Electronic Research Archive

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

ERA, 32(5): 3241–3274.
DOI: 10.3934/era.2024149
Received: 20 February 2024
Revised: 23 April 2024
Accepted: 28 April 2024
Published: 08 May 2024

Research article

Dynamic allocation of opposition-based learning in differential evolution for multi-role individuals

Jian Guan[1,2], Fei Yu[1,2,*], Hongrun Wu[1,2,*], Yingpin Chen[1,2], Zhenglong Xiang[3], Xuewen Xia[1,2] and Yuanxiang Li[4]

[1] School of Physics and Information Engineering, Minnan Normal University, Zhangzhou 363000, China

[2] Key Lab of Intelligent Optimization and Information Processing, Minnan Normal University, Zhangzhou 363000, China

[3] School of Software, Nanjing University of Information Science and Technology, Nanjing 210044, China

[4] School of Computer Science, Wuhan University, Wuhan 430072, China

* **Correspondence:** Email: yufei@whu.edu.cn, wuhongrun@whu.edu.cn.

Abstract: Opposition-based learning (OBL) is an optimization method widely applied to algorithms. Through analysis, it has been found that different variants of OBL demonstrate varying performance in solving different problems, which makes it crucial for multiple OBL strategies to co-optimize. Therefore, this study proposed a dynamic allocation of OBL in differential evolution for multi-role individuals. Before the population update in DAODE, individuals in the population played multiple roles and were stored in corresponding archives. Subsequently, different roles received respective rewards through a comprehensive ranking mechanism based on OBL, which assigned an OBL strategy to maintain a balance between exploration and exploitation within the population. In addition, a mutation strategy based on multi-role archives was proposed. Individuals for mutation operations were selected from the archives, thereby influencing the population to evolve toward more promising regions. Experimental results were compared between DAODE and state of the art algorithms on the benchmark suite presented at the 2017 IEEE conference on evolutionary computation (CEC2017). Furthermore, statistical tests were conducted to examine the significance differences between DAODE and the state of the art algorithms. The experimental results indicated that the overall performance of DAODE surpasses all state of the art algorithms on more than half of the test functions. Additionally, the results of statistical tests also demonstrated that DAODE consistently ranked first in comprehensive ranking.

Keywords: metaheuristic algorithms (MAs); opposition-based learning; differential evolution (DE); dynamic allocation; ranking mechanism

1. Introduction

Several real-world optimization problems exist, such as data analytics technology [1], medical diagnosis [2], electric load dispatch [3], truss optimization [4], wind farm layout optimization [5] and so on. Optimization methods primarily include traditional and intelligent optimization methods. Traditional optimization methods typically rely on manually crafted rules and strategies, unable to fully leverage the potential information from large-scale data and complex problems, and perform poorly when dealing with high-dimensional, nonlinear, and dynamic systems. However, intelligent optimization methods such as metaheuristic algorithms (MAs) are widely used in the real world because of their simplicity and applicability compared to traditional optimization methods, especially for various optimization problems with large-scale, multi-objective, and multi-constraint characteristics. So far, researchers have proposed several MAs, which mainly include evolutionary algorithms represented by genetic algorithms (GA) [6] and differential evolution (DE), and swarm intelligence algorithms represented by particle swarm optimization (PSO) [7], ant colony optimization (ACO) [8], artificial bee colony (ABC) [9], and gray wolf optimizer (GWO) [10]. Meanwhile, some intelligent algorithms proposed in recent years have also attracted widespread attention, such as parrot optimizer (PO) [11], rime optimization algorithm (RIME) [12], weighted mean of vectors (INFO) [13], and hunger games search (HGS) [14], among others. Among these algorithms, DE, proposed by Price and Storn in 1995 [15], has received widespread research attention due to its fewer parameters and simple structure, making it easily implementable. The strong optimization capability of DE has also been validated in various real-world applications, such as power system optimization [16], path planning problems [17], and logistics scheduling [18], among others [19].

Since the proposal of DE, researchers have made a series of improvements, such as parameter adaptation, hybrid optimization with various algorithms, and the introduction of optimization strategies. These improvement approaches have further enhanced the performance of DE and gained more attention. It is worth noting that among the optimization strategies, opposition-based learning (OBL) is a powerful technique proposed by Tizhoosh [20] for enhancing algorithm performance. OBL generates opposite solutions in evolutionary algorithms while evaluating both the original and opposite solutions based on the objective function, thereby enhancing the algorithm's ability to escape local optima and converge quickly toward the global optimum solution. Rahnamayan et al. [21] first broadened the search space of DE using OBL. Subsequently, researchers have proposed several variants of OBL to further enhance its optimization capability. However, these methods often become disadvantaged when faced with more complex problems. On the one hand, classical optimization algorithms often grapple with escaping local optima, particularly when confronted with objective functions containing numerous local optima or extreme points. On the other hand, the majority of prior studies commonly utilized a singular OBL strategy for algorithm optimization, rather than exploring multiple OBL strategies (OBLs). However, it has been discovered that different variants of OBL exhibit varying optimization performances. For example, some OBL variants are advantageous for population exploration, while others enhance algorithm exploitation. Therefore, in light of the existing gaps within the co-optimization algorithm for multiple OBL strategies, this paper argues that research on co-optimization algorithms for multiple OBL strategies has become crucial.

Dealing with the coordination of different OBL variants at various stages of the algorithm is crucial in the process of multiple OBL cooperative optimization. Exploration and exploitation are the two

cornerstones of population evolution, and their balance significantly affects the performance of evolutionary algorithms [22]. Exploration is the process of completely accessing new regions in a search space. In the early stages of an algorithm, good exploration capability can provide a good search space for the initial population. However, emphasizing only exploration and neglecting exploitation reduces the convergence speed. Exploitation is the process of accessing a search space near the previously visited points. In the later stages of the algorithm, a good exploitation ability can make the population converge to the optimal solution quickly. Simultaneously, excessive emphasis on exploitation causes the algorithm to become premature and fall into local optimal solutions. Hence, in order to achieve a better balance between exploration and exploitation capabilities of the algorithm while employing multiple OBL cooperative optimization methods, this paper recognizes that the diversity and convergence of the population directly correspond to the exploration and exploitation capabilities [23]. To this end, dynamically allocating an OBL strategy for individuals in the population based on the characteristics and variations of the population at each stage is a promising approach. Assigning an OBL strategy with different characteristics to satisfy the needs of individuals in the population can maximize the algorithm's balance state [24].

In the previous research results of this paper, the effectiveness of OBL has been successfully validated [25]. To further study the collaborative optimization of OBLs, this paper proposes a dynamic allocation of opposition-based learning in differential evolution for multi-role individuals (DAODE), to efficiently exploit the co-optimization effect among multiple OBL strategies. To begin, the population individuals of each generation are divided into multiple roles and stored in corresponding archives. Then, a comprehensive ranking mechanism based on OBL is utilized to assign corresponding rewards to different roles. Specifically, different roles are assigned different OBL strategies to maintain a balance between population exploration and exploitation. Furthermore, this paper believes that the archives of different roles represent the proximity between the roles and the global optimal solution. Therefore, this paper proposes a mutation strategy based on multi-role archives, where individuals in the mutation operation are selected from different archives. This enhances the convergence speed of the algorithm.

The main contributions of this paper are as follows:

1) Introduce a novel individual evaluation metric and assign individuals three roles: elite, commoner, and inferior.

2) Propose a mechanism for dynamically allocating OBL strategies based on the different characteristics of various OBL strategies.

3) To flexibly allocate OBL, define a comprehensive ranking mechanism for multiple OBL strategies.

4) Based on the different roles of individuals, propose a mutation strategy based on multi-role archives.

The remainder of this paper is organized as follows: Section 2 describes the traditional DE framework and OBL strategy, and reviews the application of OBL and its variants in MAs. The details of the proposed DAODE algorithm are presented in Section 3. In Section 4, the experimental results of DAODE are analyzed and discussed. Finally, Section 5 briefly discusses the conclusions and future developments of this study.

2. Background

In this section, the traditional DE framework and OBL strategies are described, and we review the application of OBL in MAs and the study of classical OBL variants.

2.1. Differential evolution

Due to its simplicity and efficiency, traditional DE has been extensively researched as an optimization algorithm. It first generates an initial population P^0 randomly and then generates an offspring by mutation, crossover, and selection operations. The details are as follows:

2.1.1. Initialization

The jth dimension of the ith vector $x_{i,j}$ in the initial population P^0, is defined as follows:

$$x_{i,j}^0 = lb_j + rand(0,1) \cdot (ub_j - lb_j) \tag{2.1}$$

where $i = [1, 2, \ldots, NP]$ and $j = [1, 2, \ldots, D]$, NP and D denote the size and dimension of the population, respectively; ub_j and lb_j denote the upper and lower boundaries of the vector, respectively; and $rand(0, 1)$ is a uniformly distributed random number in the range [0,1].

2.1.2. Mutation

The mutation operation is the most important part of the DE. In the classical DE mutation strategy ($DE/rand/1$), when the population is iteratively updated, X_i^t generates mutation vector V_i^{t+1} of the offspring using the mutation operator. Thus far, there are several types of mutation strategies, and the six most frequently used mutation strategies are as follows:

(1) "DE/rand/1"

$$V_i^{t+1} = X_{r1}^t + F \cdot (X_{r2}^t - X_{r3}^t) \tag{2.2}$$

(2) "DE/best/1"

$$V_i^{t+1} = X_{best}^t + F \cdot (X_{r1}^t - X_{r2}^t) \tag{2.3}$$

(3) "DE/rand/2"

$$V_i^{t+1} = X_{r1}^t + F_1 \cdot (X_{r2}^t - X_{r3}^t) + F_2 \cdot (X_{r4}^t - X_{r5}^t) \tag{2.4}$$

(4) "DE/best/2"

$$V_i^{t+1} = X_{best}^t + F_1 \cdot (X_{r1}^t - X_{r2}^t) + F_2 \cdot (X_{r3}^t - X_{r4}^t) \tag{2.5}$$

(5) "DE/current-to-rand/1"

$$V_i^{t+1} = X_i^t + rand \cdot (X_{r1}^t - X_i^t) + F \cdot (X_{r2}^t - X_{r3}^t) \tag{2.6}$$

(6) "DE/current-to-best/1"

$$V_i^{t+1} = X_i^t + F_1 \cdot (X_{best}^t - X_i^t) + F_2 \cdot (X_{r1}^t - X_{r2}^t) \tag{2.7}$$

where X_i^t, X_{r1}^t, X_{r2}^t, X_{r3}^t, X_{r4}^t, and X_{r5}^t denote randomly selected vectors; indices i, $r1$, $r2$, $r3$, $r4$, and $r5$ are unique integers randomly selected from the range $[1, NP]$; F denotes the control parameter of the perturbation vector; and X_{best}^t denotes the optimal vector in the t-th generation.

2.1.3. Crossover

The crossover operation of the DE population is aimed at enhancing the diversity of the population and promoting the generation of structured differences within the population. The definition of the trial vector U_i^t generated by mutation is as follows.

$$u_{i,j}^{t+1} = \begin{cases} v_{i,j}^{t+1}, & \text{if } rand(0,1) \leq \text{CR} \parallel j = j_{rand} \\ x_{i,j}^{t}, & \text{otherwise} \end{cases} \tag{2.8}$$

where $\text{CR} \in [0, 1]$ denotes the crossover parameter, $j \in [1, D]$ denotes the dimension of a vector, and j_{rand} is a random integer generated from the range $[1, D]$; $x_{i,j}^t$ denotes the j-th dimension of the original vector X_i^t, and $v_{i,j}^{t+1}$, $u_{i,j}^{t+1}$ are the j-th dimension of the $(t + 1)$-th iteration of the mutation vector V_i^t and the trial vector U_i^t, respectively.

2.1.4. Selection

In the selection operation, the DE will select the better of the original vector X_i^t and the trial vector U_i^{t+1} to maintain the next generation X_i^{t+1}. It is defined as follows:

$$X_i^{t+1} = \begin{cases} U_i^{t+1}, & \text{if } f(U_i^{t+1}) \leq f(X_i^t) \\ X_i^t, & \text{otherwise} \end{cases} \tag{2.9}$$

where $f(X_i^t)$ and $f(U_i^{t+1})$ denote the fitness values of original vector X_i^t and trial vector U_i^{t+1}, respectively. Using a greedy selection method, a better individual is selected for the next generation.

2.2. Opposition-based learning

Inspired by the opposite concept, OBL is a general and efficient method, proposed by Tizhoosh [20]. It is often used in various optimization algorithms to improve algorithm performance and effectively address real-world optimization problems [26]. The OBL is defined as follows:

First, the opposite solution is defined as follows in one-dimensional space.

Opposition number: Let $x \in [a, b]$ be a real number, where a and b, respectively, represent the minimum and maximum values of the variable x. The opposition number $\check{x}$ is defined as follows:

$$\check{x} = a + b - x \tag{2.10}$$

Similarly, the opposite solution can easily be extended to a D-dimensional space, which is defined as follows:

Opposition point: Let $X = (x_1, x_2, \ldots, x_D)$ be a point in a D-dimensional space, where $x_j \in [a_j, b_j]$ and $j \in [1, D]$. The calculation of the opposition point $\check{x}_j$ for the j-th dimension of the i-th individual is as follows:

$$\check{x}_j = a_j + b_j - x_j \tag{2.11}$$

Subsequently, to use the OBL strategy flexibly, Rahnamayan et al. [21] proposed the concept of dynamic boundaries based on the original OBL [20]. With iterations of the algorithm, the search space

can be continuously reduced. Therefore, the OBL can efficiently determine the optimal solution. The details of the OBL with D-dimensional dynamic bounds are as follows.

$$\begin{aligned} \breve{x}_{i,j}^t &= a_j^t + b_j^t - x_{i,j}^t \\ a_j^t = min(x_j^t), &\quad b_j^t = max(x_j^t) \end{aligned} \tag{2.12}$$

where $\breve{x}_{i,j}^t$ denotes the point opposite to the j-th dimension of the the i-th vector at the t-th iteration, $x_{i,j}^t$; and a_j^t and b_j^t denote the minimum and maximum values of the j-th dimension at the t-th iteration, respectively.

2.3. Application of OBL and its variants in MAs

Since the introduction of OBL, it has been used in several optimization algorithms. Several variants of OBL have been proposed based on traditional OBL to improve its optimization capability in algorithms. Therefore, this section briefly introduces studies on combining OBL with different optimization algorithms and review the concepts of some classical OBL variants.

2.3.1. OBL and MAs

In many studies, to improve the diversity and convergence of algorithm populations, OBL is often used in the population initialization and the evolutionary process of various algorithms. Details are summarized in Table 1.

Table 1. Using OBL in some MAs.

Author	Summary of the research work	Algorithm
Deng et al. [27]	Generalized OBL optimizes the population initialization of DE.	NBOLDE
Kang et al. [28]	OBL enhances the population diversity of PSO.	NOPSO
Dhargupta et al. [29]	OBL improves the diversity of the wolf swarm in GWO.	SOGWO
Chatterjee et al. [30]	OBL optimizes the initial population and generational leaps of the harmony search algorithm.	OHS
Kazemi et al. [31]	OBL optimizes the population initialization and update process of ACO.	OACO
Zhang et al. [32]	OBL optimizes the update process of the BSA algorithm.	BSA-SRL
Patel et al. [33]	OBL enhances the diversity of the multi-objective GA algorithm.	DMOGA
Tair et al. [34]	OBL enhances the population diversity of the whale optimisation algorithm (WOA).	COWOAFS-SVM
Abualigah et al. [35]	OBL improves the convergence speed of the slime mould algorithm (SMA).	OBLSMAL
Joshi et al. [36]	OBL based on mixed sequences enhances the search capability of the gravitational search algorithm (GSA).	COGSA
Pham et al. [37]	OBL enhances the optimization capability of the sine cosine algorithm (SCA).	nSCA
Bacanin et al. [38]	OBL improves the initialization process of the bat algorithm (BA).	OBI-BA
Bezdan et al. [39]	OBL optimizes the initialization and iteration processes of the salp swarm algorithm (SSA).	CBOBLSSA
Mousavirad et al. [40]	Quasi-OBL enhances the learning process of Feedforward neural networks (FFNN) in DE.	QODE-FFNN

2.3.2. OBL variants

However, as the complexity of a problem increases, traditional OBL exhibits inefficient performance in high-dimensional or multimodal problems. Therefore, several relevant variants of the OBL have been proposed, as shown in Table 2.

3. The proposed algorithm

As described in Section 2.3, OBL has been shown to be a practical and efficient optimization strategy. From these studies, it is observed that people usually use only one OBL strategy in their algorithms

Table 2. A research work of some classic OBL variants.

Author	Summary of the research work	Algorithm
Rahnamayan et al. [41]	A quasi-opposition between the center point and the opposite point.	QOBL
Ergezer et al. [42]	A quasi-reflection OBL between the orginal point and the center point.	QROBL
Tizhoosh et al. [43]	A Super-opposition between the opposite point and the boundary.	SOBL
Wang et al. [44]	A generalized OBL capable of transforming the search space.	GOBL
Ergezer et al. [45]	A fitness-based quasi-reflection OBL.	FQROBL
Hu et al. [46]	A type of OBL that focuses only on partial population opposition.	POBL
Rahnamayan et al. [47]	A population centroid-based OBL.	COBL
Liu et al. [48]	A rotating OBL capable of varying angles.	RBL
Xu et al. [49]	A center-opposition learning minimizing the distance between the original point and the opposite point to the global optimum.	OCL
Seif et al. [50]	An extended opposition and reflected extended opposition based on Quasi-opposition (QO) and Quasi-reflection (QR).	EO/REO
Xu et al. [51]	An OBL based on the current best solution.	COOBL
Park et al. [52]	A beta distribution-based OBL.	BetaOBL

and that these OBL strategies exhibit different performances in addressing different types of problems. For instance, the different OBL strategies exhibit varying population diversity and convergence rates in DE, and in the early stage of the algorithm, DE using OBL with better diversity is beneficial for the population to explore the unknown space. In the later stage of the algorithm, using OBL with better convergence will be beneficial for finding the optimal solution. Therefore, there is a lack of research on the cooperative effect of OBLs, and using OBLs with different characteristics to address the corresponding problems. Moreover, the study of dynamically allocating OBL to the population at different time periods is highly significant.

Related studies have demonstrated that dynamic allocation methods produce favorable performance for algorithms [53]. In DE, exploration and exploitation significantly affect the performance of the DE, and methods for the dynamic selection of parameters or strategies have been proposed to maintain the balance between them. Depending on the problem, the dynamic allocation mechanism can select parameters or strategies that are suitable for the current problem, thus improving the ability of the algorithm to address the problem. A dynamic allocation strategy based on the ranking mechanism is a common research method [54], and the corresponding strategy is adopted at the ranking level to achieve reasonable resource allocation and performance improvement.

Building upon the aforementioned inspiration, this paper proposes a dynamic allocation of OBL in DAODE. In DAODE, based on the characteristics of individuals in each generation of the population, this paper first assigns three roles to each individual and stores them in the corresponding archives. In the mutation operation, DAODE then employs a novel mutation strategy based on multi-role archives to expedite the search for the global optimum solution. Lastly, the OBL pool, composed of multiple OBL strategies, dynamically assigns an OBL strategy to individuals of different roles. Different OBL strategies are allocated to roles at different stages, ensuring both population diversity and improved convergence speed. The complete framework of DAODE is shown in Figure 1. Details of this process are described in the following subsections.

3.1. Multi-role individuals and archives (MIA)

The DE is a stochastic search algorithm that randomly generates populations without prior knowledge. In several studies, archiving has been used to retain the genes of elite individuals for use in future generations [53], thus reducing the impact of a lack of prior knowledge. Additionally, divid-

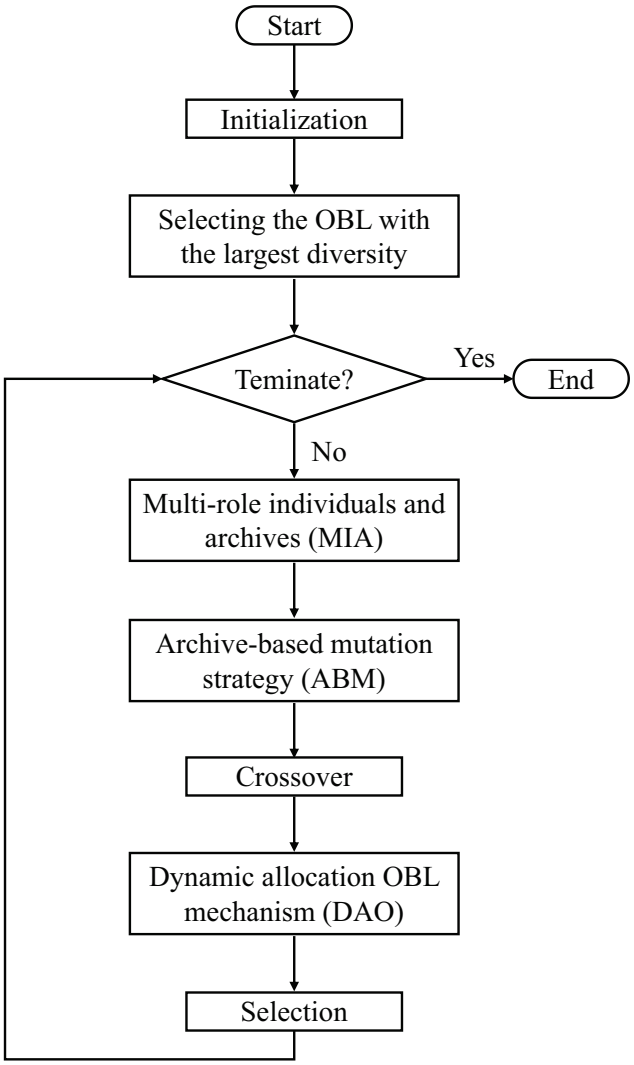


Figure 1. The complete framwork of DAODE.

ing the population into multiple roles can provide a good balance for the algorithm [55]. Therefore, this paper will assign individuals three roles: elite, commoner, and inferior before each generation's population update, and store them in the corresponding archives.

