symmetry

MDPI

Article

# Multi-Strategy Enhanced Child Drawing Development Optimization Algorithm for Global Optimization Problems and Real Problems

**Zhizi Wei [1,2,†], Sheng Wang [2,3,†], Shaojie Yin [4] and Guanjie Wang [4,*]**

[1] Design College, Jinhua University of Vocational Technology, Jinhua 321000, China; weizhizi@zjnu.edu.cn
[2] College of Education, Zhejiang Normal University, Jinhua 321000, China; jhcwsws@163.com
[3] School of Business, Jinhua University of Vocational Technology, Jinhua 321000, China
[4] Taizhou Institute, Zhejiang University, Taizhou 318000, China; yinshaojie@tzizju.cn
* Correspondence: gj-wang@zju.edu.cn
† These authors contributed equally to this work.

**Abstract**

To address the tendency of the traditional Children's Drawing Development Optimization (CDDO) algorithm to fall into local optima and converge slowly in global optimization and fire-field robot path planning, this study proposes a Multi-Strategy Enhanced Children's Drawing Development Optimization (MECDDO) algorithm. The algorithm achieves performance improvements through three core strategies: (1) an adaptive cooperative search strategy that integrates information from the global best, worst, and random individuals and guides updates via dynamic weighting, expanding the exploration of the solution space; (2) a multi-strategy adaptive selection mechanism that constructs a pool of four differentiated strategies and dynamically adjusts selection probabilities based on strategy success rates, balancing exploration and exploitation; and (3) a global-optimum guided boundary repair strategy that reduces the loss of high-quality information from out-of-bounds solutions, enhancing local exploitation efficiency. Experiments on the CEC2017 benchmark suite demonstrate that MECDDO achieves outstanding performance across 30-, 50-, and 100-dimensional spaces. Statistical significance was evaluated using the Friedman test and Wilcoxon signed-rank test at a 0.05 significance level. The Friedman test mean rankings (M.R.) are 1.63, 2.20, and 2.70, respectively, consistently outperforming traditional CDDO (M.R. = 9.83, 9.93, 9.73, ranked 10th). Applied to mobile robot path planning, MECDDO achieves an average path length of 27.95483 in 20 × 20 grid environments (rank 1), shortening paths by 8.83% compared with CDDO (30.66212, rank 10), and 61.15516 in 40 × 40 grids (rank 1), reducing paths by 37.19% versus CDDO (97.20336, rank 9), providing trajectories free of redundant turns and convergence speeds 2–3 times faster than competing algorithms. These results validate MECDDO's significant advantages in numerical optimization accuracy and practical robot path planning.

**Keywords:** child drawing development optimization; heuristic algorithm; robot path planning; differential evolution operation; global optimization

Check for updates

Academic Editors: Theodore E. Simos and Zhixun Su

Received: 8 January 2026
Revised: 25 February 2026
Accepted: 3 March 2026
Published: 11 March 2026

**Copyright:** © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

## 1. Introduction

Against the backdrop of rapid advancements in artificial intelligence and automation technologies, global optimization algorithms have emerged as core tools for addressing complex engineering problems. They have been widely applied in fields such as mobile

robot path planning [1,2], industrial parameter scheduling [3], and UAV trajectory optimization [4,5]. The central challenge of these problems lies in the fact that their solution spaces are often characterized by high dimensionality, multimodality, and strong constraints. Traditional optimization methods, such as gradient descent, are prone to being influenced by initial values and may become trapped in local optima. Although heuristic algorithms can explore a broader solution space through random search, they commonly suffer from slow convergence and an imbalance between exploration and exploitation [6–8]. This issue is particularly critical in emergency scenarios, such as fire-field robot path planning, where algorithms are required not only to rapidly generate feasible, collision-free, and short trajectories but also to ensure trajectory smoothness to reduce robotic energy consumption and motion errors [9–13]. Such demands place higher requirements on the overall performance of optimization algorithms.

As a typical application scenario of global optimization algorithms, mobile robot path planning directly affects a robot's operational efficiency and safety in complex environments. Current mainstream path planning algorithms can be categorized into traditional methods and intelligent optimization methods. Traditional methods, such as the A* algorithm [14–16] and Dijkstra algorithm, can quickly find feasible paths but exhibit poor adaptability to dynamic or densely obstructed environments and tend to generate redundant turning trajectories [17,18]. In contrast, intelligent optimization methods, such as particle swarm optimization and ant colony optimization, offer stronger environmental adaptability but suffer from slow convergence and insufficient trajectory smoothness [19–22]. Therefore, developing an optimization algorithm that simultaneously achieves high solution accuracy, fast convergence, and robust environmental adaptability is of significant importance for enhancing the performance of mobile robot path planning.

In recent years, numerous swarm intelligence optimization algorithms have been proposed to address various problems [23]. For example, Fu et al. introduced the Secretary Bird Optimization Algorithm (SBOA), inspired by the survival behaviors of secretary birds in nature, and applied it to 3D path planning for UAVs [24]. Tian-Lei Wang et al. developed a novel metaheuristic algorithm, the Cuckoo Catfish Optimizer (CCO), by simulating the searching, predation, and parasitic behaviors observed in cichlids [25]. Dler O. Hasan et al. integrated four major foraging strategies of griffon vultures in nature—following behavior, group foraging, independent search, and carrion inspection—to propose the Griffon Vulture Optimization Algorithm (GVOA) [26]. Inspired by the survival strategies of rabbits, including detour foraging and random hiding, the Artificial Rabbits Optimization (ARO) algorithm was developed [27]. The Chimpanzee Optimization Algorithm (ChOA) simulates four social behaviors of chimpanzees toward prey: attack, driving, interception, and pursuit [28]. Salah Kamel et al. proposed the Super Tornado Algorithm (STA), which draws inspiration from storm strategies such as spiral motion, tornado formation, and rapid currents, and is specifically designed to simulate the behavior of supercell thunderstorms [29]. Additionally, inspired by the Newton–Raphson method, the Newton–Raphson-based Optimizer (NRBO) employs two rules—the Newton–Raphson Search Rule (NRSR) and the Trap Avoidance Operator (TAO)—along with several matrix sets to further explore optimal solutions [30].

In addition, to address the limitations of basic algorithms in path planning, such as slow convergence and suboptimal path quality, Ran Zhang et al. proposed a UAV path planning algorithm based on an improved Harris Hawks Optimization (HHO), incorporating the Cauchy mutation strategy, adaptive weights, and sine–cosine strategies to enhance performance in complex three-dimensional (3D) flight environments [31]. Chao Zhang et al. developed a multi-strategy integrated wind-driven optimization (MEWDO) algorithm for robotic path planning, leveraging cubic spline interpolation, Chebyshev strategies, and

a dimension-wise perturbation mutation approach [32]. Yaonan Dai et al. introduced a novel Whale Optimization Algorithm (NWOA) for mobile robot path planning by employing adaptive techniques and potential field factors [33]. Ruo-Bin Wang et al., aiming to find collision-free optimal motion paths in obstacle-laden environments, proposed an Adaptive Parallel Algorithm Optimization (APAOA) with a novel parallel communication strategy [34]. To overcome major challenges faced by traditional path planning algorithms in large-scale scenarios with high-density irregular obstacles—including low search efficiency, limited obstacle avoidance capability, and a tendency to get trapped in local optima—Yuchao Wang et al. proposed a hybrid path planning algorithm combining an improved Golden Jackal Optimization (MGJO) algorithm with an Improved Dynamic Window Approach (IDWA) [35]. Xuzhen Deng et al. addressed issues such as low population diversity, imbalance between exploration and exploitation, and low optimal solution accuracy in arithmetic optimization algorithms by proposing a Multi-Strategy Enhanced Arithmetic Optimization Algorithm (MSEAOA) for mobile robot path planning [36]. Dong Wu et al. developed a hybrid path planning algorithm that utilizes Adaptive Step Rapidly-exploring Random Tree-Connect (AS-RRT-Connect) to generate initial paths and reduce the search space, thereby providing more accurate pheromone distribution information for an improved Ant Colony Optimization (IACO) algorithm and significantly enhancing convergence speed [37].

The Child Drawing Development Optimization Algorithm (CDDO), a novel metaheuristic algorithm, was proposed by Abdulhameed et al. in 2022 [38]. It draws inspiration from the cognitive development process observed in children's drawing learning and simulates five stages—doodling, development, application of the golden ratio, creativity cultivation, and pattern memorization—to iteratively update solutions. CDDO incorporates the Golden Ratio (GR) and Pattern Memory (PM) mechanisms, demonstrating certain distinct advantages in low-dimensional and simple optimization problems. For instance, in unimodal function optimization, it guides solutions toward high-quality regions through the golden ratio, achieving convergence accuracy superior to some traditional algorithms.

However, as problem complexity increases, inherent limitations of CDDO gradually emerge. Its search mechanism is primarily "enveloping" and overly reliant on the golden ratio for local guidance, lacking effective interaction among multiple individual sources within the population. This restricts exploration of the solution space and makes the algorithm prone to being trapped in local optima for multimodal or high-dimensional problems. Moreover, its fixed update strategy—combining the golden ratio with pattern memory—limits the algorithm's ability to rapidly cover a broad solution space in the early stages, while hindering precise refinement of the optimal solution in later stages, thus disrupting the balance between exploration and exploitation. Additionally, boundary handling relies solely on simple truncation, which discards potentially valuable information in out-of-bound dimensions. This not only reduces population diversity but also significantly increases the risk of premature convergence in high-dimensional or strongly constrained scenarios [39,40]. These limitations considerably restrict the application of CDDO in complex engineering problems, such as robot path planning in high-density obstacle environments.

In this context, this study proposes a Multi-Strategy Enhanced Child Drawing Development Optimization Algorithm (MECDDO), which achieves performance breakthroughs through three core improvement strategies: (1) an adaptive collaborative search strategy that integrates multi-source information from the global best, worst, and random individuals; dynamic weighting coefficients are computed based on Euclidean distances to guide individuals toward high-quality regions and broaden solution space exploration; (2) a multi-strategy adaptive selection mechanism that constructs a strategy pool comprising

four differentiated strategies, including collaborative search and differential evolution, and dynamically adjusts strategy selection probabilities according to their success rates, thereby achieving a dynamic balance between exploration and exploitation throughout the optimization process; (3) a global-best-guided boundary repair strategy that uses the position of the global best individual as a reference to intelligently bring out-of-bound solutions back into the feasible domain, preventing the loss of high-quality information and improving local exploitation efficiency.

Compared with learning-based or neuroevolutionary approaches [41], MECDDO avoids costly model training and large memory requirements, making it more suitable for high-dimensional or resource-constrained optimization tasks.

Although MECDDO follows the general paradigm of multi-strategy hybridization, its novelty lies in three fundamental aspects. First, unlike conventional ensemble or adaptive DE-based optimizers that rely solely on historical success rates, MECDDO introduces an adaptive cooperative search strategy that explicitly integrates global best, worst, and random individuals through distance-based dynamic weighting, enabling structured population-level information exchange. Second, the proposed multi-strategy selection mechanism unifies heterogeneous search behaviors (collaborative search and multiple DE variants) within a single probabilistic framework, allowing the algorithm to adaptively transition between exploration-dominated and exploitation-dominated phases. Third, the global-best-guided boundary repair strategy departs from traditional truncation or random repair schemes by preserving high-quality information near the feasibility boundaries. These mechanisms collectively differentiate MECDDO from existing adaptive DE or ensemble-based optimizers.

To validate the performance of MECDDO, this study conducted multi-dimensional experiments. First, using the CEC2017 and CEC2022 (20-dimensional) benchmark test suite—which includes 30 functions across four categories: unimodal, multimodal, hybrid, and composition—MECDDO was compared with nine mainstream algorithms, including GWO, HHO, WOA, and CDDO, in 30-, 50-, and 100-dimensional spaces. Its numerical optimization capability was evaluated through average fitness values, standard deviations, and the Friedman test. Second, two grid maps with complexities of $20 \times 20$ and $40 \times 40$ were constructed to assess MECDDO's practical applicability in mobile robot path planning, using path length, smoothness, and convergence speed as evaluation metrics. The experimental results indicate that MECDDO significantly outperforms the comparative algorithms in terms of numerical optimization accuracy, stability, and practical path planning performance, providing an efficient solution for complex global optimization problems and engineering applications.

The structure of this paper is organized as follows: Section 2 provides a detailed introduction to the principles of the traditional CDDO algorithm and the three core improvement strategies of MECDDO; Section 3 validates the numerical optimization performance of MECDDO through experiments on the CEC2017 benchmark test suite; Section 4 applies MECDDO to mobile robot path planning and analyzes its performance in different grid environments; Section 5 summarizes the research findings, highlights current limitations, and outlines directions for future work.

## 2. Child Drawing Development Optimization and the Proposed MECDDO

### *2.1. Child Drawing Development Optimization (CDDO)*

The Child Drawing Development Optimization (CDDO) algorithm is a novel metaheuristic inspired by children's learning behaviors and cognitive development, leveraging the Golden Ratio to optimize the aesthetic aspects underlying children's art [38].

(1) First Stage (Doodling Stage)

The initial doodles created by children mainly consist of random marks. At this stage, children are primarily observing and exploring hand movements and pressure. Since children notice that straight motions produce lines while other hand movements generate curves, their actions involve both linear motions and random curved motions. Hand pressure is often inconsistent—either too high or too low—but in subsequent stages, it is refined through repeated attempts and consideration of multiple factors [38,39]:

$$X_{i,j} = lb_j + rand \times \left(ub_j - lb_j\right) \tag{1}$$

In the formula above, $X$ represents the current solution, corresponding to a child's drawing. $lb_j$ and $ub_j$ denote the lower and upper bounds of the $j$th dimension, ensuring that each individual's initial position is randomly distributed within the feasible space. $rand$ is a random number in the range [0,1].

(2) Second Stage (Development Stage)

Hand pressure is one of the factors used to assess a child's performance. Appropriate hand pressure indicates a higher skill level; it also implies that the child has sufficient ability to complete the drawing with minimal pressure while accurately applying the Golden Ratio. First, a random hand pressure (RHP) is generated. RHP is a random number between the problem's lower bound $(lb)$ and upper bound $(ub)$ and is used to evaluate the current solution's hand pressure relative to the actual hand pressure (HP) [38,39].

$$\begin{aligned} RHP &= \mathrm{rand}(lb, ub) \\ HP &= X(i, \mathrm{rand}(j)) \end{aligned} \tag{2}$$

(3) Third Stage (Golden Ratio Stage)

At this stage, children begin applying the skills they have learned. They utilize feedback from past experiences, observe patterns in the actual image, attempt to assign meaning to their drawings, and create artworks through imitation, practice, and active experimentation. To model these behaviors, after evaluating the hand pressure (HP) to assess the child's skill level, the current hand pressure is compared with the random hand pressure (RHP). If the current HP is less than RHP, the solution is updated using Equation (3), while taking into account the child's Skill Rate (SR) and Level Rate (LR). SR and LR are initially random numbers between 0 and 1; if the child's hand pressure is appropriate, they are later adjusted to values between 0.6 and 1. Setting SR and LR to higher values (0.6–1) indicates that the child possesses relatively accurate knowledge and skill, which can be further enhanced by considering the Golden Ratio (GR) factor. Another factor used to update the solution and improve its performance is the Golden Ratio (GR), defined as the ratio between two selected features of the solution (i.e., the length and width of the child's drawing) as shown in Equation (5). Each of these two features is randomly selected from all problem factors using Equation (5) [38,39].

$$X_{i+1} = GR + SR \cdot (X_{ilbest} - X_i) + LR \cdot (X_{igbest} - X_i) \tag{3}$$

$$X_{iGR} = \frac{X_{iL} + X_{iW}}{X_{iL}} \tag{4}$$

$$L, W = \mathrm{rand}(0, j) \tag{5}$$

where $X_{ilbest}$ represents the child's best artwork to date, i.e., the local optimum, while $X_{igbest}$ represents the global optimum observed in the child's environment. The Golden Ratio (GR) is the ratio between the length $L$ and width $W$ of the child's drawing.

(4) Fourth Stage (Creativity Stage)

Each child possesses creativity and skills gained through experience and observation, which are key to enhancing the visual appeal of their artwork. During this stage, children integrate information and tend to adopt solutions that conform to or approximate the Golden Ratio. However, if a solution aligns with the Golden Ratio but performs poorly in practice, it indicates that the child's skill is not yet fully developed and requires further refinement through creativity and application of the Golden Ratio. Children remember the best learning methods and imitate these processes to optimize outcomes. To model this process, the algorithm establishes a Pattern Memory (PM) for each solution, whose size can be adjusted according to the problem. Solutions are randomly selected from PM to update underperforming solutions, which helps accelerate convergence, akin to speeding up a child's learning process. The Creativity Rate (CR) and Pattern Memory (PM) are both incorporated in Equation (6) to update the current solution and guide it toward the optimum. CR is experimentally set to a fixed value of 0.1. The Skill Rate (SR) and Level Rate (LR) are set to lower values (0–0.5), indicating that the child's knowledge and skills are still developing and can be further enhanced through CR and PM [38,39].

$$X_{i+1} = X_{iMP} + CR \cdot (X_{igbest}) \tag{6}$$

(5) Fifth Stage (Pattern Memory Stage)

At this stage, children use their prior knowledge and skills to add details to their drawings, enhancing precision, and compare their work with all previously best artworks. In the algorithm, this behavior is simulated by randomly selecting one solution from the top ten best solutions to update the current solution that is "proportionally accurate but exhibits improper hand pressure," focusing on improving detailed performance [38,39].

This process corresponds to the individual update mechanism of agents in the algorithm: if a better solution exists, the current solution is updated toward the global best solution. Furthermore, whenever a new global best solution is found during iterations, it is simultaneously incorporated into the Pattern Memory (PM).

### *2.2. Multi-Strategy Enhanced Child Drawing Development Optimization (MECDDO)*

#### 2.2.1. Adaptive Collaborative Search Strategy

The core inspiration of the traditional CDDO algorithm comes from children's drawing learning behaviors, relying on the Golden Ratio (GR) and the Pattern Matrix (PM) to guide individual updates. However, it exhibits significant limitations in global exploration. On one hand, its search mechanism is primarily "enveloping" and overly dependent on a single golden ratio for local guidance, lacking effective information exchange among different individuals in the population. This restricts exploration of the solution space and makes it difficult to escape the initial search region. On the other hand, the random search mechanism of CDDO is relatively blind, and individual updates do not sufficiently exploit high-quality information from the population. Consequently, the algorithm performs poorly during the exploration phase and is prone to stagnation in local optima, failing to efficiently approach the global optimum.

To address these issues, this study introduces an adaptive collaborative search strategy, which strengthens global exploration by integrating multi-source information from the population to construct a guided update mechanism.

The core logic of this strategy is as follows: for each individual $X_{\mathrm{i}}$ to be updated, the global best individual $X_{\mathrm{best}}$ and the worst individual $X_{\mathrm{worst}}$ are first determined based on population fitness ranking. Meanwhile, two distinct random individuals $X_{r1}$ and $X_{r2}$, different from the current individual, are selected to form a multi-dimensional information

exchange system. To achieve adaptive guidance, weighting coefficients $\alpha_1$ and $\alpha_2$ are computed based on the Euclidean distances among these individuals [42]:

$$\begin{aligned} \alpha_1 &= \frac{\| X_{\text{best}} - X_{\text{worst}} \|}{\| X_{\text{best}} - X_{\text{worst}} \| + \| X_{r1} - X_{r2} \|} \\ \alpha_2 &= \frac{\| X_{r1} - X_{r2} \|}{\| X_{\text{best}} - X_{\text{worst}} \| + \| X_{r1} - X_{r2} \|} \end{aligned} \tag{7}$$

Here, $\alpha_1$ reflects the contribution of the difference between the global best and worst individuals, while $\alpha_2$ captures the diversity information among the random individuals. The sum of $\alpha_1$ and $\alpha_2$ equals 1, ensuring a reasonable weight distribution. Based on this weighting system, the individual update formula is:

$$X_i^{\text{new}} = X_i + \alpha_1 \cdot (X_{\text{best}} - X_{\text{worst}}) + \alpha_2 \cdot (X_{r1} - X_{r2}) \tag{8}$$

The adaptive weighting mechanism can be interpreted as a multi-criteria decision process, where convergence guidance (global best–worst difference) and diversity preservation (random individual difference) represent two competing criteria. The worst individual provides directional contrast information, enhancing population diversity and preventing premature convergence by encouraging exploration away from poor regions. By dynamically adjusting their weights based on Euclidean distances, MECDDO balances exploitation and exploration in a principled manner.

By integrating multi-source individual information with adaptive weight guidance, this strategy effectively addresses the traditional CDDO's weaknesses in exploration and insufficient information interaction. The dynamic adjustment of weights allows individuals to move toward the global optimum while maintaining population diversity, enhancing both the breadth and efficiency of exploration in the solution space and reducing the risk of local stagnation.

#### 2.2.2. Multi-Strategy Adaptive Selection Mechanism

The traditional CDDO algorithm employs a single update strategy (Golden Ratio guidance + Pattern Matrix update), which suffers from inadequate strategy adaptability. On one hand, a single strategy cannot simultaneously meet the requirements of the entire optimization process: strong exploration is needed in the early stage to cover a broad solution space, but the traditional CDDO's exploratory capability is limited; strong exploitation is required in the later stage to refine the optimal solution, yet its local search ability is weak, resulting in an imbalance between exploration and exploitation. On the other hand, a fixed strategy lacks dynamic adjustment capability and cannot adaptively switch update modes according to the population's evolution state, reducing algorithm robustness and limiting adaptability across different types of optimization problems. To address these issues, this study proposes a multi-strategy adaptive selection mechanism, constructing a multi-dimensional strategy pool and enabling intelligent strategy selection through dynamic probability adjustment [43,44].

The core logic of this mechanism consists of two parts: strategy pool construction and adaptive selection.

(1) Strategy Pool Construction

A strategy pool containing four differentiated update strategies is constructed to cover exploration, exploitation, and balanced needs:

- Strategy 1: Adaptive Collaborative + Standard CDDO Strategy

Combines the previously described multi-source collaborative update with the standard CDDO strategy, focusing on broadening the search range.

- Strategy 2 (DE/rand/1):

Classic differential evolution strategy, constructing a differential vector from three randomly selected distinct individuals to enhance population diversity. The update formula is [43,44]

$$X_i^{\text{new}}(j) = \begin{cases} X_{r1}(j) + F\cdot(X_{r2}(j) - X_{r3}(j)), & if\ rand < CR \\ X_i(j), & otherwise \end{cases} \tag{9}$$

where $X_{r1}$, $X_{r2}$ and $X_{r3}$ are distinct random individuals different from the current individual, $F = 0.8$ is the scaling factor, and $p_{CR} = 0.95$ is the crossover probability.

- Strategy 3 (DE/current-to-best/1):

A differential strategy based on the current individual and the global best, emphasizing local exploitation:

$$X_i^{\text{new}}(j) = \begin{cases} X_i(j) + F\cdot(X_{best}(j) - X_i(j) + X_{r1}(j) - X_{r2}(j)), & if\ rand < p_{CR} \\ X_i(j), & otherwise \end{cases} \tag{10}$$

By incorporating guidance from the global best individual, this strategy accelerates convergence toward high-quality regions.

- Strategy 4 (DE/rand-to-current/1):

A differential strategy balancing exploration and exploitation:

$$X_i^{\text{new}}(j) = X_i(j) + L\cdot(X_{r1}(j) - X_i(j)) + F\cdot(X_{r2}(j) - X_{r3}(j)) \tag{11}$$

where $L = rand$ is a random weight coefficient. By integrating information from random and current individuals, this strategy achieves dynamic balance between exploration and exploitation.

(2) Adaptive Selection Mechanism

In each iteration, the "success count" $c_s$ of each strategy (i.e., the number of times a strategy improves an individual's fitness) is recorded. The strategy selection probability $p_s$ is then calculated as

$$p_s = \frac{c_s}{\sum_{s=1}^{4} c_s} \tag{12}$$

If none of the strategies succeed (i.e., $\frac{c_s}{\sum_{s=1}^{4} c_s} = 0$), equal probabilities are assigned $p_s = \frac{1}{4}$ to avoid stagnation. Individuals select a strategy via a roulette-wheel method: generate a random number $r \in [0, 1]$, and select the $s$th strategy if $\sum_{k=1}^{s-1} p_k < r \leq \sum_{k=1}^{s} p_k$.

This mechanism allows the algorithm to dynamically adjust strategy preference according to the optimization process: in early stages, exploration-oriented strategies (Strategy 1 and 2) have higher success probabilities, ensuring broad solution space coverage; in later stages, exploitation-oriented strategies (Strategy 3 and 4) gain advantage, accelerating refinement of the optimal solution. Through this adaptive adjustment, the mechanism effectively addresses the limitations of the traditional single-strategy CDDO, significantly improving robustness and adaptability across different optimization scenarios.

### 2.2.3. Global-Best Guided Boundary Repair Strategy

In the traditional CDDO algorithm, out-of-bound solutions are handled using a simple "truncation method," which directly resets the out-of-bound dimension values to the corresponding boundary via max/min functions. This approach has significant limitations. On one hand, truncation discards potentially valuable information contained in the out-of-bound solutions, sharply reducing solution diversity. This is especially problematic in

high-dimensional optimization, where it can cause the population to become trapped in local optima, increasing the risk of premature convergence. On the other hand, the single truncation mechanism lacks guidance toward high-quality regions of the solution space; repaired solutions tend to cluster near the boundary edges and fail to approach the core regions where the global optimum resides, resulting in low local exploitation efficiency and insufficient utilization of high-quality boundary-adjacent solutions [45,46].

To address these issues, this study proposes a Global-Best Guided Boundary Repair Strategy, which leverages the position of the global-best individual to intelligently guide out-of-bound solutions back into the feasible domain, ensuring both solution validity and local exploitation efficiency.

The core logic is as follows: for the $j$th dimension $X_i(j)$ of individual $X_i$, first check whether it exceeds the feasible domain (i.e., $X_i(j) < lb(j)$ or $X_i(j) > ub(j)$, where $lb(j)$ and $ub(j)$ are the lower and upper bounds for the $j$th dimension). If an out-of-bound case exists, the $j$th dimension of the global-best individual $X_{\text{best}}$ is used as a reference. A random adjustment factor guides the out-of-bound solution toward a high-quality region within the feasible domain, ensuring that the repaired solution remains feasible while approaching the region near the global optimum. The specific repair formula is [47]

$$x_i(j) = \begin{cases} X_{\text{best}}(j) + 0.5 \cdot rand \cdot (ub(j) - X_{\text{best}}(j)), & if\ X_i(j) > ub(j) \\ X_{\text{best}}(j) - 0.5 \cdot rand \cdot (X_{\text{best}}(j) - lb(j)), & if\ X_i(j) < lb(j) \end{cases} \tag{13}$$

Compared with the traditional truncation method, this strategy has two key advantages: (1) by leveraging the global-best solution as guidance, it avoids blind discarding of out-of-bound information, giving repaired solutions greater optimization potential; (2) repaired solutions are distributed near high-quality regions, significantly improving local exploitation efficiency while maintaining population diversity through random adjustment factors. This improvement effectively addresses the rough boundary handling and low exploitation efficiency of the traditional CDDO, and in high-dimensional optimization problems, it can significantly enhance convergence accuracy and stability.

The MECDDO's pseudocode is provided in Algorithm 1, and a flowchart illustrating the overall MECDDO framework and its key innovations is shown in Figure 1.

