*, **31** (2019), 7665–7683. https://doi.org/10.1007/s00521-018-3592-0

77. S. Cheng, Y. Shi, Q. Qin, Q. Zhang, R. Bai, Population diversity maintenance in brain storm optimization algorithm, *J. Artif. Intell. Soft Comput. Res.*, **4** (2014), 83–97. https://doi.org/10.1515/jaiscr-2015-0001

78. S. Mirjalili, A. Lewis, S-shaped versus V-shaped transfer functions for binary particle swarm optimization, *Swarm Evol. Comput.*, **9** (2013), 1–14. https://doi.org/10.1016/j.swevo.2012.09.002

79. J. Too, A. R. Abdullah, N. Mohd-Saad, Hybrid binary particle swarm optimization differential evolution-based feature selection for emg signals classification, *Axioms*, **8** (2019), 79. https://doi.org/10.3390/axioms8030079

80. J. Too, A. R. Abdullah, N. Mohd-Saad, N. Mohd-Ali, W. Tee, A new competitive binary grey wolf optimizer to solve the feature selection problem in emg signals classification, *Computers*, **7** (2018), 58. https://doi.org/10.3390/computers7040058

81. M. H. Aghdam, N. Ghasem-Aghaee, M. E. Basiri, Text feature selection using ant colony optimization, *Exp. Syst. Appl.*, **36** (2009), 6843–6853. https://doi.org/10.1016/j.eswa.2008.08.022

82. J. Too, A. R. Abdullah, N. Mohd-Saad, A new quadratic binary harris hawk optimization for feature selection, *Electronics*, **8** (2019), 1130. https://doi.org/10.3390/electronics8101130