In DAODE, this paper first evaluates the position and fitness of each individual. The position of an individual is determined by its euclidean distance from the current optimum solution. Then, a comprehensive characteristic of the individual is derived from the position and fitness. Finally, based on this comprehensive characteristic, this paper assigns three roles to the individuals and stores them in their corresponding archives, the details of which are provided in Eq (3.1). As in the real world, the proportion of individuals with high and low abilities is small, while those with general abilities are the majority. Thus, the proportions of elite, commoner, and inferior roles in DAODE were $m\%$, $n\%$, and $p\%$ (e.g., $m = 20$, $n = 50$, and $p = 30$, respectively), and the final proportions were generated by the experiment (see Section 4.5.1).

Elite roles provide suitable genes for their offspring and are beneficial for improving the accuracy

of the solution. Inferior roles aim to improve the exploratory ability of a population. Commoner plays a balancing role in the exploration and exploitation of the entire population. The multi-role individuals and archives are shown in Figure 2. From the figure, it can be observed that the darker the color of the individual, the smaller the fitness and the closer it is to the optimal solution.

$$E = \alpha * E_f + \beta * E_p \tag{3.1}$$

where E, E_f, E_p denote the comprehensive evaluation, fitness, and position indices for each individual, respectively. α, β correspond to scaling parameters between E_f and E_p, respectively, and they will be generated by the experiment.

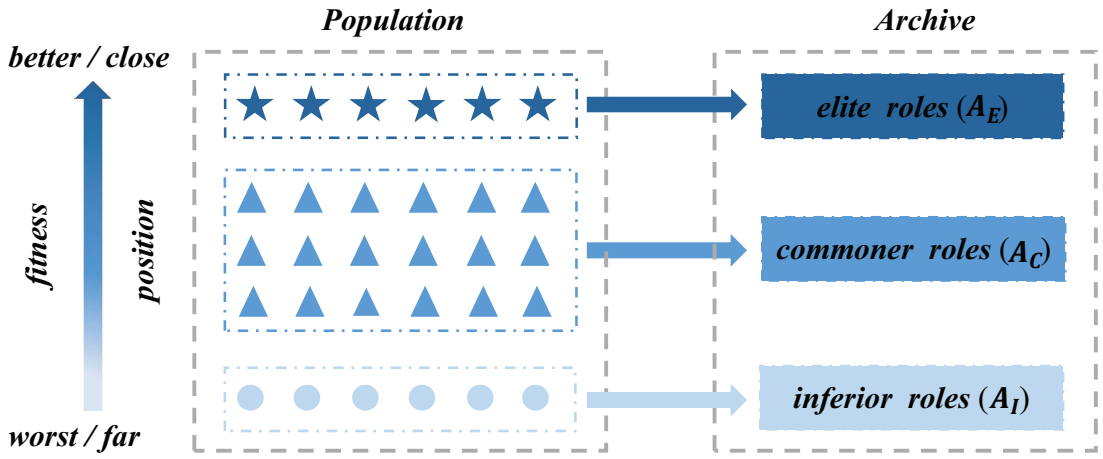


Figure 2. Multi-role individuals and archives.

3.2. Dynamic allocation OBL mechanism (DAO)

When generating trial vectors in a population, this paper identifies the roles of each individual and assigns them with corresponding OBL strategies. Therefore, this paper proposes a multiple DAO. The model of the DAO mechanism is shown in Figure 3, and the details are as follows.

To ensure the flexibility of dynamically allocating OBLs, this paper defines a comprehensive ranking mechanism for OBLs to efficiently allocate the corresponding OBLs to roles. First, the diversity and fitness of each OBL are evaluated and ranked, and the diversity is measured using Eq (3.2), as proposed by Morales-Castañeda [56].

$$\begin{aligned} Div_j &= \frac{1}{NP}\sum_{i=1}^{NP} |median(x^j) - x_i^j| \\ Div &= \frac{1}{D}\sum_{j=1}^{D} Div_j \end{aligned} \tag{3.2}$$

where x_i^j denotes the j-th dimension of the i-th vector. The $median(x^j)$ symbolizes the median of the j-th dimension of the population. The diversity Div is calculated as the average of Div_j in each

dimension. The calculation of Eq (3.2) is involved in each iteration.

$$\begin{aligned} R_D &= rank_{Div} \\ R_F &= i \end{aligned} \tag{3.3}$$

where R_D and R_F denote the diversity and fitness ranking values of each OBL, respectively, and i denotes the fitness ranking, where a smaller fitness value indicates a smaller ranking value. Therefore, the diversity and fitness rankings were calculated using Eq (3.3).

Subsequently, the diversity of the DE population gradually decreased with continuous iterations, and the fitness gradually converged to the optimal solution. Therefore, this paper designs a weight parameter for the diversity and fitness rankings of the OBL and obtained a comprehensive ranking of the OBL from the weight parameter and the ranking result. The comprehensive rank is calculated as follows:

$$\begin{aligned} R &= \omega * R_D + (1 - \omega) * R_F \\ \omega &= \frac{1}{2} + \frac{1}{2} * \cos(\frac{iter}{Max\_Iter} * \theta * \pi) \end{aligned} \tag{3.4}$$

where R denotes comprehensive ranking of OBL, ω denotes a weight parameter by the cosine function, and $iter$ and Max_Iter denote the number of current iterations and the maximum number of iterations, respectively. To normalize the parameter ω, an additional parameter θ is added here.

Finally, based on the comprehensive ranking of OBL, the pool of OBL strategies was divided into three levels: excellent OBLs, ordinary OBLs, and inferior OBLs. As shown in Figure 3, the darker color in the OBL pool indicates that this OBL has a better optimization capability at the current stage. Specifically, after each update of the algorithm population, each type of OBL evaluates the diversity and convergence of the current search environment. They are then ranked using a comprehensive ranking mechanism, and based on the individuals with different roles, an appropriate OBL strategy is assigned to each individual accordingly. To maintain the balance between the exploration and exploitation of the DE, elite roles randomly select an OBL in the inferior strategy pool that has k OBLs, ensuring the individual's superiority. By contrast, inferior roles select an OBL randomly in an excellent strategy pool with i OBLs, which drives the individual to a good region. However, for individuals with ordinary OBLs, a suitable OBL is obtained using a Gaussian distribution in an ordinary OBL strategy pool, which ensures a balanced state. This means that after each iteration of the algorithm, different individuals will change their roles, and with the continuous changes in the search environment, individuals will also select different OBL strategies from the OBL pool as optimization strategies. This achieves a state of dynamically allocating OBL throughout the entire process of the algorithm.

3.3. Archive-based mutation strategy (ABM)

Mutation operations play an important role in DE, and the selection of mutant individuals affects the offspring trend. A mutation strategy based on multi-role archiving is proposed in this study, and we named it "ABM". In ABM, the base vector of the mutation strategy is randomly selected from the archives of elite roles A_E to ensure the superiority of offspring. However, the difference vector of the mutation perturbation is the difference between two vectors randomly selected from the archives of commoner roles A_C and those of inferior roles A_I, which avoids the problem of excessive perturbation and accelerates convergence. The mutation strategy of ABM in a two-dimensional space is shown in Figure 4.

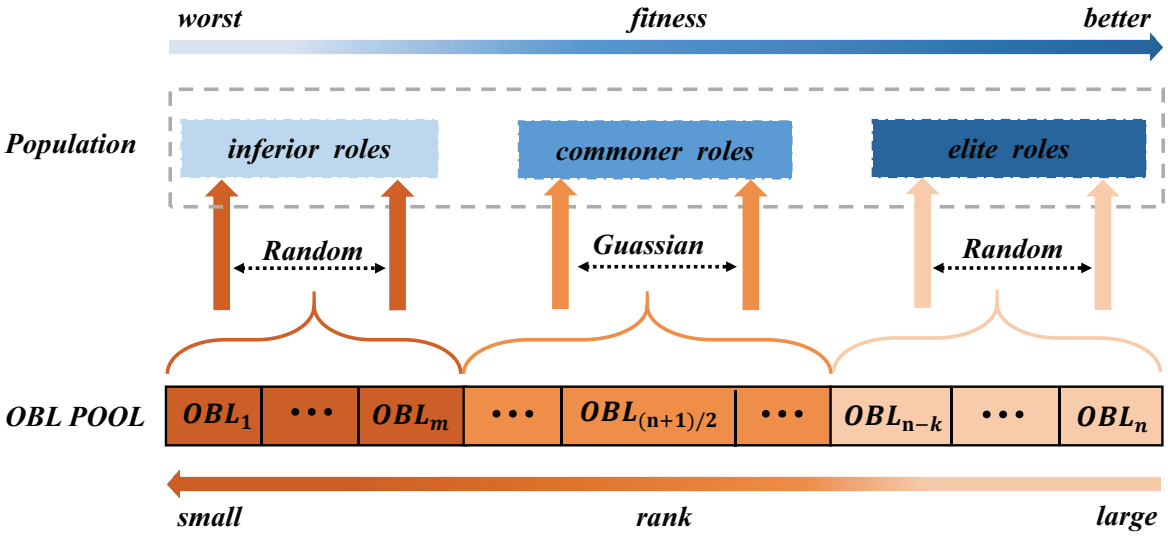


Figure 3. Dynamic allocation OBL mechanism.

In Figure 4, the darker regions on the left figure represent positions closer to the optimal solution, and the red pentagon represents the location of the optimal solution in the two-dimensional space. Different-colored regions symbolize the distribution range of different classes of individuals, and individuals with mutations were selected from these regions. As the mutation process shows in the figure on the right, the offspring will continue to approach the optimal solution.

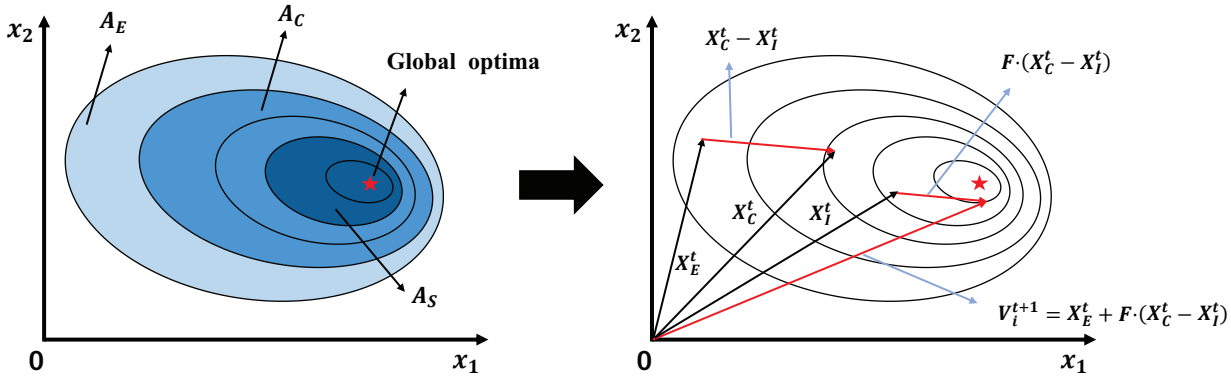


Figure 4. ABM strategy in a two-dimensional space.

3.4. The entire pseudocode of DAODE

Based on this discussion, this study constructs a dynamic allocation of OBL in DAODE. The entire pseudocode of DAODE is presented in Algorithm 1. At the beginning of DAODE, the OBL with the highest level of diversity is selected as the initial strategy to improve the diversity of the initial population in the evolutionary stage of the algorithm. To begin, in line 5, individuals in the population are assigned three roles and stored in their respective archives. Subsequently, a mutation vector is generated using the mutation strategy based on the archiving of multi-role in line 7, and a trial vector is generated using the crossover operation in line 8. At this point, the dynamic allocation OBL mechanism

is applied to the current roles in lines 9 and 10 to generate the opposite vector. Finally, the offsprings are evaluated in line 12 until the iterations of the algorithm are completed. Furthermore, the source code for this study can be found at the following link: https://github.com/gjianzx/DAODE.

Algorithm 1 DAODE

1: Randomly generate NP individuals as the initial population P^0 by Eq (2.1);
2: Select the largest diversity OBL to generate the opposite population OP^0;
3: Evaluate the fitness of P^0 and OP^0;
4: **while** the termination condition is not satisfied **do**
5: Multi-role individuals and archives (MIA);
6: **for** $i = 1$ to NP **do**
7: Generate a mutation vector V_i^{t+1} by using the ABM;
8: Generate a trial vector U_i^{t+1} by Eq (2.8);
9: Ranking OBLs by Eqs (3.3) and (3.4);
10: Generate the opposite vector OX_i^{t+1} by using DAO;
11: Evaluate the fitness of U_i^{t+1} and OX_i^{t+1};
12: Retaining excellent vectors X_i^{t+1} from X_i^t, U_i^{t+1} and OX_i^{t+1} by Eq (2.9);
13: **end for**
14: t = t + 1;
15: **end while**

3.5. Complexity analysis of DAODE

Compared with the original DE, DAODE also needs to calculate the time complexity of MIA, DAO, and ABM, and their time complexities are analyzed individually below.

In MIA, first, each individual in the population is sorted according to the complexity of $O(NP \times logNP)$. The population is then classified into three levels according to the sorting of roles, and the complexity is $O(NP \times NP)$. Finally, the archiving of individuals is performed simultaneously with the classification, and no additional time complexity was calculated. Thus, the overall complexity of the MIA is $O(NP \times logNP + NP \times NP)$.

In the DAO, the additional complexity is mainly caused by the ranking of the OBLs and the dynamic allocation OBL mechanism. The sorting of the OBLs is divided into diversity and fitness sorting, both of which have a complexity of $O(logN)$, where N denotes the total number of OBLs. In the dynamic allocation OBL mechanism, each role must select an OBL strategy with complexity $O(NP)$.

In ABM, the three vectors of the mutation operation are randomly selected from the archives, which have the same complexity as the original DE, which is $O(NP \times D)$.

The complexity of the original DE was $O(Max\_Iter \times NP \times D)$, where Max_Iter was the maximum value of the algorithm iteration. The total complexity of DAODE is $O(Max\_Iter \times (NP \times logNP + NP \times NP + logN + logN + NP + NP \times D))$, that is, $O(Max\_Iter \times NP \times max(NP, logNP, logN, D))$. Therefore, DAODE remains computationally efficient in terms of time compared to the original DE.

Table 3. CEC2017 benchmark suite.

Type	No.	Function	$f_i^* = f_i(x^*)$
Unimodal	f_1	Shifted and Rotated Bent Cigar Function	100
	f_2	Shifted and Rotated Sum of Different Power Function*	200
	f_3	Shifted and Rotated Zakharov Function	300
Multimodal	f_4	Shifted and Rotated Rosenbrock's Function	400
	f_5	Shifted and Rotated Rastrigin's Function	500
	f_6	Shifted and Rotated Expanded Scaffer's F6 Function	600
	f_7	Shifted and Rotated Lunacek Bi_Rastrigin Function	700
	f_8	Shifted and Rotated Non-Continuous Rastrigin's Function	800
	f_9	Shifted and Rotated Levy Function	900
	f_{10}	Shifted and Rotated Schwefel's Function	1000
Hybrid	f_{11}	Hybrid Function 1 (N=3)	1100
	f_{12}	Hybrid Function 2 (N = 3)	1200
	f_{13}	Hybrid Function 3 (N = 3)	1300
	f_{14}	Hybrid Function 4 (N = 4)	1400
	f_{15}	Hybrid Function 5 (N = 4)	1500
	f_{16}	Hybrid Function 6 (N = 4)	1600
	f_{17}	Hybrid Function 6 (N = 5)	1700
	f_{18}	Hybrid Function 6 (N = 5)	1800
	f_{19}	Hybrid Function 6 (N = 5)	1900
	f_{20}	Hybrid Function 6 (N = 6)	2000
Composition	f_{21}	Composition Function 1 (N=3)	2100
	f_{22}	Composition Function 2 (N = 3)	2200
	f_{23}	Composition Function 3 (N = 4)	2300
	f_{24}	Composition Function 4 (N = 4)	2400
	f_{25}	Composition Function 5 (N = 5)	2500
	f_{26}	Composition Function 6 (N = 5)	2600
	f_{27}	Composition Function 7 (N = 6)	2700
	f_{28}	Composition Function 8 (N = 6)	2800
	f_{29}	Composition Function 9 (N = 3)	2900
	f_{30}	Composition Function 10 (N = 3)	3000
		Search Range:$[-100, 100]^D$	

* f_2 has been excluded because it shows unstable behavior especially for higher dimensions, and significant performance variations for the same algorithm implemented in Matlab, C [57].

4. Experimental results and analysis

In this section, the performance of DAODE is evaluated mainly on test functions presented at the 2017 IEEE conference on evolutionary computation (CEC2017), and the experimental results are analyzed and compared in detail. The experiments included a comparison between DAODE and several state of the art algorithms and OBL variants (i.e., the improved OBL strategy introduced in Section 2.3.2). The differences between DAODE and several other algorithms were further evaluated using statistical tests. In addition, this paper analyzes the influence of the proposed strategy on DE. The experimental details are as follows.

4.1. Benchmark functions

The CEC2017 benchmark suite used in this study comprised 29 test functions for unimodal functions ($f_1 - f_3$), simple multimodal functions ($f_4 - f_{10}$), hybrid functions ($f_{11} - f_{20}$), and composition function ($f_{21} - f_{30}$). Meanwhile, these functions represented actual problems with various characteristics. The test function information is listed in Table 3, where N denotes the number of functions in the hybrid and composition functions, f_i^* and $f_i(x^*)$ denote the optimal solution for each test function, and $[-100, 100]^D$ denotes the search bound for each test function. More detailed information on the CEC2017 benchmark suite can be found in the literature [57].

4.2. Experimental settings

For a fair comparison of the experiments, the common parameters of the algorithms compared in this study were set as follows:

- *Popualtion size (NP): 100*
- *Mutation factor (F): 0.5*
- *Crossover factor (CR): 0.9*
- *The problem dimension (D): 30 and 50*
- *Maximum number of function evaluations (Max_FES): 10000×D*
- *Maximum number of algorithm iterations (Max_Iter): Max_FES/NP*
- *The independent number of runs (Run_Num): 30*

For each algorithm, the additional parameters are listed in Table 4, the details of which can be found in the corresponding literature. In addition, all simulation environments in this study included Windows 10, an Intel(R) Core(TM) i5-7300HQ CPU, RAM-16 GB, and MATLAB 2020a.

Table 4. Additional parameters for the comparison algorithm.

Algorithm	Year	Parameters setting
JaDE [58]	2009	$\mu_F = \mu_{CR} = 0.5$, $c = 0.1$, $p = 0.05$.
ACDE/F [59]	2022	$F_0 = 0.5$, $CR_0 = 0.8$, $k \in (0, 1)$; if $v = 0$, $\%P_{ks} = 0.5$.
OMLDE [60]	2021	$r_i^1, r_i^2, r_i^3 \in [0, 1]$.
DODE [61]	2021	$c = 0.1$, $\mu_j = 0.3$, $PT = 0.1$.
OBA [50]	2015	$P_s(t) = t/T$.
NBOLDE [27]	2021	$F_1 = F_2 = 0.4$, $Ns = [10, 30]$, $Jr = 0.3$.
GODE [62]	2021	$Jr = 0.3, r_1 = 10^{-2}, r_2 = 10^{-3}, r_3 = 10^{-6}$.
DAODE	—	$\alpha = 0.5, \beta = 0.5, \theta = D/10$, A_E:A_C:A_I=2 : 5 : 3.