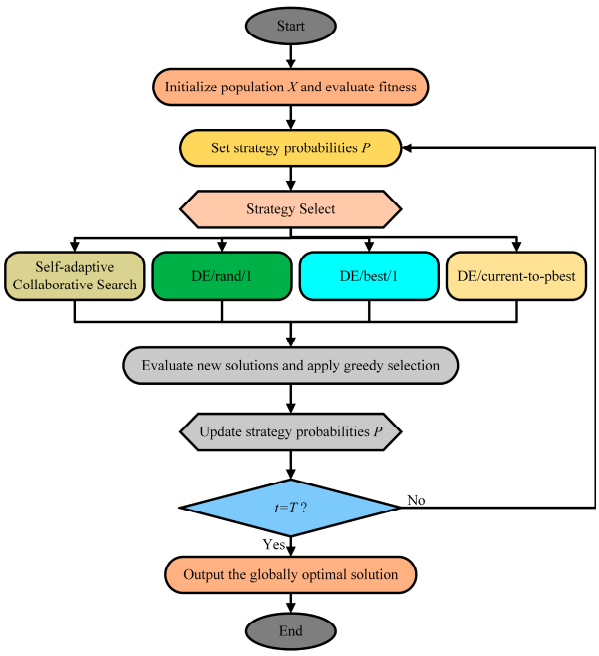


**Figure 1.** The flowchart illustrating the overall MECDDO framework.

**Algorithm 1:** the pseudo-code of the MECDDO

**Input**: *N (population size), T (maximum iterations), dim (problem dimensions), lb (lower bound), ub (upper bound), F (scaling factor = 0.8),* $p_{CR}$ *(crossover probability = 0.95).*
***Output***: $X_{best}$ *(global optimal solution).*
***Begin***
***Initialize:*** *the relevant parameters N, T, dim and initialize population X by Equation (1).*
*Compute fitness, set personal best* $X_{ilbest}$ *and global best* $X_{igbest}$.
*Calculate Golden Ratio Equations (3)–(5).*
*Initialize strategy pool parameters:* $F = 0.8$, $p_{CR} = 0.95$, $\mathbf{p} = [0.25, 0.25, 0.25, 0.25]$.
***while*** $t < T$ ***do***
  *Calculate random hand pressure RHP by Equation (2).*
  ***for*** $i = 1{:}Np$ ***do***
    *Select current strategy* via *roulette wheel and record pre-update state* $X_{\text{old}}$ *and* $fitness_{\text{old}}$.
    ***Switch strategy***
      ***Case 1: Self-adaptive Collaborative Search Strategy***
        *Determine* $X_{\text{best}}$, $X_{\text{worst}}$, *randomly select* $X_{r1}$ *and* $X_{r2}$
        *Calculate weights* $\alpha_1$ *and* $\alpha_2$ *by Equation (7).*
        *Update* $X_i^{\text{new}}$ *by Equation (8).*
        *Perform Global Best-guided Boundary Repair by Equation (13).*
      ***Case 2: DE/rand/1 Differential Strategy***
        *Update* $X_i^{\text{new}}$ *by Equation (9).*
        *Perform Global Best-guided Boundary Repair by Equation (13).*
      ***Case 3: DE/current-to-best/1 Differential Strategy***
        *Update* $X_i^{\text{new}}$ *by Equation (10).*
        *Perform Global Best-guided Boundary Repair by Equation (13).*
      ***Case 4: DE/rand-to-current/1 Differential Strategy***
        *Update* $X_i^{\text{new}}$ *by Equation (11).*
        *Perform Global Best-guided Boundary Repair by Equation (13).*
    ***end Switch***
    ***if*** $fitness_{\text{new}} < fitness_{\text{old}}$
      *Update* $X_{ilbest}$, $X_{igbest}$ *and strategy success count* $c_{strategy}$.
    ***else***
      *restore the original state* $\mathbf{p} = [0.25,\ 0.25,\ 0.25,\ 0.25]$.
    ***end if***
  ***end for***
  *Update strategy probabilities adaptively by Equation (12).*
  $t = t + 1$
***end while***
***return*** *the best solution* $X_{best}$.
***End***

### 2.3. Computational Complexity Analysis

The computational complexity of the proposed algorithm (MECDDO) is analyzed in terms of the population size $N$, the maximum number of iterations $T$, and the problem dimensionality $dim$. Similar to the original CDDO, the computational cost of the initialization phase is $O(N \times dim)$. During the iterative process, the adaptive cooperative search strategy requires the computation of Euclidean distances among four individuals, resulting in a time complexity of $O(N \times dim)$. The multi-strategy adaptive selection mechanism consists of strategy execution with a complexity of $O(N \times 4 \times dim)$ and the statistics of

successful strategies with a complexity of $O(4)$; when combined, its overall complexity remains $O(N \times dim)$.

The global-best-guided boundary repair strategy checks and corrects each dimension of every individual, and thus also incurs a computational cost of $O(N \times dim)$. Consequently, the overall time complexity of MECDDO is $O(T \times N \times dim)$, which is consistent with that of the original CDDO and other mainstream meta-heuristic optimization algorithms in terms of asymptotic complexity.

Although additional operations such as distance calculation, strategy success counting, and roulette-wheel selection are introduced, leading to a slight increase in the constant factor compared with the original CDDO, these operations incur only limited constant overhead and do not alter the algorithm's asymptotic complexity. This demonstrates that the proposed enhancement strategies effectively improve the optimization performance while maintaining computational efficiency.

## 3. Numerical Experiments

### *3.1. Competitor Algorithms and Parameters Setting*

In this subsection, the proposed MECDDO algorithm is evaluated using the currently most challenging numerical optimization benchmark, CEC2017 [48], and its performance is compared with several advanced optimization methods. The comparative algorithms are selected based on three core criteria: (1) representativeness of mainstream metaheuristic categories (swarm intelligence, evolutionary computation, etc.), including Grey Wolf Optimizer (GWO) [49], Harris Hawks Optimization(HHO) [50], Whale Optimization Algorithm (WOA) [51]; (2) recency and state-of-the-art performance, such as Dung beetle optimization algorithm (DBO) [52], Birds of Prey-Based Optimization (BPBO) [53], Artificial Lemming Algorithm(ALA) [54], Snow Geese Algorithm (SGA) [55], Hannibal Barca optimizer (HBO) [56], and the original Child Drawing Development Optimization (CDDO) [38]. The parameter settings for these algorithms are summarized in Table 1.

**Table 1.** Parameter settings of the compared algorithms.

| Algorithms | Name of the Parameter | Value of the Parameter |
|---|---|---|
| GWO | $a$ | [0,2] |
| HHO | $E0, E1$ | $[-1,1], [0,2]$ |
| WOA | $r, l, a$ | [0,1], [−1,1], Linear reduction from 2 to 1 |
| DBO | $p$ | 0.2 |
| BPBO | $P_i$ | 0.7 |
| ALA | $Prob, CF, I$ | 0.3, $[0,2], \{1,2\}$ |
| SGA | $\theta, b, d, e$ | $[0,2\pi], [-2,2], [-1.5,1.5], [-1,1]$ |
| HBO | $Coef$ | 0.94 |
| CDDO | $CR$ | $0.1, 0.15, 0.35, 0.5, 5$ |
| MECDDO | $CR, p_{CR}, F, p_s$ | $0.1, 0.95, 0.8,\ 0.25$ |

All comparative algorithms are implemented in MATLAB 2022b and run on the same hardware platform to ensure fair comparison. Each algorithm is independently executed 30 times for each function (consistent with the sample size requirement for non-parametric statistical tests), and the average fitness, standard deviation, and runtime are recorded. The source codes of the comparative algorithms are obtained from the original authors' official repositories or standardized implementations in the MATLAB Optimization Toolbox, ensuring consistency with the original papers' parameter settings (Table 1).

### 3.2. Design and Result Analysis of Ablation Experiments

To evaluate the performance enhancement of the CDDO algorithm brought by the proposed improvement strategies—Adaptive Collaborative Search Strategy, Multi-Strategy Adaptive Selection Mechanism, and Global Best-Guided Boundary Repair Strategy—this study conducts an ablation experiment. By comparing the standard CDDO algorithm with its enhanced variants, each incorporating only one individual strategy (denoted as CDDO-S1, CDDO-S2, and CDDO-S3), the independent contribution of each strategy is assessed, along with the effect of the fully integrated MECDDO. The evaluation is performed using the CEC 2017 benchmark suite (dim = 30), with all experimental parameter settings consistent with those described earlier.

The convergence curves in Figure 2 demonstrate the optimization behavior of CDDO and its enhanced variants (CDDO-S1, CDDO-S2, CDDO-S3, and MECDDO) on the CEC2017 benchmark set. As shown in the figure, the standard CDDO exhibits a significantly slower convergence rate and becomes trapped in suboptimal regions at early iterations, indicating insufficient global exploration capability. In contrast, all three single-strategy variants achieve noticeable acceleration in convergence, confirming the independent effectiveness of each improvement mechanism. Among them, CDDO-S1 shows superior early-stage descent due to strengthened global guidance, while CDDO-S3 provides more pronounced late-stage refinement benefiting from enhanced local exploitation. The full version, MECDDO, consistently delivers the fastest convergence and lowest fitness curve throughout the entire iteration range, suggesting that combining multiple strategies enables the optimizer to maintain strong exploration in early iterations and transition smoothly into fine exploitation later. This validates that the synergy between adaptive collaborative search, multi-strategy switching, and global best-guided boundary repair greatly improves convergence efficiency and solution quality.

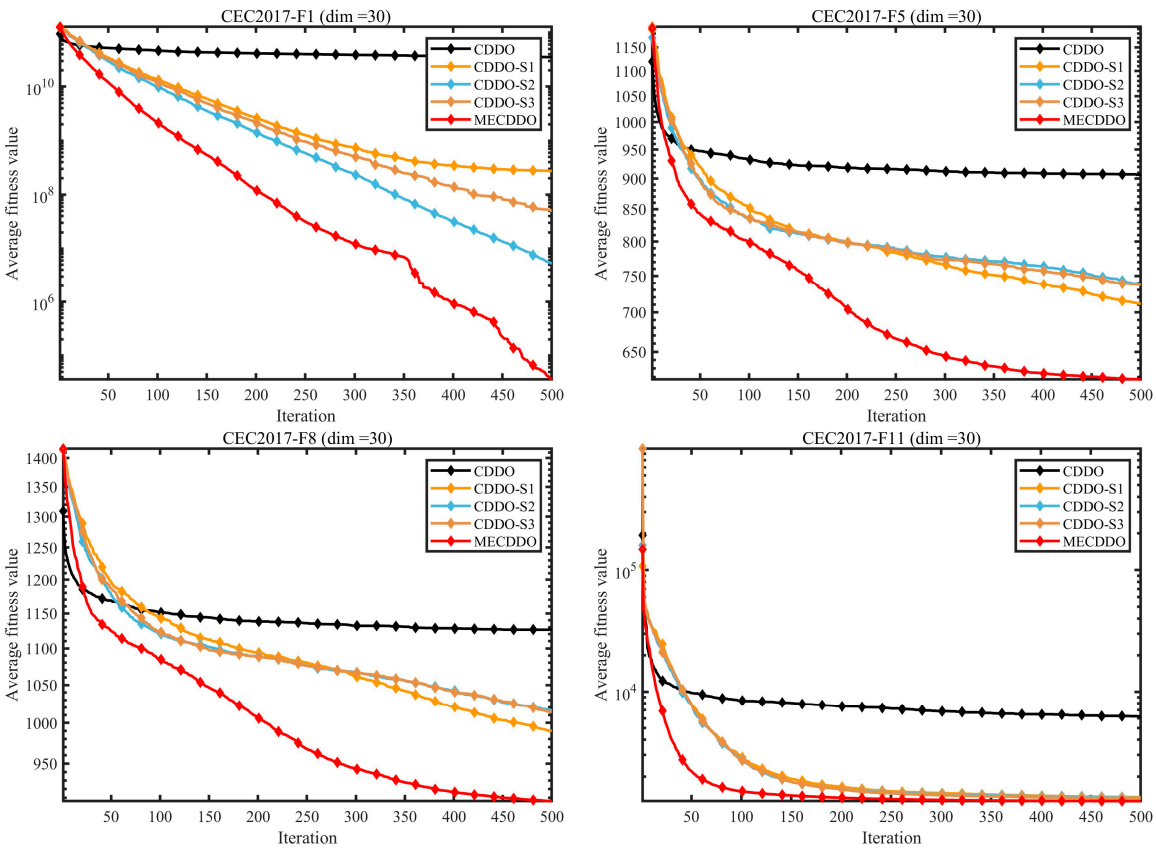


**Figure 2.** *Cont.*

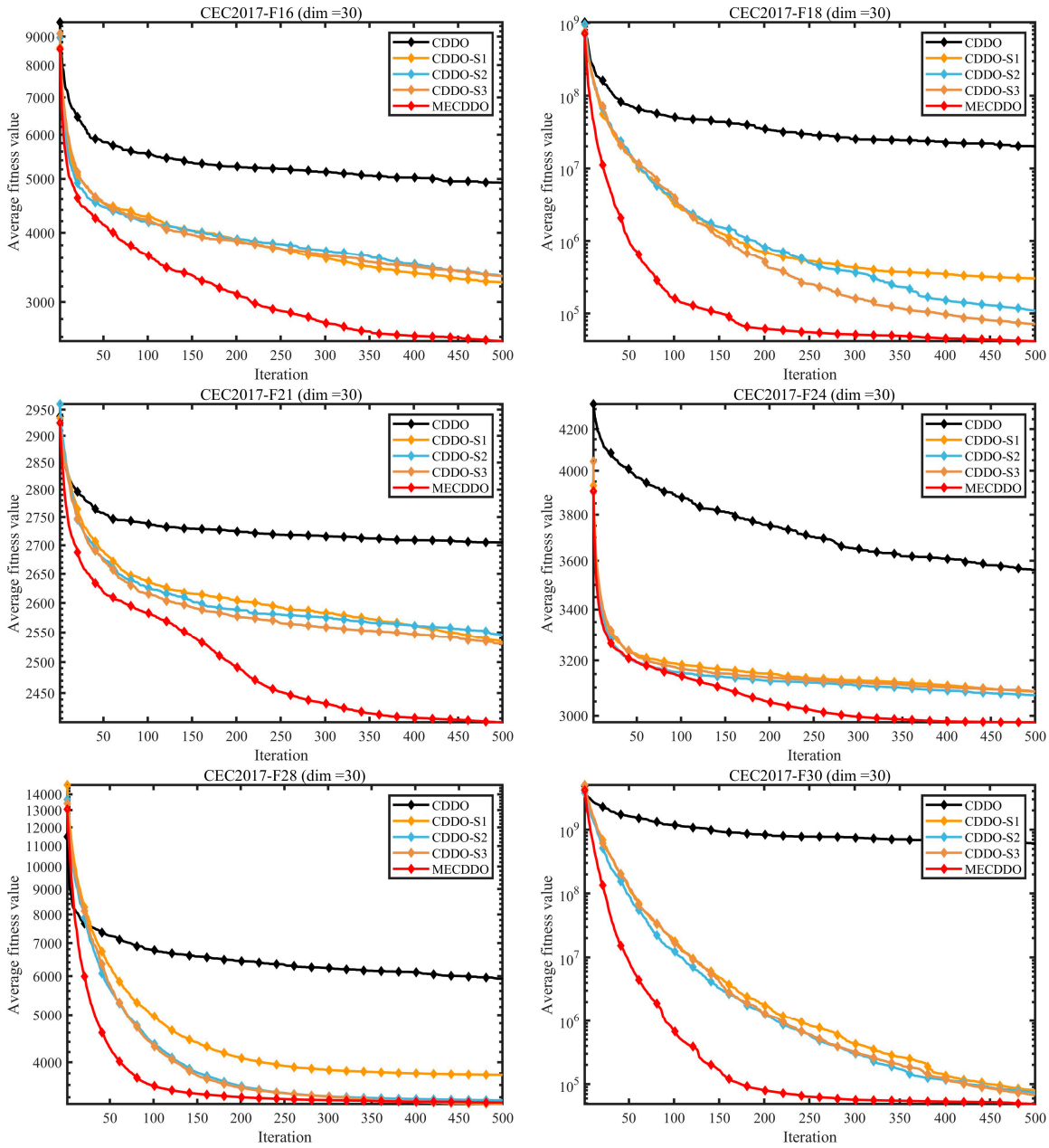


**Figure 2.** Convergence behavior of enhanced CDDO variants using distinct improvement strategies.

Figure 3 presents the average ranking performance of the five algorithms across all benchmark functions. As shown in the figure, MECDDO achieves the best overall rank with an average score of 1.10, indicating that it consistently outperforms all other competitors. The single-strategy variants show noticeable but limited improvements: CDDO-S1 obtains an average rank of 3.23, CDDO-S2 scores 2.87, and CDDO-S3 ranks at 2.90. Although these values are substantially better than the standard CDDO (4.90), they remain inferior to the fully enhanced MECDDO (1.10), demonstrating that single-strategy enhancement can improve performance but is insufficient to fully address the limitations of the original optimizer.

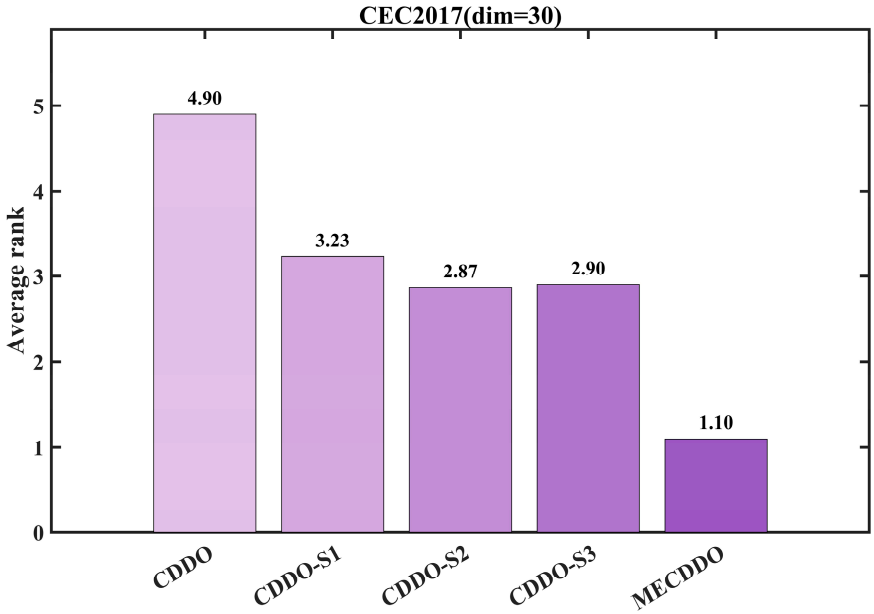


**Figure 3.** Comparative performance ranking of enhanced CDDO implementations across various improvement strategies.

In contrast, the standard CDDO obtains the worst average ranking of 4.90, confirming its weaker optimization capability and higher likelihood of premature convergence on complex benchmark functions. The clear performance gap between MECDDO and the other four implementations validates the effectiveness of the multi-strategy fusion framework, which enables MECDDO to maintain stronger global exploration in early iterations while achieving more refined exploitation in the later search stage. Therefore, the results in Figure 3 demonstrate that MECDDO delivers the most balanced and robust optimization performance among all tested algorithms.

### *3.3. Parameter Sensitivity Analysis*

The key fixed parameters of MECDDO include the scaling factor $F = 0.8$ and the crossover probability $p_{CR} = 0.95$. To conduct sensitivity analysis, each parameter was independently adjusted within a reasonable range ($F : 0.2 - 1, p_{CR} : 0.8 - 1$), while keeping the other parameters unchanged. Figure 4 illustrates the variations in the average ranking of MECDDO on the CEC2017 (30-dimensional) benchmark suite.

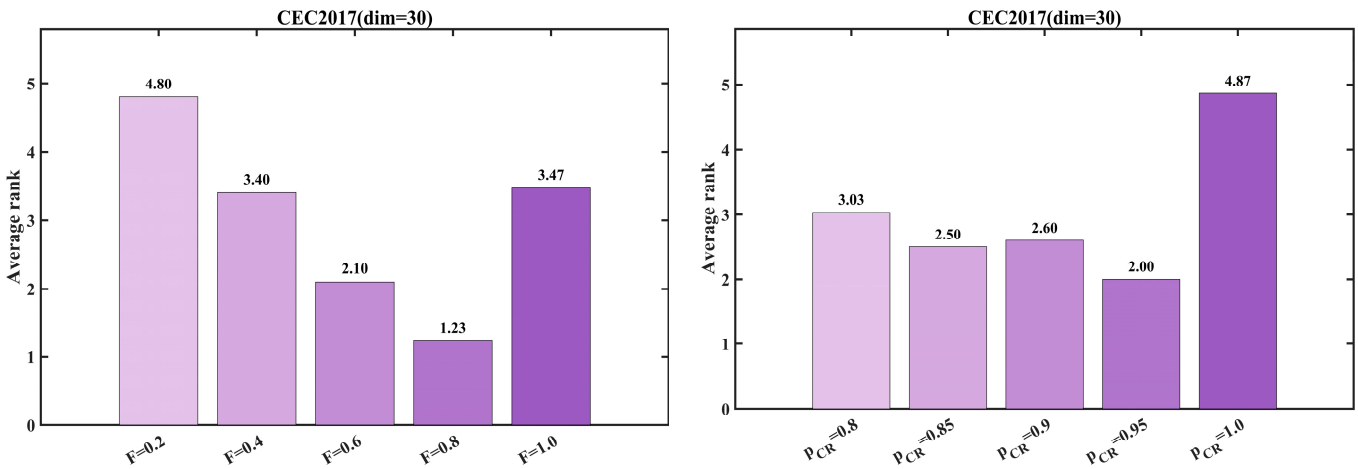


**Figure 4.** Average ranking values of the MECDDO algorithm under different parameter values.

The scaling factor FFF directly affects the magnitude of the mutation vectors in differential evolution strategies and plays a critical role in balancing exploration and exploitation. As shown in Figure 3 (left), MECDDO achieves the best average ranking of 1.23 when F = 0.8, which is significantly superior to other settings. When $F = 0.6$, the average ranking is 2.10, yielding the second-best performance. In contrast, the average rankings decrease to 3.40 and 3.47 when $F = 0.4$ and $F = 1.0$, respectively, indicating a clear performance degradation. The worst performance is observed at $F = 0.2$, with an average ranking of 4.80. These results suggest that excessively small values of $F$ (e.g., 0.2 and 0.4) lead to insufficient mutation strength, making it difficult for the algorithm to escape local optima and thereby limiting its exploration capability. Conversely, an overly large value of $F$ (e.g., 1.0) results in overly aggressive mutations, which may disrupt high-quality solutions within the population and reduce exploitation efficiency. The setting $F = 0.8$ achieves an optimal balance between mutation amplitude and solution stability, maintaining population diversity for global exploration while effectively exploiting high-quality solution information for local refinement, thus yielding the best overall performance.

The crossover probability $p_{CR}$ determines the probability of updating individual dimensions and influences the search precision in the solution space. As depicted in Figure 3 (right), MECDDO attains the best average ranking (2.00) at $p_{CR} = 0.95$, which represents the optimal configuration. When $p_{CR} = 0.90$ and $p_{CR} = 0.85$, the average rankings are 2.60 and 2.50, respectively, indicating performance close to the optimum. However, the ranking deteriorates to 3.03 at $p_{CR} = 0.80$, while the worst performance is observed at $p_{CR} = 1.0$, with an average ranking of 4.87. This trend indicates that excessively high or low crossover probabilities can adversely affect algorithm performance. Specifically, when $p_{CR} = 1.0$, all dimensions are forced to undergo crossover, which may lead to the loss of critical information from high-quality solutions and thus degrade solution quality. In contrast, when $p_{CR} = 0.80$, the crossover probability is insufficient, resulting in low update efficiency and difficulty in rapidly converging to promising regions. The setting $p_{CR} = 0.95$ provides an appropriate crossover intensity that effectively integrates useful information from different individuals while preserving the core characteristics of high-quality solutions, enabling a dynamic balance between exploring new regions and reinforcing superior solutions, and thereby achieving optimal stability and optimization accuracy.

In summary, the parameter sensitivity analysis demonstrates that the core parameters of MECDDO ($F = 0.8$ and $p_{CR} = 0.95$) are both reasonable and robust. Within the examined ranges, the impact of parameter variations on algorithm performance is consistent with the general parameter tuning principles of differential evolution strategies. Moreover, the optimal parameter combination effectively supports the cooperative search behavior and dynamic strategy selection mechanisms of MECDDO, providing a reliable basis for parameter configuration in subsequent numerical experiments and practical engineering applications.

### *3.4. Population Diversity Analysis*

Population diversity is a core factor influencing the global exploration capability and local exploitation efficiency of heuristic algorithms. It directly determines whether an algorithm can effectively escape local optima and steadily approach the global optimum. Therefore, conducting comparative experiments on the population diversity of MECDDO and the conventional CDDO can provide in-depth insights into the optimization mechanism of search behavior induced by the multi-strategy enhancement framework.

Figure 5 compares the population diversity distribution characteristics of MECDDO and the traditional CDDO on the CEC2017 benchmark functions, intuitively illustrating the optimization effect of the multi-strategy enhancement mechanism on population dynam-

ics. As observed from the diversity curves across multiple test functions, the population diversity of conventional CDDO decreases rapidly in the early iterations and remains at a relatively low level in the later stages. This phenomenon reflects its inherent deficiency of premature individual convergence and insufficient information exchange during the search process, which constitutes the primary reason for its tendency to become trapped in local optima. In contrast, MECDDO consistently maintains a higher level of population diversity throughout the entire iterative process, with a noticeably slower decay rate. This advantage is particularly pronounced in high-dimensional and complex functions (e.g., F19 and F28), indicating that the adaptive cooperative search strategy effectively preserves population heterogeneity by integrating multi-source information from the global best, global worst, and randomly selected individuals, thereby ensuring sustained exploration of the solution space.

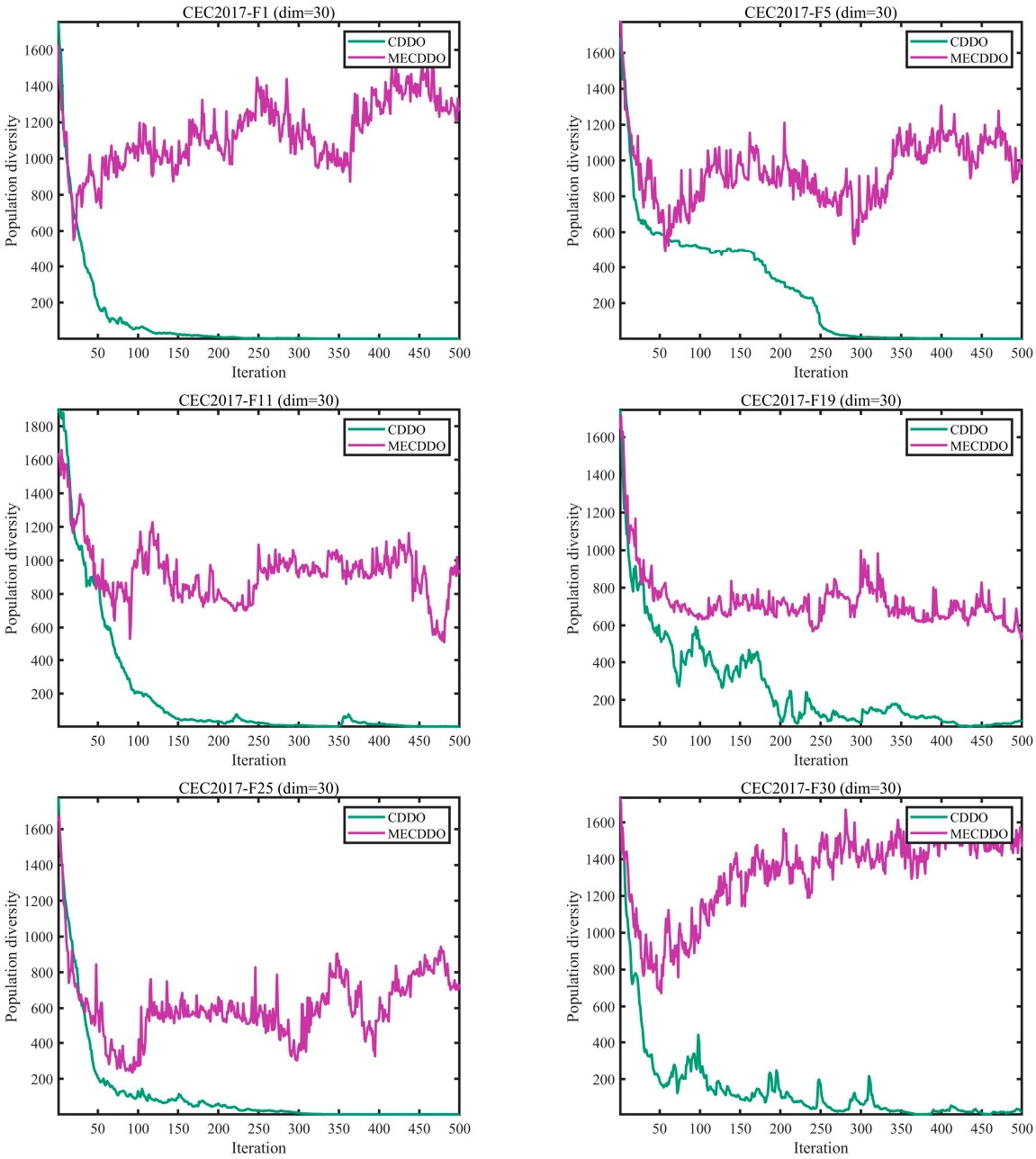


**Figure 5.** Comparative analysis of swarm diversity distribution between MECDDO and CDDO methodologies.

Further analysis reveals that MECDDO enables rapid population dispersion in the early iterations, guaranteeing extensive coverage of the solution space, while gradually converging in the later stages without sacrificing an appropriate level of diversity, thus achieving a dynamic balance between exploration and exploitation. This behavior is closely associated with the regulatory role of the multi-strategy adaptive selection mechanism. Specifically, exploration-oriented strategies (such as adaptive cooperative search and DE/rand/1) are assigned higher selection probabilities in the early phase to promote population dispersion, whereas exploitation-oriented strategies (such as DE/current-to-best/1) dominate in the later phase, guiding the population toward high-quality regions while effectively preventing premature homogenization. By comparison, due to the lack of flexibility inherent in its single update strategy, CDDO either rapidly exhausts population diversity—leading to search stagnation—or remains overly dispersed, making it difficult to converge to the optimal solution. In contrast, MECDDO achieves precise regulation of population diversity, thereby establishing a solid population foundation for the simultaneous improvement of optimization accuracy and convergence speed.

### *3.5. Experimental Results and Analysis on CEC 2017 Benchmark*

This section presents a comparative performance evaluation between the MECDDO method and mainstream benchmark algorithms using the CEC 2017 test suite. The benchmark comprises four distinct categories of mathematical functions: unimodal, multimodal, hybrid, and composition functions. Multimodal functions, characterized by multiple local optima, are primarily employed to assess an algorithm's exploration capability. In contrast, unimodal functions possess only one global optimum and are mainly used to evaluate the effectiveness of its exploitation mechanism. Hybrid and composition functions feature complex landscapes designed to test an optimizer's ability to escape deceptive local solutions.