4.3. Experimental results and analysis

To evaluate the overall performance of DAODE, the seven state of the art algorithms listed in Table 4 are compared and analyzed. Moreover, to verify the flexibility of the dynamic allocation of OBL in DAODE, the performance of each OBL in the OBL pool in DE was evaluated and compared with the experimental results of DAODE.

The accuracy of each algorithm was calculated using Eq (4.1), where $f_i(x)$ denotes the result obtained by the algorithm on the i-th test function, and $f_i(x^*)$ denotes the optimal solution that the algorithm can achieve in the test function (e.g., the optimal solution for f_1 in Table 3 is $f_1(x^*) = 100$).

$$FEv = f_i(x) - f_i(x^*) \tag{4.1}$$

In addition, each algorithm was run 30 times independently of each test function, and the mean value (Mean) and standard deviation (Std) were recorded. The data in the table is represented in scientific notation, such as 7.14E+08 and 9.47E-16, which are equivalent to 7.14×10^8 and 9.47×10^{-16}, respectively. Additionally, the optimal solution of the comparison results is highlighted in bold. Notably, "+/-/=" denotes that DAODE is better than, approximately the same as, or lower than the corresponding algorithm, respectively. In addition, this paper plots the convergence of the algorithms on several test functions, which clearly show the convergence trend of each algorithm.

4.3.1. Comparison of state of the art algorithms

These state of the art algorithms include adaptive mechanisms combined with OBL and novel optimization strategies. Therefore, they are more convincing than the comparison algorithms for DAODE. The following seven algorithms are briefly described. JaDE [58] is an adaptive DE with optional external archiving that improves optimization performance by implementing a new mutation strategy with optional external archiving and adaptively updating the control parameters. ACDE/F [59] proposed a belief space strategy incorporating a generalized OBL and a parameter adaptive adjustment mechanism, aimed at striking a balance between global exploration and local exploitation capabilities. Based on the original dynamic OBL (DOL), OMLDE [60] introduced a mutual learning (ML) method aimed at guiding individual-determined ML and proposed opposite ML. In DODE [61], OBLs are simply classified and a dual OBL is proposed that contains two OBL strategies and a protection mechanism. OBA [50] mathematically proves the probability that the original and opposite solutions are close to the optimal solution when constructing an opposition-based metaheuristic optimization algorithm. For NBODE [27], new parameters and weight factors are incorporated into the neighborhood mutation model (DE/current-to-best/1), introducing a mutation strategy (DE/neighbor-to-neighbor/1) that focuses on local mutations for enhanced efficiency over global mutations. Finally, GODE [62] believes that more promising solutions exist around the opposite solution, and thus uses Gaussian perturbation to widen the neighborhood of the opposite solution and increase the possibility of finding a better solution.

The comparison results of DAODE with the state of the art algorithms at $30D$ and $50D$ test functions are summarized in Tables 5 and 6, respectively. According to the comparison results, DAODE still exhibited good performance against the other algorithms for unimodal functions ($f_1 - f_3$) as the dimensions of the problem increased. For the multimodal function ($f_4 - f_{10}$), although DAODE is not the best among several compared algorithms, it still has the top ranking and is next to the first among several test functions (e.g., f_5, f_7, and f_8). In the hybrid ($f_{11} - f_{20}$) and composition ($f_{21} - f_{30}$) functions, DAODE outperforms most algorithms at $30D$, which is strongly related to the proposed DAO. As the dimension increased to $50D$, DAODE outperformed other state of the art algorithms on only half of the test functions. However, it still achieved a second or third ranking on the other test functions.

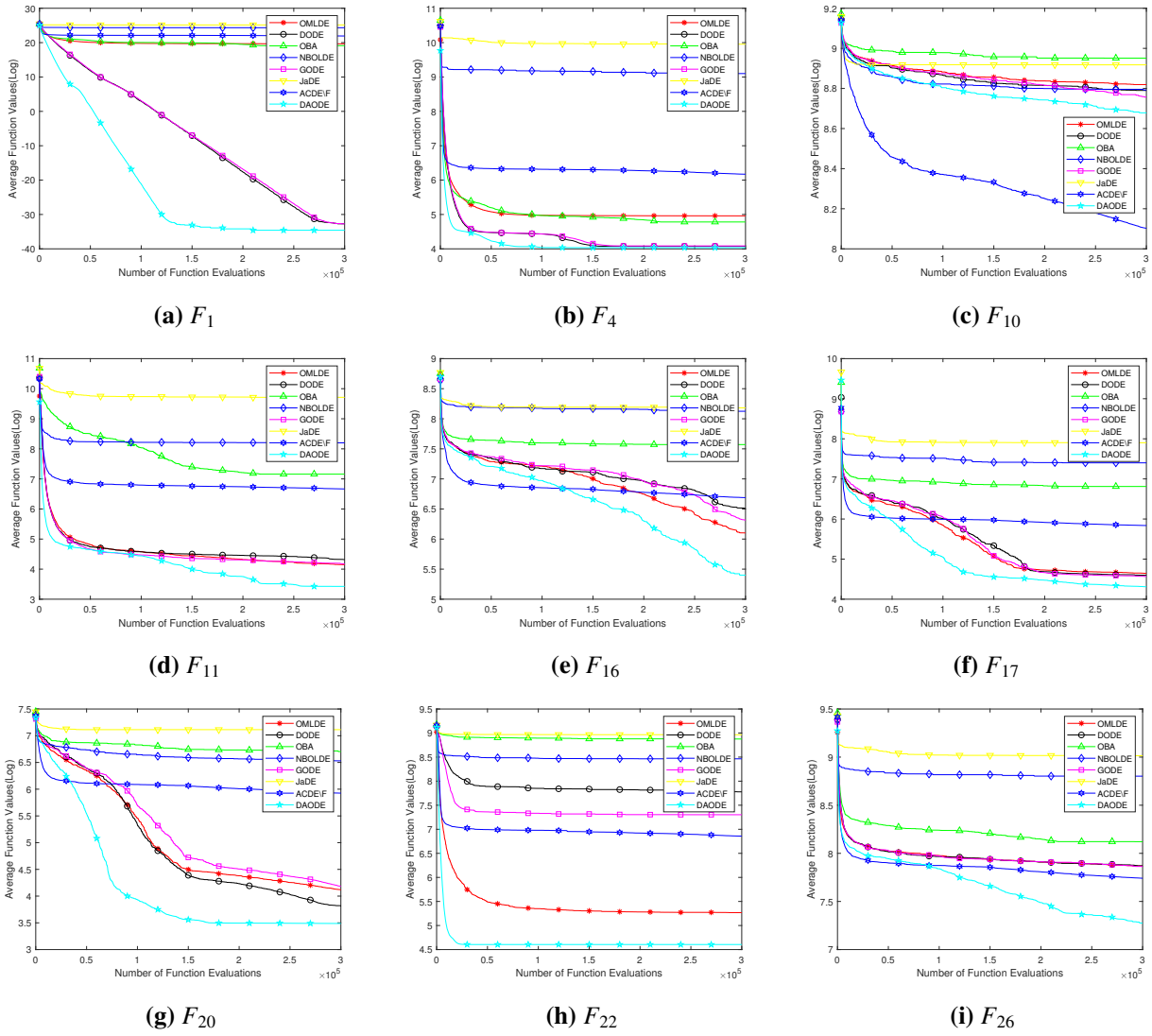
The convergence curves of the compared algorithms are shown in Figures 5 and 7 for 30 independent runs on the CEC2017 benchmark suite. DAODE outperformed the other state of the art algorithms in several test functions and exhibited a faster convergence rate. This also demonstrates that ABM can accelerate the development of populations in good areas by selecting superior individuals as base vectors for mutation operations. The convergence curves of f_{16}, f_{17}, and f_{26} at $30D$ and f_3, f_{16}, f_{20}, f_{23}, and f_{26} at $50D$ exhibited a decreasing trend. Thus, DAODE has great potential for searching optimal solutions to these functions. Meanwhile, for the convenience of observing the distribution of results from 30 runs of each algorithm, partial boxplots of the test functions are provided in Figures 6 and 8, corresponding to the convergence plots. In these test functions, it can be clearly observed that DAODE achieves superior solutions compared to other algorithms, demonstrating significant performance advantages.

Therefore, DAODE exhibits good performance compared with the state of the art algorithms in the CEC2017 benchmark suites of $30D$ and $50D$. In particular, for JaDE, ACDE/F, OBA, and NBODE, DAODE has an absolute advantage and contains various state of the art adaptive strategies, which indicate the effectiveness of the optimization strategies proposed by DAODE. Moreover, it was experi-

mentally found that the reason for the insignificant performance of DAODE on the multimodal function ($f_4 - f_{10}$) may mainly lie in the multimodal function possessing multiple local optima, which incorrectly selects locally optimal individuals as superior individuals in the MIA, reducing the possibility of searching for optimal solutions.

Table 5. Comparison results of DAODE and state-of-the-art algorithms on CEC2017 benchmark suite ($D = 30$).

Fun.		JaDE	ACDE/F	OMLDE	DODE	OBA	NBODE	GODE	DAODE
f_1	Mean	8.42E+10(8+)	3.37E+09(6+)	3.46E+08(5+)	6.16E-15(3=)	2.02E+08(4+)	3.72E+10(7+)	6.16E-15(2=)	**9.47E-16**(1)
	Std.	2.06E+10	1.00E+09	4.42E+08	7.16E-15	1.41E+08	7.14E+09	7.16E-15	**3.61E-15**
f_3	Mean	3.57E+05(6+)	3.57E+05(7+)	1.35E+03(4+)	4.62E+01(2+)	4.07E+05(8+)	7.22E+04(5+)	6.41E+01(3+)	**1.25E-12**(1)
	Std.	1.25E+05	9.91E+03	8.75E+02	4.10E+01	6.25E+05	8.14E+03	7.68E+01	**4.82E-12**
f_4	Mean	2.10E+04(8+)	4.79E+02(6+)	1.43E+02(5+)	5.87E+01(2=)	1.19E+02(4+)	8.96E+03(7+)	5.94E+01(3=)	**5.65E+01**(1)
	Std.	6.65E+03	1.77E+02	3.83E+01	1.01E+00	4.53E+01	2.21E+03	2.13E+00	**1.56E+01**
f_5	Mean	5.53E+02(8+)	**8.26E+01**(1-)	1.76E+02(3=)	1.76E+02(4=)	2.31E+02(6+)	3.76E+02(7+)	1.77E+02(5=)	1.69E+02(2)
	Std.	6.37E+01	**1.30E+01**	1.03E+01	9.18E+00	1.29E+01	2.52E+01	8.47E+00	8.14E+00
f_6	Mean	1.13E+02(8+)	1.21E+01(5+)	6.76E+00(4+)	**2.68E-10(1-)**	4.40E+01(6+)	8.21E+01(7+)	1.19E-09(2-)	1.11E-03(3)
	Std.	1.06E+01	2.22E+00	2.23E+00	**4.33E-10**	6.57E+00	6.65E+00	2.37E-09	6.08E-03
f_7	Mean	2.12E+03(5+)	**1.71E+02(1=)**	2.17E+02(6+)	2.09E+02(4=)	2.84E+02(7+)	6.00E+02(8+)	2.08E+02(3=)	1.99E+02(2)
	Std.	2.88E+02	**2.68E+01**	1.28E+01	9.27E+00	2.48E+01	5.49E+01	1.01E+01	1.07E+01
f_8	Mean	5.09E+02(8+)	**7.96E+01(1-)**	1.75E+02(3=)	1.80E+02(5=)	2.41E+02(6+)	2.98E+02(7+)	1.78E+02(4=)	1.66E+02(2)
	Std.	4.10E+01	**1.34E+01**	9.00E+00	1.03E+01	1.13E+01	2.04E+01	1.09E+01	9.66E+00
f_9	Mean	2.45E+04(8+)	6.07E+02(5+)	3.73E+01(4+)	0.00E+00(2-)	2.11E+03(6+)	8.29E+03(7+)	**0.00E+00(1-)**	8.17E-02(3)
	Std.	4.74E+03	2.26E+02	3.23E+01	0.00E+00	9.06E+02	8.37E+02	**0.00E+00**	1.71E-01
f_{10}	Mean	7.46E+03(7+)	**3.30E+03(1-)**	6.76E+03(6+)	6.57E+03(4+)	7.72E+03(8+)	6.58E+03(5+)	6.35E+03(3=)	5.87E+03(2)
	Std.	4.72E+02	**5.97E+02**	4.55E+02	5.56E+02	4.38E+02	3.82E+02	7.75E+02	1.19E+03
f_{11}	Mean	1.65E+04(8+)	7.86E+02(5+)	6.34E+01(2+)	7.51E+01(3+)	1.28E+03(6+)	3.64E+03(7+)	6.48E+01(4+)	**3.08E+01(1)**
	Std.	5.88E+03	4.42E+02	3.08E+01	3.66E+01	5.27E+02	8.28E+02	2.62E+01	**2.97E+01**
f_{12}	Mean	1.36E+10(8+)	1.72E+08(6+)	3.07E+05(4+)	9.77E+03(2=)	6.09E+07(5+)	7.65E+09(7+)	**7.63E+03(1-)**	1.02E+04(3)
	Std.	5.33E+09	1.24E+08	1.09E+06	7.78E+03	3.54E+07	2.11E+09	**6.09E+03**	1.27E+04
f_{13}	Mean	1.05E+10(8+)	1.42E+08(6+)	2.81E+03(4+)	8.14E+01(2=)	9.71E+06(5+)	4.81E+09(7+)	**7.93E+01(1=)**	8.66E+01(3)
	Std.	4.38E+09	2.16E+08	3.89E+03	9.49E+00	1.71E+07	2.34E+09	**7.99E+00**	2.66E+01
f_{14}	Mean	1.38E+07(8+)	2.58E+05(5+)	6.98E+01(4=)	6.26E+01(3=)	3.85E+05(6+)	1.09E+06(7+)	6.06E+01(2=)	**5.01E+01(1)**
	Std.	8.86E+06	3.78E+05	3.96E+01	5.47E+00	2.24E+05	5.18E+05	8.92E+00	**1.53E+01**
f_{15}	Mean	2.40E+09(8+)	1.19E+06(5+)	8.32E+01(4+)	3.47E+01(2+)	2.26E+07(6+)	1.43E+08(7+)	3.69E+01(3+)	**1.54E+01(1)**
	Std.	1.45E+09	3.88E+06	1.65E+02	1.06E+01	5.32E+07	1.22E+08	5.04E+00	**3.60E+01**
f_{16}	Mean	3.57E+03(8+)	8.03E+02(5+)	4.47E+02(2+)	6.73E+02(4+)	1.94E+03(6+)	3.38E+03(7+)	5.53E+02(3+)	**2.22E+02(1)**
	Std.	4.78E+02	2.40E+02	2.94E+02	4.33E+02	1.69E+02	4.30E+02	3.85E+02	**3.17E+02**
f_{17}	Mean	2.70E+03(8+)	3.43E+02(5+)	1.04E+02(4+)	9.91E+01(3+)	9.08E+02(6+)	1.64E+03(7+)	9.63E+01(2+)	**7.45E+01(1)**
	Std.	7.36E+02	1.24E+02	9.67E+01	5.43E+01	1.40E+02	4.00E+02	3.81E+01	**7.29E+01**
f_{18}	Mean	9.82E+07(8+)	6.70E+05(5+)	8.49E+03(4+)	3.69E+01(2+)	1.58E+07(7+)	1.28E+07(6+)	3.87E+01(3+)	**2.56E+01(1)**
	Std.	6.22E+07	1.17E+06	5.83E+03	4.68E+00	1.27E+07	8.53E+06	4.08E+00	**2.14E+00**
f_{19}	Mean	3.61E+09(8+)	1.55E+06(5+)	2.93E+01(4+)	1.70E+01(2+)	2.94E+06(6+)	2.41E+08(7+)	1.81E+01(3+)	**7.34E+00(1)**
	Std.	1.44E+09	3.25E+06	3.93E+01	6.70E+00	5.24E+06	1.69E+08	6.19E+00	**2.57E+00**
f_{20}	Mean	1.23E+03(8+)	3.75E+02(5+)	6.15E+01(3+)	4.53E+01(2+)	8.15E+02(7+)	6.86E+02(6+)	6.56E+01(4+)	**3.27E+01(1)**
	Std.	2.32E+02	1.40E+02	5.74E+01	5.11E+01	1.17E+02	1.06E+02	7.70E+01	**2.72E+01**
f_{21}	Mean	6.85E+02(8+)	**2.77E+02(1-)**	3.65E+02(3=)	3.67E+02(4=)	4.25E+02(6+)	5.50E+02(7+)	3.70E+02(5=)	3.50E+02(2)
	Std.	3.60E+01	**9.53E+00**	1.23E+01	1.04E+01	1.41E+01	4.41E+01	9.74E+00	2.75E+01
f_{22}	Mean	7.82E+03(8+)	9.51E+02(3+)	1.94E+02(2+)	2.39E+03(5+)	7.16E+03(7+)	4.74E+03(6+)	1.48E+03(4+)	**1.00E+02(1)**
	Std.	8.64E+02	4.37E+02	6.40E+01	3.14E+03	1.53E+03	1.26E+03	2.82E+03	**0.00E+00**
f_{23}	Mean	9.81E+02(7+)	4.64E+02(2=)	5.29E+02(5+)	5.27E+02(4+)	5.92E+02(6+)	1.12E+03(8+)	5.25E+02(3+)	**4.58E+02(1)**
	Std.	1.04E+02	2.54E+01	1.31E+01	8.60E+00	1.84E+01	1.50E+02	1.09E+01	**6.48E+01**
f_{24}	Mean	8.99E+02(7+)	5.92E+02(3=)	5.87E+02(2=)	5.94E+02(5=)	6.67E+02(6+)	1.12E+03(8+)	5.93E+02(4=)	**5.64E+02(1)**
	Std.	7.38E+01	7.64E+01	2.73E+01	9.34E+00	2.15E+01	1.82E+02	1.01E+01	**4.57E+01**
f_{25}	Mean	9.42E+03(8+)	5.76E+02(6+)	4.37E+02(5=)	3.87E+02(3=)	4.18E+02(4=)	1.79E+03(7+)	3.87E+02(2=)	**3.87E+02(1)**
	Std.	1.98E+03	5.52E+01	2.70E+01	2.23E-02	1.62E+01	4.19E+02	2.61E-02	**3.79E-02**
f_{26}	Mean	8.22E+03(8+)	2.30E+03(2+)	2.61E+03(4+)	2.62E+03(5+)	3.36E+03(6+)	6.65E+03(7+)	2.60E+03(3+)	**1.44E+03(1)**
	Std.	1.12E+03	2.83E+02	1.77E+02	1.58E+02	5.91E+02	5.94E+02	1.96E+02	**5.10E+02**
f_{27}	Mean	7.77E+02(7+)	6.23E+02(6+)	5.18E+02(5=)	4.97E+02(2=)	5.00E+02(3=)	1.52E+03(8+)	**4.96E+02(1-)**	5.02E+02(4)
	Std.	9.82E+01	6.18E+01	9.49E+00	7.72E+00	6.52E-05	2.74E+02	**9.70E+00**	1.08E+01
f_{28}	Mean	5.71E+03(8+)	8.74E+02(6+)	4.63E+02(4+)	3.50E+02(3=)	5.00E+02(5+)	3.11E+03(7+)	**3.14E+02(1=)**	3.28E+02(2)
	Std.	1.41E+03	2.46E+02	2.44E+01	7.58E+01	6.60E-05	5.83E+02	**3.57E+01**	4.77E+01
f_{29}	Mean	2.86E+03(7+)	8.91E+02(5+)	5.37E+02(4+)	5.29E+02(3+)	1.70E+03(6+)	3.32E+03(8+)	5.28E+02(2+)	**4.29E+02(1)**
	Std.	5.57E+02	1.90E+02	1.25E+02	1.31E+02	2.58E+02	6.96E+02	1.19E+02	**7.05E+01**
f_{30}	Mean	1.52E+09(8+)	5.73E+06(6+)	4.63E+03(4+)	2.00E+03(2=)	1.61E+06(5+)	6.67E+08(7+)	**1.99E+03(1=)**	2.07E+03(3)
	Std.	8.41E+08	8.55E+06	1.12E+03	6.78E+01	2.12E+06	3.17E+08	**5.12E+01**	1.21E+02
"+/=/-"		29/0/0	22/3/4	22/7/0	14/14/1	27/2/0	29/0/0	12/13/4	-



(a) F_1 **(b)** F_4 **(c)** F_{10}

(d) F_{11} **(e)** F_{16} **(f)** F_{17}

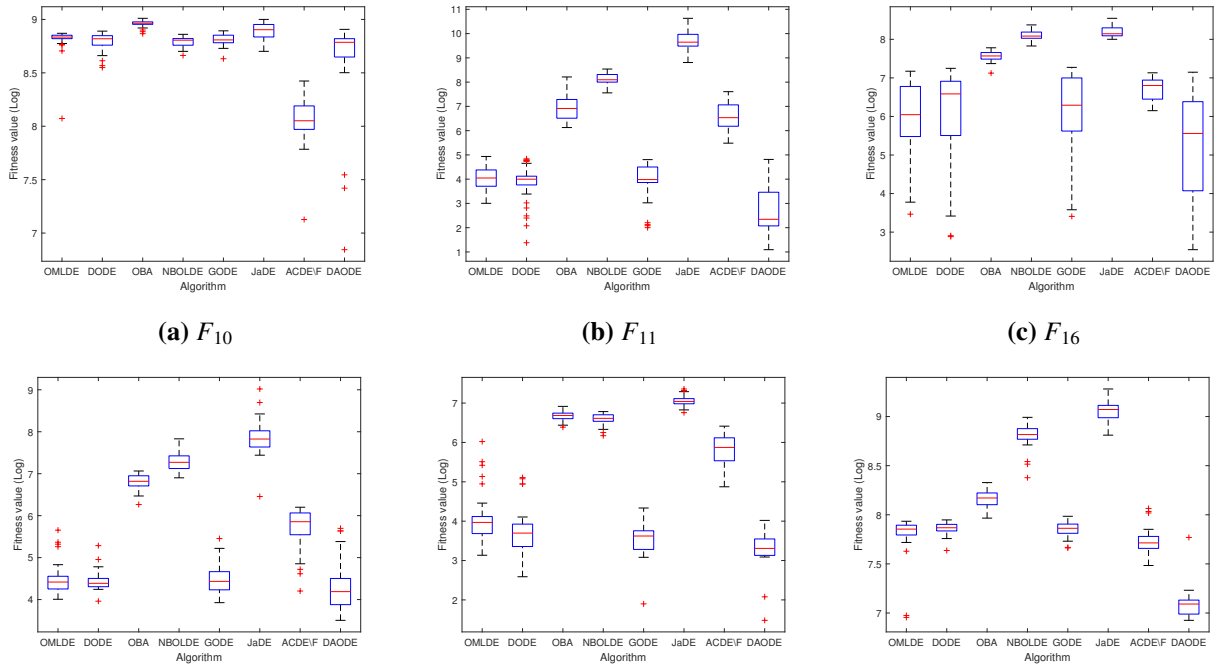
(g) F_{20} **(h)** F_{22} **(i)** F_{26}

Figure 5. The convergence curves of 30 independent runs for the comparison algorithm on CEC2017 benchmark suite ($D = 30$).

4.3.2. Comparison of OBL strategies

To test the performance of the dynamically allocated OBL mechanism, this paper evaluates the experimental results of each OBL combined with DE in the OBL pool and compared their performance with that of DAODE. A total of 13 strategies are included: original, QOBL [41], QROBL [42], SOBL [43], GOBL [44], FQROBL [45], POBL [46], COBL [47], RBL [48], OCL [49], EO/REO [50] and COOBL [51]. Their information is briefly described in Table 2, and detailed information can be found in the corresponding literature. Note that, when combined with DE, the algorithm names are: ODE(OBL), QODE(QOBL), QRODE(QROBL), EODE(EO), REODE(REO), FQRODE(FQROBL), GODE(GOBL), RODE(RBL), CODE(COBL), PODE(POBL), SODE(SOBL), OCDE(OCL), and COODE(COOBL).

The comparison results of DAODE with the OBL variants are listed in Tables 7 and 8. For unimodal



(a) F_{10} **(b)** F_{11} **(c)** F_{16}

(d) F_{17} **(e)** F_{20} **(f)** F_{26}

Figure 6. The boxplots of 30 independent runs for the comparison algorithm on CEC2017 benchmark suite ($D = 30$).

functions ($f_1 - f_3$), both the DAODE and OBL variants performed well on the test function of $30D$. When the number of dimensions increased to 50, the performance of the OBL variants exhibited a decreasing trend, but DAODE still found effective solutions. When solving the multimodal functions ($f_4 - f_{10}$) problem, the performance of DAODE was not outstanding and only outperformed the OBL variants on f_{10} for $30D$ and f_7 for $50D$. Notably, DAODE had a significant effect on the hybrid and composition functions of $30D$. However, when $D = 50$, although the results of DAODE are not optimal, it can be observed from the data in Table 8 that DAODE is extremely close to the OBL variants with the optimal solution. In addition, the results from the sign test ("+/=/-") can be obtained such that the comprehensive performance of DAODE outperforms each OBL variant and has a significant effect compared to ODE, QODE, GODE, RODE, PODE, SODE, OCDE, and COODE, which verifies the robustness of the dynamically allocated OBL mechanism proposed in this study.

4.4. Results of statistic tests

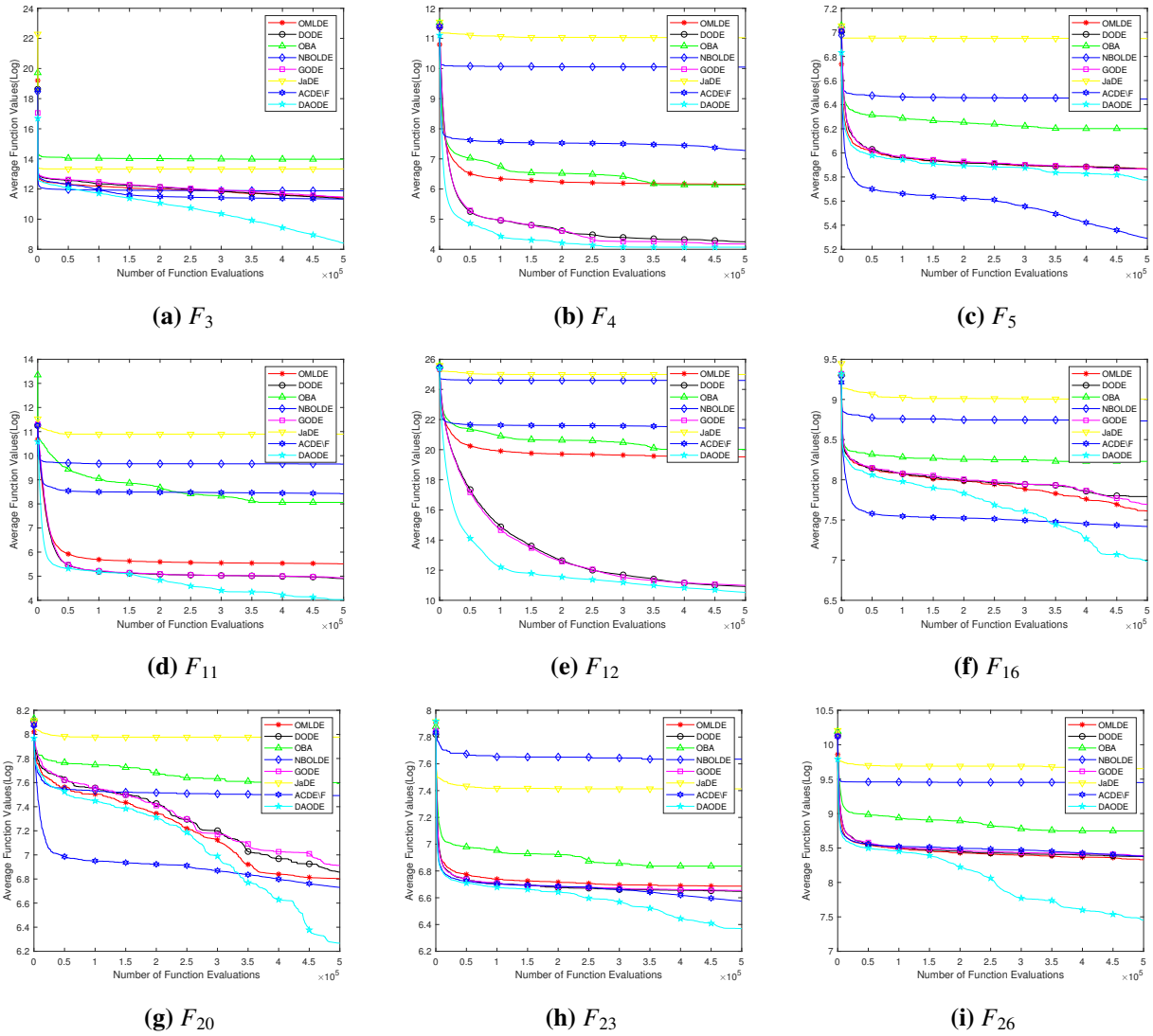
In recent years, statistical tests have been widely used in the field of computational intelligence to further evaluate the performance of algorithms and determine whether the proposed algorithm is significantly different from existing state of the art algorithms [63]. In this study, two nonparametric tests, the Wilcoxon and Friedman tests, were used to verify whether DAODE is improved compared to state of the art algorithms.

Table 6. The comparison results of DAODE and state of the art algorithms on CEC2017 benchmark suite (D = 50).

Fun.		JaDE	ACDE/F	OMLDE	DODE	OBA	NBODE	GODE	DAODE
f_1	Mean	1.94E+11(8+)	1.24E+10(6+)	2.83E+09(5+)	3.09E+00(2+)	1.45E+09(4+)	8.50E+10(7+)	1.61E+01(3+)	**5.45E-14(1)**
	Std.	3.39E+10	2.59E+09	2.09E+09	1.40E+01	6.00E+08	8.61E+09	4.98E+01	**7.75E-14**
f_3	Mean	6.20E+05(7+)	8.35E+04(2+)	9.20E+04(4+)	8.43E+04(3+)	1.18E+06(8+)	1.45E+05(6+)	9.53E+04(5+)	**4.52E+03(1)**
	Std.	1.74E+05	1.35E+04	1.60E+04	1.99E+04	1.77E+06	1.11E+04	1.49E+04	**2.28E+03**
f_4	Mean	6.13E+04(8+)	1.45E+03(6+)	4.73E+02(5+)	7.02E+01(3+)	4.63E+02(4+)	2.31E+04(7+)	6.51E+01(2=)	**5.85E+01(1)**
	Std.	1.80E+04	4.72E+02	1.10E+02	5.26E+01	8.52E+01	5.50E+03	5.30E+01	**4.93E+01**
f_5	Mean	1.04E+03(8+)	**1.98E+02(1=)**	3.53E+02(5=)	3.52E+02(3=)	4.94E+02(6+)	6.30E+02(7+)	3.52E+02(4=)	3.22E+02(2)
	Std.	5.63E+01	**2.61E+01**	1.73E+01	1.11E+01	3.00E+01	3.15E+01	1.26E+01	7.85E+01
f_6	Mean	1.27E+02(8+)	1.96E+01(5+)	1.19E+01(4+)	6.15E-07(2-)	6.48E+01(6+)	9.60E+01(7+)	**3.99E-08(1-)**	4.42E-04(3)
	Std.	8.58E+00	2.79E+00	3.98E+00	2.32E-06	7.33E+00	4.71E+00	**1.17E-07**	1.48E-03
f_7	Mean	4.12E+03(8+)	**3.83E+02(1=)**	4.44E+02(5=)	4.08E+02(4=)	5.92E+02(6+)	1.22E+03(7+)	4.00E+02(3=)	3.90E+02(2)
	Std.	4.48E+02	**5.71E+01**	2.33E+01	1.02E+01	3.57E+01	7.07E+01	1.29E+01	1.43E+01
f_8	Mean	1.03E+03(8+)	**2.06E+02(1-)**	3.56E+02(5=)	3.53E+02(4=)	5.16E+02(6+)	6.58E+02(7+)	3.50E+02(3=)	3.25E+02(2)
	Std.	8.38E+01	**2.16E+01**	1.62E+01	1.19E+01	2.54E+01	3.07E+01	1.21E+01	5.47E+01
f_9	Mean	6.99E+04(8+)	3.81E+03(5+)	5.00E+02(4+)	**6.36E-02(1-)**	1.34E+04(6+)	3.21E+04(7+)	1.82E-01(2-)	2.30E+00(3)
	Std.	9.58E+03	1.09E+03	2.37E+02	**1.97E-01**	3.24E+03	2.44E+03	8.32E-01	7.87E+00
f_{10}	Mean	1.33E+04(7=)	**6.53E+03(1-)**	1.31E+04(6=)	1.30E+04(5=)	1.42E+04(8=)	1.27E+04(2=)	1.30E+04(4=)	1.27E+04(3)
	Std.	1.26E+03	**3.80E+02**	3.83E+02	4.26E+02	3.62E+02	5.10E+02	7.03E+02	1.02E+03
f_{11}	Mean	5.39E+04(8+)	4.53E+03(6+)	2.48E+02(4+)	1.34E+02(2+)	3.17E+03(5+)	1.55E+04(7+)	1.38E+02(3+)	**5.63E+01(1)**
	Std.	1.13E+04	1.52E+03	5.06E+01	3.34E+01	1.26E+03	2.99E+03	2.70E+01	**1.84E+01**
f_{12}	Mean	7.14E+10(8+)	2.04E+09(6+)	3.02E+08(4+)	5.53E+04(2+)	4.96E+08(5+)	4.81E+10(7+)	5.92E+04(3+)	**3.72E+04(1)**
	Std.	1.75E+10	1.02E+09	1.94E+08	3.03E+04	2.39E+08	1.39E+10	4.34E+04	**3.58E+04**
f_{13}	Mean	4.12E+10(8+)	6.53E+08(6+)	7.03E+03(4+)	**2.94E+02(1-)**	4.32E+06(5+)	2.29E+10(7+)	3.34E+02(2-)	1.62E+03(3)
	Std.	1.44E+10	5.55E+08	5.05E+03	**7.91E+01**	2.09E+06	7.79E+09	1.78E+02	6.79E+03
f_{14}	Mean	5.65E+07(8+)	1.94E+06(5+)	3.28E+03(4+)	**1.26E+02(1-)**	7.47E+06(6+)	3.05E+07(7+)	1.26E+02(2=)	1.29E+02(3)
	Std.	2.43E+07	1.31E+06	5.54E+03	**8.01E+00**	5.21E+06	1.74E+07	6.26E+00	3.80E+01
f_{15}	Mean	1.76E+10(8+)	6.48E+07(6+)	1.11E+03(4+)	1.10E+02(2+)	1.14E+07(5+)	3.63E+09(7+)	1.11E+02(3+)	**6.09E+01(1)**
	Std.	6.09E+09	7.48E+07	1.65E+03	6.87E+00	1.98E+07	2.00E+09	6.17E+00	**6.83E+01**
f_{16}	Mean	8.14E+03(8+)	1.67E+03(2=)	2.02E+03(3+)	2.42E+03(5+)	3.75E+03(6+)	6.20E+03(7+)	2.19E+03(4+)	**1.08E+03(1)**
	Std.	1.49E+03	3.38E+02	6.92E+02	7.26E+02	3.39E+02	6.98E+02	7.99E+02	**1.06E+03**
f_{17}	Mean	2.70E+05(8+)	1.08E+03(3+)	**6.33E+02(1-)**	1.10E+03(5+)	2.55E+03(6+)	4.10E+03(7+)	1.08E+03(4+)	8.04E+02(2)
	Std.	2.73E+05	2.14E+02	**3.36E+02**	5.38E+02	2.95E+02	1.04E+03	4.64E+02	5.68E+02
f_{18}	Mean	2.16E+08(8+)	4.16E+06(5+)	2.28E+04(4+)	**6.65E+02(1-)**	5.37E+07(6+)	6.42E+07(7+)	5.71E+02(2-)	1.05E+03(3)
	Std.	1.24E+08	3.19E+06	1.43E+04	**7.23E+02**	3.28E+07	3.35E+07	3.52E+02	1.19E+03
f_{19}	Mean	6.78E+09(8+)	1.25E+07(6+)	1.86E+03(4+)	5.90E+01(2+)	3.82E+05(5+)	1.67E+09(7+)	6.33E+01(3+)	**3.96E+01(1)**
	Std.	2.25E+09	1.97E+07	1.78E+03	1.19E+01	1.35E+06	7.18E+08	4.25E+00	**2.48E+01**
f_{20}	Mean	2.91E+03(8+)	8.37E+02(2+)	9.01E+02(3+)	9.50E+02(4+)	1.98E+03(7+)	1.79E+03(6+)	1.01E+03(5+)	**5.27E+02(1)**
	Std.	2.74E+02	2.54E+02	4.31E+02	4.89E+02	1.75E+02	1.57E+02	3.85E+02	**5.20E+02**
f_{21}	Mean	1.24E+03(8+)	**4.04E+02(1-)**	5.54E+02(4=)	5.55E+02(5=)	6.68E+02(6+)	9.52E+02(7+)	5.51E+02(3=)	5.14E+02(2)
	Std.	6.96E+01	**2.78E+01**	2.39E+01	1.31E+01	3.06E+01	5.87E+01	1.69E+01	6.65E+01
f_{22}	Mean	1.44E+04(8+)	**7.35E+03(1-)**	1.06E+04(3+)	1.34E+04(5+)	1.42E+04(7+)	1.38E+04(6+)	1.33E+04(4+)	8.12E+03(2)
	Std.	1.01E+03	**7.77E+02**	5.27E+03	3.16E+02	4.00E+02	6.96E+02	3.52E+02	6.21E+03
f_{23}	Mean	1.66E+03(7+)	7.16E+02(2+)	8.02E+02(5+)	7.72E+02(3+)	9.31E+02(6+)	2.07E+03(8+)	7.76E+02(4+)	**5.83E+02(1)**
	Std.	1.88E+02	3.47E+01	2.34E+01	1.51E+01	4.18E+01	3.42E+02	1.45E+01	**1.43E+02**
f_{24}	Mean	1.53E+03(7+)	9.27E+02(5+)	8.61E+02(4+)	8.47E+02(3=)	1.04E+03(6+)	2.04E+03(8+)	8.46E+02(2=)	**7.81E+02(1)**
	Std.	1.76E+02	1.57E+02	2.35E+01	1.12E+01	3.83E+01	2.43E+02	1.31E+01	**1.08E+02**
f_{25}	Mean	3.42E+04(8+)	2.06E+03(6+)	9.01E+02(5+)	4.92E+02(2=)	6.62E+02(4=)	9.17E+03(7+)	**4.89E+02(1=)**	5.09E+02(3)
	Std.	1.00E+04	3.61E+02	2.23E+02	2.60E+01	6.74E+01	1.09E+03	**2.50E+01**	3.81E+01
f_{26}	Mean	1.56E+04(8+)	4.36E+03(4+)	4.14E+03(2+)	4.34E+03(3+)	6.30E+03(6+)	1.27E+04(7+)	4.40E+03(5+)	**1.72E+03(1)**
	Std.	2.34E+03	4.96E+02	6.90E+02	2.33E+02	5.81E+02	6.67E+02	1.92E+02	**1.60E+02**
f_{27}	Mean	1.34E+03(7+)	1.19E+03(6+)	7.76E+02(5+)	**5.00E+02(1-)**	5.00E+02(2-)	3.88E+03(8+)	5.56E+02(3=)	5.90E+02(4)
	Std.	2.33E+02	1.64E+02	7.31E+01	**1.37E+00**	6.19E-05	8.30E+02	8.08E+01	4.48E+01
f_{28}	Mean	1.12E+04(8+)	2.30E+03(6+)	8.83E+02(5+)	4.88E+02(3=)	5.00E+02(4=)	7.76E+03(7+)	**4.66E+02(1=)**	4.77E+02(2)
	Std.	2.99E+03	8.57E+02	1.56E+02	1.93E+01	5.25E-05	7.68E+02	**1.79E+01**	2.33E+01
f_{29}	Mean	3.33E+04(8+)	1.80E+03(5+)	1.47E+03(4+)	1.09E+03(3+)	3.15E+03(6+)	1.53E+04(7+)	9.18E+02(2+)	**4.30E+02(1)**
	Std.	3.77E+04	2.63E+02	3.83E+02	5.33E+02	3.82E+02	5.71E+03	5.21E+02	**9.93E+01**
f_{30}	Mean	7.76E+09(8+)	1.10E+08(6+)	2.52E+06(4+)	**5.94E+05(1-)**	4.60E+06(5+)	2.94E+09(7+)	5.95E+05(2-)	6.44E+05(3)
	Std.	2.75E+09	5.85E+07	4.75E+05	**2.34E+04**	6.59E+06	1.23E+09	2.32E+04	5.90E+04
"+/=/-"		29/0/0	21/3/4	23/5/1	14/8/7	26/3/0	28/1/0	15/11/3	-

4.4.1. Wilcoxon-test results

The Wilcoxon test tests whether there is a significant difference between two datasets. In this study, a significance level of 0.05 was used to test the difference between DAODE and other algorithms,