To ensure the fairness of the experimental process and minimize the interference of random factors, all algorithms involved in the comparison adopt a consistent parameter configuration: the population size is fixed at 30, and the maximum number of iterations is set to 500. Each algorithm is independently executed 30 times in repeated experiments, with the average value (Ave) and standard deviation (Std) of the results recorded, and the optimal values are highlighted in bold. All experiments are conducted on a unified computing platform with the following environmental configuration: Windows 11 operating system, AMD Ryzen 7 7700X 4.5GHz processor, 32GB DDR5-6000 memory, NVIDIA GeForce RTX 4070 Ti graphics card, Western Digital SN850X 2TB NVMe solid-state drive, and MATLAB 2022b as the programming environment. Tables 2–4 present the numerical optimization results of MECDDO and nine comparative algorithms on the CEC 2017 test suite in 30-, 50-, and 100-dimensional search spaces, with convergence curves and box plots shown in Figures 3 and 4, respectively. In the tables, bold values are used to indicate the best-performing result in that row or column.

**Table 2.** Experimental results of CEC2017 (dim = 30).

| Function | Metric | GWO | HHO | WOA | DBO | BPBO | ALA | SGA | HBO | CDDO | MECDDO |
|---|---|---|---|---|---|---|---|---|---|---|---|
| F1 | Ave | $2.6605 \times 10^{9}$ | $4.5442 \times 10^{8}$ | $5.5160 \times 10^{9}$ | $2.3155 \times 10^{8}$ | $2.6293 \times 10^{7}$ | $2.8929 \times 10^{6}$ | $3.5389 \times 10^{9}$ | $1.4160 \times 10^{9}$ | $3.6030 \times 10^{10}$ | $\mathbf{7.6981 \times 10^{3}}$ |
| | Std | $1.8530 \times 10^{9}$ | $3.3645 \times 10^{8}$ | $1.6939 \times 10^{9}$ | $1.3982 \times 10^{8}$ | $1.7398 \times 10^{7}$ | $2.4309 \times 10^{6}$ | $2.5361 \times 10^{9}$ | $5.0193 \times 10^{8}$ | $3.4076 \times 10^{9}$ | $\mathbf{8.4717 \times 10^{3}}$ |
| F2 | Ave | $2.9978 \times 10^{33}$ | $7.2630 \times 10^{36}$ | $7.3693 \times 10^{34}$ | $7.6477 \times 10^{32}$ | $3.5479 \times 10^{23}$ | $1.4607 \times 10^{26}$ | $9.5286 \times 10^{32}$ | $4.6855 \times 10^{29}$ | $2.3889 \times 10^{41}$ | $\mathbf{2.6607 \times 10^{19}}$ |
| | Std | $1.1363 \times 10^{34}$ | $3.9405 \times 10^{37}$ | $1.6929 \times 10^{35}$ | $2.7364 \times 10^{33}$ | $1.5272 \times 10^{24}$ | $7.4850 \times 10^{26}$ | $4.9378 \times 10^{33}$ | $1.4769 \times 10^{30}$ | $5.8129 \times 10^{41}$ | $\mathbf{1.4572 \times 10^{20}}$ |
| F3 | Ave | $6.5135 \times 10^{4}$ | $5.7966 \times 10^{4}$ | $2.4055 \times 10^{5}$ | $9.7834 \times 10^{4}$ | $5.4688 \times 10^{4}$ | $3.4402 \times 10^{4}$ | $5.9426 \times 10^{4}$ | $5.9451 \times 10^{4}$ | $8.3081 \times 10^{4}$ | $\mathbf{1.2314 \times 10^{4}}$ |
| | Std | $1.3875 \times 10^{4}$ | $\mathbf{4.6516 \times 10^{3}}$ | $7.4739 \times 10^{4}$ | $3.2352 \times 10^{4}$ | $7.1831 \times 10^{3}$ | $9.4724 \times 10^{3}$ | $1.0267 \times 10^{4}$ | $1.3483 \times 10^{4}$ | $6.7156 \times 10^{3}$ | $5.9773 \times 10^{3}$ |
| F4 | Ave | $6.3954 \times 10^{2}$ | $7.4804 \times 10^{2}$ | $1.2910 \times 10^{3}$ | $6.9685 \times 10^{2}$ | $5.3247 \times 10^{2}$ | $5.2366 \times 10^{2}$ | $9.3682 \times 10^{2}$ | $6.8509 \times 10^{2}$ | $9.3462 \times 10^{3}$ | $\mathbf{4.8724 \times 10^{2}}$ |
| | Std | $7.9011 \times 10^{1}$ | $1.1105 \times 10^{2}$ | $4.0263 \times 10^{2}$ | $1.7203 \times 10^{2}$ | $2.6838 \times 10^{1}$ | $2.2960 \times 10^{1}$ | $3.1082 \times 10^{2}$ | $9.3350 \times 10^{1}$ | $1.8116 \times 10^{3}$ | $\mathbf{1.9335 \times 10^{1}}$ |
| F5 | Ave | $6.2434 \times 10^{2}$ | $7.7767 \times 10^{2}$ | $8.5151 \times 10^{2}$ | $7.5196 \times 10^{2}$ | $7.2143 \times 10^{2}$ | $\mathbf{6.0373 \times 10^{2}}$ | $7.7521 \times 10^{2}$ | $6.9042 \times 10^{2}$ | $9.0472 \times 10^{2}$ | $6.1521 \times 10^{2}$ |
| | Std | $3.7939 \times 10^{1}$ | $2.7655 \times 10^{1}$ | $4.2748 \times 10^{1}$ | $6.0640 \times 10^{1}$ | $3.6969 \times 10^{1}$ | $2.6748 \times 10^{1}$ | $3.8282 \times 10^{1}$ | $4.5833 \times 10^{1}$ | $\mathbf{2.2550 \times 10^{1}}$ | $6.4372 \times 10^{1}$ |
| F6 | Ave | $6.1267 \times 10^{2}$ | $6.6817 \times 10^{2}$ | $6.8336 \times 10^{2}$ | $6.4601 \times 10^{2}$ | $6.5511 \times 10^{2}$ | $6.1267 \times 10^{2}$ | $6.6958 \times 10^{2}$ | $6.3095 \times 10^{2}$ | $6.9057 \times 10^{2}$ | $\mathbf{6.0256 \times 10^{2}}$ |
| | Std | $4.6182 \times 10^{0}$ | $6.8654 \times 10^{0}$ | $1.4215 \times 10^{1}$ | $8.8558 \times 10^{0}$ | $1.0232 \times 10^{1}$ | $4.4035 \times 10^{0}$ | $8.8278 \times 10^{0}$ | $7.7193 \times 10^{0}$ | $4.5472 \times 10^{0}$ | $\mathbf{2.6282 \times 10^{0}}$ |

**Table 2.** *Cont.*

| Function | Metric | GWO | HHO | WOA | DBO | BPBO | ALA | SGA | HBO | CDDO | MECDDO |
|---|---|---|---|---|---|---|---|---|---|---|---|
| F7 | Ave | $9.1701 \times 10^{2}$ | $1.2979 \times 10^{3}$ | $1.3190 \times 10^{3}$ | $1.0327 \times 10^{3}$ | $1.1440 \times 10^{3}$ | $\mathbf{9.0065 \times 10^{2}}$ | $1.2385 \times 10^{3}$ | $1.0185 \times 10^{3}$ | $1.3527 \times 10^{3}$ | $9.3934 \times 10^{2}$ |
| | Std | $4.3581 \times 10^{1}$ | $6.9105 \times 10^{1}$ | $7.4748 \times 10^{1}$ | $6.6987 \times 10^{1}$ | $8.3434 \times 10^{1}$ | $4.5967 \times 10^{1}$ | $9.1196 \times 10^{1}$ | $6.2757 \times 10^{1}$ | $\mathbf{4.2379 \times 10^{1}}$ | $6.7150 \times 10^{1}$ |
| F8 | Ave | $\mathbf{9.0810 \times 10^{2}}$ | $9.8960 \times 10^{2}$ | $1.0627 \times 10^{3}$ | $1.0165 \times 10^{3}$ | $9.5742 \times 10^{2}$ | $9.0823 \times 10^{2}$ | $1.0173 \times 10^{3}$ | $9.6226 \times 10^{2}$ | $1.1273 \times 10^{3}$ | $9.2843 \times 10^{2}$ |
| | Std | $2.5848 \times 10^{1}$ | $2.0597 \times 10^{1}$ | $5.1541 \times 10^{1}$ | $5.2386 \times 10^{1}$ | $2.5186 \times 10^{1}$ | $2.3901 \times 10^{1}$ | $3.6235 \times 10^{1}$ | $2.9573 \times 10^{1}$ | $\mathbf{1.7317 \times 10^{1}}$ | $7.3968 \times 10^{1}$ |
| F9 | Ave | $2.7068 \times 10^{3}$ | $8.5554 \times 10^{3}$ | $1.2197 \times 10^{4}$ | $6.7404 \times 10^{3}$ | $5.8754 \times 10^{3}$ | $2.1794 \times 10^{3}$ | $7.5764 \times 10^{3}$ | $5.5959 \times 10^{3}$ | $1.1489 \times 10^{4}$ | $\mathbf{1.4105 \times 10^{3}}$ |
| | Std | $9.5894 \times 10^{2}$ | $9.9791 \times 10^{2}$ | $3.2777 \times 10^{3}$ | $1.9134 \times 10^{3}$ | $1.8864 \times 10^{3}$ | $9.2573 \times 10^{2}$ | $1.6404 \times 10^{3}$ | $1.5919 \times 10^{3}$ | $1.1826 \times 10^{3}$ | $\mathbf{8.8352 \times 10^{2}}$ |
| F10 | Ave | $5.5675 \times 10^{3}$ | $6.3233 \times 10^{3}$ | $7.6089 \times 10^{3}$ | $6.7606 \times 10^{3}$ | $6.3981 \times 10^{3}$ | $6.2803 \times 10^{3}$ | $6.7431 \times 10^{3}$ | $\mathbf{5.3515 \times 10^{3}}$ | $8.9226 \times 10^{3}$ | $7.1647 \times 10^{3}$ |
| | Std | $1.8329 \times 10^{3}$ | $6.3159 \times 10^{2}$ | $6.7695 \times 10^{2}$ | $1.1099 \times 10^{3}$ | $1.2707 \times 10^{3}$ | $7.2269 \times 10^{2}$ | $8.0957 \times 10^{2}$ | $5.3272 \times 10^{2}$ | $\mathbf{4.6944 \times 10^{2}}$ | $1.7543 \times 10^{3}$ |
| F11 | Ave | $2.4534 \times 10^{3}$ | $1.6293 \times 10^{3}$ | $1.0069 \times 10^{4}$ | $1.8561 \times 10^{3}$ | $1.3304 \times 10^{3}$ | $1.2704 \times 10^{3}$ | $2.0634 \times 10^{3}$ | $1.8111 \times 10^{3}$ | $6.2712 \times 10^{3}$ | $\mathbf{1.2138 \times 10^{3}}$ |
| | Std | $1.1557 \times 10^{3}$ | $2.4095 \times 10^{2}$ | $4.7596 \times 10^{3}$ | $6.9368 \times 10^{2}$ | $6.9796 \times 10^{1}$ | $\mathbf{4.3831 \times 10^{1}}$ | $4.1270 \times 10^{2}$ | $3.0496 \times 10^{2}$ | $9.0392 \times 10^{2}$ | $5.0400 \times 10^{1}$ |
| F12 | Ave | $1.0163 \times 10^{8}$ | $7.3440 \times 10^{7}$ | $4.1530 \times 10^{8}$ | $6.2137 \times 10^{7}$ | $1.2820 \times 10^{7}$ | $3.0874 \times 10^{6}$ | $1.5197 \times 10^{8}$ | $4.9246 \times 10^{7}$ | $6.3347 \times 10^{9}$ | $\mathbf{1.8072 \times 10^{5}}$ |
| | Std | $1.0264 \times 10^{8}$ | $6.4798 \times 10^{7}$ | $3.0725 \times 10^{8}$ | $9.0610 \times 10^{7}$ | $1.2403 \times 10^{7}$ | $2.7101 \times 10^{6}$ | $1.1688 \times 10^{8}$ | $3.4240 \times 10^{7}$ | $1.3267 \times 10^{9}$ | $\mathbf{4.8311 \times 10^{5}}$ |
| F13 | Ave | $1.9983 \times 10^{7}$ | $3.7066 \times 10^{6}$ | $1.4053 \times 10^{7}$ | $7.9961 \times 10^{6}$ | $9.7150 \times 10^{4}$ | $4.2400 \times 10^{4}$ | $1.1895 \times 10^{6}$ | $9.3332 \times 10^{5}$ | $1.3714 \times 10^{9}$ | $\mathbf{6.5573 \times 10^{3}}$ |
| | Std | $5.5471 \times 10^{7}$ | $1.4374 \times 10^{7}$ | $1.1411 \times 10^{7}$ | $2.7825 \times 10^{7}$ | $6.5602 \times 10^{4}$ | $4.0877 \times 10^{4}$ | $2.1214 \times 10^{6}$ | $2.2061 \times 10^{6}$ | $4.2588 \times 10^{8}$ | $\mathbf{1.5293 \times 10^{4}}$ |
| F14 | Ave | $8.0634 \times 10^{5}$ | $1.1934 \times 10^{6}$ | $2.5737 \times 10^{6}$ | $2.7875 \times 10^{5}$ | $1.2484 \times 10^{5}$ | $1.9494 \times 10^{3}$ | $5.3941 \times 10^{5}$ | $4.6480 \times 10^{5}$ | $2.1453 \times 10^{6}$ | $\mathbf{1.5054 \times 10^{3}}$ |
| | Std | $9.3946 \times 10^{5}$ | $1.5594 \times 10^{6}$ | $2.5689 \times 10^{6}$ | $5.8602 \times 10^{5}$ | $1.1942 \times 10^{5}$ | $3.6127 \times 10^{2}$ | $6.0511 \times 10^{5}$ | $5.3923 \times 10^{5}$ | $8.5517 \times 10^{5}$ | $\mathbf{6.7243 \times 10^{1}}$ |
| F15 | Ave | $4.5899 \times 10^{6}$ | $9.5039 \times 10^{4}$ | $1.1878 \times 10^{7}$ | $5.6060 \times 10^{4}$ | $1.6763 \times 10^{4}$ | $1.7924 \times 10^{4}$ | $6.9681 \times 10^{4}$ | $1.2546 \times 10^{4}$ | $1.0125 \times 10^{8}$ | $\mathbf{1.8823 \times 10^{3}}$ |
| | Std | $1.3699 \times 10^{7}$ | $4.2648 \times 10^{4}$ | $2.0978 \times 10^{7}$ | $5.1701 \times 10^{4}$ | $8.5094 \times 10^{3}$ | $8.7746 \times 10^{3}$ | $7.0822 \times 10^{4}$ | $9.6876 \times 10^{3}$ | $5.6515 \times 10^{7}$ | $\mathbf{9.5879 \times 10^{2}}$ |
| F16 | Ave | $2.6379 \times 10^{3}$ | $3.6857 \times 10^{3}$ | $4.1793 \times 10^{3}$ | $3.1687 \times 10^{3}$ | $3.0665 \times 10^{3}$ | $2.8429 \times 10^{3}$ | $3.8212 \times 10^{3}$ | $2.9744 \times 10^{3}$ | $4.9498 \times 10^{3}$ | $\mathbf{2.4557 \times 10^{3}}$ |
| | Std | $3.2081 \times 10^{2}$ | $4.3717 \times 10^{2}$ | $6.3866 \times 10^{2}$ | $4.4355 \times 10^{2}$ | $3.4097 \times 10^{2}$ | $\mathbf{3.0578 \times 10^{2}}$ | $5.5792 \times 10^{2}$ | $3.5104 \times 10^{2}$ | $3.0675 \times 10^{2}$ | $5.8085 \times 10^{2}$ |
| F17 | Ave | $2.0537 \times 10^{3}$ | $2.6385 \times 10^{3}$ | $2.8084 \times 10^{3}$ | $2.6422 \times 10^{3}$ | $2.4107 \times 10^{3}$ | $2.1937 \times 10^{3}$ | $2.6487 \times 10^{3}$ | $2.3483 \times 10^{3}$ | $3.1762 \times 10^{3}$ | $\mathbf{2.0420 \times 10^{3}}$ |
| | Std | $1.9635 \times 10^{2}$ | $3.1953 \times 10^{2}$ | $3.0411 \times 10^{2}$ | $2.9011 \times 10^{2}$ | $2.3270 \times 10^{2}$ | $\mathbf{1.5592 \times 10^{2}}$ | $3.1970 \times 10^{2}$ | $2.1746 \times 10^{2}$ | $2.7603 \times 10^{2}$ | $1.5691 \times 10^{2}$ |
| F18 | Ave | $1.5958 \times 10^{6}$ | $3.3601 \times 10^{6}$ | $1.1045 \times 10^{7}$ | $5.8236 \times 10^{6}$ | $5.1167 \times 10^{5}$ | $6.9205 \times 10^{4}$ | $8.6242 \times 10^{6}$ | $3.7282 \times 10^{6}$ | $2.1310 \times 10^{7}$ | $\mathbf{2.0689 \times 10^{3}}$ |
| | Std | $1.7245 \times 10^{6}$ | $5.0256 \times 10^{6}$ | $9.5463 \times 10^{6}$ | $7.5165 \times 10^{6}$ | $5.9491 \times 10^{5}$ | $3.3179 \times 10^{4}$ | $8.4257 \times 10^{6}$ | $4.5366 \times 10^{6}$ | $1.0801 \times 10^{7}$ | $\mathbf{3.8794 \times 10^{2}}$ |
| F19 | Ave | $2.8303 \times 10^{6}$ | $2.2364 \times 10^{6}$ | $1.0688 \times 10^{7}$ | $9.1244 \times 10^{6}$ | $1.3469 \times 10^{5}$ | $1.9110 \times 10^{4}$ | $1.7152 \times 10^{6}$ | $6.0835 \times 10^{4}$ | $1.2964 \times 10^{8}$ | $\mathbf{1.9960 \times 10^{3}}$ |
| | Std | $7.8432 \times 10^{6}$ | $1.5867 \times 10^{6}$ | $1.2342 \times 10^{7}$ | $2.0168 \times 10^{7}$ | $2.0056 \times 10^{5}$ | $1.6062 \times 10^{4}$ | $1.8873 \times 10^{6}$ | $2.4119 \times 10^{5}$ | $5.8854 \times 10^{7}$ | $\mathbf{6.7308 \times 10^{1}}$ |
| F20 | Ave | $2.4723 \times 10^{3}$ | $2.8286 \times 10^{3}$ | $2.9200 \times 10^{3}$ | $2.7512 \times 10^{3}$ | $2.5745 \times 10^{3}$ | $2.5238 \times 10^{3}$ | $2.7288 \times 10^{3}$ | $2.5193 \times 10^{3}$ | $3.0168 \times 10^{3}$ | $\mathbf{2.3523 \times 10^{3}}$ |
| | Std | $1.8979 \times 10^{2}$ | $1.9870 \times 10^{2}$ | $2.4599 \times 10^{2}$ | $1.7427 \times 10^{2}$ | $1.7591 \times 10^{2}$ | $1.6783 \times 10^{2}$ | $2.0554 \times 10^{2}$ | $2.0455 \times 10^{2}$ | $\mathbf{1.4168 \times 10^{2}}$ | $2.2171 \times 10^{2}$ |
| F21 | Ave | $2.4064 \times 10^{3}$ | $2.5839 \times 10^{3}$ | $2.6433 \times 10^{3}$ | $2.5573 \times 10^{3}$ | $2.4774 \times 10^{3}$ | $\mathbf{2.4001 \times 10^{3}}$ | $2.5679 \times 10^{3}$ | $2.4683 \times 10^{3}$ | $2.6972 \times 10^{3}$ | $2.4212 \times 10^{3}$ |
| | Std | $3.5512 \times 10^{1}$ | $5.3113 \times 10^{1}$ | $6.7225 \times 10^{1}$ | $5.2541 \times 10^{1}$ | $4.6240 \times 10^{1}$ | $3.2223 \times 10^{1}$ | $6.7601 \times 10^{1}$ | $3.3755 \times 10^{1}$ | $\mathbf{2.7120 \times 10^{1}}$ | $5.5614 \times 10^{1}$ |
| F22 | Ave | $5.9872 \times 10^{3}$ | $6.6127 \times 10^{3}$ | $8.5522 \times 10^{3}$ | $5.2909 \times 10^{3}$ | $\mathbf{3.3616 \times 10^{3}}$ | $5.8878 \times 10^{3}$ | $6.7320 \times 10^{3}$ | $4.7355 \times 10^{3}$ | $7.3379 \times 10^{3}$ | $5.9448 \times 10^{3}$ |
| | Std | $2.0867 \times 10^{3}$ | $1.8967 \times 10^{3}$ | $1.3148 \times 10^{3}$ | $2.6190 \times 10^{3}$ | $2.0941 \times 10^{3}$ | $2.5773 \times 10^{3}$ | $2.2450 \times 10^{3}$ | $2.0894 \times 10^{3}$ | $\mathbf{9.7391 \times 10^{2}}$ | $3.5185 \times 10^{3}$ |
| F23 | Ave | $2.7704 \times 10^{3}$ | $3.2910 \times 10^{3}$ | $3.1404 \times 10^{3}$ | $3.0293 \times 10^{3}$ | $2.8728 \times 10^{3}$ | $2.7765 \times 10^{3}$ | $3.1816 \times 10^{3}$ | $2.8252 \times 10^{3}$ | $3.3257 \times 10^{3}$ | $\mathbf{2.7536 \times 10^{3}}$ |
| | Std | $3.9660 \times 10^{1}$ | $1.6821 \times 10^{2}$ | $9.8631 \times 10^{1}$ | $1.0424 \times 10^{2}$ | $5.7038 \times 10^{1}$ | $\mathbf{3.0319 \times 10^{1}}$ | $1.3544 \times 10^{2}$ | $4.6436 \times 10^{1}$ | $9.8252 \times 10^{1}$ | $5.1093 \times 10^{1}$ |
| F24 | Ave | $\mathbf{2.9486 \times 10^{3}}$ | $3.5253 \times 10^{3}$ | $3.2902 \times 10^{3}$ | $3.1891 \times 10^{3}$ | $3.0143 \times 10^{3}$ | $2.9607 \times 10^{3}$ | $3.3155 \times 10^{3}$ | $3.0075 \times 10^{3}$ | $3.5564 \times 10^{3}$ | $2.9537 \times 10^{3}$ |
| | Std | $4.6338 \times 10^{1}$ | $1.6721 \times 10^{2}$ | $9.3484 \times 10^{1}$ | $8.5319 \times 10^{1}$ | $5.5593 \times 10^{1}$ | $\mathbf{4.2364 \times 10^{1}}$ | $1.4482 \times 10^{2}$ | $4.2432 \times 10^{1}$ | $1.4896 \times 10^{2}$ | $5.9379 \times 10^{1}$ |
| F25 | Ave | $3.0200 \times 10^{3}$ | $3.0203 \times 10^{3}$ | $3.2602 \times 10^{3}$ | $3.0006 \times 10^{3}$ | $2.9706 \times 10^{3}$ | $2.9092 \times 10^{3}$ | $3.0765 \times 10^{3}$ | $3.0351 \times 10^{3}$ | $4.0450 \times 10^{3}$ | $\mathbf{2.8870 \times 10^{3}}$ |
| | Std | $6.5741 \times 10^{1}$ | $2.8367 \times 10^{1}$ | $6.6005 \times 10^{1}$ | $5.7150 \times 10^{1}$ | $2.7027 \times 10^{1}$ | $1.5518 \times 10^{1}$ | $5.0525 \times 10^{1}$ | $6.3630 \times 10^{1}$ | $1.1068 \times 10^{2}$ | $\mathbf{2.3366 \times 10^{0}}$ |
| F26 | Ave | $4.9605 \times 10^{3}$ | $8.0823 \times 10^{3}$ | $8.4965 \times 10^{3}$ | $6.9611 \times 10^{3}$ | $6.4608 \times 10^{3}$ | $5.1849 \times 10^{3}$ | $8.3300 \times 10^{3}$ | $5.0938 \times 10^{3}$ | $9.5361 \times 10^{3}$ | $\mathbf{4.7434 \times 10^{3}}$ |
| | Std | $5.2449 \times 10^{2}$ | $9.9289 \times 10^{2}$ | $7.2975 \times 10^{2}$ | $9.8260 \times 10^{2}$ | $1.7240 \times 10^{3}$ | $6.5062 \times 10^{2}$ | $1.1932 \times 10^{3}$ | $1.1404 \times 10^{3}$ | $\mathbf{3.9691 \times 10^{2}}$ | $5.8371 \times 10^{2}$ |
| F27 | Ave | $3.2683 \times 10^{3}$ | $3.5426 \times 10^{3}$ | $3.5062 \times 10^{3}$ | $3.3330 \times 10^{3}$ | $3.3602 \times 10^{3}$ | $3.2394 \times 10^{3}$ | $3.4282 \times 10^{3}$ | $3.2560 \times 10^{3}$ | $4.2736 \times 10^{3}$ | $\mathbf{3.2315 \times 10^{3}}$ |
| | Std | $3.3618 \times 10^{1}$ | $1.3457 \times 10^{2}$ | $1.3750 \times 10^{2}$ | $8.4229 \times 10^{1}$ | $8.1811 \times 10^{1}$ | $2.3507 \times 10^{1}$ | $1.3709 \times 10^{2}$ | $\mathbf{1.9883 \times 10^{1}}$ | $2.2840 \times 10^{2}$ | $2.3009 \times 10^{1}$ |
| F28 | Ave | $3.4679 \times 10^{3}$ | $3.5379 \times 10^{3}$ | $3.8412 \times 10^{3}$ | $3.5966 \times 10^{3}$ | $3.3157 \times 10^{3}$ | $3.6103 \times 10^{3}$ | $3.5978 \times 10^{3}$ | $3.4442 \times 10^{3}$ | $5.9162 \times 10^{3}$ | $\mathbf{3.2950 \times 10^{3}}$ |
| | Std | $1.3915 \times 10^{2}$ | $1.0857 \times 10^{2}$ | $2.6837 \times 10^{2}$ | $4.9505 \times 10^{2}$ | $\mathbf{2.3480 \times 10^{1}}$ | $9.7768 \times 10^{2}$ | $2.1013 \times 10^{2}$ | $6.9594 \times 10^{1}$ | $3.2772 \times 10^{2}$ | $4.0526 \times 10^{2}$ |
| F29 | Ave | $3.9101 \times 10^{3}$ | $4.9147 \times 10^{3}$ | $5.3939 \times 10^{3}$ | $4.4567 \times 10^{3}$ | $4.6175 \times 10^{3}$ | $3.9504 \times 10^{3}$ | $5.1136 \times 10^{3}$ | $4.0695 \times 10^{3}$ | $6.3370 \times 10^{3}$ | $\mathbf{3.6262 \times 10^{3}}$ |
| | Std | $\mathbf{2.0827 \times 10^{2}}$ | $4.4377 \times 10^{2}$ | $5.7960 \times 10^{2}$ | $4.5182 \times 10^{2}$ | $3.2053 \times 10^{2}$ | $2.6481 \times 10^{2}$ | $4.7903 \times 10^{2}$ | $2.2367 \times 10^{2}$ | $3.4739 \times 10^{2}$ | $2.4590 \times 10^{2}$ |
| F30 | Ave | $1.0443 \times 10^{7}$ | $1.6774 \times 10^{7}$ | $7.7775 \times 10^{7}$ | $3.6753 \times 10^{6}$ | $1.3626 \times 10^{6}$ | $7.1430 \times 10^{4}$ | $1.8148 \times 10^{7}$ | $4.3507 \times 10^{5}$ | $6.2241 \times 10^{8}$ | $\mathbf{6.7748 \times 10^{3}}$ |
| | Std | $6.5886 \times 10^{6}$ | $1.8815 \times 10^{7}$ | $7.6390 \times 10^{7}$ | $6.3748 \times 10^{6}$ | $1.1777 \times 10^{6}$ | $9.0456 \times 10^{4}$ | $1.5750 \times 10^{7}$ | $3.7494 \times 10^{5}$ | $2.0564 \times 10^{8}$ | $\mathbf{2.1549 \times 10^{3}}$ |

**Table 3.** Experimental results of CEC2017 (dim=50).

| Function | Metric | GWO | HHO | WOA | DBO | BPBO | ALA | SGA | HBO | CDDO | MECDDO |
|---|---|---|---|---|---|---|---|---|---|---|---|
| F1 | Ave | $1.2138 \times 10^{10}$ | $5.4881 \times 10^{9}$ | $2.1747 \times 10^{10}$ | $8.3045 \times 10^{9}$ | $5.7488 \times 10^{8}$ | $5.5660 \times 10^{8}$ | $2.3263 \times 10^{10}$ | $1.2593 \times 10^{10}$ | $8.7082 \times 10^{10}$ | $\mathbf{2.8863 \times 10^{6}}$ |
| | Std | $4.3668 \times 10^{9}$ | $1.5850 \times 10^{9}$ | $5.2661 \times 10^{9}$ | $1.2143 \times 10^{10}$ | $2.1478 \times 10^{8}$ | $2.6084 \times 10^{8}$ | $6.1364 \times 10^{9}$ | $3.2761 \times 10^{9}$ | $5.4063 \times 10^{9}$ | $\mathbf{2.5340 \times 10^{6}}$ |
| F2 | Ave | $2.7349 \times 10^{62}$ | $2.2171 \times 10^{68}$ | $1.2933 \times 10^{79}$ | $1.2788 \times 10^{66}$ | $2.6421 \times 10^{54}$ | $6.4064 \times 10^{54}$ | $1.4351 \times 10^{66}$ | $5.1761 \times 10^{58}$ | $7.7661 \times 10^{76}$ | $\mathbf{6.2483 \times 10^{45}}$ |
| | Std | $1.4980 \times 10^{63}$ | $1.1967 \times 10^{69}$ | $6.2216 \times 10^{79}$ | $6.0848 \times 10^{66}$ | $8.2946 \times 10^{54}$ | $3.5067 \times 10^{55}$ | $7.5368 \times 10^{66}$ | $2.5408 \times 10^{59}$ | $3.8076 \times 10^{77}$ | $\mathbf{3.3379 \times 10^{46}}$ |
| F3 | Ave | $1.7018 \times 10^{5}$ | $1.7491 \times 10^{5}$ | $2.8672 \times 10^{5}$ | $2.7778 \times 10^{5}$ | $1.5270 \times 10^{5}$ | $1.1926 \times 10^{5}$ | $1.8641 \times 10^{5}$ | $1.8334 \times 10^{5}$ | $2.0879 \times 10^{5}$ | $\mathbf{1.0698 \times 10^{5}}$ |
| | Std | $2.9670 \times 10^{4}$ | $1.6055 \times 10^{4}$ | $1.0038 \times 10^{5}$ | $7.1378 \times 10^{4}$ | $\mathbf{1.3768 \times 10^{4}}$ | $3.1542 \times 10^{4}$ | $4.8158 \times 10^{4}$ | $1.9322 \times 10^{4}$ | $2.6291 \times 10^{4}$ | $2.5859 \times 10^{4}$ |
| F4 | Ave | $1.6522 \times 10^{3}$ | $2.0008 \times 10^{3}$ | $5.1401 \times 10^{3}$ | $1.8038 \times 10^{3}$ | $8.1523 \times 10^{2}$ | $7.5427 \times 10^{2}$ | $3.0748 \times 10^{3}$ | $1.8903 \times 10^{3}$ | $2.4032 \times 10^{4}$ | $\mathbf{5.9977 \times 10^{2}}$ |
| | Std | $6.9206 \times 10^{2}$ | $5.0184 \times 10^{2}$ | $1.4384 \times 10^{3}$ | $1.5255 \times 10^{3}$ | $7.9754 \times 10^{1}$ | $8.4429 \times 10^{1}$ | $7.8497 \times 10^{2}$ | $5.7520 \times 10^{2}$ | $2.2133 \times 10^{3}$ | $\mathbf{4.1347 \times 10^{1}}$ |
| F5 | Ave | $\mathbf{7.6341 \times 10^{2}}$ | $9.4014 \times 10^{2}$ | $1.1101 \times 10^{3}$ | $9.8926 \times 10^{2}$ | $8.8570 \times 10^{2}$ | $7.7184 \times 10^{2}$ | $9.8817 \times 10^{2}$ | $9.3574 \times 10^{2}$ | $1.1791 \times 10^{3}$ | $8.1579 \times 10^{2}$ |
| | Std | $4.4737 \times 10^{1}$ | $3.3865 \times 10^{1}$ | $8.6935 \times 10^{1}$ | $8.5879 \times 10^{1}$ | $3.3887 \times 10^{1}$ | $5.5400 \times 10^{1}$ | $4.2895 \times 10^{1}$ | $3.8839 \times 10^{1}$ | $\mathbf{1.9630 \times 10^{1}}$ | $1.3802 \times 10^{2}$ |
| F6 | Ave | $6.2553 \times 10^{2}$ | $6.8092 \times 10^{2}$ | $6.9373 \times 10^{2}$ | $6.6784 \times 10^{2}$ | $6.6976 \times 10^{2}$ | $6.3181 \times 10^{2}$ | $6.8199 \times 10^{2}$ | $6.4692 \times 10^{2}$ | $7.0279 \times 10^{2}$ | $\mathbf{6.1560 \times 10^{2}}$ |
| | Std | $5.3444 \times 10^{0}$ | $4.4297 \times 10^{0}$ | $7.9878 \times 10^{0}$ | $1.0579 \times 10^{1}$ | $7.7585 \times 10^{0}$ | $1.0068 \times 10^{1}$ | $6.7870 \times 10^{0}$ | $1.0077 \times 10^{1}$ | $\mathbf{3.1771 \times 10^{0}}$ | $6.1672 \times 10^{0}$ |
| F7 | Ave | $\mathbf{1.1625 \times 10^{3}}$ | $1.8942 \times 10^{3}$ | $1.8906 \times 10^{3}$ | $1.4039 \times 10^{3}$ | $1.6808 \times 10^{3}$ | $1.2258 \times 10^{3}$ | $1.8221 \times 10^{3}$ | $1.4904 \times 10^{3}$ | $1.9415 \times 10^{3}$ | $1.3313 \times 10^{3}$ |
| | Std | $9.0709 \times 10^{1}$ | $\mathbf{5.8957 \times 10^{1}}$ | $9.8866 \times 10^{1}$ | $1.4100 \times 10^{2}$ | $1.2005 \times 10^{2}$ | $8.5490 \times 10^{1}$ | $9.7944 \times 10^{1}$ | $1.0427 \times 10^{2}$ | $6.9589 \times 10^{1}$ | $7.0731 \times 10^{1}$ |
| F8 | Ave | $1.0760 \times 10^{3}$ | $1.2376 \times 10^{3}$ | $1.4284 \times 10^{3}$ | $1.3173 \times 10^{3}$ | $1.2123 \times 10^{3}$ | $\mathbf{1.0722 \times 10^{3}}$ | $1.3090 \times 10^{3}$ | $1.2221 \times 10^{3}$ | $1.4774 \times 10^{3}$ | $1.0916 \times 10^{3}$ |
| | Std | $4.6545 \times 10^{1}$ | $3.8020 \times 10^{1}$ | $9.0298 \times 10^{1}$ | $9.0820 \times 10^{1}$ | $4.3926 \times 10^{1}$ | $4.4846 \times 10^{1}$ | $4.5233 \times 10^{1}$ | $4.9660 \times 10^{1}$ | $\mathbf{2.3266 \times 10^{1}}$ | $1.2427 \times 10^{2}$ |
| F9 | Ave | $1.0416 \times 10^{4}$ | $3.1750 \times 10^{4}$ | $3.8504 \times 10^{4}$ | $2.8432 \times 10^{4}$ | $2.5432 \times 10^{4}$ | $8.8818 \times 10^{3}$ | $2.3547 \times 10^{4}$ | $1.8101 \times 10^{4}$ | $4.0472 \times 10^{4}$ | $\mathbf{8.5774 \times 10^{3}}$ |
| | Std | $3.7972 \times 10^{3}$ | $3.3877 \times 10^{3}$ | $1.0690 \times 10^{4}$ | $7.5904 \times 10^{3}$ | $6.0361 \times 10^{3}$ | $3.0851 \times 10^{3}$ | $4.1751 \times 10^{3}$ | $3.2782 \times 10^{3}$ | $\mathbf{2.3411 \times 10^{3}}$ | $5.1479 \times 10^{3}$ |
| F10 | Ave | $\mathbf{8.8935 \times 10^{3}}$ | $1.0162 \times 10^{4}$ | $1.3650 \times 10^{4}$ | $1.1823 \times 10^{4}$ | $1.0452 \times 10^{4}$ | $1.1174 \times 10^{4}$ | $1.1935 \times 10^{4}$ | $9.6421 \times 10^{3}$ | $1.5197 \times 10^{4}$ | $1.4411 \times 10^{4}$ |
| | Std | $2.5043 \times 10^{3}$ | $9.4416 \times 10^{2}$ | $9.2150 \times 10^{2}$ | $2.4094 \times 10^{3}$ | $2.1412 \times 10^{3}$ | $1.5606 \times 10^{3}$ | $1.4073 \times 10^{3}$ | $8.5382 \times 10^{2}$ | $\mathbf{3.7009 \times 10^{2}}$ | $1.9311 \times 10^{3}$ |
| F11 | Ave | $6.4942 \times 10^{3}$ | $3.1121 \times 10^{3}$ | $8.6713 \times 10^{3}$ | $4.7746 \times 10^{3}$ | $2.1660 \times 10^{3}$ | $1.8480 \times 10^{3}$ | $6.1616 \times 10^{3}$ | $3.7430 \times 10^{3}$ | $2.0566 \times 10^{4}$ | $\mathbf{1.3663 \times 10^{3}}$ |
| | Std | $2.7705 \times 10^{3}$ | $7.7274 \times 10^{2}$ | $2.2779 \times 10^{3}$ | $2.7643 \times 10^{3}$ | $2.8655 \times 10^{2}$ | $2.6508 \times 10^{2}$ | $1.9984 \times 10^{3}$ | $9.9596 \times 10^{2}$ | $1.8058 \times 10^{3}$ | $\mathbf{9.2122 \times 10^{1}}$ |
| F12 | Ave | $1.5381 \times 10^{9}$ | $1.0409 \times 10^{9}$ | $4.2456 \times 10^{9}$ | $7.1799 \times 10^{8}$ | $9.9911 \times 10^{7}$ | $4.7879 \times 10^{7}$ | $1.7094 \times 10^{9}$ | $9.8366 \times 10^{8}$ | $5.3421 \times 10^{10}$ | $\mathbf{5.1430 \times 10^{6}}$ |
| | Std | $1.4450 \times 10^{9}$ | $7.3175 \times 10^{8}$ | $2.1171 \times 10^{9}$ | $3.8640 \times 10^{8}$ | $6.2295 \times 10^{7}$ | $3.6793 \times 10^{7}$ | $9.3560 \times 10^{8}$ | $4.8805 \times 10^{8}$ | $6.2707 \times 10^{9}$ | $\mathbf{5.4436 \times 10^{6}}$ |
| F13 | Ave | $2.2706 \times 10^{8}$ | $3.7784 \times 10^{7}$ | $5.5663 \times 10^{8}$ | $1.4603 \times 10^{8}$ | $1.6741 \times 10^{6}$ | $6.5288 \times 10^{4}$ | $3.3352 \times 10^{8}$ | $6.2602 \times 10^{7}$ | $2.3932 \times 10^{10}$ | $\mathbf{2.5958 \times 10^{4}}$ |
| | Std | $2.5291 \times 10^{8}$ | $7.6704 \times 10^{7}$ | $3.4308 \times 10^{8}$ | $2.2106 \times 10^{8}$ | $2.1535 \times 10^{6}$ | $3.3603 \times 10^{4}$ | $4.8468 \times 10^{8}$ | $5.4241 \times 10^{7}$ | $6.3884 \times 10^{9}$ | $\mathbf{1.2559 \times 10^{4}}$ |
| F14 | Ave | $2.3547 \times 10^{6}$ | $4.6786 \times 10^{6}$ | $6.1745 \times 10^{6}$ | $3.8375 \times 10^{6}$ | $7.7870 \times 10^{5}$ | $5.4380 \times 10^{4}$ | $3.5832 \times 10^{6}$ | $3.8996 \times 10^{6}$ | $3.2584 \times 10^{7}$ | $\mathbf{1.7306 \times 10^{3}}$ |
| | Std | $3.4391 \times 10^{6}$ | $3.1187 \times 10^{6}$ | $4.9877 \times 10^{6}$ | $3.9584 \times 10^{6}$ | $6.3448 \times 10^{5}$ | $5.8184 \times 10^{4}$ | $2.7235 \times 10^{6}$ | $3.6055 \times 10^{6}$ | $1.3618 \times 10^{7}$ | $\mathbf{8.7460 \times 10^{1}}$ |
| F15 | Ave | $4.4759 \times 10^{7}$ | $1.9152 \times 10^{6}$ | $5.6180 \times 10^{7}$ | $1.0140 \times 10^{8}$ | $8.5055 \times 10^{4}$ | $2.1632 \times 10^{4}$ | $9.4002 \times 10^{7}$ | $1.4216 \times 10^{6}$ | $2.5386 \times 10^{9}$ | $\mathbf{7.0815 \times 10^{3}}$ |
| | Std | $7.2725 \times 10^{7}$ | $1.2177 \times 10^{6}$ | $4.4391 \times 10^{7}$ | $3.2467 \times 10^{8}$ | $7.3622 \times 10^{4}$ | $1.1240 \times 10^{4}$ | $2.7463 \times 10^{8}$ | $2.3488 \times 10^{6}$ | $6.1949 \times 10^{8}$ | $\mathbf{9.8211 \times 10^{3}}$ |
| F16 | Ave | $\mathbf{3.3609 \times 10^{3}}$ | $5.0673 \times 10^{3}$ | $6.7326 \times 10^{3}$ | $4.7897 \times 10^{3}$ | $3.8590 \times 10^{3}$ | $3.8988 \times 10^{3}$ | $5.2376 \times 10^{3}$ | $4.0938 \times 10^{3}$ | $7.8078 \times 10^{3}$ | $3.6370 \times 10^{3}$ |
| | Std | $5.5881 \times 10^{2}$ | $8.9915 \times 10^{2}$ | $9.3664 \times 10^{2}$ | $6.7691 \times 10^{2}$ | $4.8414 \times 10^{2}$ | $\mathbf{4.2915 \times 10^{2}}$ | $7.4071 \times 10^{2}$ | $4.8018 \times 10^{2}$ | $5.2040 \times 10^{2}$ | $9.1419 \times 10^{2}$ |
| F17 | Ave | $3.1960 \times 10^{3}$ | $3.8825 \times 10^{3}$ | $4.6996 \times 10^{3}$ | $4.1940 \times 10^{3}$ | $3.8375 \times 10^{3}$ | $3.3582 \times 10^{3}$ | $4.2878 \times 10^{3}$ | $3.4535 \times 10^{3}$ | $5.7166 \times 10^{3}$ | $\mathbf{3.1952 \times 10^{3}}$ |
| | Std | $4.2059 \times 10^{2}$ | $3.6890 \times 10^{2}$ | $6.4412 \times 10^{2}$ | $5.2109 \times 10^{2}$ | $4.6662 \times 10^{2}$ | $3.2876 \times 10^{2}$ | $6.3450 \times 10^{2}$ | $\mathbf{3.2623 \times 10^{2}}$ | $7.9318 \times 10^{2}$ | $6.0116 \times 10^{2}$ |
| F18 | Ave | $1.2252 \times 10^{7}$ | $1.0251 \times 10^{7}$ | $6.4702 \times 10^{7}$ | $1.1550 \times 10^{7}$ | $2.7132 \times 10^{6}$ | $9.3011 \times 10^{5}$ | $3.6668 \times 10^{7}$ | $1.1111 \times 10^{7}$ | $8.5404 \times 10^{7}$ | $\mathbf{4.3812 \times 10^{4}}$ |
| | Std | $1.2396 \times 10^{7}$ | $1.1660 \times 10^{7}$ | $4.9285 \times 10^{7}$ | $1.5844 \times 10^{7}$ | $1.6344 \times 10^{6}$ | $1.2868 \times 10^{6}$ | $2.6865 \times 10^{7}$ | $7.6190 \times 10^{6}$ | $3.9449 \times 10^{7}$ | $\mathbf{2.9728 \times 10^{4}}$ |
| F19 | Ave | $2.6892 \times 10^{7}$ | $2.0270 \times 10^{6}$ | $2.9199 \times 10^{7}$ | $4.8281 \times 10^{6}$ | $2.8857 \times 10^{5}$ | $1.4618 \times 10^{4}$ | $1.4593 \times 10^{7}$ | $9.0000 \times 10^{5}$ | $1.5491 \times 10^{9}$ | $\mathbf{2.6365 \times 10^{3}}$ |
| | Std | $7.1801 \times 10^{7}$ | $1.7379 \times 10^{6}$ | $2.8580 \times 10^{7}$ | $5.3976 \times 10^{6}$ | $3.2140 \times 10^{5}$ | $1.4408 \times 10^{4}$ | $1.4649 \times 10^{7}$ | $8.8140 \times 10^{5}$ | $3.4403 \times 10^{8}$ | $\mathbf{2.0528 \times 10^{3}}$ |

**Table 3.** *Cont.*

| Function | Metric | GWO | HHO | WOA | DBO | BPBO | ALA | SGA | HBO | CDDO | MECDDO |
|---|---|---|---|---|---|---|---|---|---|---|---|
| F20 | Ave | **$3.1489 \times 10^{3}$** | $3.5275 \times 10^{3}$ | $3.9478 \times 10^{3}$ | $3.8466 \times 10^{3}$ | $3.4178 \times 10^{3}$ | $3.5998 \times 10^{3}$ | $3.7148 \times 10^{3}$ | $3.4623 \times 10^{3}$ | $4.0950 \times 10^{3}$ | $3.8054 \times 10^{3}$ |
| | Std | $3.6869 \times 10^{2}$ | $3.0574 \times 10^{2}$ | $3.5501 \times 10^{2}$ | $3.5501 \times 10^{2}$ | $3.1238 \times 10^{2}$ | $2.8782 \times 10^{2}$ | $3.0620 \times 10^{2}$ | $3.7304 \times 10^{2}$ | **$1.8117 \times 10^{2}$** | $4.4620 \times 10^{2}$ |
| F21 | Ave | **$2.5640 \times 10^{3}$** | $2.9260 \times 10^{3}$ | $3.1007 \times 10^{3}$ | $2.8902 \times 10^{3}$ | $2.6847 \times 10^{3}$ | $2.5678 \times 10^{3}$ | $2.9460 \times 10^{3}$ | $2.7262 \times 10^{3}$ | $3.1097 \times 10^{3}$ | $2.6020 \times 10^{3}$ |
| | Std | $6.7678 \times 10^{1}$ | $8.0276 \times 10^{1}$ | $1.3793 \times 10^{2}$ | $6.8136 \times 10^{1}$ | $6.7512 \times 10^{1}$ | $5.4519 \times 10^{1}$ | $1.0721 \times 10^{2}$ | $5.1574 \times 10^{1}$ | **$4.8247 \times 10^{1}$** | $1.3246 \times 10^{2}$ |
| F22 | Ave | **$1.0175 \times 10^{4}$** | $1.2464 \times 10^{4}$ | $1.4921 \times 10^{4}$ | $1.3246 \times 10^{4}$ | $1.2505 \times 10^{4}$ | $1.3149 \times 10^{4}$ | $1.3494 \times 10^{4}$ | $1.1433 \times 10^{4}$ | $1.6815 \times 10^{4}$ | $1.5697 \times 10^{4}$ |
| | Std | $1.6640 \times 10^{3}$ | $1.0124 \times 10^{3}$ | $1.0460 \times 10^{3}$ | $2.2736 \times 10^{3}$ | $2.8345 \times 10^{3}$ | $1.1976 \times 10^{3}$ | $1.2261 \times 10^{3}$ | $6.9388 \times 10^{2}$ | **$4.6976 \times 10^{2}$** | $1.9017 \times 10^{3}$ |
| F23 | Ave | **$3.0354 \times 10^{3}$** | $4.0512 \times 10^{3}$ | $3.8836 \times 10^{3}$ | $3.5550 \times 10^{3}$ | $3.3193 \times 10^{3}$ | $3.0544 \times 10^{3}$ | $3.8974 \times 10^{3}$ | $3.1887 \times 10^{3}$ | $4.2805 \times 10^{3}$ | $3.0725 \times 10^{3}$ |
| | Std | $6.9734 \times 10^{1}$ | $2.2229 \times 10^{2}$ | $1.9584 \times 10^{2}$ | $1.3210 \times 10^{2}$ | $1.4358 \times 10^{2}$ | $6.6903 \times 10^{1}$ | $2.2192 \times 10^{2}$ | **$6.1647 \times 10^{1}$** | $2.0114 \times 10^{2}$ | $1.3450 \times 10^{2}$ |
| F24 | Ave | **$3.2306 \times 10^{3}$** | $4.4571 \times 10^{3}$ | $3.9225 \times 10^{3}$ | $3.6998 \times 10^{3}$ | $3.3711 \times 10^{3}$ | $3.2314 \times 10^{3}$ | $4.0860 \times 10^{3}$ | $3.3418 \times 10^{3}$ | $4.5322 \times 10^{3}$ | $3.2787 \times 10^{3}$ |
| | Std | $1.0028 \times 10^{2}$ | $2.8300 \times 10^{2}$ | $1.5787 \times 10^{2}$ | $1.2137 \times 10^{2}$ | $9.1319 \times 10^{1}$ | $8.3439 \times 10^{1}$ | $2.8366 \times 10^{2}$ | **$8.0392 \times 10^{1}$** | $1.7714 \times 10^{2}$ | $1.6363 \times 10^{2}$ |
| F25 | Ave | $4.0166 \times 10^{3}$ | $3.8037 \times 10^{3}$ | $5.2722 \times 10^{3}$ | $4.0538 \times 10^{3}$ | $3.3191 \times 10^{3}$ | $3.2649 \times 10^{3}$ | $4.8939 \times 10^{3}$ | $4.1626 \times 10^{3}$ | $1.1977 \times 10^{4}$ | **$3.0648 \times 10^{3}$** |
| | Std | $3.6595 \times 10^{2}$ | $2.0221 \times 10^{2}$ | $4.9646 \times 10^{2}$ | $2.1703 \times 10^{3}$ | $6.4740 \times 10^{1}$ | $9.3386 \times 10^{1}$ | $5.2722 \times 10^{2}$ | $3.2333 \times 10^{2}$ | $8.4696 \times 10^{2}$ | **$2.9872 \times 10^{1}$** |
| F26 | Ave | $7.3205 \times 10^{3}$ | $1.1845 \times 10^{4}$ | $1.5103 \times 10^{4}$ | $1.1339 \times 10^{4}$ | $1.0879 \times 10^{4}$ | $7.2419 \times 10^{3}$ | $1.3439 \times 10^{4}$ | $9.3381 \times 10^{3}$ | $1.5278 \times 10^{4}$ | **$6.5112 \times 10^{3}$** |
| | Std | $9.4543 \times 10^{2}$ | $9.5940 \times 10^{2}$ | $1.3203 \times 10^{3}$ | $1.2160 \times 10^{3}$ | $2.3080 \times 10^{3}$ | $9.1042 \times 10^{2}$ | $1.6825 \times 10^{3}$ | $2.3052 \times 10^{3}$ | **$4.7082 \times 10^{2}$** | $8.0521 \times 10^{2}$ |
| F27 | Ave | $3.7136 \times 10^{3}$ | $5.0163 \times 10^{3}$ | $4.9410 \times 10^{3}$ | $3.9523 \times 10^{3}$ | $4.0349 \times 10^{3}$ | $3.5953 \times 10^{3}$ | $4.3469 \times 10^{3}$ | $3.6349 \times 10^{3}$ | $6.6335 \times 10^{3}$ | **$3.5053 \times 10^{3}$** |
| | Std | $1.2976 \times 10^{2}$ | $6.5553 \times 10^{2}$ | $5.1662 \times 10^{2}$ | $2.1153 \times 10^{2}$ | $2.4084 \times 10^{2}$ | $1.4894 \times 10^{2}$ | $5.2163 \times 10^{2}$ | **$1.1333 \times 10^{2}$** | $5.4273 \times 10^{2}$ | $1.4271 \times 10^{2}$ |
| F28 | Ave | $4.5605 \times 10^{3}$ | $4.9487 \times 10^{3}$ | $6.2368 \times 10^{3}$ | $5.7328 \times 10^{3}$ | $3.7597 \times 10^{3}$ | $4.8586 \times 10^{3}$ | $5.8884 \times 10^{3}$ | $4.8141 \times 10^{3}$ | $1.0352 \times 10^{4}$ | **$3.6309 \times 10^{3}$** |
| | Std | $5.5751 \times 10^{2}$ | $4.3865 \times 10^{2}$ | $6.1406 \times 10^{2}$ | $1.9514 \times 10^{3}$ | **$1.2752 \times 10^{2}$** | $2.0788 \times 10^{3}$ | $6.1872 \times 10^{2}$ | $5.1036 \times 10^{2}$ | $5.5977 \times 10^{2}$ | $1.0214 \times 10^{3}$ |
| F29 | Ave | $4.9308 \times 10^{3}$ | $7.4302 \times 10^{3}$ | $1.0083 \times 10^{4}$ | $6.1923 \times 10^{3}$ | $6.1103 \times 10^{3}$ | $4.9399 \times 10^{3}$ | $8.4899 \times 10^{3}$ | $4.8578 \times 10^{3}$ | $1.8414 \times 10^{4}$ | **$4.4341 \times 10^{3}$** |
| | Std | $4.5135 \times 10^{2}$ | $7.6749 \times 10^{2}$ | $1.9806 \times 10^{3}$ | $6.9735 \times 10^{2}$ | $5.4675 \times 10^{2}$ | $4.0803 \times 10^{2}$ | $1.3128 \times 10^{3}$ | **$3.9769 \times 10^{2}$** | $2.3833 \times 10^{3}$ | $5.2834 \times 10^{2}$ |
| F30 | Ave | $1.5602 \times 10^{8}$ | $1.2944 \times 10^{8}$ | $3.7291 \times 10^{8}$ | $5.2542 \times 10^{7}$ | $6.3072 \times 10^{7}$ | $3.9845 \times 10^{6}$ | $2.4310 \times 10^{8}$ | $2.4906 \times 10^{7}$ | $3.2496 \times 10^{9}$ | **$1.4685 \times 10^{6}$** |
| | Std | $6.5572 \times 10^{7}$ | $4.7254 \times 10^{7}$ | $1.4798 \times 10^{8}$ | $4.0679 \times 10^{7}$ | $1.9574 \times 10^{7}$ | $2.3028 \times 10^{6}$ | $1.2129 \times 10^{8}$ | $1.3336 \times 10^{7}$ | $8.1141 \times 10^{8}$ | **$5.7457 \times 10^{5}$** |