(a) F_3 **(b)** F_4 **(c)** F_5

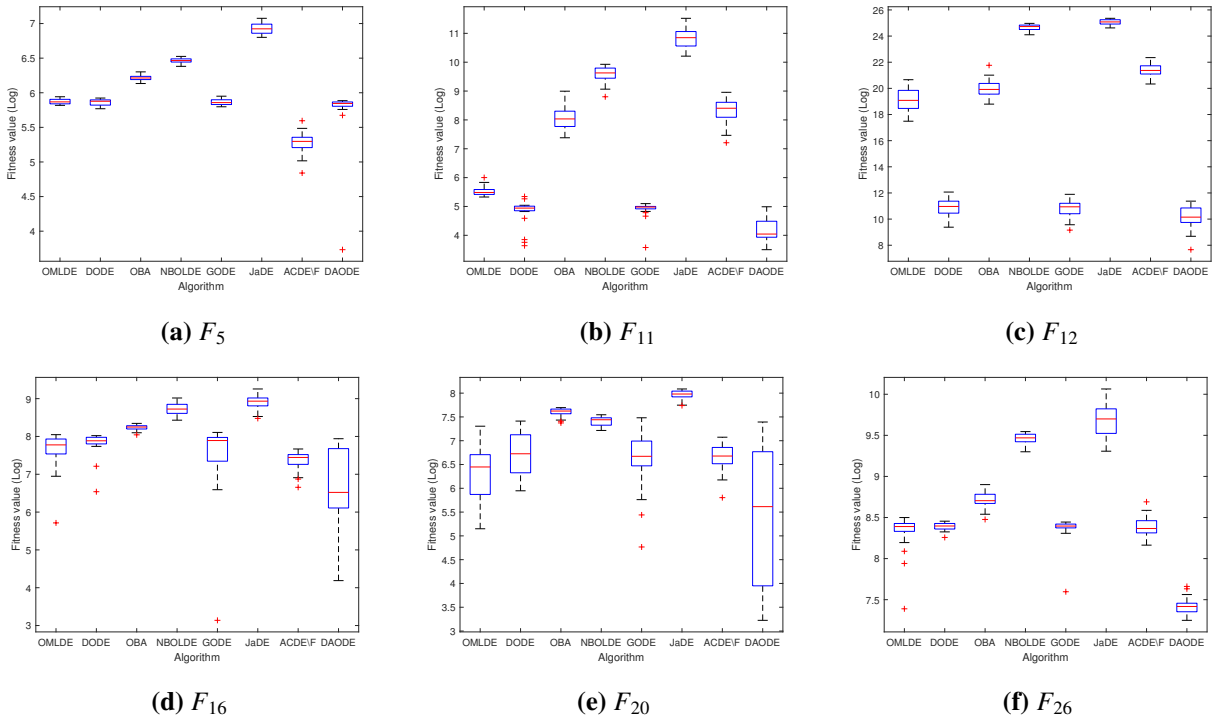
(d) F_{11} **(e)** F_{12} **(f)** F_{16}

(g) F_{20} **(h)** F_{23} **(i)** F_{26}

Figure 7. The convergence curves of 30 independent runs for the comparison algorithm on CEC2017 benchmark suite ($D = 50$).

and the comparison results between DAODE and the state of the art algorithms, OBL variants, are respectively presented in Tables 9 and 10. Terms R^+ and R^- denote the sum of the rankings that DAODE outperforms or underperforms the competitors, respectively, and when the p-value is less than 0.05 it implies that DAODE is significantly different from the competitors. The computational formulas for them can be found in [63].

From the results in Table 9, it can be observed that the R^+ values of DAODE compared to the competitors are much larger than the R^- values whenever $D = 30$ or $D = 50$, and the p-values are also less than 0.05. Therefore, DAODE differs significantly from these algorithms, which corresponds to the comparison results in Section 4.3.1. Furthermore, this study exhibited significant differences between the DAODE and OBL variants (Table 10). When $D = 30$, it can be observed that DAODE outperforms all the OBL variants according to the values of R^+, R^-, and p-value. However, when



(a) F_5 **(b)** F_{11} **(c)** F_{12}

(d) F_{16} **(e)** F_{20} **(f)** F_{26}

Figure 8. The boxplots of 30 independent runs for the comparison algorithm on CEC2017 benchmark suite ($D = 50$).

$D = 50$, although the R^+ values of DAODE were greater than the R^- values compared to the QROBL, FQROBL, and COBL variants, the p-values were all greater than 0.05; thus, there was no significant difference between DAODE and these three variants. Overall, it can be concluded that DAODE is effective for high-dimensional problems.

4.4.2. Friedman-test results

The Friedman test determines whether a significant difference exists between the algorithms using rank. The test results of DAODE with the state of the art algorithms and the OBL variants are summarized in Tables 11 and 12, respectively. Note that "ARV" is the average ranking value of each algorithm in the CEC2017 benchmark suite. For ease of observation, the "ARV" values are ranked to obtain the actual ranking values denoted as "Rank", where a smaller "Rank" value implies better algorithm performance.

From the data in the table, it can be observed that DAODE is ranked first in both tables, indicating that its overall performance is better than that of several state of the art algorithms and OBL variants. In addition, as the problem dimension increases, the ranking of DAODE remains at the top position, although the "ARV" of DAODE becomes larger, and the performance may decrease. Thus, the overall performance of DAODE was significantly better than that of the other algorithms.

Table 7. The comparison results of DAODE with various OBLs on CEC2017 benchmark suite (D = 30).

Fun.		ODE	QODE	QRODE	EODE	REODE	FQRODE	GODE	RODE	CODE	PODE	SODE	OCDE	COODE	DAODE
f_1	Mean	8.53E-15(=)	3.79E-15(=)	4.26E-15(=)	5.68E-15(=)	5.21E-15(=)	2.46E-14(=)	1.29E+10(+)	5.68E-15(=)	6.63E-15(=)	9.00E-15(=)	5.21E-15(=)	6.63E-15(=)	3.39E+00(+)	**9.47E-16**
	Std.	7.08E-15	6.39E-15	6.62E-15	7.08E-15	6.97E-15	8.98E-14	2.99E+10	7.08E-15	7.21E-15	6.97E-15	6.97E-15	7.21E-15	1.86E+01	**3.61E-15**
f_3	Mean	7.15E+01(+)	7.38E+01(+)	5.78E+02(+)	6.63E+01(+)	5.48E+01(+)	1.03E+01(+)	4.03E+01(+)	4.58E+01(+)	1.26E+02(+)	9.48E+01(+)	7.44E+01(+)	3.73E+01(+)	7.80E+02(+)	**1.25E-12**
	Std.	1.17E+02	6.81E+01	2.74E+03	1.02E+02	6.20E+01	1.28E+01	7.93E+01	3.29E+01	1.36E+02	7.88E+01	7.98E+01	3.67E+01	6.67E+02	**4.82E-12**
f_4	Mean	5.87E+01(=)	5.87E+01(=)	2.43E+02(+)	5.91E+01(=)	5.89E+01(=)	4.04E+02(+)	6.46E+01(+)	5.95E+01(=)	5.66E+01(=)	5.66E+01(=)	5.86E+01(=)	5.93E+01(=)	8.59E+01(+)	**5.65E+01**
	Std.	1.01E+00	1.01E+00	1.07E+03	1.70E+00	1.41E+00	2.05E+03	1.49E+01	2.11E+00	1.07E+01	1.07E+01	1.04E-13	1.92E+00	1.01E+00	**1.56E+01**
f_5	Mean	1.75E+02(+)	1.63E+02(-)	**1.46E+02(-)**	1.76E+02(+)	1.82E+02(+)	1.18E+02(-)	1.54E+02(-)	1.81E+02(+)	1.69E+02(=)	1.81E+02(+)	1.75E+02(+)	1.78E+02(+)	1.75E+02(+)	1.69E+02
	Std.	1.05E+01	3.33E+01	**5.48E+01**	1.58E+01	8.78E+00	5.85E+01	1.07E+02	8.84E+00	1.14E+01	9.61E+00	8.85E+00	9.35E+00	1.04E+01	8.14E+00
f_6	Mean	4.93E-10(-)	9.37E-10(-)	8.60E-10(-)	1.35E-09(-)	**1.60E-10(-)**	3.51E-01(+)	4.92E-02(+)	2.65E-10(-)	3.52E-09(-)	6.55E-10(-)	2.33E-10(-)	3.93E-10(-)	1.73E-08(-)	1.11E-03
	Std.	1.23E-09	1.65E-09	1.70E-09	6.65E-09	**2.10E-10**	1.66E+00	2.64E-01	4.84E-10	9.30E-09	9.03E-10	4.05E-10	1.33E-09	4.75E-08	6.08E-03
f_7	Mean	2.12E+02(=)	2.06E+02(=)	2.13E+02(=)	2.11E+02(=)	2.08E+02(=)	2.06E+02(=)	**1.95E+02(-)**	2.07E+02(=)	2.07E+02(=)	2.09E+02(=)	2.11E+02(=)	2.12E+02(=)	2.10E+02(=)	1.99E+02
	Std.	8.54E+00	1.11E+01	9.24E+00	8.99E+00	7.59E+00	2.78E+01	**1.54E+02**	1.03E+01	1.08E+01	1.47E+01	9.24E+00	9.41E+00	1.18E+01	1.07E+01
f_8	Mean	1.78E+02(+)	1.76E+02(+)	8.02E+01(-)	1.78E+02(+)	1.80E+02(+)	**7.59E+01(-)**	1.50E+02(-)	1.77E+02(+)	1.58E+02(-)	1.83E+02(+)	1.80E+02(+)	1.81E+02(+)	1.79E+02(+)	1.66E+02
	Std.	7.84E+00	1.26E+01	5.14E+01	9.99E+00	8.75E+00	**4.21E+01**	9.25E+01	1.26E+01	3.83E+01	1.01E+01	8.47E+00	9.61E+00	1.04E+01	9.66E+00
f_9	Mean	**0.00E+00(-)**	**0.00E+00(-)**	3.79E-15(-)	**0.00E+00(-)**	**0.00E+00(-)**	2.82E+02(+)	1.91E+01(+)	**0.00E+00(-)**	3.79E-15(-)	3.79E-15(-)	3.79E-15(-)	**0.00E+00(-)**	**0.00E+00(-)**	8.17E-02
	Std.	**0.00E+00**	**0.00E+00**	2.08E-14	**0.00E+00**	**0.00E+00**	7.59E+02	1.04E+02	**0.00E+00**	2.08E-14	2.08E-14	2.08E-14	**0.00E+00**	**0.00E+00**	1.71E-01
f_{10}	Mean	6.64E+03(+)	6.61E+03(+)	6.66E+03(+)	6.34E+03(+)	6.59E+03(+)	6.24E+03(+)	5.66E+03(+)	6.51E+03(+)	6.57E+03(+)	6.66E+03(+)	6.52E+03(+)	6.69E+03(+)	5.90E+03(+)	**5.87E+03**
	Std.	3.37E+02	4.72E+02	6.21E+02	5.32E+02	5.12E+02	1.12E+03	1.59E+03	4.54E+02	6.18E+02	3.90E+02	5.92E+02	3.51E+02	1.15E+03	**1.19E+03**
f_{11}	Mean	5.60E+01(+)	6.68E+01(+)	5.56E+01(+)	7.17E+01(+)	5.94E+01(+)	6.13E+01(+)	7.02E+01(+)	7.03E+01(+)	7.91E+01(+)	6.42E+01(+)	7.14E+01(+)	6.51E+01(+)	6.14E+01(+)	**3.08E+01**
	Std.	2.30E+01	2.79E+01	3.32E+01	3.98E+01	3.26E+01	3.16E+01	1.52E+02	3.74E+01	3.05E+01	3.16E+01	3.07E+01	2.57E+01	3.10E+01	**2.97E+01**
f_{12}	Mean	9.05E+03(-)	9.72E+03(-)	1.04E+04(=)	1.13E+04(+)	8.20E+03(-)	4.32E+08(+)	5.92E+04(+)	8.97E+03(-)	1.15E+04(+)	1.16E+04(+)	**7.70E+03(-)**	8.41E+03(-)	7.91E+03(-)	1.02E+04
	Std.	7.97E+03	7.65E+03	6.79E+03	1.03E+04	6.06E+03	1.65E+09	2.38E+05	6.85E+03	8.21E+03	8.78E+03	**6.02E+03**	8.55E+03	7.00E+03	1.27E+04
f_{13}	Mean	7.96E+01(-)	7.97E+01(-)	7.91E+01(-)	8.47E+01(=)	8.08E+01(-)	8.37E+01(=)	1.48E+08(+)	8.04E+01(-)	8.19E+01(-)	7.98E+01(-)	8.18E+01(-)	**7.74E+01(-)**	8.38E+01(=)	8.66E+01
	Std.	9.24E+00	6.99E+00	6.75E+00	1.83E+01	6.88E+00	1.75E+01	8.11E+08	6.29E+00	7.16E+00	6.15E+00	7.75E+00	**7.99E+00**	7.87E+00	2.66E+01
f_{14}	Mean	6.33E+01(+)	6.30E+01(+)	6.56E+01(+)	6.25E+01(+)	6.36E+01(+)	6.41E+01(+)	5.36E+01(=)	6.34E+01(+)	6.50E+01(+)	6.21E+01(+)	6.17E+01(+)	6.36E+01(+)	6.38E+01(+)	**5.01E+01**
	Std.	4.75E+00	4.75E+00	4.12E+00	5.21E+00	4.17E+00	4.34E+00	3.49E+01	5.85E+00	5.18E+00	6.47E+00	7.66E+00	5.64E+00	4.20E+00	**1.53E+01**
f_{15}	Mean	3.49E+01(+)	3.64E+01(+)	3.83E+01(+)	3.74E+01(+)	3.72E+01(+)	3.68E+01(+)	2.51E+01(+)	3.65E+01(+)	3.68E+01(+)	3.51E+01(+)	3.41E+01(+)	3.51E+01(+)	3.68E+01(+)	**1.54E+01**
	Std.	9.00E+00	7.13E+00	6.51E+00	6.82E+00	6.23E+00	6.75E+00	1.59E+01	5.90E+00	4.68E+00	6.24E+00	9.57E+00	9.42E+00	5.70E+00	**3.60E+01**
f_{16}	Mean	7.10E+02(+)	6.78E+02(+)	4.91E+02(+)	5.48E+02(+)	6.75E+02(+)	6.03E+02(+)	6.06E+02(+)	7.13E+02(+)	4.17E+02(+)	5.34E+02(+)	6.18E+02(+)	6.20E+02(+)	5.80E+02(+)	**2.22E+02**
	Std.	4.34E+02	3.95E+02	3.32E+02	4.17E+02	4.09E+02	4.48E+02	3.52E+02	4.17E+02	1.99E+02	3.88E+02	4.37E+02	4.93E+02	3.89E+02	**3.17E+02**
f_{17}	Mean	9.22E+01(+)	9.31E+01(+)	7.95E+01(=)	9.37E+01(+)	1.07E+02(+)	1.26E+02(+)	1.01E+02(+)	9.84E+01(+)	1.10E+02(+)	1.03E+02(+)	1.31E+02(+)	8.68E+01(+)	8.47E+01(+)	**7.45E+01**
	Std.	3.51E+01	3.18E+01	2.77E+01	4.59E+01	5.04E+01	9.65E+01	7.56E+01	5.09E+01	7.46E+01	6.68E+01	7.44E+01	1.58E+01	1.68E+01	**7.29E+01**
f_{18}	Mean	3.66E+01(+)	3.69E+01(+)	3.76E+01(+)	3.72E+01(+)	3.67E+01(+)	3.53E+01(+)	1.23E+03(+)	3.86E+01(+)	3.80E+01(+)	3.51E+01(+)	3.63E+01(+)	3.70E+01(+)	7.38E+02(+)	**2.56E+01**
	Std.	5.49E+00	4.77E+00	4.68E+00	3.29E+00	5.78E+00	5.83E+00	6.17E+03	6.07E+00	4.28E+00	5.14E+00	4.79E+00	5.76E+00	1.59E+03	**2.14E+00**
f_{19}	Mean	2.11E+01(+)	1.69E+01(+)	1.44E+01(+)	1.69E+01(+)	1.61E+01(+)	1.75E+01(+)	2.47E+01(+)	1.63E+01(+)	1.75E+01(+)	1.65E+01(+)	1.57E+01(+)	1.61E+01(+)	2.07E+01(+)	**7.34E+00**
	Std.	5.93E+00	6.96E+00	6.09E+00	7.15E+00	5.53E+00	6.76E+00	4.77E+01	5.96E+00	6.78E+00	6.31E+00	6.39E+00	6.45E+00	6.02E+00	**2.57E+00**
f_{20}	Mean	7.09E+01(+)	4.02E+01(+)	4.94E+01(+)	6.38E+01(+)	5.18E+01(+)	7.17E+01(+)	1.25E+02(+)	4.85E+01(+)	5.96E+01(+)	7.37E+01(+)	4.15E+01(+)	3.10E+01(=)	5.07E+01(+)	**3.27E+01**
	Std.	1.24E+02	2.93E+01	5.52E+01	5.67E+01	4.85E+01	6.46E+01	1.94E+02	5.07E+01	6.85E+01	7.35E+01	2.66E+01	1.72E+01	3.85E+01	**2.72E+01**
f_{21}	Mean	3.70E+02(+)	3.70E+02(+)	**3.32E+02(-)**	3.67E+02(+)	3.70E+02(+)	3.37E+02(-)	3.36E+02(-)	3.64E+02(+)	3.57E+02(+)	3.63E+02(+)	3.67E+02(+)	3.65E+02(+)	3.70E+02(+)	3.50E+02
	Std.	7.86E+00	9.96E+00	**5.31E+01**	1.24E+01	1.02E+01	7.74E+01	9.18E+01	8.49E+00	2.87E+01	1.02E+01	1.06E+01	1.05E+01	8.29E+00	2.75E+01
f_{22}	Mean	1.48E+03(+)	**1.00E+02(=)**	**1.00E+02(=)**	4.67E+03(+)	2.91E+02(+)	**1.00E+02(=)**	8.56E+02(+)	3.62E+03(+)	**1.00E+02(=)**	9.80E+02(+)	4.16E+03(+)	3.39E+02(+)	3.79E+03(+)	**1.00E+02**
	Std.	2.81E+03	**0.00E+00**	**1.45E-13**	3.29E+03	1.05E+03	**0.00E+00**	1.89E+03	3.37E+03	**1.15E-13**	2.29E+03	3.39E+03	1.31E+03	2.75E+03	**0.00E+00**
f_{23}	Mean	5.21E+02(+)	5.30E+02(+)	5.25E+02(+)	5.24E+02(+)	5.23E+02(+)	5.25E+02(+)	4.94E+02(+)	5.24E+02(+)	5.02E+02(+)	5.25E+02(+)	5.26E+02(+)	5.25E+02(+)	5.27E+02(+)	**4.58E+02**
	Std.	9.91E+00	1.16E+01	1.10E+01	1.07E+01	1.09E+01	1.36E+01	8.63E+01	1.31E+01	3.99E+01	1.05E+01	1.05E+01	1.26E+01	9.79E+00	**6.48E+01**
f_{24}	Mean	5.91E+02(+)	5.93E+02(+)	5.91E+02(+)	5.96E+02(+)	5.95E+02(+)	5.90E+02(+)	**5.48E+02(-)**	5.95E+02(+)	5.93E+02(+)	5.93E+02(+)	5.95E+02(+)	5.91E+02(+)	5.93E+02(+)	5.64E+02
	Std.	1.09E+01	9.22E+00	2.63E+01	1.01E+01	8.89E+00	1.09E+01	**6.59E+01**	8.77E+00	1.38E+01	8.01E+00	7.47E+00	1.31E+01	9.01E+00	4.57E+01
f_{25}	Mean	**3.87E+02(=)**	**3.87E+02(=)**	4.01E+02(+)	**3.87E+02(=)**	**3.87E+02(=)**	4.32E+02(+)	5.99E+02(+)	**3.87E+02(=)**	**3.87E+02(=)**	**3.87E+02(=)**	**3.87E+02(=)**	**3.87E+02(=)**	**3.87E+02(=)**	**3.87E+02**
	Std.	**2.92E-02**	**3.07E-02**	8.02E+01	**2.78E-02**	**3.20E-02**	2.56E+02	1.16E+03	**3.16E-02**	**1.82E+00**	**2.37E-02**	**3.02E-02**	**2.57E-02**	**5.31E-02**	**3.79E-02**
f_{26}	Mean	2.58E+03(+)	2.71E+03(+)	2.32E+03(+)	2.57E+03(+)	2.61E+03(+)	1.59E+03(+)	1.99E+03(+)	2.61E+03(+)	2.55E+03(+)	2.62E+03(+)	2.58E+03(+)	2.60E+03(+)	2.59E+03(+)	**1.44E+03**
	Std.	3.18E+02	9.77E+01	8.26E+02	1.28E+02	1.58E+02	1.18E+03	6.51E+02	1.29E+02	1.53E+02	1.34E+02	1.28E+02	1.27E+02	1.29E+02	**5.10E+02**
f_{27}	Mean	4.97E+02(-)	4.98E+02(-)	4.98E+02(-)	4.96E+02(-)	4.97E+02(-)	4.97E+02(-)	5.20E+02(+)	4.97E+02(-)	4.96E+02(-)	4.96E+02(-)	**4.94E+02(-)**	4.96E+02(-)	5.10E+02(+)	5.02E+02
	Std.	7.27E+00	1.04E+01	8.64E+00	7.84E+00	1.04E+01	9.65E+00	7.41E+01	6.78E+00	6.97E+00	8.40E+00	**7.48E+00**	9.07E+00	2.03E+01	1.08E+01
f_{28}	Mean	3.17E+02(-)	3.27E+02(=)	3.18E+02(-)	3.33E+02(+)	3.16E+02(-)	3.22E+02(=)	3.74E+02(+)	3.11E+02(-)	3.18E+02(-)	3.20E+02(-)	3.18E+02(-)	**3.07E+02(-)**	4.72E+02(+)	3.28E+02
	Std.	3.92E+01	5.03E+01	4.00E+01	5.65E+01	4.16E+01	4.43E+01	6.58E+01	3.26E+01	4.16E+01	4.53E+01	4.16E+01	**2.62E+01**	6.40E+01	4.77E+01
f_{29}	Mean	5.16E+02(+)	5.89E+02(+)	5.13E+02(+)	5.48E+02(+)	5.24E+02(+)	5.30E+02(+)	5.83E+02(+)	5.33E+02(+)	5.73E+02(+)	5.94E+02(+)	5.47E+02(+)	5.52E+02(+)	5.79E+02(+)	**4.29E+02**
	Std.	1.06E+02	1.61E+02	1.15E+02	1.28E+02	1.15E+02	1.13E+02	1.97E+02	1.36E+02	1.92E+02	1.39E+02	1.48E+02	1.34E+02	1.28E+02	**7.05E+01**
f_{30}	Mean	2.01E+03(-)	**1.99E+03(-)**	2.00E+03(-)	2.01E+03(-)	2.01E+03(-)	1.98E+07(+)	2.28E+03(+)	2.01E+03(-)	2.01E+03(-)	2.00E+03(-)	2.01E+03(-)	**1.99E+03(-)**	2.38E+03(+)	2.07E+03
	Std.	7.20E+01	**5.47E+01**	7.30E+01	7.33E+01	6.83E+01	1.09E+08	1.19E+03	5.36E+01	7.15E+01	6.37E+01	5.61E+01	**5.49E+01**	2.71E+02	1.21E+02
"+/=/-"		18/4/7	17/6/6	16/5/8	21/4/4	18/4/7	20/5/4	23/1/5	18/4/7	16/6/7	19/4/6	18/4/7	17/5/7	23/3/3	

4.5. Performance analysis of newly introduced strategies and the parameter

In this section, the effectiveness and robustness of the strategies proposed in this study, such as ABM, MIA, and their candidate strategies, are tested for a comparative performance analysis. The effectiveness of the DAO is verified in Section 4.3.2, therefore, it is not analyzed here. Owing to space limitations, the results were tested using only the CEC2017 benchmark suite of 30D.

4.5.1. Effectiveness of MIA

It is well-known that elite and inferior roles make up a small proportion of the population, while commoner roles make up the majority of the population. Therefore, in MIA, the individuals of DAODE are classified and archived in three levels: elite (A_E), commoner (A_C), and inferior roles (A_I), and set A_E:A_C:A_I to 2:5:3. To test the effectiveness of this proportion, MIA was compared with two other candidate proportions for performance: MIA-2 (3:5:2) and MIA-3 (1:5:4). Here, the percentage of commoner roles is set to 50%, such that the impact of elite and inferior roles with different proportions

Table 8. The comparison results of DAODE with various OBLs on CEC2017 benchmark suite (D = 50).

Fun.		ODE	QODE	QRODE	EODE	REODE	FQRODE	GODE	RODE	CODE	PODE	SODE	OCDE	COODE	DAODE
f_1	Mean	2.56E+00(+)	9.82E+01(+)	3.14E+00(+)	1.76E+00(+)	5.41E+01(+)	4.81E+09(+)	1.36E+10(+)	1.28E+01(+)	6.99E+00(+)	1.27E+01(+)	1.19E+01(+)	6.36E+00(+)	6.18E+03(+)	**5.45E-14**
	Std.	4.16E+00	4.07E+02	5.11E+00	3.33E+00	2.15E+02	2.15E+10	4.18E+10	4.26E+01	1.19E+01	3.67E+01	3.96E+01	1.03E+01	6.18E+03	**7.75E-14**
f_3	Mean	9.29E+04(+)	6.60E+04(+)	8.25E+04(+)	8.82E+04(+)	8.96E+04(+)	8.53E+04(+)	6.68E+04(+)	9.26E+04(+)	1.02E+05(+)	9.02E+04(+)	8.99E+04(+)	8.94E+04(+)	9.91E+04(+)	**4.52E+03**
	Std.	1.95E+04	1.07E+04	1.41E+04	1.98E+04	1.72E+04	2.01E+04	3.02E+04	1.86E+04	1.71E+04	1.46E+04	1.58E+04	1.66E+04	1.95E+04	**2.28E+03**
f_4	Mean	8.34E+01(+)	6.83E+01(+)	5.07E+01(-)	8.77E+01(+)	8.71E+01(+)	**5.04E+01(-)**	5.77E+03(+)	7.54E+01(+)	7.21E+01(+)	8.47E+01(+)	7.48E+01(+)	7.83E+01(+)	1.44E+02(+)	5.85E+01
	Std.	4.64E+01	5.50E+01	4.55E+01	5.02E+01	5.39E+01	**3.83E+01**	1.75E+04	4.81E+01	4.69E+01	5.17E+01	4.91E+01	4.99E+01	3.70E+01	4.93E+01
f_5	Mean	3.50E+02(+)	3.21E+02(=)	2.34E+02(+)	3.49E+02(+)	3.54E+02(+)	**2.13E+02(-)**	2.75E+02(-)	3.53E+02(+)	3.37E+02(+)	3.52E+02(+)	3.51E+02(+)	3.46E+02(+)	3.49E+02(+)	3.22E+02
	Std.	1.38E+01	8.12E+01	1.26E+02	1.70E+01	1.23E+01	**1.09E+02**	1.91E+02	1.35E+01	4.38E+01	1.47E+01	1.55E+01	1.63E+01	1.30E+01	7.85E+01
f_6	Mean	**2.40E-08(-)**	1.06E-07(-)	6.49E-04(+)	8.19E-08(-)	3.17E-08(-)	1.74E-01(+)	1.46E-02(+)	3.96E-07(-)	5.03E-08(-)	3.43E-07(-)	6.94E-08(-)	1.66E-07(-)	3.29E-07(-)	4.42E-04
	Std.	**8.58E-08**	3.71E-07	2.90E-03	3.66E-07	9.34E-08	7.24E-01	2.97E-02	1.15E-06	1.17E-07	1.16E-06	1.42E-07	5.03E-07	8.70E-07	1.48E-03
f_7	Mean	4.03E+02(=)	4.05E+02(=)	4.10E+02(+)	4.00E+02(=)	4.04E+02(=)	4.18E+02(+)	4.24E+02(+)	4.00E+02(=)	4.06E+02(=)	4.00E+02(=)	4.02E+02(=)	4.02E+02(=)	4.05E+02(=)	**3.90E+02**
	Std.	1.23E+01	1.76E+01	1.36E+01	1.46E+01	1.82E+01	6.55E+01	1.93E+02	1.20E+01	1.70E+01	1.67E+01	1.21E+01	1.67E+01	9.59E+00	**1.43E+01**
f_8	Mean	3.50E+02(+)	3.47E+02(+)	**1.25E+02(-)**	3.54E+02(+)	3.56E+02(+)	2.34E+02(-)	2.84E+02(-)	3.53E+02(+)	3.10E+02(-)	3.49E+02(+)	3.48E+02(+)	3.56E+02(+)	3.46E+02(+)	3.25E+02
	Std.	1.18E+01	1.45E+01	**5.43E+01**	1.34E+01	1.36E+01	1.01E+02	1.94E+02	1.56E+01	7.88E+01	1.53E+01	1.48E+01	1.47E+01	1.55E+01	5.47E+01
f_9	Mean	2.27E-02(-)	2.27E-02(-)	5.22E+02(+)	7.26E-02(-)	**4.48E-03(-)**	5.31E+02(+)	2.38E+02(+)	6.81E-02(-)	5.44E-02(-)	9.98E-02(-)	5.44E-02(-)	4.54E-02(-)	1.27E-01(-)	2.30E+00
	Std.	1.02E-01	1.02E-01	2.34E+03	1.66E-01	**2.00E-02**	1.67E+03	9.44E+02	1.66E-01	1.39E-01	2.36E-01	1.39E-01	1.40E-01	2.07E-01	7.87E+00
f_{10}	Mean	1.32E+04(+)	1.33E+04(+)	1.15E+04(-)	1.31E+04(+)	1.33E+04(+)	1.23E+04(-)	**1.10E+04(-)**	1.30E+04(+)	1.31E+04(+)	1.32E+04(+)	1.29E+04(+)	1.31E+04(+)	1.29E+04(+)	1.27E+04
	Std.	3.06E+02	3.35E+02	8.59E+02	2.74E+02	3.15E+02	1.20E+03	**3.53E+03**	7.03E+02	2.82E+02	3.17E+02	6.87E+02	3.90E+02	1.76E+03	1.02E+03
f_{11}	Mean	1.44E+02(+)	1.35E+02(+)	1.39E+02(+)	1.29E+02(+)	1.39E+02(+)	1.43E+02(+)	1.26E+03(+)	1.38E+02(+)	1.44E+02(+)	1.41E+02(+)	1.31E+02(+)	1.32E+02(+)	1.36E+02(+)	**5.63E+01**
	Std.	1.04E+01	2.77E+01	2.60E+01	3.05E+01	2.29E+01	2.90E+01	4.05E+03	2.17E+01	6.92E+00	2.37E+01	3.32E+01	2.66E+01	3.10E+01	**1.84E+01**
f_{12}	Mean	9.33E+04(+)	7.41E+04(+)	5.92E+04(+)	5.16E+04(+)	5.94E+04(+)	7.82E+09(+)	7.80E+07(+)	5.77E+04(+)	6.91E+04(+)	6.24E+04(+)	6.61E+04(+)	6.91E+04(+)	1.44E+05(+)	**3.72E+04**
	Std.	6.76E+04	6.08E+04	3.88E+04	3.47E+04	4.14E+04	1.77E+10	3.45E+08	3.48E+04	4.01E+04	3.69E+04	7.98E+04	4.51E+04	1.27E+05	**3.58E+04**
f_{13}	Mean	4.19E+02(-)	3.40E+02(-)	1.05E+03(-)	**3.05E+02(-)**	3.63E+02(-)	1.42E+03(-)	3.65E+03(+)	3.24E+02(-)	3.17E+02(-)	3.99E+02(-)	3.50E+02(-)	5.20E+02(-)	1.39E+03(-)	1.62E+03
	Std.	3.60E+02	8.43E+01	3.12E+03	**8.61E+01**	3.07E+02	4.32E+03	5.30E+03	1.07E+02	1.06E+02	2.15E+02	1.43E+02	9.98E+02	4.18E+03	6.79E+03
f_{14}	Mean	1.27E+02(=)	1.28E+02(=)	**1.26E+02(-)**	1.28E+02(=)	**1.26E+02(-)**	1.30E+02(=)	3.40E+02(+)	1.30E+02(=)	**1.26E+02(-)**	1.28E+02(=)	1.29E+02(=)	1.28E+02(=)	1.31E+02(=)	1.29E+02
	Std.	5.89E+00	1.05E+01	**8.31E+00**	1.42E+01	**8.86E+00**	7.40E+00	9.59E+02	6.90E+00	**7.58E+00**	7.68E+00	6.37E+00	9.83E+00	4.64E+00	3.80E+01
f_{15}	Mean	1.09E+02(+)	1.07E+02(+)	1.07E+02(+)	1.13E+02(+)	1.08E+02(+)	1.13E+02(+)	1.20E+09(+)	1.09E+02(+)	1.14E+02(+)	1.06E+02(+)	1.14E+02(+)	1.11E+02(+)	1.07E+02(+)	**6.09E+01**
	Std.	1.15E+01	8.49E+00	1.03E+01	9.78E+00	5.20E+00	9.97E+00	5.36E+09	8.39E+00	8.52E+00	8.65E+00	9.20E+00	9.29E+00	7.54E+00	**6.83E+01**
f_{16}	Mean	2.69E+03(+)	2.50E+03(+)	1.50E+03(+)	2.39E+03(+)	2.27E+03(+)	1.65E+03(+)	1.75E+03(+)	2.29E+03(+)	**9.19E+02(-)**	2.57E+03(+)	2.45E+03(+)	2.67E+03(+)	2.56E+03(+)	1.08E+03
	Std.	3.28E+02	5.85E+02	9.22E+02	6.53E+02	7.44E+02	1.26E+03	8.77E+02	1.02E+03	**5.17E+02**	6.07E+02	6.65E+02	5.50E+02	6.53E+02	1.06E+03
f_{17}	Mean	1.23E+03(+)	7.37E+02(-)	1.21E+03(+)	1.13E+03(+)	1.16E+03(+)	9.01E+02(+)	1.07E+03(+)	9.01E+02(+)	**7.21E+02(-)**	1.08E+03(+)	1.10E+03(+)	1.09E+03(+)	1.24E+03(+)	8.04E+02
	Std.	4.58E+02	2.84E+02	4.54E+02	4.71E+02	5.38E+02	3.86E+02	3.87E+02	4.21E+02	**2.13E+02**	5.28E+02	4.31E+02	4.10E+02	4.78E+02	5.68E+02
f_{18}	Mean	7.12E+02(-)	1.07E+03(=)	**4.61E+02(-)**	5.81E+02(-)	6.82E+02(-)	1.24E+03(+)	5.01E+04(+)	5.51E+02(-)	8.10E+02(-)	6.94E+02(-)	6.53E+02(-)	4.62E+02(-)	1.31E+04(+)	1.05E+03
	Std.	7.33E+02	1.36E+03	**2.79E+02**	2.94E+02	3.53E+02	1.78E+03	1.94E+05	2.99E+02	6.06E+02	6.50E+02	4.88E+02	2.84E+02	2.85E+04	1.19E+03
f_{19}	Mean	6.16E+01(+)	6.26E+01(+)	2.23E+07(+)	6.20E+01(+)	6.02E+01(+)	6.11E+01(+)	6.58E+07(+)	6.28E+01(+)	6.61E+01(+)	5.98E+01(+)	5.78E+01(+)	6.26E+01(+)	5.99E+01(+)	**3.96E+01**
	Std.	1.09E+01	6.67E+00	9.98E+07	6.18E+00	1.11E+01	1.30E+01	2.91E+08	5.26E+00	1.41E+01	1.19E+01	1.52E+01	6.20E+00	1.05E+01	**2.48E+01**
f_{20}	Mean	8.68E+02(+)	9.15E+02(+)	**3.72E+02(-)**	8.39E+02(+)	1.12E+03(+)	3.85E+02(-)	1.00E+03(+)	1.16E+03(+)	5.77E+02(+)	9.46E+02(+)	1.07E+03(+)	1.07E+03(+)	8.60E+02(+)	5.27E+02
	Std.	4.15E+02	4.59E+02	**1.94E+02**	3.91E+02	4.02E+02	1.87E+02	4.10E+02	4.41E+02	2.34E+02	4.16E+02	4.64E+02	4.38E+02	6.33E+02	5.20E+02
f_{21}	Mean	5.51E+02(+)	5.52E+02(+)	**4.81E+02(-)**	5.52E+02(+)	5.46E+02(+)	4.76E+02(-)	5.00E+02(-)	5.52E+02(+)	5.48E+02(+)	5.54E+02(+)	5.53E+02(+)	5.58E+02(+)	5.49E+02(+)	5.14E+02
	Std.	1.08E+01	1.67E+01	**1.21E+02**	2.05E+01	2.00E+01	1.29E+02	9.84E+01	1.74E+01	1.92E+01	1.11E+01	1.58E+01	8.92E+00	1.34E+01	6.65E+01
f_{22}	Mean	1.32E+04(+)	7.84E+03(-)	9.21E+03(+)	1.33E+04(+)	1.32E+04(+)	**3.95E+03(-)**	1.18E+04(+)	1.33E+04(+)	1.32E+04(+)	1.34E+04(+)	1.31E+04(+)	1.32E+04(+)	1.33E+04(+)	8.12E+03
	Std.	5.68E+02	6.50E+03	6.13E+03	3.39E+02	3.23E+02	**6.04E+03**	2.70E+03	3.30E+02	3.73E+02	3.70E+02	4.57E+02	5.33E+02	4.16E+02	6.21E+03
f_{23}	Mean	7.70E+02(+)	7.67E+02(+)	7.77E+02(+)	7.69E+02(+)	7.74E+02(+)	7.74E+02(+)	6.38E+02(+)	7.75E+02(+)	7.41E+02(+)	7.75E+02(+)	7.70E+02(+)	7.69E+02(+)	7.69E+02(+)	**5.83E+02**
	Std.	1.88E+01	1.58E+01	1.59E+01	2.04E+01	1.80E+01	1.51E+01	1.39E+02	1.49E+01	8.22E+01	1.61E+01	1.77E+01	1.69E+01	2.44E+01	**1.43E+02**
f_{24}	Mean	8.38E+02(+)	8.43E+02(+)	8.51E+02(+)	8.46E+02(+)	8.40E+02(+)	8.46E+02(+)	8.24E+02(+)	8.47E+02(+)	8.53E+02(+)	8.45E+02(+)	8.40E+02(+)	8.44E+02(+)	8.44E+02(+)	**7.81E+02**
	Std.	1.61E+01	1.53E+01	1.38E+01	1.62E+01	1.25E+01	8.85E+00	2.96E+02	1.12E+01	1.16E+01	1.73E+01	1.57E+01	1.87E+01	1.34E+01	**1.08E+02**
f_{25}	Mean	5.03E+02(=)	5.09E+02(=)	5.05E+02(=)	**4.85E+02(-)**	4.89E+02(-)	5.54E+02(+)	6.23E+02(+)	4.89E+02(-)	5.04E+02(=)	4.87E+02(-)	4.90E+02(-)	4.93E+02(-)	5.09E+02(=)	5.09E+02
	Std.	3.68E+01	3.40E+01	4.05E+01	**1.85E+01**	2.53E+01	3.25E+01	5.22E+02	2.54E+01	3.67E+01	2.03E+01	2.52E+01	3.01E+01	2.22E+01	3.81E+01
f_{26}	Mean	4.40E+03(+)	4.23E+03(+)	3.63E+03(+)	4.32E+03(+)	4.25E+03(+)	**5.97E+02(-)**	4.30E+03(+)	4.43E+03(+)	4.08E+03(+)	4.33E+03(+)	4.32E+03(+)	4.44E+03(+)	4.37E+03(+)	1.72E+03
	Std.	1.77E+02	6.84E+02	1.72E+03	2.63E+02	6.87E+02	**1.33E+03**	4.96E+03	1.38E+02	9.16E+02	2.04E+02	6.92E+02	1.42E+02	4.01E+02	5.20E+02
f_{27}	Mean	5.33E+02(-)	5.44E+02(-)	**5.30E+02(-)**	5.43E+02(-)	5.35E+02(-)	5.37E+02(-)	5.81E+02(-)	5.33E+02(-)	5.48E+02(-)	5.42E+02(-)	5.64E+02(-)	5.59E+02(-)	5.56E+02(-)	5.90E+02
	Std.	2.32E+01	3.56E+01	**3.05E+01**	2.93E+01	3.50E+01	2.52E+01	9.80E+01	2.22E+01	4.65E+01	3.76E+01	6.25E+01	8.85E+01	8.16E+01	4.48E+01
f_{28}	Mean	4.69E+02(-)	4.69E+02(-)	6.86E+02(+)	4.71E+02(=)	4.70E+02(=)	1.60E+03(+)	1.90E+03(+)	4.76E+02(=)	4.75E+02(=)	4.67E+02(-)	4.68E+02(-)	**4.66E+02(-)**	5.25E+02(+)	4.77E+02
	Std.	2.00E+01	2.00E+01	9.60E+02	2.17E+01	2.03E+01	2.77E+03	4.47E+03	2.39E+01	2.29E+01	1.77E+01	1.84E+01	**1.70E+01**	1.25E+02	2.33E+01
f_{29}	Mean	1.14E+03(+)	1.07E+03(+)	1.09E+03(+)	8.24E+02(+)	9.80E+02(+)	9.68E+02(+)	6.76E+02(+)	9.58E+02(+)	1.08E+03(+)	1.12E+03(+)	1.12E+03(+)	1.19E+03(+)	1.01E+03(+)	**4.30E+02**
	Std.	5.00E+02	5.44E+02	6.00E+02	4.54E+02	5.34E+02	5.46E+02	3.91E+02	4.56E+02	5.87E+02	5.30E+02	4.56E+02	4.71E+02	4.99E+02	**9.93E+01**
f_{30}	Mean	5.93E+05(-)	5.93E+05(-)	6.04E+05(-)	5.89E+05(-)	6.00E+05(-)	3.81E+08(+)	7.36E+05(+)	6.01E+05(-)	5.86E+05(-)	5.92E+05(-)	5.94E+05(-)	5.96E+05(-)	**5.32E+05(-)**	6.44E+05
	Std.	2.32E+04	2.26E+04	2.67E+04	2.17E+04	2.80E+04	1.20E+09	4.52E+05	2.93E+04	1.68E+04	2.69E+04	3.17E+04	2.44E+04	**7.99E+04**	5.90E+04
"+/=/-"		19/3/7	16/5/8	18/1/10	19/3/7	18/2/8	18/1/10	24/0/5	19/3/7	16/3/10	19/2/8	19/2/8	19/2/8	21/3/5	

on the performance of the algorithm can be tested.