**Table 4.** Experimental results of CEC2017 (dim=100).

| Function | Metric | GWO | HHO | WOA | DBO | BPBO | ALA | SGA | HBO | CDDO | MECDDO |
|---|---|---|---|---|---|---|---|---|---|---|---|
| F1 | Ave | $5.7040 \times 10^{10}$ | $4.8299 \times 10^{10}$ | $1.1027 \times 10^{11}$ | $8.4887 \times 10^{10}$ | $1.2774 \times 10^{10}$ | $2.2202 \times 10^{10}$ | $1.1389 \times 10^{11}$ | $8.8172 \times 10^{10}$ | $2.3706 \times 10^{11}$ | **$3.9066 \times 10^{9}$** |
| | Std | $1.1772 \times 10^{10}$ | $6.4559 \times 10^{9}$ | $9.6275 \times 10^{9}$ | $6.5992 \times 10^{10}$ | $2.4771 \times 10^{9}$ | $5.0254 \times 10^{9}$ | $1.5463 \times 10^{10}$ | $1.2937 \times 10^{10}$ | $7.7224 \times 10^{9}$ | **$2.2139 \times 10^{9}$** |
| F2 | Ave | $4.3435 \times 10^{136}$ | $1.9639 \times 10^{155}$ | $1.1553 \times 10^{178}$ | $1.1504 \times 10^{157}$ | $9.7684 \times 10^{144}$ | $6.6357 \times 10^{136}$ | $3.0991 \times 10^{166}$ | $4.2203 \times 10^{148}$ | $2.0662 \times 10^{166}$ | **$1.1137 \times 10^{119}$** |
| | Std | $2.3785 \times 10^{137}$ | **$6.5535 \times 10^{4}$** | **$6.5535 \times 10^{4}$** | **$6.5535 \times 10^{4}$** | $5.0011 \times 10^{145}$ | $3.6207 \times 10^{137}$ | **$6.5535 \times 10^{4}$** | $2.3116 \times 10^{149}$ | **$6.5535 \times 10^{4}$** | $6.0985 \times 10^{119}$ |
| F3 | Ave | $5.4384 \times 10^{5}$ | $3.5033 \times 10^{5}$ | $9.1439 \times 10^{5}$ | $6.1044 \times 10^{5}$ | $3.3923 \times 10^{5}$ | **$3.1037 \times 10^{5}$** | $3.9690 \times 10^{5}$ | $3.7556 \times 10^{5}$ | $3.7216 \times 10^{5}$ | $5.9311 \times 10^{5}$ |
| | Std | $8.4989 \times 10^{4}$ | $3.9129 \times 10^{4}$ | $1.4411 \times 10^{5}$ | $2.8836 \times 10^{5}$ | **$2.0002 \times 10^{4}$** | $3.5665 \times 10^{4}$ | $8.1813 \times 10^{4}$ | $3.6023 \times 10^{4}$ | $3.5354 \times 10^{4}$ | $8.6130 \times 10^{4}$ |
| F4 | Ave | $6.1578 \times 10^{3}$ | $9.2368 \times 10^{3}$ | $2.1047 \times 10^{4}$ | $1.4590 \times 10^{4}$ | $2.4771 \times 10^{3}$ | $3.2130 \times 10^{3}$ | $2.0012 \times 10^{4}$ | $9.7931 \times 10^{3}$ | $7.5721 \times 10^{4}$ | **$1.3155 \times 10^{3}$** |
| | Std | $1.6997 \times 10^{3}$ | $1.2242 \times 10^{3}$ | $3.7627 \times 10^{3}$ | $1.5608 \times 10^{4}$ | $3.5896 \times 10^{2}$ | $7.1609 \times 10^{2}$ | $4.1364 \times 10^{3}$ | $2.1755 \times 10^{3}$ | $5.5117 \times 10^{3}$ | **$2.0289 \times 10^{2}$** |
| F5 | Ave | **$1.2416 \times 10^{3}$** | $1.6655 \times 10^{3}$ | $1.9860 \times 10^{3}$ | $1.7791 \times 10^{3}$ | $1.5578 \times 10^{3}$ | $1.4219 \times 10^{3}$ | $1.8306 \times 10^{3}$ | $1.7082 \times 10^{3}$ | $2.0882 \times 10^{3}$ | $1.3456 \times 10^{3}$ |
| | Std | $5.0770 \times 10^{1}$ | $4.3395 \times 10^{1}$ | $1.2310 \times 10^{2}$ | $2.2206 \times 10^{2}$ | $5.2484 \times 10^{1}$ | $9.9356 \times 10^{1}$ | $8.5028 \times 10^{1}$ | $8.0812 \times 10^{1}$ | **$3.3687 \times 10^{1}$** | $1.9389 \times 10^{2}$ |
| F6 | Ave | **$6.4612 \times 10^{2}$** | $6.9024 \times 10^{2}$ | $7.0689 \times 10^{2}$ | $6.7785 \times 10^{2}$ | $6.8185 \times 10^{2}$ | $6.5887 \times 10^{2}$ | $6.9613 \times 10^{2}$ | $6.7407 \times 10^{2}$ | $7.1198 \times 10^{2}$ | $6.5197 \times 10^{2}$ |
| | Std | $4.4002 \times 10^{0}$ | $3.3686 \times 10^{0}$ | $7.4467 \times 10^{0}$ | $1.2467 \times 10^{1}$ | $6.1671 \times 10^{0}$ | $7.9905 \times 10^{0}$ | $4.5783 \times 10^{0}$ | $6.7100 \times 10^{0}$ | **$3.1735 \times 10^{0}$** | $8.8692 \times 10^{0}$ |
| F7 | Ave | **$2.1497 \times 10^{3}$** | $3.7567 \times 10^{3}$ | $3.8624 \times 10^{3}$ | $2.9586 \times 10^{3}$ | $3.4329 \times 10^{3}$ | $2.5262 \times 10^{3}$ | $3.6278 \times 10^{3}$ | $3.1479 \times 10^{3}$ | $3.9223 \times 10^{3}$ | $2.8201 \times 10^{3}$ |
| | Std | $1.4594 \times 10^{2}$ | $9.8883 \times 10^{1}$ | $1.5837 \times 10^{2}$ | $2.9121 \times 10^{2}$ | $1.2901 \times 10^{2}$ | $1.9577 \times 10^{2}$ | $1.8654 \times 10^{2}$ | $1.8329 \times 10^{2}$ | **$9.2464 \times 10^{1}$** | $2.4790 \times 10^{2}$ |
| F8 | Ave | **$1.5575 \times 10^{3}$** | $2.1335 \times 10^{3}$ | $2.3952 \times 10^{3}$ | $2.1810 \times 10^{3}$ | $2.0203 \times 10^{3}$ | $1.7296 \times 10^{3}$ | $2.2857 \times 10^{3}$ | $2.0693 \times 10^{3}$ | $2.5616 \times 10^{3}$ | $1.6848 \times 10^{3}$ |
| | Std | $1.4147 \times 10^{2}$ | $4.7974 \times 10^{1}$ | $1.3126 \times 10^{2}$ | $1.9531 \times 10^{2}$ | $7.5826 \times 10^{1}$ | $1.0222 \times 10^{2}$ | $8.0433 \times 10^{1}$ | $7.3279 \times 10^{1}$ | **$3.7835 \times 10^{1}$** | $2.1987 \times 10^{2}$ |
| F9 | Ave | $5.1760 \times 10^{4}$ | $6.9746 \times 10^{4}$ | $7.7324 \times 10^{4}$ | $7.7567 \times 10^{4}$ | $5.4734 \times 10^{4}$ | **$4.6516 \times 10^{4}$** | $5.8451 \times 10^{4}$ | $5.4164 \times 10^{4}$ | $8.1580 \times 10^{4}$ | $5.7968 \times 10^{4}$ |
| | Std | $1.2037 \times 10^{4}$ | $4.9452 \times 10^{3}$ | $1.3650 \times 10^{4}$ | $9.5781 \times 10^{3}$ | $1.2761 \times 10^{4}$ | $1.3257 \times 10^{4}$ | $5.1781 \times 10^{3}$ | $6.7642 \times 10^{3}$ | **$3.8190 \times 10^{3}$** | $1.2374 \times 10^{4}$ |
| F10 | Ave | **$2.0765 \times 10^{4}$** | $2.4490 \times 10^{4}$ | $2.9364 \times 10^{4}$ | $3.0473 \times 10^{4}$ | $2.4067 \times 10^{4}$ | $2.7883 \times 10^{4}$ | $2.7056 \times 10^{4}$ | $2.4126 \times 10^{4}$ | $3.2529 \times 10^{4}$ | $3.2700 \times 10^{4}$ |
| | Std | $5.8573 \times 10^{3}$ | $2.2446 \times 10^{3}$ | $1.4057 \times 10^{3}$ | $3.1779 \times 10^{3}$ | $4.0917 \times 10^{3}$ | $2.1667 \times 10^{3}$ | $1.9813 \times 10^{3}$ | $1.2282 \times 10^{3}$ | $8.3947 \times 10^{2}$ | **$8.3262 \times 10^{2}$** |
| F11 | Ave | $8.9270 \times 10^{4}$ | $1.5430 \times 10^{5}$ | $2.6553 \times 10^{5}$ | $2.2431 \times 10^{5}$ | $8.3310 \times 10^{4}$ | $7.2779 \times 10^{4}$ | $1.3226 \times 10^{5}$ | $1.2479 \times 10^{5}$ | $2.6994 \times 10^{5}$ | **$2.3343 \times 10^{4}$** |
| | Std | $1.8012 \times 10^{4}$ | $3.0366 \times 10^{4}$ | $1.0129 \times 10^{5}$ | $5.0931 \times 10^{4}$ | $1.7664 \times 10^{4}$ | $2.0733 \times 10^{4}$ | $2.4436 \times 10^{4}$ | $2.5441 \times 10^{4}$ | $7.2545 \times 10^{4}$ | **$1.1150 \times 10^{4}$** |
| F12 | Ave | $1.3755 \times 10^{10}$ | $1.1215 \times 10^{10}$ | $3.1098 \times 10^{10}$ | $7.0947 \times 10^{9}$ | $1.6116 \times 10^{9}$ | $1.5093 \times 10^{9}$ | $2.8883 \times 10^{10}$ | $1.2229 \times 10^{10}$ | $1.5884 \times 10^{11}$ | **$1.6416 \times 10^{8}$** |
| | Std | $6.9453 \times 10^{9}$ | $4.2111 \times 10^{9}$ | $7.5377 \times 10^{9}$ | $2.5221 \times 10^{9}$ | $5.0882 \times 10^{8}$ | $5.7910 \times 10^{8}$ | $6.3237 \times 10^{9}$ | $3.3899 \times 10^{9}$ | $1.2374 \times 10^{10}$ | **$1.1645 \times 10^{8}$** |
| F13 | Ave | $1.4568 \times 10^{9}$ | $3.3346 \times 10^{8}$ | $3.0589 \times 10^{9}$ | $3.3608 \times 10^{8}$ | $9.6709 \times 10^{6}$ | $3.3356 \times 10^{6}$ | $1.6708 \times 10^{9}$ | $7.4827 \times 10^{8}$ | $3.4573 \times 10^{10}$ | **$6.4756 \times 10^{4}$** |
| | Std | $1.3609 \times 10^{9}$ | $2.6698 \times 10^{8}$ | $1.3255 \times 10^{9}$ | $2.7639 \times 10^{8}$ | $4.8806 \times 10^{6}$ | $3.2708 \times 10^{6}$ | $1.7567 \times 10^{9}$ | $2.9919 \times 10^{8}$ | $3.9226 \times 10^{9}$ | **$7.5624 \times 10^{4}$** |
| F14 | Ave | $9.4231 \times 10^{6}$ | $1.0139 \times 10^{7}$ | $2.2456 \times 10^{7}$ | $1.8972 \times 10^{7}$ | $5.9333 \times 10^{6}$ | $2.8565 \times 10^{6}$ | $1.8111 \times 10^{7}$ | $1.6644 \times 10^{7}$ | $2.8286 \times 10^{7}$ | **$2.5560 \times 10^{5}$** |
| | Std | $5.6649 \times 10^{6}$ | $4.5098 \times 10^{6}$ | $1.2360 \times 10^{7}$ | $1.5619 \times 10^{7}$ | $2.3933 \times 10^{6}$ | $1.8725 \times 10^{6}$ | $1.0912 \times 10^{7}$ | $6.5066 \times 10^{6}$ | $4.3129 \times 10^{6}$ | **$2.4608 \times 10^{5}$** |
| F15 | Ave | $3.4736 \times 10^{8}$ | $1.8631 \times 10^{7}$ | $4.8946 \times 10^{8}$ | $6.7565 \times 10^{7}$ | $7.4953 \times 10^{5}$ | $6.8849 \times 10^{5}$ | $2.4504 \times 10^{8}$ | $9.3443 \times 10^{7}$ | $1.5005 \times 10^{10}$ | **$1.8041 \times 10^{4}$** |
| | Std | $4.5077 \times 10^{8}$ | $1.4582 \times 10^{7}$ | $2.4820 \times 10^{8}$ | $6.9443 \times 10^{7}$ | $6.4535 \times 10^{5}$ | $1.8892 \times 10^{6}$ | $1.9500 \times 10^{8}$ | $5.2515 \times 10^{7}$ | $2.2957 \times 10^{9}$ | **$1.1803 \times 10^{4}$** |
| F16 | Ave | **$6.5591 \times 10^{3}$** | $1.0089 \times 10^{4}$ | $1.6226 \times 10^{4}$ | $9.3011 \times 10^{3}$ | $8.4097 \times 10^{3}$ | $8.0226 \times 10^{3}$ | $1.3396 \times 10^{4}$ | $9.3652 \times 10^{3}$ | $1.9001 \times 10^{4}$ | $7.2185 \times 10^{3}$ |
| | Std | **$6.9262 \times 10^{2}$** | $1.1037 \times 10^{3}$ | $2.3603 \times 10^{3}$ | $1.2663 \times 10^{3}$ | $1.1221 \times 10^{3}$ | $9.7174 \times 10^{2}$ | $2.1947 \times 10^{3}$ | $9.0492 \times 10^{2}$ | $1.1955 \times 10^{3}$ | $2.2335 \times 10^{3}$ |
| F17 | Ave | $5.7086 \times 10^{3}$ | $8.3049 \times 10^{3}$ | $3.3071 \times 10^{4}$ | $9.2045 \times 10^{3}$ | $6.4404 \times 10^{3}$ | $6.1942 \times 10^{3}$ | $1.5261 \times 10^{4}$ | $8.0035 \times 10^{3}$ | $7.9104 \times 10^{5}$ | **$5.7007 \times 10^{3}$** |
| | Std | $8.4173 \times 10^{2}$ | $1.3835 \times 10^{3}$ | $2.6279 \times 10^{4}$ | $1.4580 \times 10^{3}$ | $6.9338 \times 10^{2}$ | **$5.8913 \times 10^{2}$** | $9.6354 \times 10^{3}$ | $7.9895 \times 10^{2}$ | $4.9180 \times 10^{5}$ | $1.3304 \times 10^{3}$ |
| F18 | Ave | $8.8921 \times 10^{6}$ | $9.4368 \times 10^{6}$ | $2.0199 \times 10^{7}$ | $2.7045 \times 10^{7}$ | $4.6516 \times 10^{6}$ | $4.5248 \times 10^{6}$ | $1.9172 \times 10^{7}$ | $2.0375 \times 10^{7}$ | $8.8624 \times 10^{7}$ | **$9.4780 \times 10^{5}$** |
| | Std | $6.7183 \times 10^{6}$ | $4.1232 \times 10^{6}$ | $1.1157 \times 10^{7}$ | $1.9287 \times 10^{7}$ | $1.7299 \times 10^{6}$ | $2.3909 \times 10^{6}$ | $1.0487 \times 10^{7}$ | $8.8515 \times 10^{6}$ | $2.7961 \times 10^{7}$ | **$6.8640 \times 10^{5}$** |
| F19 | Ave | $2.4536 \times 10^{8}$ | $3.3437 \times 10^{7}$ | $5.3697 \times 10^{8}$ | $8.7279 \times 10^{7}$ | $5.7667 \times 10^{6}$ | $1.2008 \times 10^{6}$ | $2.2772 \times 10^{8}$ | $1.1738 \times 10^{8}$ | $1.2712 \times 10^{10}$ | **$2.4733 \times 10^{4}$** |
| | Std | $2.6772 \times 10^{8}$ | $1.7276 \times 10^{7}$ | $3.8490 \times 10^{8}$ | $9.5539 \times 10^{7}$ | $5.1361 \times 10^{6}$ | $3.8876 \times 10^{6}$ | $2.0082 \times 10^{8}$ | $7.0463 \times 10^{7}$ | $2.1813 \times 10^{9}$ | **$2.0121 \times 10^{4}$** |
| F20 | Ave | **$5.4728 \times 10^{3}$** | $6.1555 \times 10^{3}$ | $7.0938 \times 10^{3}$ | $7.3771 \times 10^{3}$ | $6.1517 \times 10^{3}$ | $6.5977 \times 10^{3}$ | $6.5121 \times 10^{3}$ | $6.1404 \times 10^{3}$ | $7.7982 \times 10^{3}$ | $7.4823 \times 10^{3}$ |
| | Std | $9.7510 \times 10^{2}$ | $5.7229 \times 10^{2}$ | $4.8008 \times 10^{2}$ | $6.1927 \times 10^{2}$ | $6.5695 \times 10^{2}$ | $5.6113 \times 10^{2}$ | $6.1230 \times 10^{2}$ | $5.6014 \times 10^{2}$ | **$4.2861 \times 10^{2}$** | $1.0003 \times 10^{3}$ |
| F21 | Ave | **$3.0920 \times 10^{3}$** | $4.3536 \times 10^{3}$ | $4.5021 \times 10^{3}$ | $4.0250 \times 10^{3}$ | $3.5931 \times 10^{3}$ | $3.3055 \times 10^{3}$ | $4.4510 \times 10^{3}$ | $3.7052 \times 10^{3}$ | $4.7828 \times 10^{3}$ | $3.2890 \times 10^{3}$ |
| | Std | $1.5098 \times 10^{2}$ | $1.5689 \times 10^{2}$ | $1.8666 \times 10^{2}$ | $1.2920 \times 10^{2}$ | $1.3840 \times 10^{2}$ | $1.2340 \times 10^{2}$ | $1.9747 \times 10^{2}$ | **$8.9436 \times 10^{1}$** | $2.0559 \times 10^{2}$ | $2.1841 \times 10^{2}$ |
| F22 | Ave | **$2.3374 \times 10^{4}$** | $2.7384 \times 10^{4}$ | $3.2601 \times 10^{4}$ | $2.8303 \times 10^{4}$ | $2.7575 \times 10^{4}$ | $3.0069 \times 10^{4}$ | $2.9507 \times 10^{4}$ | $2.6113 \times 10^{4}$ | $3.4565 \times 10^{4}$ | $3.4431 \times 10^{4}$ |
| | Std | $5.4229 \times 10^{3}$ | $1.6735 \times 10^{3}$ | $1.6446 \times 10^{3}$ | $4.4523 \times 10^{3}$ | $3.5445 \times 10^{3}$ | $2.3128 \times 10^{3}$ | $1.1895 \times 10^{3}$ | $2.5001 \times 10^{3}$ | **$8.4030 \times 10^{2}$** | $1.6826 \times 10^{3}$ |
| F23 | Ave | **$3.7294 \times 10^{3}$** | $5.8032 \times 10^{3}$ | $5.3476 \times 10^{3}$ | $4.8729 \times 10^{3}$ | $4.2496 \times 10^{3}$ | $3.7978 \times 10^{3}$ | $5.5471 \times 10^{3}$ | $4.0442 \times 10^{3}$ | $7.6883 \times 10^{3}$ | $3.7539 \times 10^{3}$ |
| | Std | **$1.2162 \times 10^{2}$** | $4.2828 \times 10^{2}$ | $2.6746 \times 10^{2}$ | $2.3086 \times 10^{2}$ | $2.3805 \times 10^{2}$ | $1.2362 \times 10^{2}$ | $3.5774 \times 10^{2}$ | $1.4484 \times 10^{2}$ | $6.2515 \times 10^{2}$ | $1.3104 \times 10^{2}$ |
| F24 | Ave | **$4.5035 \times 10^{3}$** | $8.5368 \times 10^{3}$ | $6.8103 \times 10^{3}$ | $6.0757 \times 10^{3}$ | $5.1379 \times 10^{3}$ | $4.5608 \times 10^{3}$ | $7.2204 \times 10^{3}$ | $4.8156 \times 10^{3}$ | $1.1066 \times 10^{4}$ | $4.5825 \times 10^{3}$ |
| | Std | **$1.1950 \times 10^{2}$** | $8.5337 \times 10^{2}$ | $4.7379 \times 10^{2}$ | $4.8905 \times 10^{2}$ | $3.3974 \times 10^{2}$ | $2.5518 \times 10^{2}$ | $6.6796 \times 10^{2}$ | $1.4261 \times 10^{2}$ | $1.2959 \times 10^{3}$ | $2.2799 \times 10^{2}$ |
| F25 | Ave | $7.1133 \times 10^{3}$ | $6.7409 \times 10^{3}$ | $1.0903 \times 10^{4}$ | $9.9105 \times 10^{3}$ | $9.4984 \times 10^{3}$ | $5.8376 \times 10^{3}$ | $1.1062 \times 10^{4}$ | $9.8190 \times 10^{3}$ | $2.2650 \times 10^{4}$ | **$3.9532 \times 10^{3}$** |
| | Std | $1.1639 \times 10^{3}$ | $6.4806 \times 10^{2}$ | $1.0164 \times 10^{3}$ | $6.4884 \times 10^{3}$ | $3.4697 \times 10^{2}$ | $7.4519 \times 10^{2}$ | $1.3743 \times 10^{3}$ | $1.3189 \times 10^{3}$ | $1.1187 \times 10^{3}$ | **$1.9995 \times 10^{2}$** |
| F26 | Ave | $1.7869 \times 10^{4}$ | $3.2250 \times 10^{4}$ | $3.7681 \times 10^{4}$ | $2.5034 \times 10^{4}$ | $2.9760 \times 10^{4}$ | **$1.7304 \times 10^{4}$** | $4.0214 \times 10^{4}$ | $2.5530 \times 10^{4}$ | $4.5911 \times 10^{4}$ | $1.7463 \times 10^{4}$ |
| | Std | $1.7284 \times 10^{3}$ | $2.7110 \times 10^{3}$ | $3.5825 \times 10^{3}$ | $3.4136 \times 10^{3}$ | $2.8989 \times 10^{3}$ | $1.9505 \times 10^{3}$ | $3.1077 \times 10^{3}$ | $3.3301 \times 10^{3}$ | **$1.4408 \times 10^{3}$** | $2.5130 \times 10^{3}$ |
| F27 | Ave | $4.3672 \times 10^{3}$ | $6.9730 \times 10^{3}$ | $6.1925 \times 10^{3}$ | $4.5961 \times 10^{3}$ | $4.9970 \times 10^{3}$ | $3.8121 \times 10^{3}$ | $5.4086 \times 10^{3}$ | $4.1393 \times 10^{3}$ | $1.1975 \times 10^{4}$ | **$3.6546 \times 10^{3}$** |
| | Std | $2.2335 \times 10^{2}$ | $1.0988 \times 10^{3}$ | $7.8237 \times 10^{2}$ | $6.0887 \times 10^{2}$ | $4.6225 \times 10^{2}$ | $1.9738 \times 10^{2}$ | $7.1129 \times 10^{2}$ | **$1.7091 \times 10^{2}$** | $9.4548 \times 10^{2}$ | $1.7366 \times 10^{2}$ |
| F28 | Ave | $9.4116 \times 10^{3}$ | $9.2859 \times 10^{3}$ | $1.4890 \times 10^{4}$ | $1.7891 \times 10^{4}$ | $6.0838 \times 10^{3}$ | $9.2569 \times 10^{3}$ | $1.5338 \times 10^{4}$ | $1.2581 \times 10^{4}$ | $2.5667 \times 10^{4}$ | **$6.0150 \times 10^{3}$** |
| | Std | $1.3143 \times 10^{3}$ | $8.4722 \times 10^{2}$ | $1.1911 \times 10^{3}$ | $6.3886 \times 10^{3}$ | **$4.8880 \times 10^{2}$** | $3.7930 \times 10^{3}$ | $1.3507 \times 10^{3}$ | $1.1450 \times 10^{3}$ | $7.5539 \times 10^{2}$ | $2.8250 \times 10^{3}$ |
| F29 | Ave | $9.4361 \times 10^{3}$ | $1.3040 \times 10^{4}$ | $2.0131 \times 10^{4}$ | $1.4089 \times 10^{4}$ | $1.1455 \times 10^{4}$ | $8.6197 \times 10^{3}$ | $1.8154 \times 10^{4}$ | $1.0693 \times 10^{4}$ | $1.5067 \times 10^{5}$ | **$7.2435 \times 10^{3}$** |
| | Std | $1.1928 \times 10^{3}$ | $9.5891 \times 10^{2}$ | $2.4687 \times 10^{3}$ | $1.1088 \times 10^{4}$ | $9.2852 \times 10^{2}$ | **$5.7418 \times 10^{2}$** | $3.2323 \times 10^{3}$ | $1.1400 \times 10^{3}$ | $7.0139 \times 10^{4}$ | $1.1652 \times 10^{3}$ |
| F30 | Ave | $1.7748 \times 10^{9}$ | $7.4491 \times 10^{8}$ | $3.1803 \times 10^{9}$ | $2.7962 \times 10^{8}$ | $1.3624 \times 10^{8}$ | $6.4877 \times 10^{6}$ | $1.8486 \times 10^{9}$ | $3.7076 \times 10^{8}$ | $3.0463 \times 10^{10}$ | **$1.3492 \times 10^{6}$** |
| | Std | $1.3744 \times 10^{9}$ | $4.3978 \times 10^{8}$ | $1.1164 \times 10^{9}$ | $1.7586 \times 10^{8}$ | $6.9742 \times 10^{7}$ | $4.7120 \times 10^{6}$ | $8.9440 \times 10^{8}$ | $1.2341 \times 10^{8}$ | $3.4749 \times 10^{9}$ | **$2.0996 \times 10^{6}$** |

Table 2 presents the performance of 10 algorithms on the 30-dimensional CEC2017 test set, spanning unimodal functions (F1–F7), multimodal functions (F8–F13), hybrid functions (F14–F20), and composition functions (F21–F30). MECDDO demonstrates significant precision advantages and stability across all categories. For unimodal functions, F1 (Sphere function), which is a typical function with a unique global optimum and no local extrema, primarily tests the algorithm's local exploitation capability. The mean fitness value of MECDDO ($7.6981 \times 10^3$) is 7 orders of magnitude lower than that of traditional CDDO ($3.6030 \times 10^{10}$) and 3 orders of magnitude lower than that of the second-best algorithm ALA ($2.8929 \times 10^6$). Moreover, its standard deviation ($8.4717 \times 10^3$) is only 1/40,000 of that of CDDO ($3.4076 \times 10^9$), indicating its ability to accurately approach the global optimum with minimal result fluctuation in low-dimensional unimodal scenarios. For the multimodal function F9 (Rastrigin function), which contains numerous local extrema and demands strong exploration capability from algorithms, the mean fitness value of MECDDO ($1.4105 \times 10^3$) is 8 times lower than that of CDDO ($1.1489 \times 10^4$) and 6 times lower than that of HHO ($8.5554 \times 10^3$). Its standard deviation ($8.8352 \times 10^2$) is also lower than all competing algorithms, demonstrating its effectiveness in escaping local extrema traps through an adaptive collaborative search strategy. In hybrid and composition functions, MECDDO's advantages are further highlighted. For F18 (Hybrid function 1), which combines complex characteristics of different function types, the mean fitness value of MECDDO ($2.0689 \times 10^3$) is 4 orders of magnitude lower than that of CDDO ($2.1310 \times 10^7$) and 2 orders of magnitude lower than that of BPBO ($5.1167 \times 10^5$). For the composition function F30 (Composition function 10), where the solution space involves multimodal and nonlinear superposition, traditional algorithms generally perform poorly. In contrast, the mean fitness value of MECDDO ($6.7748 \times 10^3$) is only 1/91,800 of that of CDDO ($6.2241 \times 10^8$), and its standard deviation ($2.1549 \times 10^3$) is significantly lower than algorithms such as WOA ($7.6390 \times 10^7$) and GWO ($6.5886 \times 10^6$). These results fully validate MECDDO's ability to balance exploration and exploitation in complex solution spaces.

The 50-dimensional test results in Table 3 show that as the dimensionality increases, most comparative algorithms exhibit significant performance degradation, while MECDDO maintains a stable advantage. In the unimodal function F4 (Schwefel 2.22 function), the average fitness of GWO increases from $6.3954 \times 10^2$ in 30 dimensions to $1.6522 \times 10^3$ (a 1.58-fold increase), and WOA rises from $1.2910 \times 10^3$ to $5.1401 \times 10^3$ (a 3-fold increase). In contrast, MECDDO increases only from $4.8724 \times 10^2$ to $5.9977 \times 10^2$ (a 23% increase), and its standard deviation ($4.1347 \times 10^1$) is just 1/37 of DBO ($1.5255 \times 10^3$), indicating stronger anti-degradation capability in local exploitation under medium-dimensional settings. The multimodal function F11 (Griewank function) imposes higher requirements on global exploration ability. CDDO's average fitness increases from $6.2712 \times 10^3$ to $2.0566 \times 10^4$ (2.28-fold growth), and HHO rises from $1.6293 \times 10^3$ to $3.1121 \times 10^3$ (91% growth). However, MECDDO increases only from $1.2138 \times 10^3$ to $1.3663 \times 10^3$ (12.6% growth), and its standard deviation ($9.2122 \times 10^1$) is just 1/30 of GWO ($2.7705 \times 10^3$), demonstrating that its multi-strategy adaptive selection mechanism can continuously and efficiently utilize population information in medium-dimensional scenarios, avoiding the decline in exploration efficiency. In the hybrid function F14 (Hybrid function 2), MECDDO's average fitness increases only slightly from $1.5054 \times 10^3$ (30D) to $1.7306 \times 10^3$ (15% growth), while CDDO increases from $2.1453 \times 10^6$ to $3.2584 \times 10^7$ (14.2× growth), and SGA increases from $5.3941 \times 10^5$ to $3.5832 \times 10^6$ (5.6× growth). For the composition function F25 (Composition function 5), MECDDO achieves an average fitness of $3.0648 \times 10^3$, which is 74% lower than CDDO ($1.1977 \times 10^4$), and its standard deviation ($2.9872 \times 10^1$) is significantly lower than all comparative algorithms. These results further highlight MECDDO's stability advantage in medium-dimensional complex optimization problems.

Table 4 presents the 100-dimensional test results, which serve as a critical benchmark for algorithmic performance. Most comparative algorithms experience a sharp decline in optimization accuracy due to the curse of dimensionality, while MECDDO achieves a performance breakthrough through the synergy of its three improvement strategies. For the unimodal function F2 (Rosenbrock function), which imposes extremely high requirements on local search accuracy, the average fitness of CDDO increases from $2.3889 \times 10^{41}$ in 30 dimensions to $2.0662 \times 10^{166}$ in 100 dimensions (an increase of $8.65 \times 10^{124}$ times). WOA rises from $7.3693 \times 10^{34}$ to $1.1553 \times 10^{178}$ (an increase of $1.57 \times 10^{143}$ times), whereas MECDDO increases from $2.6607 \times 10^{19}$ to $1.1137 \times 10^{119}$ (an increase of $4.19 \times 10^{99}$ times) and is the only algorithm whose fitness does not exceed $10^{120}$ on this function. This confirms that its global-optimality-guided boundary repair mechanism effectively reduces the loss of high-quality information in high-dimensional spaces. In the multimodal function F11 (Griewank function), MECDDO achieves an average fitness of $2.3343 \times 10^4$, which is 91.3% lower than CDDO ($2.6994 \times 10^5$) and 84.9% lower than HHO ($1.5430 \times 10^5$). Its standard deviation ($1.1150 \times 10^4$) is only one-ninth that of WOA ($1.0129 \times 10^5$), indicating that MECDDO can still efficiently explore the search space in high-dimensional multimodal environments. For the hybrid function F15 (Hybrid function 3), MECDDO's average fitness increases from $1.8823 \times 10^3$ (30-dimensional) to $1.8041 \times 10^4$ (8.6× growth), while CDDO increases from $1.0125 \times 10^8$ to $1.5005 \times 10^{10}$ (147× increase) and DBO increases from $5.6060 \times 10^4$ to $6.7565 \times 10^7$ (1205× increase). In the composition function F30 (Composition function 10), MECDDO achieves an average fitness of $1.3492 \times 10^6$, which is 99.995% lower than CDDO ($3.0463 \times 10^{10}$), and its standard deviation ($2.0996 \times 10^6$) is only 1/655 of GWO ($1.3744 \times 10^9$). These results fully demonstrate that MECDDO exhibits significantly superior global optimization capability in high-dimensional complex scenarios compared with conventional algorithms.

A horizontal comparison of Table 2 (30D), Table 3 (50D), and Table 4 (100D) shows that MECDDO exhibits a significantly lower performance degradation rate than the comparative algorithms, highlighting its dimensional adaptability advantage. For key test functions, in F1, the average fitness of MECDDO increases 481-fold from 30D to 100D, whereas CDDO, WOA, and GWO increase 658-fold, 505-fold, and 388-fold, respectively. In F9, MECDDO increases 41-fold, while CDDO, HHO, and SGA increase 3.4-fold, 8.1-fold, and 7.7-fold, respectively (although CDDO's growth factor is smaller, its absolute values remain far higher than MECDDO's). Regarding standard deviation, MECDDO maintains average values of $1.24 \times 10^3$, $2.87 \times 10^4$, and $1.12 \times 10^5$ across the three dimensionalities, which are far lower than those of CDDO ($1.21 \times 10^9$, $6.27 \times 10^9$, $3.47 \times 10^9$) and WOA ($1.69 \times 10^9$, $5.27 \times 10^9$, $1.12 \times 10^9$). This indicates that MECDDO retains stable optimization consistency as dimensionality increases. Such robustness is attributed to the synergy of its three improvement strategies: adaptive cooperative search expands the high-dimensional search breadth, multi-strategy adaptive selection balances exploration and exploitation, and the global-optimality-guided boundary repair reduces the loss of high-quality information. Together, these mechanisms endow MECDDO with strong dimensional adaptability.

The convergence curves in Figure 6 provide an intuitive comparison of algorithmic convergence efficiency, showing that MECDDO consistently achieves faster convergence across all test functions. In the 30-dimensional F1 function, MECDDO approaches the optimal value within 100 iterations, whereas CDDO, GWO, WOA, and other algorithms remain in a slow-decreasing stage. In higher-dimensional spaces, MECDDO's convergence advantage becomes even more pronounced: for the 100-dimensional F30 function, the fitness value drops by over 99% from its initial value after 500 iterations, significantly outperforming comparative algorithms, which achieve reductions of only 60–85%. This performance is attributed to MECDDO's multi-strategy adaptive selection mechanism,

which rapidly explores the solution space in the early stage and switches to exploitation strategies in the later stage to refine the optimal solution, thus dynamically balancing exploration and exploitation.

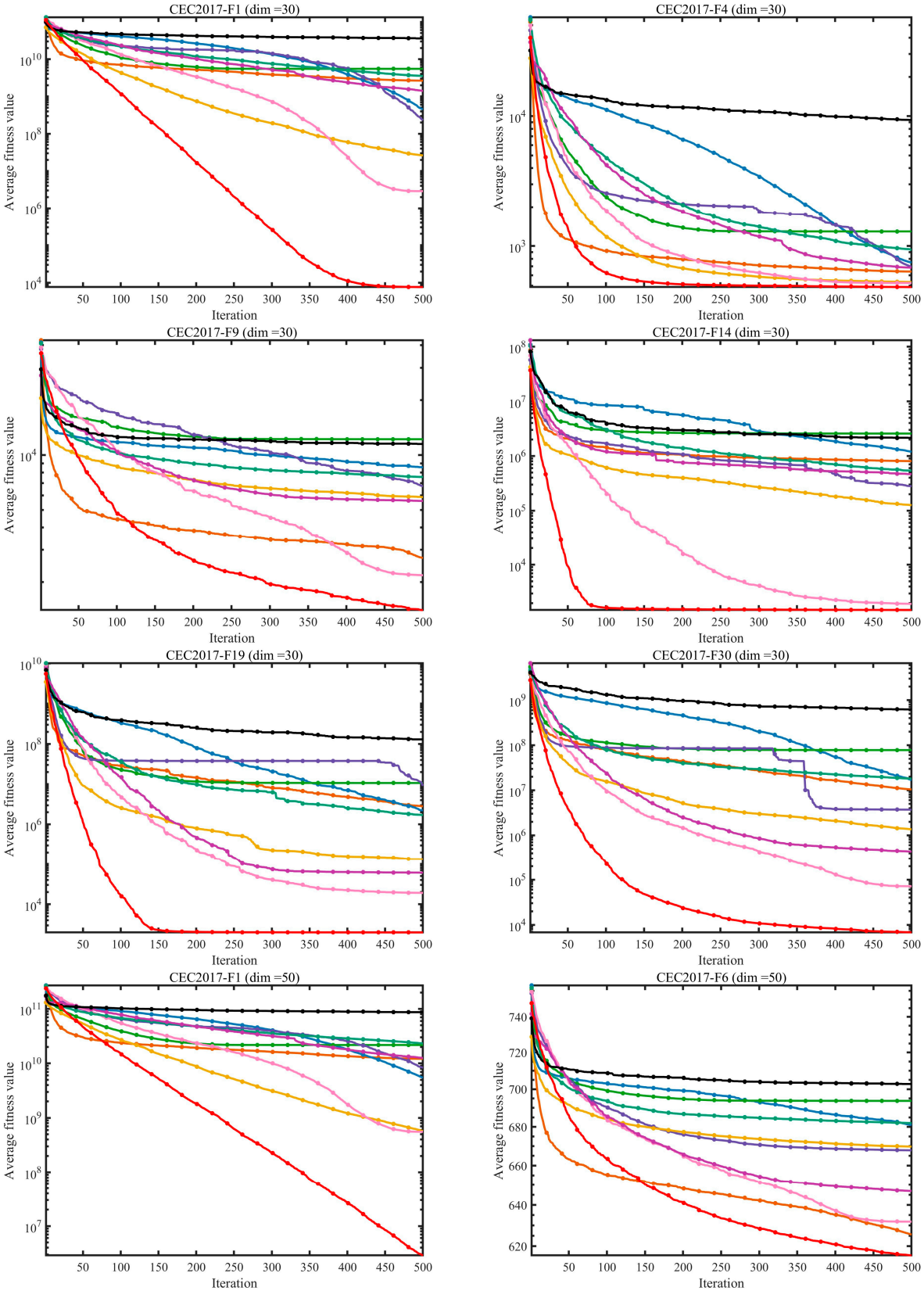


**Figure 6.** *Cont.*

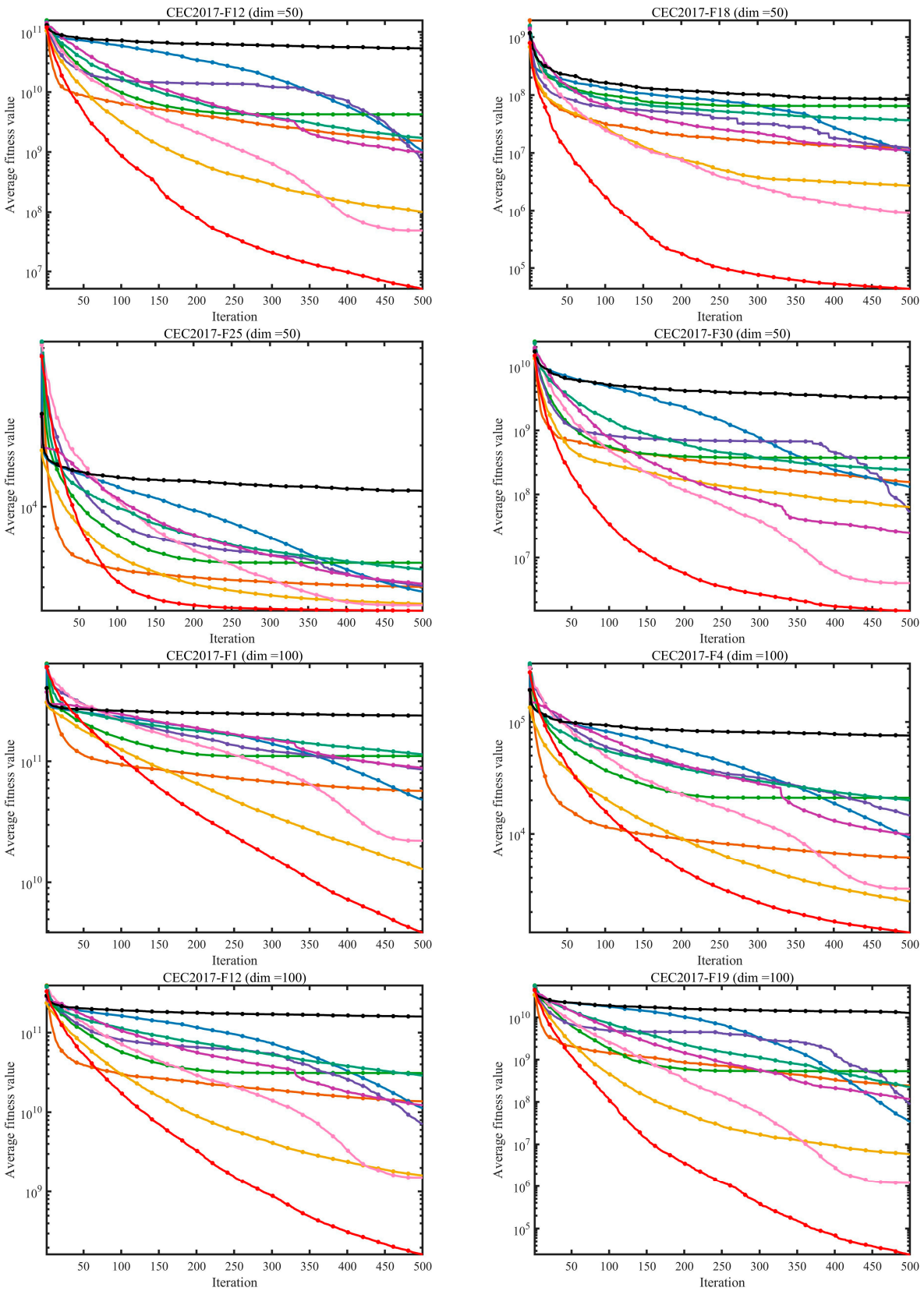


**Figure 6.** *Cont.*

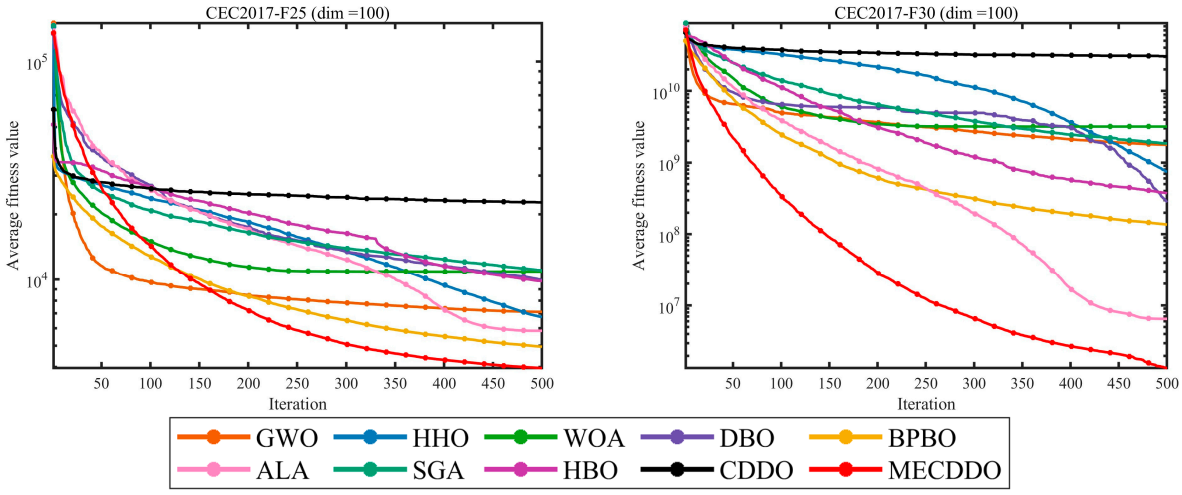


**Figure 6.** Convergence speed comparison among competing algorithms on CEC2017 benchmark test set.

The fitness value distributions in Figure 7 further corroborate MECDDO's superiority in optimization accuracy. For unimodal functions (e.g., F1, F6), multimodal functions (e.g., F9, F11), and hybrid functions (e.g., F23, F28), MECDDO's fitness values are more tightly concentrated near the optimal value, with the smallest dispersion. In the 30-dimensional F9 function, MECDDO's fitness values are concentrated around $1.4 \times 10^3$, whereas CDDO, WOA, and others are spread between $2.1 \times 10^3$ and $1.2 \times 10^4$. In the 50-dimensional F26 function, MECDDO's median fitness is over 30% lower than that of HBO, SGA, and other algorithms, demonstrating its ability to escape local optima and approach the global optimum in complex solution spaces.

### *3.6. Experimental Results and Analysis on CEC 2022 Benchmark*

To further validate the generalization performance of MECDDO across different benchmark test suites, supplementary experiments were conducted on the CEC2022 benchmark set with 20 dimensions. This benchmark consists of 12 functions with complex characteristics, covering unimodal, multimodal, hybrid, and other function types, which enables a more comprehensive evaluation of the algorithm's adaptability under diverse optimization scenarios. The experiments adopted the same parameter settings and evaluation criteria as those used in CEC2017. MECDDO was compared with nine mainstream algorithms, and the corresponding results are reported in Table 5, Figures 8 and 9.

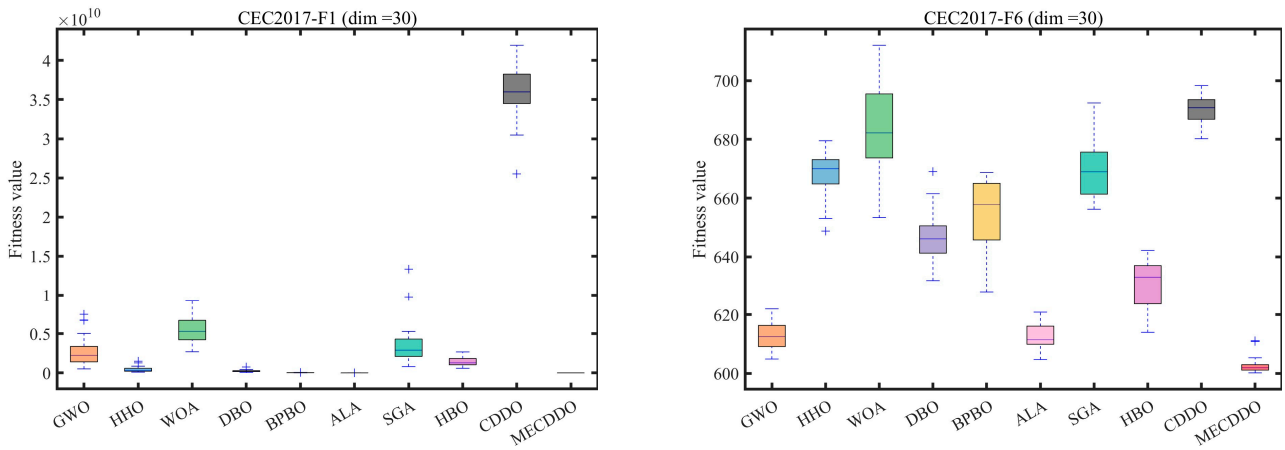


**Figure 7.** *Cont.*

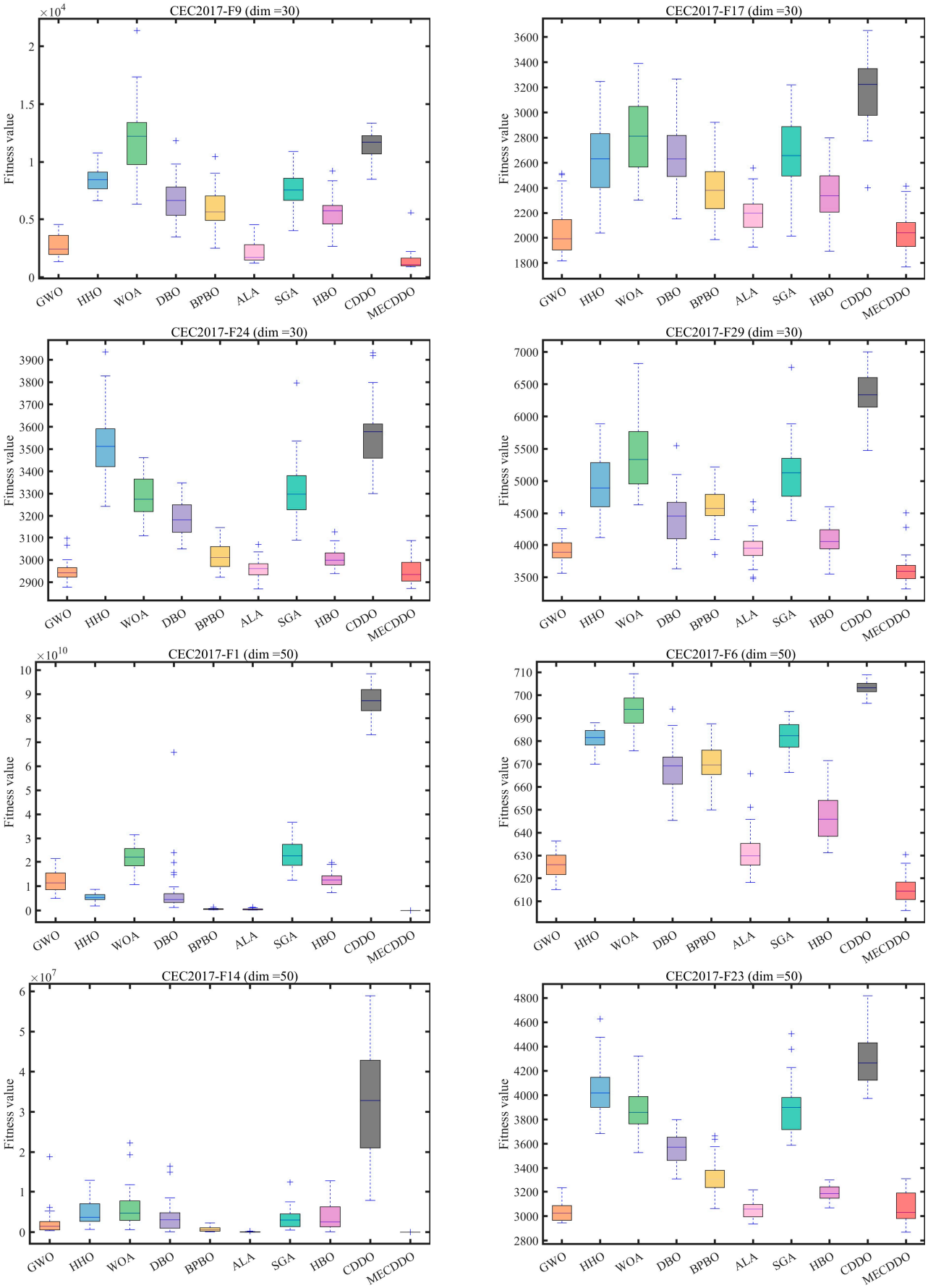


**Figure 7.** *Cont.*

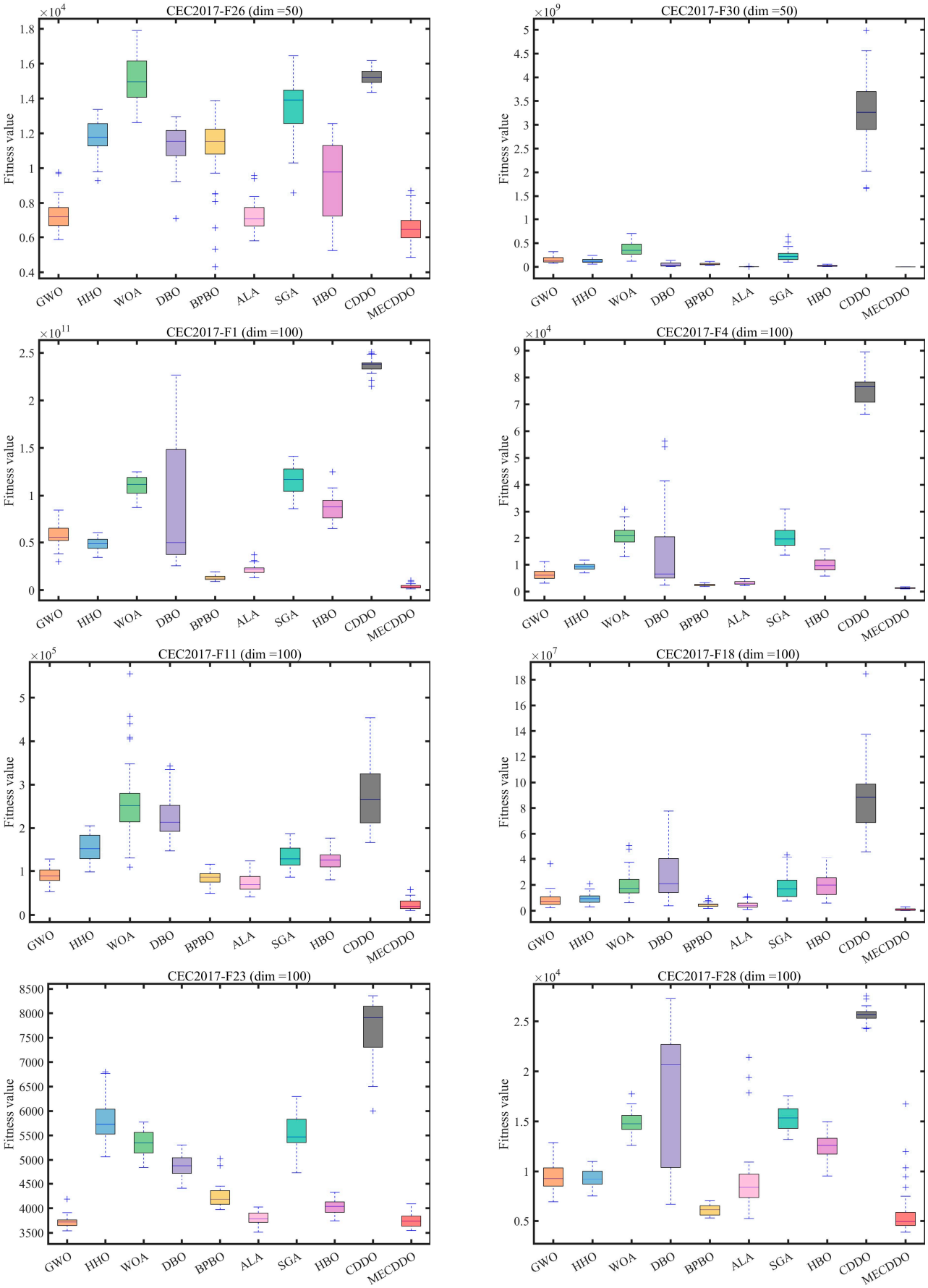


**Figure 7.** Statistical distribution analysis of all algorithms on CEC2017 benchmark set.

**Table 5.** Experimental results of CEC2022 (dim=20).

| Function | Metric | GWO | HHO | WOA | DBO | BPBO | ALA | SGA | HBO | CDDO | MECDDO |
|---|---|---|---|---|---|---|---|---|---|---|---|
| F1 | Ave | $1.4689 \times 10^{4}$ | $2.6226 \times 10^{4}$ | $3.4175 \times 10^{4}$ | $3.6966 \times 10^{4}$ | $1.0592 \times 10^{4}$ | $4.6858 \times 10^{3}$ | $1.7616 \times 10^{4}$ | $2.2552 \times 10^{4}$ | $3.9804 \times 10^{4}$ | $\mathbf{3.5209 \times 10^{2}}$ |
| | Std | $4.1993 \times 10^{3}$ | $8.8971 \times 10^{3}$ | $9.3465 \times 10^{3}$ | $9.5039 \times 10^{3}$ | $3.6213 \times 10^{3}$ | $6.3772 \times 10^{3}$ | $8.3482 \times 10^{3}$ | $7.9100 \times 10^{3}$ | $1.3060 \times 10^{4}$ | $\mathbf{7.3873 \times 10^{1}}$ |
| F2 | Ave | $5.1318 \times 10^{2}$ | $5.4285 \times 10^{2}$ | $6.2590 \times 10^{2}$ | $4.9316 \times 10^{2}$ | $4.8605 \times 10^{2}$ | $4.5695 \times 10^{2}$ | $5.6285 \times 10^{2}$ | $4.9014 \times 10^{2}$ | $1.3757 \times 10^{3}$ | $\mathbf{4.3544 \times 10^{2}}$ |
| | Std | $6.0001 \times 10^{1}$ | $6.0584 \times 10^{1}$ | $8.3146 \times 10^{1}$ | $3.8220 \times 10^{1}$ | $3.7929 \times 10^{1}$ | $2.5310 \times 10^{1}$ | $5.6914 \times 10^{1}$ | $4.1368 \times 10^{1}$ | $1.9574 \times 10^{2}$ | $\mathbf{2.0536 \times 10^{1}}$ |
| F3 | Ave | $6.0786 \times 10^{2}$ | $6.6482 \times 10^{2}$ | $6.6800 \times 10^{2}$ | $6.3734 \times 10^{2}$ | $6.4340 \times 10^{2}$ | $6.0530 \times 10^{2}$ | $6.6439 \times 10^{2}$ | $6.1729 \times 10^{2}$ | $6.8151 \times 10^{2}$ | $\mathbf{6.0038 \times 10^{2}}$ |
| | Std | $5.6020 \times 10^{0}$ | $8.6721 \times 10^{0}$ | $1.2417 \times 10^{1}$ | $1.1267 \times 10^{1}$ | $1.2883 \times 10^{1}$ | $4.5892 \times 10^{0}$ | $1.0605 \times 10^{1}$ | $1.0138 \times 10^{1}$ | $8.3798 \times 10^{0}$ | $\mathbf{8.1370 \times 10^{-1}}$ |
| F4 | Ave | $8.6198 \times 10^{2}$ | $8.8924 \times 10^{2}$ | $9.3386 \times 10^{2}$ | $9.0942 \times 10^{2}$ | $8.7315 \times 10^{2}$ | $\mathbf{8.5846 \times 10^{2}}$ | $8.9981 \times 10^{2}$ | $8.8937 \times 10^{2}$ | $9.6358 \times 10^{2}$ | $8.6941 \times 10^{2}$ |
| | Std | $3.3446 \times 10^{1}$ | $1.7581 \times 10^{1}$ | $2.9474 \times 10^{1}$ | $2.7334 \times 10^{1}$ | $1.4088 \times 10^{1}$ | $2.2698 \times 10^{1}$ | $1.6542 \times 10^{1}$ | $2.3461 \times 10^{1}$ | $\mathbf{1.2403 \times 10^{1}}$ | $4.1692 \times 10^{1}$ |
| F5 | Ave | $1.2215 \times 10^{3}$ | $2.9213 \times 10^{3}$ | $4.0321 \times 10^{3}$ | $2.2909 \times 10^{3}$ | $2.1663 \times 10^{3}$ | $1.1061 \times 10^{3}$ | $2.7269 \times 10^{3}$ | $2.3236 \times 10^{3}$ | $3.7449 \times 10^{3}$ | $\mathbf{9.0784 \times 10^{2}}$ |
| | Std | $3.0369 \times 10^{2}$ | $3.5575 \times 10^{2}$ | $1.3534 \times 10^{3}$ | $7.2116 \times 10^{2}$ | $5.0004 \times 10^{2}$ | $2.0267 \times 10^{2}$ | $4.2845 \times 10^{2}$ | $5.9444 \times 10^{2}$ | $3.3075 \times 10^{2}$ | $\mathbf{2.1937 \times 10^{1}}$ |
| F6 | Ave | $3.4521 \times 10^{6}$ | $2.4862 \times 10^{5}$ | $1.0018 \times 10^{7}$ | $7.0727 \times 10^{5}$ | $4.0140 \times 10^{3}$ | $1.7314 \times 10^{4}$ | $1.9721 \times 10^{4}$ | $3.1645 \times 10^{4}$ | $4.3134 \times 10^{8}$ | $\mathbf{3.4992 \times 10^{3}}$ |
| | Std | $1.1978 \times 10^{7}$ | $1.0006 \times 10^{5}$ | $1.4292 \times 10^{7}$ | $1.1898 \times 10^{6}$ | $\mathbf{2.1241 \times 10^{3}}$ | $9.0587 \times 10^{3}$ | $4.1409 \times 10^{4}$ | $4.6776 \times 10^{4}$ | $1.0880 \times 10^{8}$ | $5.8162 \times 10^{3}$ |
| F7 | Ave | $2.1098 \times 10^{3}$ | $2.2051 \times 10^{3}$ | $2.2193 \times 10^{3}$ | $2.1504 \times 10^{3}$ | $2.1242 \times 10^{3}$ | $2.0854 \times 10^{3}$ | $2.2015 \times 10^{3}$ | $2.1018 \times 10^{3}$ | $2.2263 \times 10^{3}$ | $\mathbf{2.0384 \times 10^{3}}$ |
| | Std | $5.1235 \times 10^{1}$ | $6.8826 \times 10^{1}$ | $5.5928 \times 10^{1}$ | $6.2223 \times 10^{1}$ | $3.8173 \times 10^{1}$ | $3.0153 \times 10^{1}$ | $3.2794 \times 10^{1}$ | $4.1418 \times 10^{1}$ | $3.5114 \times 10^{1}$ | $\mathbf{1.6389 \times 10^{1}}$ |
| F8 | Ave | $2.2707 \times 10^{3}$ | $2.3241 \times 10^{3}$ | $2.3161 \times 10^{3}$ | $2.2939 \times 10^{3}$ | $2.2663 \times 10^{3}$ | $\mathbf{2.2349 \times 10^{3}}$ | $2.3177 \times 10^{3}$ | $2.2553 \times 10^{3}$ | $2.2641 \times 10^{3}$ | $2.2359 \times 10^{3}$ |
| | Std | $5.6014 \times 10^{1}$ | $1.0592 \times 10^{2}$ | $7.9127 \times 10^{1}$ | $8.2163 \times 10^{1}$ | $6.1457 \times 10^{1}$ | $\mathbf{1.3220 \times 10^{1}}$ | $6.2793 \times 10^{1}$ | $4.1811 \times 10^{1}$ | $1.5104 \times 10^{1}$ | $3.2943 \times 10^{1}$ |
| F9 | Ave | $2.5218 \times 10^{3}$ | $2.5792 \times 10^{3}$ | $2.5888 \times 10^{3}$ | $2.5080 \times 10^{3}$ | $2.4856 \times 10^{3}$ | $2.4808 \times 10^{3}$ | $2.5963 \times 10^{3}$ | $2.4938 \times 10^{3}$ | $2.7579 \times 10^{3}$ | $\mathbf{2.4808 \times 10^{3}}$ |
| | Std | $2.7910 \times 10^{1}$ | $5.6165 \times 10^{1}$ | $4.5715 \times 10^{1}$ | $2.5786 \times 10^{1}$ | $4.8621 \times 10^{0}$ | 1.0093E-01 | $5.0794 \times 10^{1}$ | $1.3337 \times 10^{1}$ | $5.7722 \times 10^{1}$ | $\mathbf{4.0773 \times 10^{-3}}$ |
| F10 | Ave | $3.6757 \times 10^{3}$ | $4.2734 \times 10^{3}$ | $4.7705 \times 10^{3}$ | $\mathbf{2.9541 \times 10^{3}}$ | $3.3652 \times 10^{3}$ | $3.9349 \times 10^{3}$ | $4.5126 \times 10^{3}$ | $3.3116 \times 10^{3}$ | $4.7194 \times 10^{3}$ | $3.2544 \times 10^{3}$ |
| | Std | $1.1425 \times 10^{3}$ | $8.0021 \times 10^{2}$ | $1.3562 \times 10^{3}$ | $8.2494 \times 10^{2}$ | $1.1034 \times 10^{3}$ | $8.0113 \times 10^{2}$ | $1.2513 \times 10^{3}$ | $\mathbf{4.7208 \times 10^{2}}$ | $1.8754 \times 10^{3}$ | $8.4148 \times 10^{2}$ |
| F11 | Ave | $3.5486 \times 10^{3}$ | $3.4683 \times 10^{3}$ | $4.0301 \times 10^{3}$ | $3.1164 \times 10^{3}$ | $2.9531 \times 10^{3}$ | $3.0001 \times 10^{3}$ | $3.4167 \times 10^{3}$ | $3.2804 \times 10^{3}$ | $7.0783 \times 10^{3}$ | $\mathbf{2.9167 \times 10^{3}}$ |
| | Std | $4.0069 \times 10^{2}$ | $5.8916 \times 10^{2}$ | $8.1769 \times 10^{2}$ | $1.4884 \times 10^{2}$ | $7.6286 \times 10^{1}$ | $1.5277 \times 10^{2}$ | $2.3744 \times 10^{2}$ | $1.5309 \times 10^{2}$ | $4.8036 \times 10^{2}$ | $\mathbf{7.4675 \times 10^{1}}$ |
| F12 | Ave | $2.9854 \times 10^{3}$ | $3.2820 \times 10^{3}$ | $3.1027 \times 10^{3}$ | $3.0349 \times 10^{3}$ | $3.0165 \times 10^{3}$ | $2.9694 \times 10^{3}$ | $3.1238 \times 10^{3}$ | $2.9847 \times 10^{3}$ | $3.5108 \times 10^{3}$ | $\mathbf{2.9597 \times 10^{3}}$ |
| | Std | $2.7453 \times 10^{1}$ | $2.2570 \times 10^{2}$ | $1.0298 \times 10^{2}$ | $6.3812 \times 10^{1}$ | $4.9970 \times 10^{1}$ | $3.4626 \times 10^{1}$ | $1.8759 \times 10^{2}$ | $3.5255 \times 10^{1}$ | $1.6746 \times 10^{2}$ | $\mathbf{2.1867 \times 10^{1}}$ |