The comparison results in Table 13 show that DAODE/MIA performed better than DAODE/MIA-2. Only a few test functions of DAODE/MIA-2 outperform DAODE/MIA, such as f_{16}, f_{28}, and f_{30}; however, a careful observation shows that there is no significant difference between them. This indicates that increasing the proportion of A_E can have a negative impact on the algorithm, which may be because some roles in A_E are not elite roles, causing the population to deviate from more promising areas. For DAODE/MIA-3, reducing the proportion of A_E while increasing the proportion of A_I, and the overall performance of DAODE/MIA-3 is enhanced compared with that of DAODE/MIA-2. Notably, DAODE/MIA-3 exhibits better performance than DAODE for some simple multimodal functions (e.g., f_5, f_8, and f_{10}) and composition functions (e.g., f_{21} and f_{22}), indicating that decreasing the A_E proportion is beneficial to achieve better solutions for some functions. However, increasing the A_I proportion also affects the algorithm performance. According to the result of "+/=/-", DAODE will perform better. The convergence curves of the three proportions in the MIA are shown in Figure 9.

Table 9. Wilcoxon-test results for comparing algorithms on CEC2017 benchmark suite.

DAODE vs.	$D = 30$			$D = 50$		
	R^+	R^-	p-value	R^+	R^-	p-value
JaDE	435	0	0.000003	435	0	0.000003
ACDE/F	394	41	0.000135	390	45	0.000191
OMLDE	435	0	0.000003	428	7	0.000005
DODE	366	69	0.001323	335	100	0.011062
OBA	434	1	0.000003	432	3	0.000004
NBODE	435	0	0.000003	434	1	0.000003
GODE	347	88	0.005107	331	104	0.014119

Table 10. Wilcoxon-test results for comparing algorithms on CEC2017 benchmark suite.

DAODE vs.	$D = 30$			$D = 50$		
	R^+	R^-	p-value	R^+	R^-	p-value
ODE(OBL)	354	81	0.003162	336	99	0.010397
QODE(QOBL)	320	86	0.007716	312	123	0.041014
QRODE(QROBL)	331	104	0.014119	276	159	0.205887
EODE(EO)	391	44	0.000176	328	107	0.016878
REODE(REO)	350	85	0.004169	334	101	0.011765
FQRODE(GQROBL)	347	59	0.001041	290	145	0.116955
GODE(GOBL)	381	54	0.000407	400	35	0.000079
RODE(RBL)	347	88	0.005107	333	102	0.012508
CODE(COBL)	363	72	0.001654	289	146	0.122090
PODE(POBL)	383	52	0.000345	332	103	0.013292
SODE(SOBL)	351	84	0.003892	336	99	0.010397
OCDE(OCL)	334	101	0.011765	333	102	0.012508
COODE(COOBL)	399	36	0.000087	373	62	0.000773

Therefore, the experiment proved that the proportions of A_E and A_I were smaller and similar, thereby verifying the effectiveness of MIA in improving the performance of DAODE.

4.5.2. Effectiveness of ABM

In DE, the strategy of the mutation operation significantly affects the performance of the algorithm. In ABM, the individuals of the mutation operation are selected from the archives of the MIA, the base vectors of the mutation operation from the A_E, and the difference vector of the perturbation is generated by the difference between the vectors selected from A_C and A_I. The details of ABM are described in Section 3.3. To test the effectiveness of ABM, the performance of DAODE/ABM is compared with two other variation strategies: the first is the classical DE/rand/1, and the other mutation strategy (ABM-2) is similar to ABM, except that the base vector of the mutation operation is selected from A_C, and the

Table 11. Friedman-test results for comparing algorithms on CEC2017 benchmark suite.

Algorithms	$D = 30$		$D = 50$	
	ARV	Rank	ARV	Rank
JaDE	7.79	8	7.83	8
ACDE/F	4.24	5	4.07	4
OMLDE	3.86	4	4.10	5
DODE	3.00	3	2.90	3
OBA	5.79	6	5.55	6
NBODE	6.93	7	6.83	7
GODE	2.66	2	2.83	2
DAODE	**1.72**	**1**	**1.90**	**1**

Table 12. Friedman-test results for OBL variants on CEC2017 benchmark suite.

OBLs	$D = 30$		$D = 50$	
	ARV	Rank	ARV	Rank
ODE(OBL)	7.26	7	8.00	9
QODE(QOBL)	7.26	6	6.66	4
QRODE(QROBL)	6.81	3	6.52	3
EODE(EO)	8.57	13	6.52	2
REODE(REO)	7.19	4	7.28	6
FQEODE(FQROBL)	7.93	9	7.86	8
GODE(GOBL)	8.41	12	9.31	14
RODE(RBL)	7.53	8	8.21	11
CODE(COBL)	7.97	10	6.95	5
PODE(POBL)	8.19	11	8.10	10
SODE(SOBL)	7.21	5	7.36	7
OCDE(OCL)	6.64	2	8.24	12
COODE(COOBL)	9.69	14	8.97	13
DAODE(DAOBL)	**4.34**	**1**	**5.03**	**1**

Table 13. Comparison results of MIA and candidate strategies on cec2017 benchmark suite ($D = 30$).

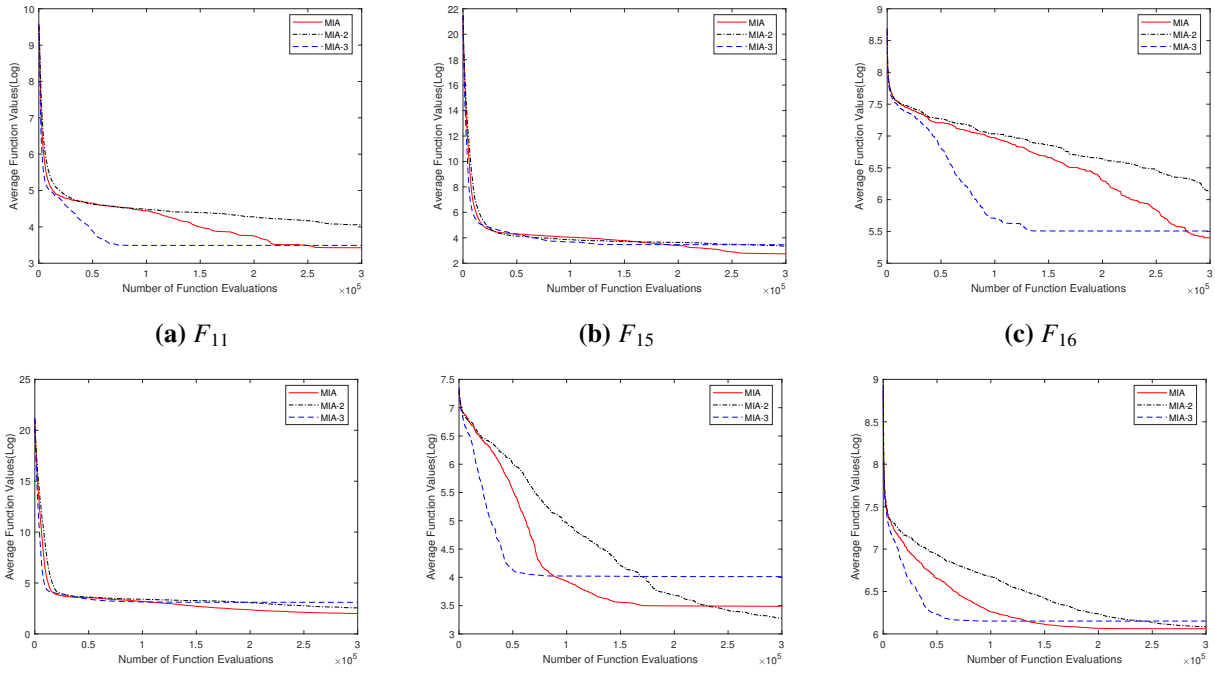
Fun	DAODE/MIA	DAODE/MIA-2	DAODE/MIA-3
	Mean ± Std	Mean ± Std	Mean ± Std
f_1	**9.47E-16 ± 3.61E-15**	**0.00E+00 ± 0.00E+00**	**5.21E-15 ± 6.97E-15**
f_3	**1.25E-12 ± 4.82E-12**	2.01E-04 ± 4.63E-04	**5.31E-14 ± 1.44E-14**
f_4	5.65E+01 ± 1.56E+01	5.87E+01 ± 6.92E-01	**4.85E+01 ± 2.38E+01**
f_5	1.69E+02 ± 8.14E+00	1.72E+02 ± 7.74E+00	**3.26E+01 ± 9.14E+00**
f_6	1.11E-03 ± 6.08E-03	**4.56E-09 ± 2.50E-08**	1.11E-02 ± 3.32E-02
f_7	1.99E+02 ± 1.07E+01	2.01E+02 ± 1.03E+01	**1.35E+02 ± 6.53E+01**
f_8	1.66E+02 ± 9.66E+00	1.68E+02 ± 1.16E+01	**4.51E+01 ± 4.27E+01**
f_9	8.17E-02 ± 1.71E-01	**0.00E+00 ± 0.00E+00**	9.75E-01 ± 1.47E+00
f_{10}	5.87E+03 ± 1.19E+03	6.17E+03 ± 7.46E+02	**3.70E+03 ± 2.26E+03**
f_{11}	**3.08E+01 ± 2.97E+01**	5.78E+01 ± 4.42E+01	3.28E+01 ± 2.74E+01
f_{12}	1.02E+04 ± 1.27E+04	**9.61E+03 ± 1.07E+04**	1.04E+04 ± 1.06E+04
f_{13}	8.66E+01 ± 2.66E+01	7.75E+01 ± 1.49E+01	**4.20E+01 ± 4.99E+01**
f_{14}	5.01E+01 ± 1.53E+01	6.10E+01 ± 4.60E+00	**4.62E+01 ± 3.69E+01**
f_{15}	**1.54E+01 ± 3.60E+01**	2.78E+01 ± 1.02E+01	3.17E+01 ± 3.47E+01
f_{16}	**2.22E+02 ± 3.17E+02**	4.59E+02 ± 4.29E+02	2.46E+02 ± 1.36E+02
f_{17}	7.45E+01 ± 7.29E+01	**7.43E+01 ± 1.02E+01**	8.92E+01 ± 7.00E+01
f_{18}	**2.56E+01 ± 2.14E+00**	3.00E+01 ± 6.00E+00	1.08E+02 ± 2.06E+02
f_{19}	**7.34E+00 ± 2.57E+00**	1.29E+01 ± 6.59E+00	2.23E+01 ± 2.95E+01
f_{20}	3.27E+01 ± 2.72E+01	**2.65E+01 ± 1.29E+01**	5.53E+01 ± 4.62E+01
f_{21}	3.50E+02 ± 2.75E+01	3.64E+02 ± 1.05E+01	**2.32E+02 ± 1.23E+01**
f_{22}	**1.00E+02 ± 0.00E+00**	**1.00E+02 ± 0.00E+00**	**1.00E+02 ± 0.00E+00**
f_{23}	4.58E+02 ± 6.48E+01	5.15E+02 ± 9.26E+00	**3.84E+02 ± 8.85E+00**
f_{24}	5.64E+02 ± 4.57E+01	5.86E+02 ± 1.28E+01	**4.57E+02 ± 1.19E+01**
f_{25}	3.87E+02 ± 3.79E-02	3.87E+02 ± 2.95E-02	**3.86E+02 ± 1.62E+00**
f_{26}	1.44E+03 ± 5.10E+02	2.28E+03 ± 6.20E+02	**1.34E+03 ± 2.41E+02**
f_{27}	**5.02E+02 ± 1.08E+01**	**5.02E+02 ± 9.68E+00**	5.14E+02 ± 1.28E+01
f_{28}	3.28E+02 ± 4.77E+01	**3.22E+02 ± 5.06E+01**	3.52E+02 ± 5.73E+01
f_{29}	**4.29E+02 ± 7.05E+01**	4.39E+02 ± 5.57E+01	4.69E+02 ± 5.86E+01
f_{30}	2.07E+03 ± 1.21E+02	**2.01E+03 ± 6.50E+01**	2.22E+03 ± 2.26E+02
"+/=/-"	-	15/9/5	13/6/10

difference vector of perturbation is the difference between the vectors selected from A_E and A_I.

The comparison results of the different mutation strategies are summarized in Table 14, where DAODE/ABM outperformed DAODE/rand/1 and DAODE/ABM-2 for most test functions. In particular, for the unimodal function f_3, simple multimodal functions f_8, f_{10}, hybrid functions f_{16}, f_{17}, and composition functions f_{20}, f_{29}, DAODE/ABM has an overwhelming advantage in terms of per-

Table 14. Comparison results of ABM and candidate strategies on CEC2017 benchmark suite ($D = 30$).

Fun	DAODE/ABM	DAODE/rand/1	DAODE/ABM-2
	Mean ± Std	Mean ± Std	Mean ± Std
f_1	**9.47E-16 ± 3.61E-15**	5.21E-15 ± 6.97E-15	3.30E-11 ± 4.37E-11
f_3	**1.25E-12 ± 4.82E-12**	4.94E+01 ± 4.80E+01	1.54E+02 ± 8.83E+01
f_4	**5.65E+01 ± 1.56E+01**	5.91E+01 ± 1.70E+00	5.89E+01 ± 1.41E+00
f_5	**1.69E+02 ± 8.14E+00**	1.76E+02 ± 1.15E+01	1.76E+02 ± 1.19E+01
f_6	1.11E-03 ± 6.08E-03	**2.27E-10 ± 5.75E-10**	2.85E-07 ± 1.27E-07
f_7	**1.99E+02 ± 1.07E+01**	2.10E+02 ± 7.13E+00	2.07E+02 ± 1.03E+01
f_8	**1.66E+02 ± 9.66E+00**	1.79E+02 ± 1.08E+01	1.77E+02 ± 9.94E+00
f_9	8.17E-02 ± 1.71E-01	**0.00E+00 ± 0.00E+00**	3.79E-15 ± 2.08E-14
f_{10}	**5.87E+03 ± 1.19E+03**	6.77E+03 ± 3.87E+02	6.32E+03 ± 8.92E+02
f_{11}	**3.08E+01 ± 2.97E+01**	6.62E+01 ± 2.98E+01	7.47E+01 ± 2.81E+01
f_{12}	1.02E+04 ± 1.27E+04	9.17E+03 ± 7.93E+03	**4.54E+03 ± 4.35E+03**
f_{13}	8.66E+01 ± 2.66E+01	**7.93E+01 ± 9.66E+00**	8.02E+01 ± 9.85E+00
f_{14}	**5.01E+01 ± 1.53E+01**	6.21E+01 ± 5.07E+00	6.39E+01 ± 4.74E+00
f_{15}	**1.54E+01 ± 3.60E+01**	3.47E+01 ± 7.38E+00	3.78E+01 ± 5.29E+00
f_{16}	**2.22E+02 ± 3.17E+02**	7.92E+02 ± 3.88E+02	9.74E+02 ± 2.63E+02
f_{17}	**7.45E+01 ± 7.29E+01**	9.09E+01 ± 3.05E+01	1.74E+02 ± 1.12E+02
f_{18}	**2.56E+01 ± 2.14E+00**	3.71E+01 ± 6.21E+00	3.81E+01 ± 3.16E+00
f_{19}	**7.34E+00 ± 2.57E+00**	1.53E+01 ± 6.31E+00	2.57E+01 ± 4.81E+00
f_{20}	**3.27E+01 ± 2.72E+01**	3.64E+01 ± 2.00E+01	4.77E+01 ± 1.20E+01
f_{21}	**3.50E+02 ± 2.75E+01**	3.68E+02 ± 9.14E+00	3.71E+02 ± 8.52E+00
f_{22}	**1.00E+02 ± 0.00E+00**	**1.00E+02 ± 1.15E-13**	**1.00E+02 ± 5.09E-13**
f_{23}	**4.58E+02 ± 6.48E+01**	5.25E+02 ± 1.10E+01	5.27E+02 ± 8.63E+00
f_{24}	**5.64E+02 ± 4.57E+01**	5.95E+02 ± 9.80E+00	5.94E+02 ± 8.89E+00
f_{25}	**3.87E+02 ± 3.79E-02**	**3.87E+02 ± 6.15E-01**	**3.87E+02 ± 2.56E-02**
f_{26}	**1.44E+03 ± 5.10E+02**	2.63E+03 ± 1.63E+02	2.65E+03 ± 1.41E+02
f_{27}	5.02E+02 ± 1.08E+01	4.96E+02 ± 8.98E+00	**4.95E+02 ± 6.06E+00**
f_{28}	3.28E+02 ± 4.77E+01	**3.18E+02 ± 4.16E+01**	**3.18E+02 ± 4.00E+01**
f_{29}	**4.29E+02 ± 7.05E+01**	5.10E+02 ± 9.02E+01	6.35E+02 ± 1.75E+02
f_{30}	2.07E+03 ± 1.21E+02	2.01E+03 ± 7.90E+01	**1.99E+03 ± 6.16E+01**
"+/=/-"	-	15/10/4	16/10/3



(a) F_{11} **(b)** F_{15} **(c)** F_{16}

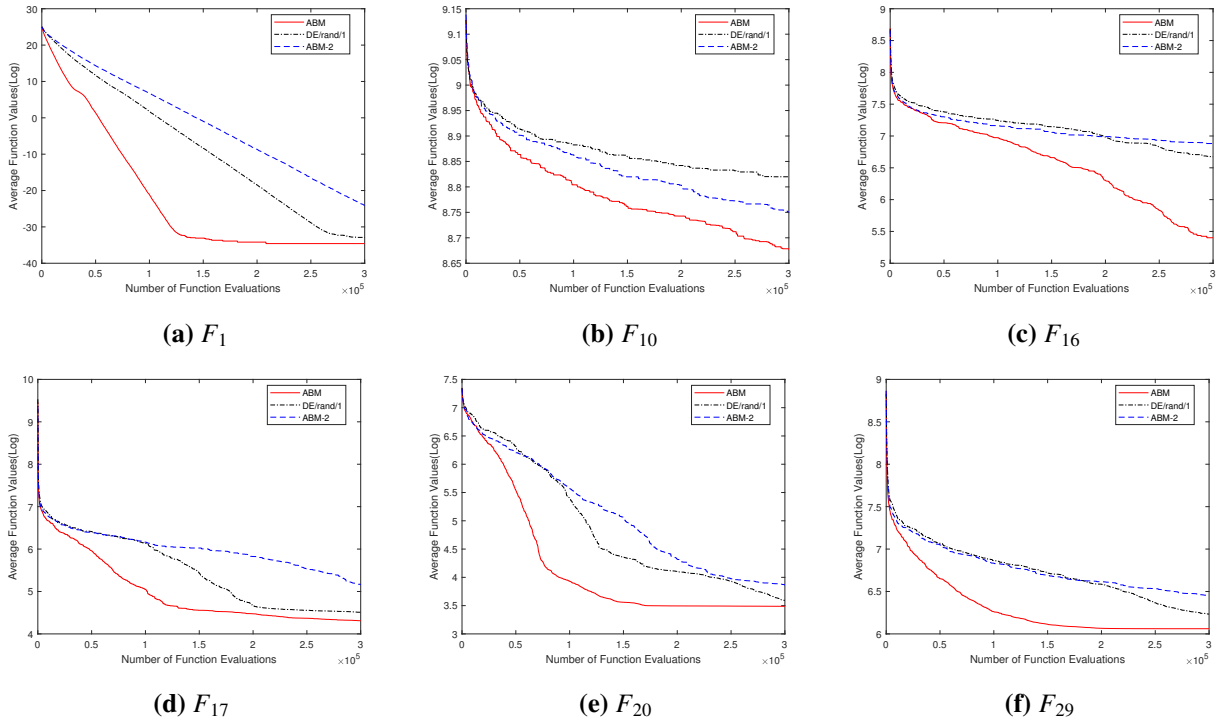
(d) F_{19} **(e)** F_{20} **(f)** F_{29}

Figure 9. The convergence curves of 30 independent runs for MIA and candidate strategies on CEC2017 benchmark suite ($D = 30$).

formance, as observed from the convergence curves in Figure 10. However, DAODE/rand/1 only performed better on individual test functions (e.g., f_6, f_9, and f_{13}), as did DAODE/ABM-2 (e.g., f_{12}, f_{21}). In addition, the comparison results of DAODE/rand/1 and DAODE/ABM-2 demonstrate that DAODE/ABM-2 appears to have reduced performance on several test functions, and the convergence curves in Figure 10 show that DAODE/ABM-2 has a slower convergence rate (e.g., f_{16}, f_{20}, and f_{29}). This is because the difference vector of the perturbation of DAODE/ABM-2 is the difference between the vectors selected from A_E and A_I, which causes the algorithm to be in a large range of perturbations at all times, and it appears that the algorithm converges excessively slow. Therefore, DAODE/ABM has two main advantages. In contrast, the base vector of the mutation operation is selected from A_E, which will benefit the offspring individuals in inheriting good genes and make the population develop into a more promising area. Conversely, the difference vector of perturbation is the difference between the vectors selected from A_C and A_I, which avoids the phenomenon of a large perturbation.