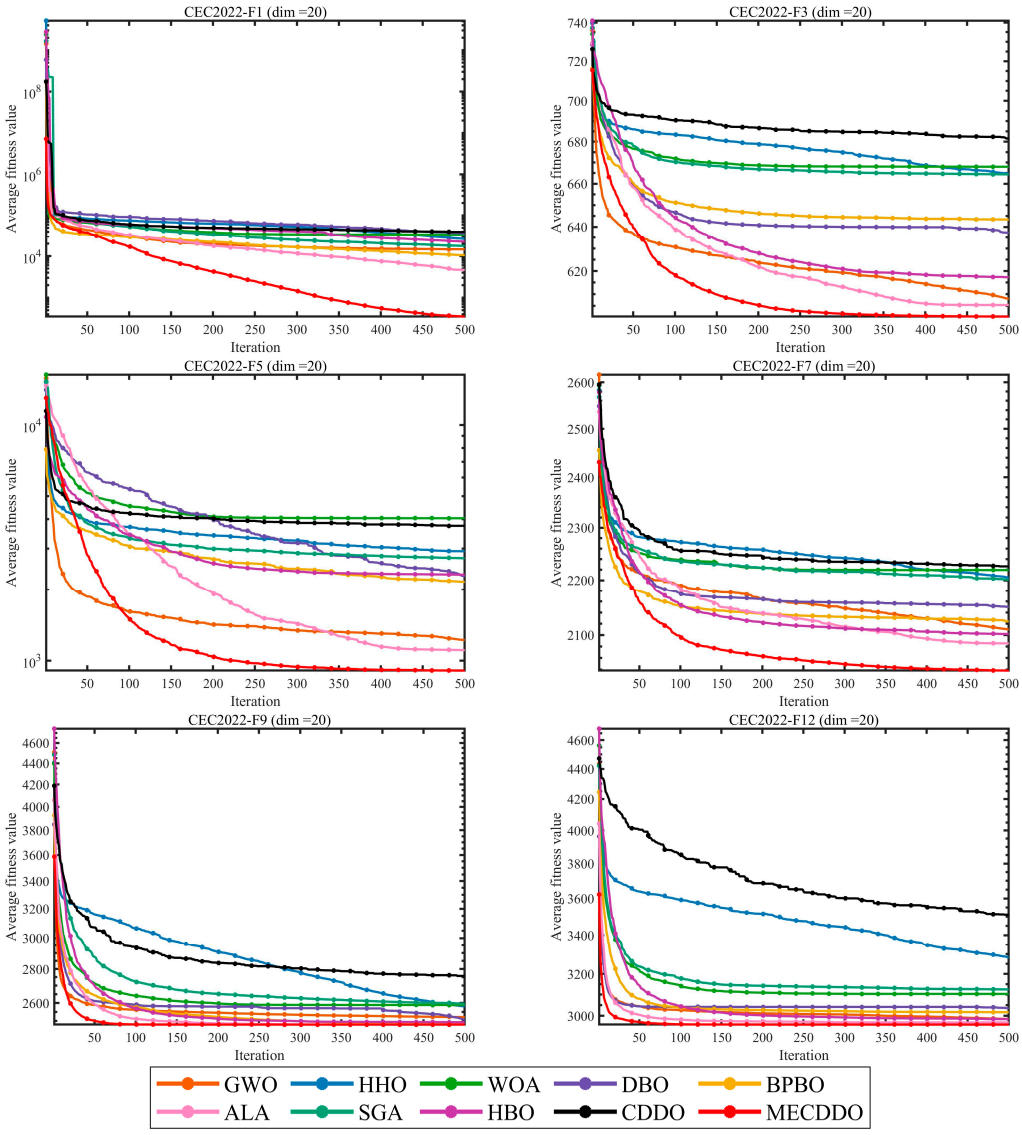


**Figure 8.** Convergence speed comparison among competing algorithms on CEC2022 benchmark test set.

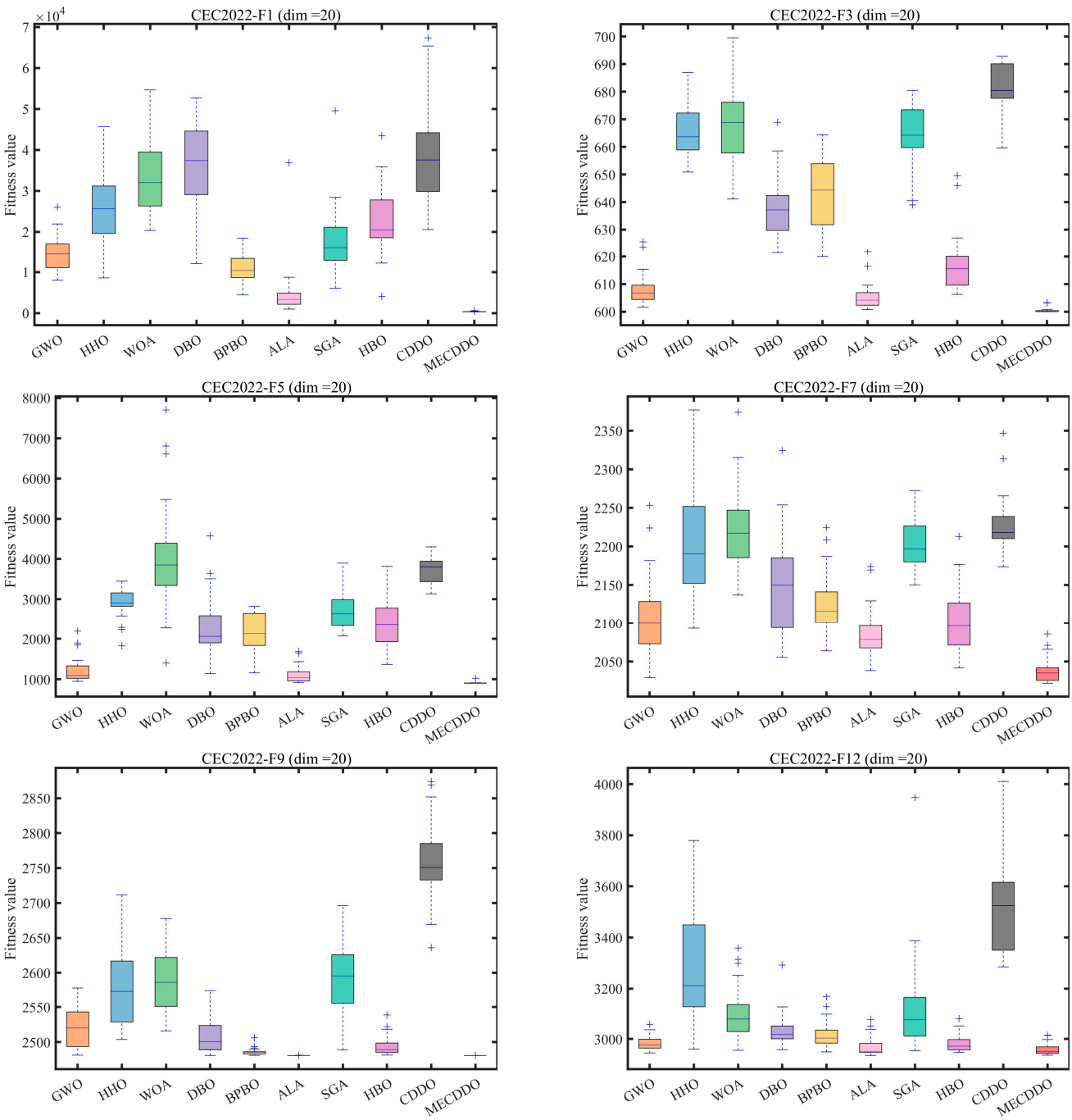


**Figure 9.** Statistical distribution analysis of all algorithms on CEC2022 benchmark set.

The data in Table 5 indicate that MECDDO outperforms the comparative algorithms in terms of both average fitness values (Ave) and standard deviations (Std) on most test functions. For the unimodal function F1, the Ave value of MECDDO ($3.5209 \times 10^2$) is only 0.88% of that obtained by CDDO ($3.9804 \times 10^4$), while its Std value ($7.3873 \times 10^1$) is merely 0.57% of that of CDDO, demonstrating an exceptionally strong capability for precise convergence.

For the multimodal function F5, the Ave value of MECDDO ($9.0784 \times 10^2$) is 75.7% lower than that of CDDO ($3.7449 \times 10^3$), and the corresponding Std value ($2.1937 \times 10^1$) is only 6.6% of that of CDDO, verifying its effectiveness in escaping local optima.

For the hybrid and composition functions, the Ave value of MECDDO on F6 ($3.4992 \times 10^3$) is significantly lower than that of CDDO ($4.3134 \times 10^8$), and on F11, the

Ave value ($2.9167 \times 10^3$) is 58.8% lower than that of CDDO ($7.0783 \times 10^3$), highlighting its superior adaptability in complex optimization scenarios.

The convergence curves shown in Figure 8 intuitively illustrate the efficiency advantage of MECDDO. For function F1, the fitness value of MECDDO decreases to below 400 within 100 iterations and stabilizes after 200 iterations, whereas algorithms such as CDDO and WOA still maintain fitness values above 10,000 even after 500 iterations.

For function F5, MECDDO completes the major fitness reduction within the first 100 iterations, decreasing from approximately 10,000 to around 900, followed by only minor refinements in subsequent iterations. In contrast, algorithms such as HHO and SGA continue to decrease slowly after 500 iterations and converge to fitness values that remain much higher than those of MECDDO, demonstrating that the multi-strategy cooperative mechanism effectively improves search efficiency.

The boxplots in Figure 9 further reveal that MECDDO exhibits a more concentrated fitness value distribution with the smallest dispersion. For function F1, MECDDO presents the narrowest box and no outliers, indicating highly consistent results over 30 independent runs, whereas CDDO shows a very wide box with numerous outliers, reflecting severe performance fluctuations.

For function F3, the box of MECDDO is almost entirely concentrated around 600, with an interquartile range significantly smaller than those of the other algorithms, confirming its stable performance in high condition number scenarios.

For function F5, MECDDO exhibits the lowest box position and the shortest whiskers, indicating not only superior average performance but also the ability to consistently generate high-quality solutions even under extreme conditions, thereby demonstrating a clear advantage in optimization stability.

Overall, MECDDO demonstrates accurate, fast, and stable optimization performance on the CEC2022 benchmark suite. The effectiveness of its core improvement strategies is consistently validated across different benchmark test sets.

### *3.7. Runtime Comparison Analysis*

To evaluate the computational efficiency of MECDDO, the CEC2017 benchmark suite with 30 dimensions was employed as the experimental platform. The average running time of MECDDO and the other algorithms on each test function was recorded, with each algorithm independently executed 30 times to ensure result reliability. The corresponding results are illustrated in Table 6.

As shown in Table 6, under the CEC2017 (dim = 30) benchmark environment, the average running times of all algorithms across the 30 test functions are within the same order of magnitude ($10^{-1}$ to $10^0$ s). This indicates that, under the unified parameter settings of population size ($N = 30$) and maximum iteration number ($T = 500$), all compared algorithms exhibit relatively high computational efficiency. Overall, WOA, SGA, and ALA demonstrate lower execution times on most functions. For example, on F1–F5, the running time of WOA remains consistently within 0.04–0.06 s, placing it among the fastest algorithms. This efficiency primarily results from its relatively simple update mechanism, where the position updating equations involve lower computational overhead, thereby reducing the time cost per iteration.

In contrast, the average running time of MECDDO is slightly higher than that of conventional algorithms on most functions. For instance, on F1, the average runtime of MECDDO is 0.1457 s, which is higher than that of WOA (0.0431 s) and SGA (0.0472 s); on F6, MECDDO requires 0.2583 s, also exceeding GWO (0.2035 s) and ALA (0.1503 s). This time difference mainly arises from the additional mechanisms introduced in MECDDO during each iteration, including the multi-strategy adaptive selection mechanism, Euclidean

distance-based weight computation, and the global-best-guided boundary repair strategy. Compared with the original CDDO, which adopts a single update strategy, MECDDO requires additional operations such as strategy success counting, roulette-wheel selection, and multi-source individual information interaction, resulting in a moderate increase in the constant time factor.

**Table 6.** Average running time of different algorithms on CEC2017(dim = 30).

| Function | Metric | GWO | HHO | WOA | DBO | BPBO | ALA | SGA | HBO | CDDO | MECDDO |
|---|---|---|---|---|---|---|---|---|---|---|---|
| F1 | Ave | 0.1185 | 0.1010 | 0.0431 | 0.0746 | 0.0907 | 0.0663 | 0.0472 | 0.0868 | 0.1277 | 0.1457 |
| F2 | Ave | 0.1334 | 0.1322 | 0.0557 | 0.0892 | 0.1098 | 0.0816 | 0.0589 | 0.1060 | 0.1480 | 0.1618 |
| F3 | Ave | 0.1236 | 0.1172 | 0.0459 | 0.0819 | 0.0912 | 0.0689 | 0.0497 | 0.0914 | 0.1360 | 0.1634 |
| F4 | Ave | 0.1243 | 0.1079 | 0.0460 | 0.0770 | 0.0949 | 0.0683 | 0.0492 | 0.0912 | 0.1350 | 0.1523 |
| F5 | Ave | 0.1368 | 0.1430 | 0.0582 | 0.0887 | 0.1070 | 0.0811 | 0.0620 | 0.1091 | 0.1485 | 0.1749 |
| F6 | Ave | 0.2035 | 0.3142 | 0.1276 | 0.1611 | 0.1780 | 0.1503 | 0.1295 | 0.2070 | 0.2217 | 0.2583 |
| F7 | Ave | 0.1379 | 0.1540 | 0.0615 | 0.0909 | 0.1098 | 0.0838 | 0.0649 | 0.1133 | 0.1513 | 0.1730 |
| F8 | Ave | 0.1370 | 0.1472 | 0.0601 | 0.0908 | 0.1086 | 0.0823 | 0.0636 | 0.1111 | 0.1501 | 0.1808 |
| F9 | Ave | 0.1377 | 0.1569 | 0.0615 | 0.0926 | 0.1073 | 0.0838 | 0.0649 | 0.1119 | 0.1479 | 0.1596 |
| F10 | Ave | 0.1516 | 0.1874 | 0.0738 | 0.1113 | 0.1228 | 0.0994 | 0.0785 | 0.1325 | 0.1676 | 0.1825 |
| F11 | Ave | 0.1306 | 0.1305 | 0.0531 | 0.0856 | 0.1023 | 0.0756 | 0.0564 | 0.1012 | 0.1457 | 0.1799 |
| F12 | Ave | 0.1423 | 0.1573 | 0.0670 | 0.0988 | 0.1163 | 0.0891 | 0.0687 | 0.1204 | 0.1548 | 0.1828 |
| F13 | Ave | 0.1317 | 0.1348 | 0.0546 | 0.0866 | 0.1043 | 0.0779 | 0.0580 | 0.1031 | 0.1398 | 0.1605 |
| F14 | Ave | 0.1544 | 0.1966 | 0.0782 | 0.1132 | 0.1255 | 0.1007 | 0.0818 | 0.1381 | 0.1684 | 0.1923 |
| F15 | Ave | 0.1283 | 0.1316 | 0.0519 | 0.0830 | 0.1013 | 0.0747 | 0.0553 | 0.0993 | 0.1420 | 0.1623 |
| F16 | Ave | 0.1366 | 0.1499 | 0.0599 | 0.0913 | 0.1081 | 0.0832 | 0.0630 | 0.1121 | 0.1538 | 0.1605 |
| F17 | Ave | 0.1825 | 0.2546 | 0.1038 | 0.1381 | 0.1561 | 0.1283 | 0.1077 | 0.1752 | 0.1977 | 0.2173 |
| F18 | Ave | 0.1398 | 0.1549 | 0.0613 | 0.0952 | 0.1109 | 0.0846 | 0.0665 | 0.1146 | 0.1527 | 0.2246 |
| F19 | Ave | 0.3988 | 0.7881 | 0.3229 | 0.3624 | 0.3844 | 0.3472 | 0.3258 | 0.4878 | 0.4268 | 0.4697 |
| F20 | Ave | 0.2709 | 0.2835 | 0.1143 | 0.1536 | 0.1648 | 0.1391 | 0.1185 | 0.1903 | 0.2106 | 0.2576 |
| F21 | Ave | 0.2118 | 0.3281 | 0.1354 | 0.1687 | 0.1865 | 0.1584 | 0.1377 | 0.2180 | 0.2293 | 0.2478 |
| F22 | Ave | 0.2173 | 0.3435 | 0.1407 | 0.1750 | 0.1892 | 0.1659 | 0.1464 | 0.2259 | 0.2335 | 0.2302 |
| F23 | Ave | 0.2389 | 0.3838 | 0.1593 | 0.1937 | 0.2106 | 0.1837 | 0.1627 | 0.2545 | 0.2588 | 0.2644 |
| F24 | Ave | 0.2595 | 0.4381 | 0.1829 | 0.2157 | 0.2323 | 0.2069 | 0.1883 | 0.2838 | 0.2819 | 0.2975 |
| F25 | Ave | 0.2331 | 0.3593 | 0.1530 | 0.1845 | 0.2030 | 0.1770 | 0.1565 | 0.2452 | 0.2437 | 0.2801 |
| F26 | Ave | 0.2720 | 0.4604 | 0.1915 | 0.2252 | 0.2420 | 0.2170 | 0.1977 | 0.3010 | 0.2885 | 0.3041 |
| F27 | Ave | 0.3101 | 0.5286 | 0.2321 | 0.2641 | 0.2840 | 0.2511 | 0.2324 | 0.3522 | 0.3202 | 0.3485 |
| F28 | Ave | 0.2661 | 0.4357 | 0.1882 | 0.2220 | 0.2385 | 0.2134 | 0.1915 | 0.2939 | 0.2792 | 0.3188 |
| F29 | Ave | 0.2349 | 0.3813 | 0.1570 | 0.1938 | 0.2115 | 0.1817 | 0.1603 | 0.2505 | 0.2603 | 0.2992 |
| F30 | Ave | 0.4610 | 0.9235 | 0.3844 | 0.4224 | 0.4449 | 0.4081 | 0.3853 | 0.5714 | 0.4912 | 0.5186 |

Nevertheless, it should be emphasized that the increase in runtime overhead of MECDDO is limited and controllable. From an overall perspective, its running time remains stable within the range of 0.15–0.30 s, and no order-of-magnitude growth is observed even for complex functions (e.g., F19 and F30). This observation is consistent with the previous computational complexity analysis: the time complexity of MECDDO remains $O(N \times T \times d)$, which is asymptotically identical to that of the original CDDO and other mainstream swarm intelligence algorithms, with only a slight increase in the constant factor. Therefore, while significantly enhancing optimization performance, the proposed method does not sacrifice scalability or practical applicability.

Combining the accuracy and stability results presented in Section 3.5, it can be observed that although MECDDO incurs a slightly higher runtime than some structurally simpler algorithms, it exhibits substantial advantages in solution accuracy, convergence speed, and robustness. For example, on high-complexity functions (e.g., F18 and F30), MECDDO improves the optimization results by several orders of magnitude compared with CDDO, while the additional runtime increases by only approximately 0.02–0.05 s. This characteristic—achieving significant performance gains with only a marginal computational

cost—demonstrates the effectiveness and practical value of the proposed multi-strategy cooperative framework.

In summary, MECDDO maintains the same asymptotic time complexity as mainstream swarm intelligence algorithms. Although its runtime is slightly increased, the growth is minor and stable. Meanwhile, its substantial improvements in optimization accuracy and robustness far outweigh the additional computational cost. Consequently, MECDDO achieves superior overall performance in terms of accuracy, stability, and efficiency, and is capable of providing high-quality solutions for complex global optimization problems while maintaining acceptable computational expense.

### *3.8. Friedman Test Analysis and Results*

To construct a complete performance hierarchy for the MECDDO method, statistical tools that can evaluate multiple interrelated algorithms simultaneously are indispensable. Customized for this application, the Friedman test operates as a non-parametric technique comparing algorithms according to their ranked outcomes over diverse datasets. This methodology intrinsically avoids any presuppositions about the fundamental distribution of performance metrics. Such a property makes it especially powerful for comparative studies examining multiple algorithms tested on the same set of benchmark functions. The computation of the test statistic adheres to the following expression [3,7]:

$$Q = \frac{12}{kn(k+1)} \sum_{j=1}^{k} R_j^2 - 3n(k+1) \tag{14}$$

In the presented equation, $n$ refers to the total number of blocks, $k$ indicates the count of experimental treatments, and $R_j$ represents the summed ranking value for treatment $j$th. Provided that the sample sizes for $n$ and $k$ meet sufficient magnitude requirements, the calculated $Q$ statistic follows a $\chi^2$ probability distribution possessing $k-1$ degrees of freedom [7].

Table 7 clearly presents the mean ranking (M.R.) and total ranking (T.R.) of all algorithms on the CEC2017 (30/50/100 dimensions) and CEC2022 (20 dimensions) benchmark test suites. MECDDO consistently ranks first across all test scenarios, demonstrating a significant and stable performance advantage.

**Table 7.** Friedman test ranking results with $p$-values.

| **Suites** | **CEC2017** | | | | | | **CEC2022** | |
|---|---|---|---|---|---|---|---|---|
| **Dimensions** | **30** | | **50** | | **100** | | **20** | |
| **Algorithms** | ***M.R*** | ***T.R*** | ***M.R*** | ***T.R*** | ***M.R*** | ***T.R*** | ***M.R*** | ***T.R*** |
| GWO | 3.90 | 3 | 3.60 | 3 | 3.30 | 3 | 4.75 | 5 |
| HHO | 6.90 | 7 | 6.10 | 7 | 5.83 | 6 | 7.17 | 7 |
| WOA | 8.97 | 9 | 8.77 | 9 | 8.43 | 9 | 8.83 | 9 |
| DBO | 5.63 | 6 | 5.80 | 6 | 6.03 | 7 | 5.50 | 6 |
| BPBO | 3.97 | 4 | 3.97 | 4 | 3.63 | 4 | 3.75 | 3 |
| ALA | 2.53 | 2 | 2.43 | 2 | 2.77 | 2 | 2.50 | 2 |
| SGA | 7.37 | 8 | 7.57 | 8 | 7.50 | 8 | 7.25 | 8 |
| HBO | 4.27 | 5 | 4.63 | 5 | 5.07 | 5 | 4.33 | 4 |
| CDDO | 9.83 | 10 | 9.93 | 10 | 9.73 | 10 | 9.67 | 10 |
| MECDDO | 1.63 | 1 | 2.20 | 1 | 2.70 | 1 | 1.25 | 1 |
| $p$-value | 0.0453 < 0.05 | | 0.0382 < 0.05 | | 0.0329 < 0.05 | | 0.0366 < 0.05 | |

On the CEC2017 benchmark, in the 30-dimensional scenario, MECDDO achieves an average ranking of only 1.63, which is substantially better than that of the second-ranked ALA (2.53) and the third-ranked GWO (3.90). In contrast, the conventional CDDO ranks tenth with an average ranking of 9.83, showing a pronounced performance gap compared with MECDDO. As the dimensionality increases to 50 and 100 dimensions, the average ranking of MECDDO slightly increases to 2.20 and 2.70, respectively, yet it still maintains the top position. Moreover, the gap between MECDDO and the second-ranked ALA (2.43 and 2.77) remains very small, indicating strong adaptability to high-dimensional optimization problems. Conversely, the average rankings of CDDO remain at the bottom, with values of 9.93 and 9.73 for the 50- and 100-dimensional cases, respectively, highlighting its clear performance disadvantage in high-dimensional scenarios.

On the CEC2022 (20-dimensional) benchmark, MECDDO further improves its average ranking to 1.25, which is the best result among all tested scenarios. This value is markedly superior to that of the second-ranked ALA (2.50) and the third-ranked BPBO (3.75), while CDDO again ranks last with an average ranking of 9.67. These results demonstrate that regardless of the benchmark characteristics, MECDDO consistently outperforms the comparative algorithms in terms of overall optimization performance, and its core improvement strategies remain effective across diverse optimization scenarios.

Figure 10 visualizes the relative ranking distributions of the algorithms across different test functions, providing an intuitive illustration of the ranking stability and dominance of MECDDO.