5. Conclusions and future work

In previous research, there was a focus on utilizing an OBL strategy optimization algorithm, however, different OBLs exhibit different performances in addressing different problems, making it crucial to study multiple OBL co-optimization algorithms. Therefore, this study proposes a dynamic allocation of opposition-based learning in differential evolution for multi-role individuals (DAODE). Before each generation of population updates, individuals in the population are assigned multiple roles and stored in their corresponding archives (MIA), and each role is identified based on its comprehensive



(a) F_1 **(b)** F_{10} **(c)** F_{16}

(d) F_{17} **(e)** F_{20} **(f)** F_{29}

Figure 10. The convergence curves of 30 independent runs for ABM and candidate strategies on CEC2017 benchmark suite ($D = 30$).

characteristics, including its fitness value and position. Subsequently, this paper classifies each role in the current population and selects the corresponding OBL strategy for each role using the dynamic allocation OBL mechanism (DAO). In the DAO, this paper defines a ranking mechanism to determine the characteristics of OBLs. This mechanism analyzes the diversity and fitness values of each OBL to obtain a comprehensive ranking of OBLs, providing a foundation for the adaptive selection of individuals. In addition, this paper proposes an individual-based archival mutation strategy (ABM) by MIA to drive populations to more promising regions.

To evaluate the performance of the proposed algorithm, DAODE is compared with several other algorithms on the CEC2017 benchmark suite and further test for significant differences between them. First, the experimental comparison results and statistical tested of DAODE with state of the art algorithms show that DAODE achieves significant performance results. The performance of DAODE is again compared with that of the OBL variants, and the experimental results confirm the effectiveness of the dynamic allocation OBL mechanism proposed by DAODE. Finally, the performance of the proposed new strategies is analyzed in this study, and the experiments demonstrate the effectiveness of MIA and ABM, which significantly improve the performance of the algorithms.

The experiment demonstrates that the comprehensive performance of DAODE surpasses that of its competitors and exhibits significant differences from them. However, DAODE still has some shortcomings, such as premature convergence when dealing with multimodal functions. First, further research is needed to determine the optimal ratio of the three roles in MIA, to obtain more accurate proportions. Second, it is worth exploring in detail whether the mutation operation in ABM is suitable for every individual in the population. Lastly, and most importantly, a more flexible OBL dynamic allocation

method must be developed in DAO to effectively utilize each OBL strategy. Therefore, in future work, this study will focus on enhancing DAODE's performance and applying it to real-world problems like industrial safety inspection, image classification, and feature selection. When used for feature selection, DAODE can select appropriate OBL strategies to address different types of data, thereby filtering out the most redundant features from the original data and improving the final classification performance.

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 was supported by National Natural Science Foundation of China (62106092); Natural Science Foundation of Fujian Province (2020J05169, 2022J01916); Natural Science Foundation of the Jiangsu Higher Education Institutions of China (22KJB520023).

Conflict of interest

No conflict of interest exit in the submission of this manuscript, and manuscript is approved by all authors for publication.

References

1. L. Migliorelli, D. Berardini, K. Cela, M. Coccia, L. Villani, E. Frontoni, et al., A store-and-forward cloud-based telemonitoring system for automatic assessing dysarthria evolution in neurological diseases from video-recording analysis, *Comput. Biol. Med.*, **163** (2023), 107194. https://doi.org/10.1016/j.compbiomed.2023.107194
2. W. Zhu, L. Fang, X. Ye, M. Medani, J. Escorcia-Gutierrez, IDRM: Brain tumor image segmentation with boosted rime optimization, *Comput. Biol. Med.*, **166** (2023), 107551. https://doi.org/10.1016/j.compbiomed.2023.107551
3. X. Zhang, Z. Wang, Z. Lu, Multi-objective load dispatch for microgrid with electric vehicles using modified gravitational search and particle swarm optimization algorithm, *Appl. Energy*, **306** (2022), 118018. http://dx.doi.org/10.1016/j.apenergy.2021.118018
4. S. Yin, Q. Luo, Y. Zhou, IBMSMA: An indicator-based multi-swarm slime mould algorithm for multi-objective truss optimization problems, *J. Bionic Eng.*, **20** (2023), 1333–1360. http://dx.doi.org/10.1007/s42235-022-00307-9
5. X. Ju, F. Liu, L. Wang, W. J. Lee, Wind farm layout optimization based on support vector regression guided genetic algorithm with consideration of participation among landowners, *Energy Convers. Manage.*, **196** (2019), 1267–1281. http://dx.doi.org/10.1016/j.enconman.2019.06.082
6. J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, MIT press, 1992.

7. J. Kennedy, R. Eberhart, Particle swarm optimization, in *Proceedings of ICNN'95-International Conference on Neural Networks*, **4** (1995), 1942–1948. http://dx.doi.org/10.1109/ICNN.1995.488968
8. M. Dorigo, V. Maniezzo, A. Colorni, Ant system: Optimization by a colony of cooperating agents, *IEEE Trans. Syst. Man Cybern. Part B*, **26** (1996), 29–41. http://dx.doi.org/10.1109/3477.484436
9. D. Karaboga, *An Idea Based on Honey Bee Swarm for Numerical Optimization*, Report, Technical report-tr06, Erciyes university, engineering faculty, computer, 2005.
10. 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
11. J. Lian, G. Hui, L. Ma, T. Zhu, X. Wu, A. A. Heidari, et al., Parrot optimizer: Algorithm and applications to medical problems, *Comput. Biol. Med.*, **172** (2024), 108064. https://doi.org/10.1016/j.compbiomed.2024.108064
12. H. Su, D. Zhao, A. A. Heidari, L. Liu, X. Zhang, M. Mafarja, et al., Rime: A physics-based optimization, *Neurocomputing*, **532** (2023), 183–214. https://doi.org/10.1016/j.neucom.2023.02.010
13. I. Ahmadianfar, A. A. Heidari, S. Noshadian, H. Chen, A. H. Gandomi, INFO: An efficient optimization algorithm based on weighted mean of vectors, *Expert Syst. Appl.*, **195** (2022), 116516. https://doi.org/10.1016/j.eswa.2022.116516
14. Y. Yang, H. Chen, A. A. Heidari, A. H. Gandomi, Hunger games search: Visions, conception, implementation, deep analysis, perspectives, and towards performance shifts, *Expert Syst. Appl.*, **177** (2021), 114864. https://doi.org/10.1016/j.eswa.2021.114864
15. R. Storn, K. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, *J. Global Optim.*, **11** (1997), 341–359. http://dx.doi.org/10.1023/A:1008202821328
16. D. Liu, Z. Hu, Q. Su, Neighborhood-based differential evolution algorithm with direction induced strategy for the large-scale combined heat and power economic dispatch problem, *Inf. Sci.*, **613** (2022), 469–493. https://doi.org/10.1016/j.ins.2022.09.025
17. C. Zhang, W. Zhou, W. Qin, W. Tang, A novel UAV path planning approach: Heuristic crossing search and rescue optimization algorithm, *Expert Syst. Appl.*, **215** (2023), 119243. https://doi.org/10.1016/j.eswa.2022.119243
18. M. Sajid, H. Mittal, S. Pare, M. Prasad, Routing and scheduling optimization for UAV assisted delivery system: A hybrid approach, *Appl. Soft Comput.*, **126** (2022), 109225. https://doi.org/10.1016/j.asoc.2022.109225
19. L. Abualigah, M. A. Elaziz, D. Yousri, M. A. A. Al-qaness, A. A. Ewees, R. A. Zitar, Augmented arithmetic optimization algorithm using opposite-based learning and lévy flight distribution for global optimization and data clustering, *J. Intell. Manuf.*, **34** (2023), 3523–3561. http://dx.doi.org/10.1007/s10845-022-02016-w
20. 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. http://dx.doi.org/10.1109/CIMCA.2005.1631345

21. S. Rahnamayan, H. R. Tizhoosh, M. M. A. Salama, Opposition-based differential evolution, *IEEE Trans. Evol. Comput.*, **12** (2008), 64–79. http://dx.doi.org/10.1109/TEVC.2007.894200
22. M. Črepinšek, S. H. Liu, M. Mernik, Exploration and exploitation in evolutionary algorithms, *ACM Comput. Surv.*, **45** (2013), 1–33. http://dx.doi.org/10.1145/2480741.2480752
23. H. L. Kwa, J. Philippot, R. Bouffanais, Effect of swarm density on collective tracking performance, *Swarm Intell.*, **17** (2023), 253–281. http://dx.doi.org/10.1007/s11721-023-00225-4
24. P. Jośko, B. M. Ombuki-Berman, A. P. Engelbrecht, Multi-guide particle swarm optimisation archive management strategies for dynamic optimisation problems, *Swarm Intell.*, **16** (2022), 143–168. http://dx.doi.org/10.1007/s11721-022-00210-3
25. F. Yu, J. Guan, H. R. Wu, C. Y. Chen, X. W. Xia, Lens imaging opposition-based learning for differential evolution with cauchy perturbation, *Appl. Soft Comput.*, **152** (2023), 111211. https://doi.org/10.1016/j.asoc.2023.111211
26. S. Mahdavi, S. Rahnamayan, K. Deb, Opposition based learning: A literature review, *Swarm Evol. Comput.*, **39** (2018), 1–23. http://dx.doi.org/10.1016/j.swevo.2017.09.010
27. W. Deng, S. F. Shang, X. Cai, H. M. Zhao, Y. J. Song, J. J. Xu, An improved differential evolution algorithm and its application in optimization problem, *Soft Comput.*, **25** (2021), 5277–5298. http://dx.doi.org/10.1007/s00500-020-05527-x
28. L. L. Kang, R. S. Chen, W. L. Cao, Y. C. Chen, Non-inertial opposition-based particle swarm optimization and its theoretical analysis for deep learning applications, *Appl. Soft Comput.*, **88** (2020), 10. http://dx.doi.org/10.1016/j.asoc.2019.106038
29. S. Dhargupta, M. Ghosh, S. Mirjalili, R. Sarkar, Selective opposition based grey wolf optimization, *Expert Syst. Appl.*, **151** (2020), 13. http://dx.doi.org/10.1016/j.eswa.2020.113389
30. A. Chatterjee, S. Ghoshal, V. Mukherjee, Solution of combined economic and emission dispatch problems of power systems by an opposition-based harmony search algorithm, *Int. J. Electr. Power Energy Syst.*, **39** (2012), 9–20. https://doi.org/10.1016/j.ijepes.2011.12.004
31. B. Kazemi, M. Ahmadi, S. Talebi, Optimum and reliable routing in VANETs: An opposition based ant colony algorithm scheme, in *2013 International Conference on Connected Vehicles and Expo (ICCVE)*, (2013), 926–930.
32. Y. Y. Zhang, Backtracking search algorithm with specular reflection learning for global optimization, *Knowl.-Based Syst.*, **212** (2021), 17. https://doi.org/10.1016/j.knosys.2020.106546
33. R. Patel, M. M. Raghuwanshi, L. G. Malik, Decomposition based multi-objective genetic algorithm (DMOGA) with opposition based learning, in *2012 Fourth International Conference on Computational Intelligence and Communication Networks*, (2012), 605–610. https://doi.org/10.1109/cicn.2012.79
34. M. Tair, N. Bacanin, M. Zivkovic, K. Venkatachalam, A chaotic oppositional whale optimisation algorithm with firefly search for medical diagnostics, *Comput. Mater. Continua*, **72** (2022). https://doi.org/10.32604/cmc.2022.024989

35. L. Abualigah, A. Diabat, M. A. Elaziz, Improved slime mould algorithm by opposition-based learning and Levy flight distribution for global optimization and advances in real-world engineering problems, *J. Ambient Intell. Humanized Comput.*, **14** (2023), 1163–1202. https://doi.org/10.1007/s12652-021-03372-w
36. S. K. Joshi, Chaos embedded opposition based learning for gravitational search algorithm, *Appl. Intell.*, **53** (2023), 5567–5586. https://doi.org/10.1007/s10489-022-03786-9
37. V. H. S. Pham, N. T. N. Dang, V. N. Nguyen, Hybrid sine cosine algorithm with integrated roulette wheel selection and opposition-based learning for engineering optimization problems, *Int. J. Comput. Intell. Syst.*, **16** (2023), 171. https://doi.org/10.1007/s44196-023-00350-2
38. N. Bacanin, U. Arnaut, M. Zivkovic, T. Bezdan, T. A. Rashid, Energy efficient clustering in wireless sensor networks by opposition-based initialization bat algorithm, in *Computer Networks and Inventive Communication Technologies . Lecture Notes on Data Engineering and Communications Technologies* (eds. S. Smys, R. Bestak, R. Palanisamy and I. Kotuliak), (2022), 1–16. https://doi.org/10.1007/978-981-16-3728-5_1
39. T. Bezdan, A. Petrovic, M. Zivkovic, I. Strumberger, V. K. Devi, N. Bacanin, Current best opposition-based learning salp swarm algorithm for global numerical optimization, in *2021 Zooming Innovation in Consumer Technologies Conference (ZINC)*, (2021), 5–10. https://doi.org/10.1109/ZINC52049.2021.9499275
40. S. J. Mousavirad, D. Oliva, S. Hinojosa, G. Schaefer, Differential evolution-based neural network training incorporating a centroid-based strategy and dynamic opposition-based learning, in *2021 IEEE Congress on Evolutionary Computation (CEC)*, (2021), 1233–1240. https://doi.org/10.1109/CEC45853.2021.9504801
41. S. Rahnamayan, H. R. Tizhoosh, M. M.A. Salama, Quasi-oppositional differential evolution, in *2007 IEEE Congress on Evolutionary Computation*, (2007), 2229–2236. https://doi.org/10.1109/CEC.2007.4424748
42. M. Ergezer, D. Simon, D. Du Oppositional biogeography-based optimization, in *2009 IEEE International Conference on Systems, Man and Cybernetics*, (2009), 1009–1014. https://doi.org/10.1109/ICSMC.2009.5346043
43. H. R. Tizhoosh, M. Ventresca, S. Rahnamayan, Opposition-based computing, in *Oppositional Concepts in Computational Intelligence* (eds. H. R. Tizhoosh and M. Ventresca), Springer, (2008), 11–28. https://doi.org/10.1007/978-3-540-70829-2_2
44. H. Wang, Z. Wu, S. Rahnamayan, Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems, *Soft Comput.*, **15** (2010), 2127–2140. http://dx.doi.org/10.1007/s00500-010-0642-7
45. M. Ergezer, D. Simon, Probabilistic properties of fitness-based quasi-reflection in evolutionary algorithms, *Comput. Oper. Res.*, **63** (2015), 114–124. http://https://doi.org/10.1016/j.cor.2015.03.013
46. Z. Hu, Y. Bao, T. Xiong, Partial opposition-based adaptive differential evolution algorithms: Evaluation on the CEC 2014 benchmark set for real-parameter optimization, in *2014 IEEE congress on evolutionary computation (CEC)*, (2014), 2259–2265. http://dx.doi.org/10.1109/CEC.2014.6900489

47. S. Rahnamayan, J. Jesuthasan, F. Bourennani, G. F. Naterer, H. Salehinejad, Centroid opposition-based differential evolution, *Int. J. Appl. Metaheuristic Comput.*, **5** (2014), 1–25. http://dx.doi.org/10.4018/ijamc.2014100101

48. H. Liu, Z. Wu, H. Li, H. Wang, S. Rahnamayan, C. Deng, Rotation-based learning: A novel extension of opposition-based learning, in *PRICAI 2014: Trends in Artificial Intelligence. PRICAI 2014. Lecture Notes in Computer Science* (eds. D. N. Pham and S. B. Park), Springer International Publishing, (2014), 511–522. https://doi.org/10.1007/978-3-319-13560-1_41

49. H. Xu, C. D. Erdbrink, V. V. Krzhizhanovskaya, How to speed up optimization? Opposite-center learning and its application to differential evolution, *Proc. Comput. Sci.*, **51** (2015), 805–814. http://doi.org/10.1016/j.procs.2015.05.203

50. Z. Seif, M. B. Ahmadi, An opposition-based algorithm for function optimization, *Eng. Appl. Artifi. Intell.*, **37** (2015), 293–306. http://dx.doi.org/10.1016/j.engappai.2014.09.009

51. Q. Xu, L. Wang, B. He, N. Wang, Modified opposition-based differential evolution for function optimization, *J. Comput. Inf. Syst.*, **7** (2011), 1582–1591.

52. S. Y. Park, J. J. Lee, Stochastic opposition-based learning using a beta distribution in differential evolution, *IEEE Trans. Cybern.*, **46** (2016), 2184–2194. http://dx.doi.org/10.1109/TCYB.2015.2469722

53. X. Xia, L. Gui, Y. Zhang, X. Xu, F. Yu, H. Wu, et al., A fitness-based adaptive differential evolution algorithm, *Inf. Sci.*, **549** (2021), 116–141. http://dx.doi.org/10.1016/j.ins.2020.11.015

54. H. Deng, L. Peng, H. Zhang, B. Yang, Z. Chen, Ranking-based biased learning swarm optimizer for large-scale optimization, *Inf. Sci.*, **493** (2019), 120–137. http://dx.doi.org/10.1016/j.ins.2019.04.037

55. L. Gui, X. Xia, F. Yu, H. Wu, R. Wu, B. Wei, et al., A multi-role based differential evolution, *Swarm Evol. Comput.*, **50** (2019), 100508. https://doi.org/10.1016/j.swevo.2019.03.003

56. B. Morales-Castañeda, D. Zaldívar, E. Cuevas, F. Fausto, A. Rodríguez, A better balance in metaheuristic algorithms: Does it exist?, *Swarm Evol. Comput.*, **54** (2020), 100671. http://dx.doi.org/10.1016/j.swevo.2020.100671

57. G. Wu, R. Mallipeddi, P. Suganthan, Problem definitions and evaluation criteria for the CEC 2017 competition and special session on constrained single objective real-parameter optimization, *Nanyang Technol. Univ. Singapore Tech. Rep.*, (2016), 1–18.

58. J. Zhang, A. C. Sanderson, JADE: Adaptive differential evolution with optional external archive, *IEEE Trans. Evol. Comput.*, **13** (2009), 945–958. http://dx.doi.org/10.1109/tevc.2009.2014613

59. W. Deng, H. C. Ni, Y. Liu, H. L. Chen, H. M. Zhao, An adaptive differential evolution algorithm based on belief space and generalized opposition-based learning for resource allocation, *Appl. Soft Comput.*, **127** (2022), 20. http://dx.doi.org/10.1016/j.asoc.2022.109419

60. Y. L. Xu, X. F. Yang, Z. L. Yang, X. P. Li, P. Wang, R. Z. Ding, et al., An enhanced differential evolution algorithm with a new oppositional-mutual learning strategy, *Neurocomputing*, **435** (2021), 162–175. http://dx.doi.org/10.1016/j.neucom.2021.01.003

61. J. Li, Y. Gao, K. Wang, Y. Sun, A dual opposition-based learning for differential evolution with protective mechanism for engineering optimization problems, *Appl. Soft Comput.*, **113** (2021), 107942. http://dx.doi.org/10.1016/j.asoc.2021.107942

62. X. C. Zhao, S. Feng, J. L. Hao, X. Q. Zuo, Y. Zhang, Neighborhood opposition-based differential evolution with gaussian perturbation, *Soft Comput.*, **25** (2021), 27–46. http://dx.doi.org/10.1007/s00500-020-05425-2

63. J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, *Swarm Evol. Comput.*, **1** (2011), 3–18. http://dx.doi.org/10.1016/j.swevo.2011.02.002

AIMS Press