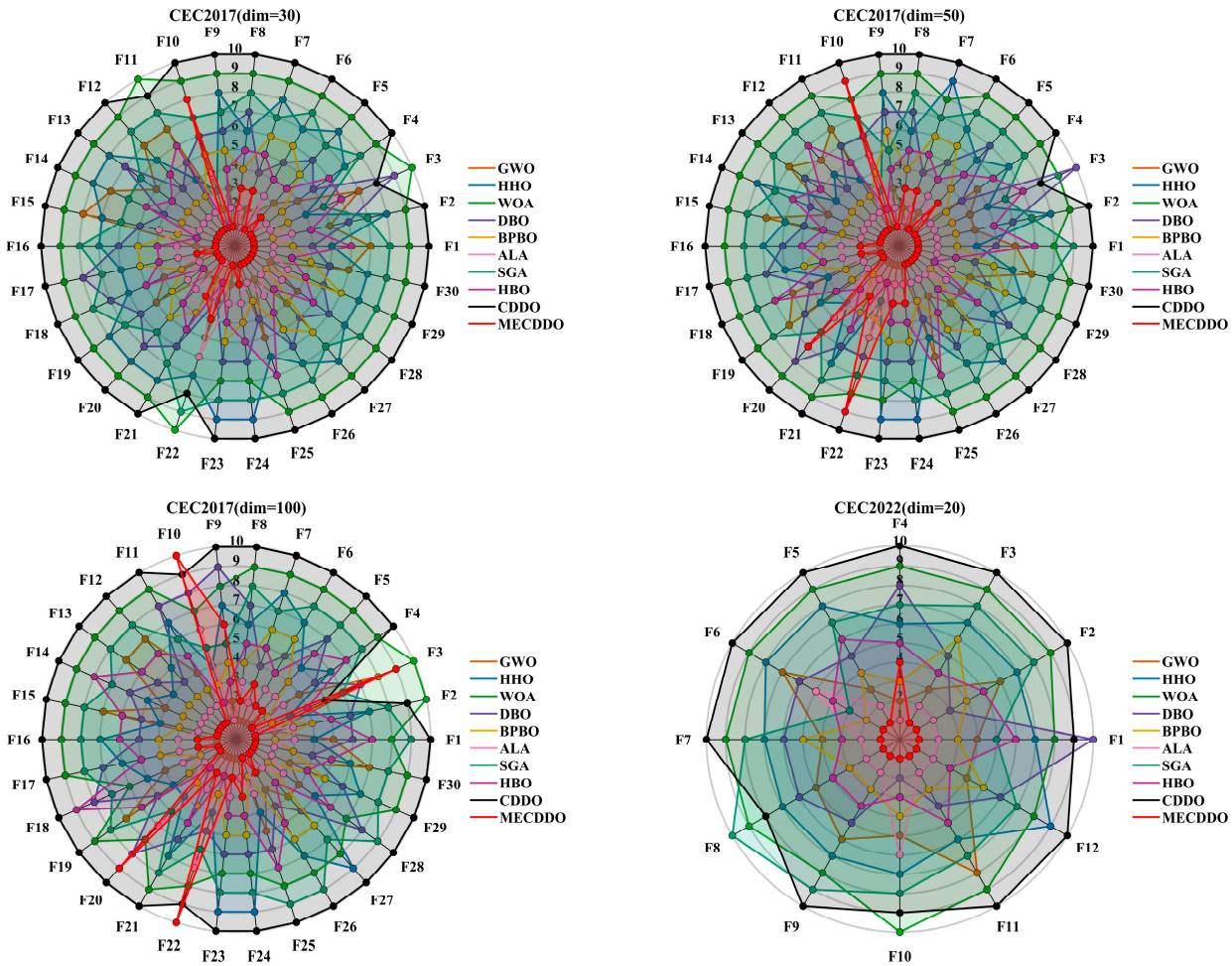


**Figure 10.** Distribution of relative performance rankings across all evaluated algorithms.

In the CEC2017 (50-dimensional) scenario, the rankings of MECDDO are almost entirely concentrated at the top positions across all functions, with only marginal competition from ALA and BPBO on a few functions. The ranking distribution of MECDDO is highly concentrated and consistently leading, whereas algorithms such as CDDO and WOA exhibit dispersed rankings mainly located in the middle-to-lower range, with most function rankings between 8th and 10th.

In the 100-dimensional scenario, MECDDO continues to maintain a concentrated ranking advantage. Even on complex high-dimensional functions (e.g., F2, F12, and F30), MECDDO consistently remains within the top three positions. In contrast, the ranking fluctuations of other algorithms increase significantly, with some algorithms experiencing sharp ranking degradations on specific functions.

In the CEC2022 (20-dimensional) scenario, MECDDO ranks first or second on all 12 test functions, with a compact and stable ranking distribution and no evident fluctuations. By comparison, algorithms such as CDDO and WOA exhibit dispersed rankings mostly ranging from 6th to 10th and rank last on multiple functions. This distribution pattern clearly indicates that MECDDO not only achieves overall superior performance but also maintains consistent advantages across functions with different characteristics, effectively avoiding the "specialization bias" observed in some algorithms. These results further confirm its strong generalization capability and robustness.

Overall, the Friedman test results, from both quantitative ranking metrics and distribution characteristics, comprehensively confirm the superior overall performance of MECDDO across various test scenarios. Its stable ranking behavior and broad function adaptability further validate the effectiveness and synergistic interaction of the three core strategies: adaptive cooperative search, multi-strategy adaptive selection, and global-optimum-guided boundary repair.

All Friedman test results yielded $p$-values $< 0.05$ (Table 7), indicating statistically significant differences in optimization performance among the ten algorithms across all dimensional settings. The results showed that MECDDO's mean ranking is significantly better than CDDO, WOA, and HHO in all dimensions, while no significant difference was found between MECDDO and ALA, confirming that MECDDO achieves performance comparable to the second-best algorithm while maintaining a stable leading advantage over other competitors.

### *3.9. Wilcoxon Signed-Rank Test*

To further verify the statistical significance of MECDDO's performance advantage, the Wilcoxon signed-rank test with Holm correction (significance level $\alpha = 0.05$) is conducted between MECDDO and each comparative algorithm on CEC2017(30/50/100 dimensions) and CEC2022(20 dimensions) benchmark. The results are shown in Table 8, where "Win" indicates MECDDO is significantly better, "Loss" indicates significantly worse, and "Tie" indicates no significant difference.

Based on the results obtained on the CEC2017 benchmark with 30 dimensions, MECDDO demonstrates a statistically significant advantage over all comparative algorithms. When compared with the original CDDO, MECDDO achieves 29 wins and 1 case with no significant difference, with no failures observed, indicating a highly reliable and statistically robust improvement over the baseline algorithm. Against the competitive ALA algorithm, MECDDO still records 24 wins and 6 cases with no significant difference, highlighting its superiority even among strong competitors. Furthermore, when compared with the well-performing GWO, MECDDO attains 22 wins and 8 cases with no significant difference, further confirming the statistical significance of its optimization performance in the 30-dimensional scenario.

**Table 8.** Wilcoxon signed-rank test results (α = 0.05).

| **Suites** | **CEC2017(Win/Tie/Loss)** | | | **CEC2022(Win/Tie/Loss)** |
|---|---|---|---|---|
| **Dimensions** | **30** | **50** | **50** | **20** |
| MECDDO vs. GWO | (22/0/8) | (25/0/5) | (23/0/7) | (10/0/2) |
| MECDDO vs. HHO | (29/0/1) | (27/1/2) | (30/0/0) | (11/0/1) |
| MECDDO vs. WOA | (29/0/1) | (30/0/0) | (30/0/0) | (12/0/0) |
| MECDDO vs. DBO | (28/0/2) | (25/0/5) | (27/0/3) | (11/0/1) |
| MECDDO vs. BPBO | (28/0/2) | (22/0/8) | (29/0/1) | (10/0/2) |
| MECDDO vs. ALA | (24/0/6) | (20/0/10) | (25/0/5) | (10/0/2) |
| MECDDO vs. SGA | (28/0/2) | (27/0/3) | (29/0/1) | (12/0/0) |
| MECDDO vs. HBO | (28/0/2) | (25/0/5) | (29/0/1) | (10/0/2) |
| MECDDO vs. CDDO | (29/0/1) | (26/0/4) | (27/0/3) | (12/0/0) |
| First-place finishes of comparative algorithm | 22 | 19 | 15 | 11 |

In the CEC2017 (50-dimensional) scenario, the performance advantage of MECDDO becomes even more pronounced. In comparison with WOA, MECDDO achieves 30 complete wins, with no cases of no significant difference or failure, indicating that its optimization performance is far superior in medium-dimensional complex search spaces. Against HHO, MECDDO records 27 wins, 1 case with no significant difference, and only 2 failures, with failures being extremely rare and the overall advantage remaining dominant. When compared with CDDO, MECDDO maintains an absolute lead with 26 wins and 4 cases with no significant difference, once again demonstrating the effectiveness of the proposed improvement strategies in medium-dimensional scenarios.

In the high-dimensional CEC2017 (100-dimensional) scenario, the significant advantage of MECDDO does not diminish with increasing dimensionality. In comparisons with algorithms such as WOA and SGA, MECDDO achieves 30 complete wins, demonstrating outstanding stability in high-dimensional optimization. When compared with CDDO, MECDDO still achieves 27 wins and 3 cases with no significant difference, indicating that the performance improvements over the original algorithm remain statistically significant in high-dimensional spaces. Even against the relatively robust BPBO, MECDDO secures 29 wins and 1 case with no significant difference, maintaining a clear and decisive advantage.

On the CEC2022 benchmark with 20 dimensions, the statistical superiority of MECDDO is likewise confirmed. In comparisons with WOA, SGA, and CDDO, MECDDO achieves 12 complete wins, with no cases of no significant difference or failure, demonstrating that its generalization performance remains statistically significant across different benchmark test suites. Against algorithms such as GWO, BPBO, and ALA, MECDDO consistently records 10 wins and 2 cases with no significant difference, further validating the stability and reliability of its optimization performance.

Overall, the results of the Wilcoxon signed-rank test provide strong statistical evidence supporting the superior performance of MECDDO. Across different dimensional settings and benchmark test suites, the performance differences between MECDDO and the comparative algorithms are statistically significant, with an overwhelmingly high proportion of wins and no notable failure cases. These results comprehensively verify the effectiveness and synergistic contribution of the three core strategies—adaptive cooperative search, multi-strategy adaptive selection, and global-optimum-guided boundary repair.

## 4. Mobile Robot Path Planning

### *4.1. Map Model Construction*

Common path planning representations mainly include the grid-based method, topological method, free-space method, and visibility graph method. The grid-based method

discretizes the external environment into uniform cells according to a given resolution. Each cell is represented by a state, either occupied or free, indicating whether the cell corresponds to an obstacle [57,58]. The path planning algorithm occupies a free cell and searches for a path by traversing free cells while avoiding obstacles, forming a path composed of multiple connected cells.

The topological method divides the robot's workspace into several smaller regions and establishes a topological network by connecting these regions with edges. Path planning is performed by searching through this network to form a path along the topological connections. The free-space method divides the environment into two regions: the movable region and the obstacle region [59]. A map model is generated by connecting the midpoints of line segments in the movable region, and the robot plans its path within this model. The visibility graph method connects the start position, obstacle vertices, and the goal position pairwise to construct a multi-segment path structure. Path planning algorithms then generate a complete path from the start to the goal along these segments [60].

Among these methods, the grid-based method is one of the simplest and most widely used for mobile robot path planning. It is straightforward, effective, highly adaptable to obstacles, and significantly reduces the complexity of environment modeling. Therefore, the robot path planning environment in this work is established based on the grid method. Each grid cell in the environment is classified as either a free cell or an obstacle cell, depending on whether it contains an obstacle.

Figure 11 illustrates the robot's possible movements within the grid map. In the figure, red dots represent the robot, and arrows indicate the directions in which the robot can move. For the robot on the left, there are no surrounding obstacles, so all eight directions are accessible. In contrast, for the robot on the right, obstacles are present above and at the upper-left, leaving only five feasible directions for movement.

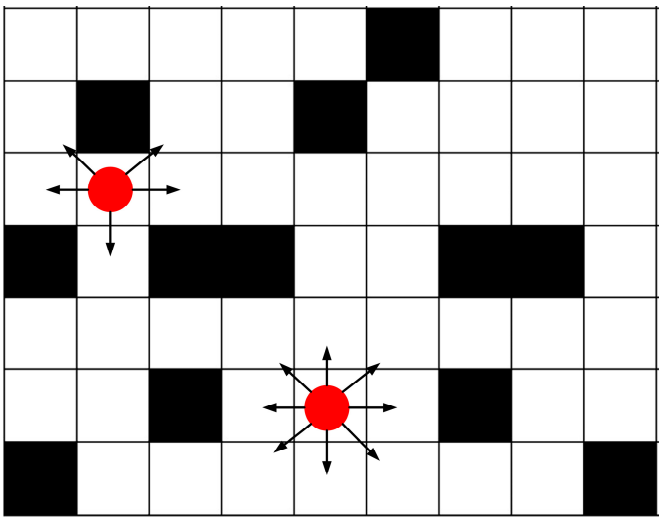

**Figure 11.** Representation of the 2-dimensional grid environment.

### *4.2. Experimental Analysis of Mobile Robot Path Planning*

To evaluate the feasibility and effectiveness of MECDDO in planning paths for mobile robots in complex fire-field grid environments, experiments were conducted on 20 × 20 and 40 × 40 grid maps. MECDDO was compared with nine other algorithms through simulation experiments. To minimize the influence of algorithmic randomness on the simulation results, each algorithm was independently run 30 times under the same conditions. In the tables, the average value represents the mean path length planned by the algorithm across 30 runs, while the maximum and minimum values indicate the longest and shortest path

lengths obtained in these runs, respectively. The minimum value among the three metrics is highlighted in bold.

#### 4.2.1. Experimental Analysis in 20 × 20 Grid Environment

In the 20 × 20 grid environment, the starting point was set at (1,1) and the goal at (20,20). Simulation experiments were conducted using MECDDO and nine comparison algorithms. The planned robot paths are shown in Figure 12, while the evolution of fitness values during iterations is presented in Figure 13. A comparison of the quantitative results for all ten algorithms is summarized in Table 9.

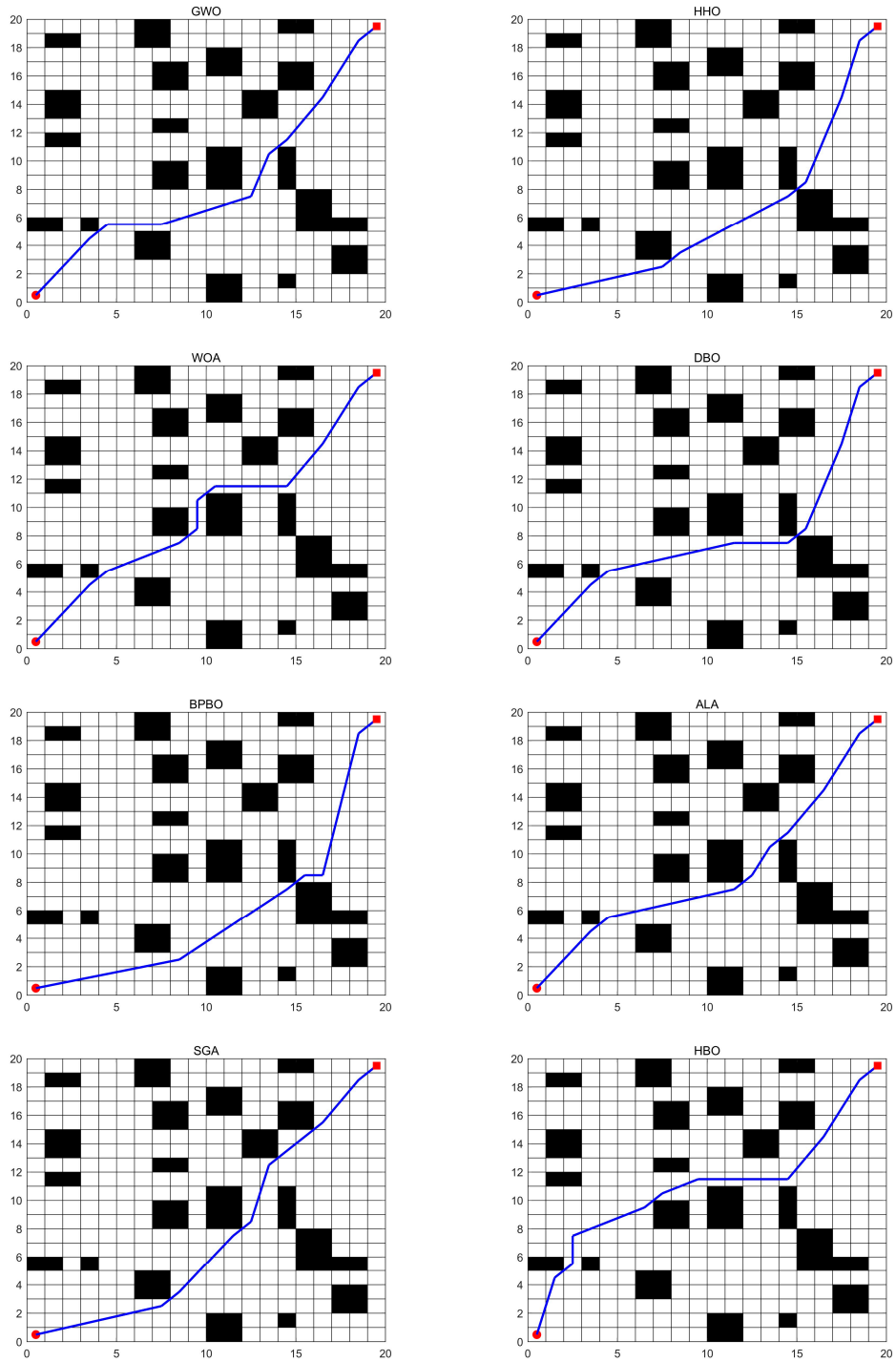


**Figure 12.** *Cont.*

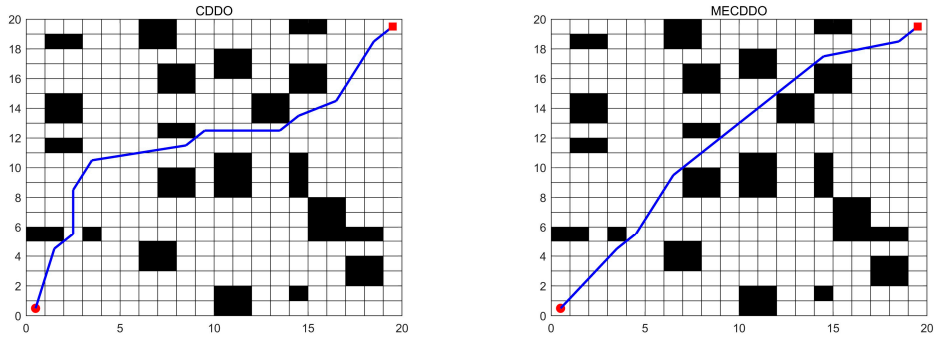


**Figure 12.** Planned trajectories in a 20 × 20 grid-based environment.

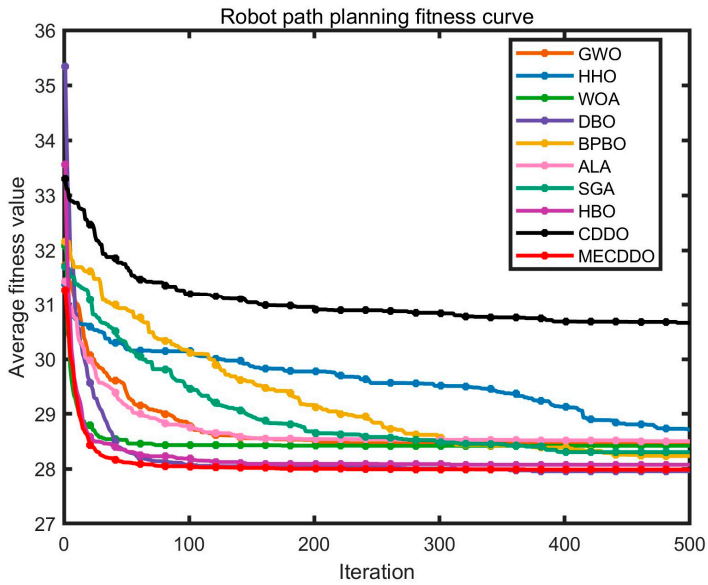


**Figure 13.** Fitness evolution across algorithms in the 20 × 20 grid environment.

**Table 9.** Performance metrics in 20 × 20 grid environment.

| Algorithm | Ave | Std | Max | Min | Rank |
|---|---|---|---|---|---|
| GWO | 28.46562 | 0.535233 | 29.93189 | 27.7589 | 7 |
| HHO | 28.71597 | 1.028369 | 32.07765 | 27.73738 | 9 |
| WOA | 28.41905 | 0.594928 | 29.99558 | 27.73738 | 6 |
| DBO | 27.98592 | 0.24286 | 28.45072 | 27.77219 | 2 |
| BPBO | 28.23401 | 0.431781 | 29.45356 | 27.7589 | 4 |
| ALA | 28.50333 | 0.629048 | 29.97041 | 27.87219 | 8 |
| SGA | 28.30385 | 0.353848 | 29.37382 | 27.7589 | 5 |
| HBO | 28.07402 | 0.296509 | 29.02853 | 27.97219 | 3 |
| CDDO | 30.66212 | 0.614306 | 32.18488 | 29.17303 | 10 |
| MECDDO | **27.95483** | **0.234255** | **28.25072** | **27.67219** | **1** |

Figure 12 provides a visual comparison of the mobile robot paths planned by ten algorithms in a 20 × 20 grid environment, clearly demonstrating MECDDO's superior performance. The paths planned by MECDDO are characterized by minimal turns and near-linear trajectories. From the start point (1,1) to the goal (20,20), the path includes only 2–3 necessary turns, each close to 90°, with no redundant detours or sharp-angle turns. This effectively reduces frequent directional adjustments during actual robot movement, minimizing energy consumption and motion errors. In contrast, CDDO produces paths with obvious redundancies, showing multiple unnecessary turns in the grid region (8,10)–(12,15), resulting in a "zigzag" trajectory and significantly longer travel distance. While GWO and

WOA generate smoother paths than CDDO, they still involve 1–2 unnecessary turns in the (5,5)–(10,10) region, leading to longer path lengths compared with MECDDO.

Regarding obstacle avoidance, MECDDO maintains a safe distance of 1–2 grid units from obstacles. In the dense obstacle area (12,8)–(15,12), MECDDO smoothly navigates along the obstacle edges without approaching too closely or risking collisions. In contrast, HHO and SGA generate paths that come dangerously close to obstacles (e.g., HHO passes only 0.5 grid units from an obstacle at (13,10)), posing potential collision hazards. These observations further confirm MECDDO's ability to jointly optimize "safety" and "smoothness" in path planning.

Figure 13 illustrates the fitness (path length) evolution of ten algorithms in a 20 × 20 grid environment, highlighting MECDDO's advantages in both convergence speed and accuracy. Regarding convergence speed, MECDDO exhibits a rapid decline in the early iterations (0–100), with the fitness value dropping below 28.5 by iteration 50 and approaching the optimal value (~27.95) by iteration 100. In contrast, CDDO requires 300 iterations to reach below 29.5, GWO and WOA need 200 iterations to approach 28.5, and HHO remains above 28.7 even after 500 iterations, demonstrating MECDDO's ability to quickly identify high-quality paths and reduce computational time. In terms of convergence accuracy, MECDDO's fitness curve stabilizes around 27.95 after 200 iterations, showing minimal fluctuation. Meanwhile, algorithms such as CDDO and ALA continue to exhibit slight oscillations in the later iterations (e.g., CDDO fluctuates between 30.7 and 30.5 during iterations 400–500), indicating a tendency to linger near local optima and difficulty in achieving stable convergence to the global optimum. Furthermore, MECDDO's fitness curve consistently lies below all comparative algorithms, with the gap widening over iterations, further validating its dual advantage in path planning accuracy and convergence stability. These observations are fully consistent with the quantitative results in Table 8 and the trajectory visualizations in Figure 10, collectively confirming MECDDO's effectiveness and superiority in mobile robot path planning within a 20 × 20 grid environment.

Table 9 quantitatively validates MECDDO's path planning performance, with key metrics including average path length (Ave), standard deviation (Std), maximum path length (Max), minimum path length (Min), and overall rank (Rank). In terms of average path length, MECDDO achieves 27.95483, ranking first, which is 0.03109 shorter than the second-ranked DBO (27.98592) and 2.70729 shorter than the traditional CDDO (30.66212), representing an 8.83% reduction. This indicates a substantial decrease in energy consumption and travel time for the robot in practical applications. Regarding extremal performance, MECDDO attains the shortest minimum path length (27.67219) among all algorithms, and its maximum path length (28.25072) exceeds the minimum by only 0.57853, much smaller than comparative algorithms—for instance, HHO's maximum (32.07765) exceeds its minimum (27.73738) by 4.34027, and CDDO's maximum (32.18488) exceeds its minimum (29.17303) by 3.01185—demonstrating MECDDO's superior stability and lower sensitivity to initial parameters or environmental randomness.

The standard deviation further confirms MECDDO's stability advantage. Its Std value (0.234255) is only 38.1% of CDDO's (0.614306) and 22.8% of HHO's (1.028369). Even compared with the second-ranked DBO (0.24286), MECDDO is lower by 0.008605, indicating that across 30 independent trials, the path length fluctuations remain minimal, ensuring high reliability for robotic path planning. In overall ranking, MECDDO holds a clear first place with a significant gap from other algorithms, further confirming its optimal comprehensive performance in the 20 × 20 grid environment.

#### 4.2.2. Experimental Analysis in 40 × 40 Grid Environment

In the 40 × 40 grid environment, the starting point was set at (1,1) and the goal at (40,40). Simulation experiments were conducted using MECDDO and nine comparison algorithms. The planned robot paths are shown in Figure 14, while the evolution of fitness values during iterations is presented in Figure 15. A comparison of the quantitative results for all ten algorithms is summarized in Table 10.

**Table 10.** Performance metrics in 40 × 40 grid environment.

| Algorithm | Ave | Std | Max | Min | Rank |
|---|---|---|---|---|---|
| GWO | 100.0443 | 15.01177 | 129.7612 | 67.29792 | 10 |
| HHO | 70.01516 | **0.724759** | 71.64656 | 68.7475 | 5 |
| WOA | 67.35359 | 2.591567 | 70.59169 | 59.66034 | 4 |
| DBO | 64.48274 | 3.528144 | 75.1708 | 59.4203 | 2 |
| BPBO | 82.31536 | 13.5643 | 113.426 | 68.47481 | 7 |
| ALA | 91.03177 | 8.921237 | 108.3541 | 68.26577 | 8 |
| SGA | 77.14263 | 1.487526 | 77.41421 | 69.2667 | 6 |
| HBO | 65.01684 | 4.275003 | 76.90015 | 59.96258 | 3 |
| CDDO | 97.20336 | 16.02579 | 121.1821 | 74.81734 | 9 |
| MECDDO | **61.15516** | 1.815313 | **65.30184** | **58.24101** | **1** |

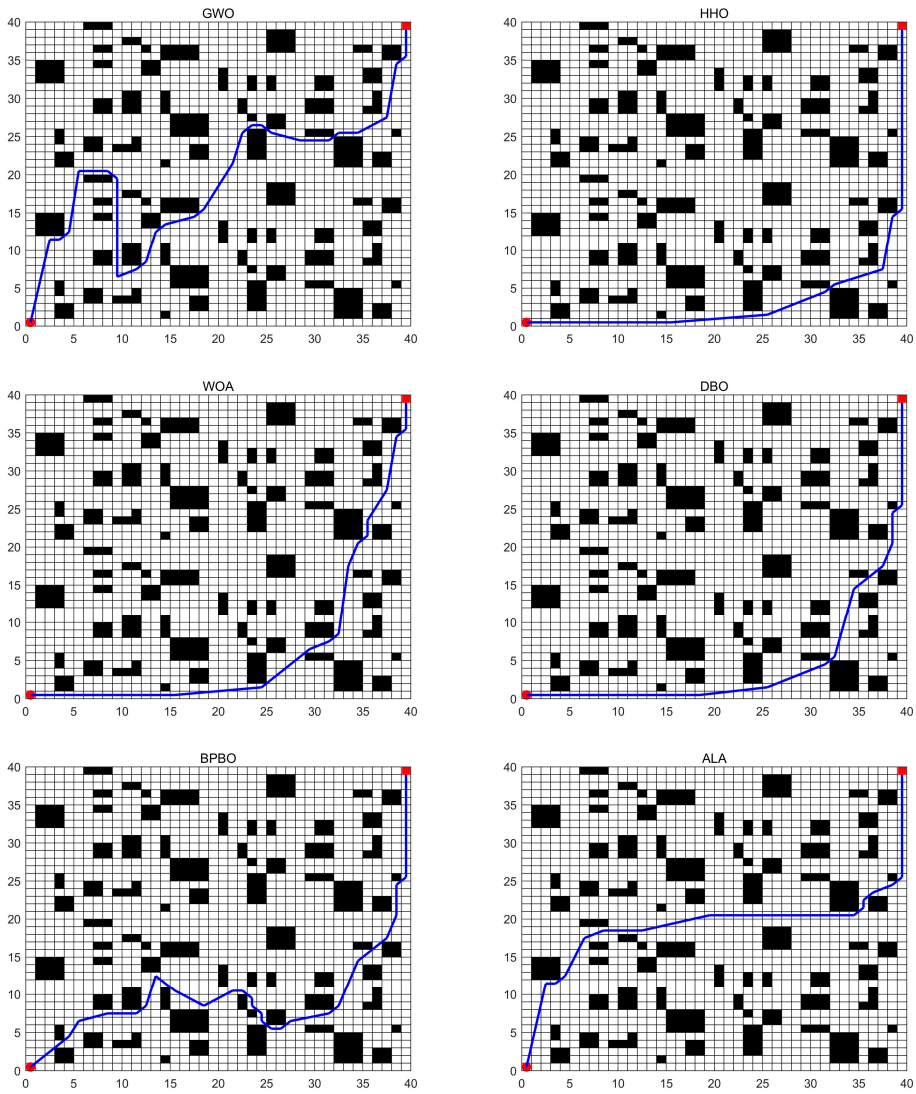


**Figure 14.** *Cont.*

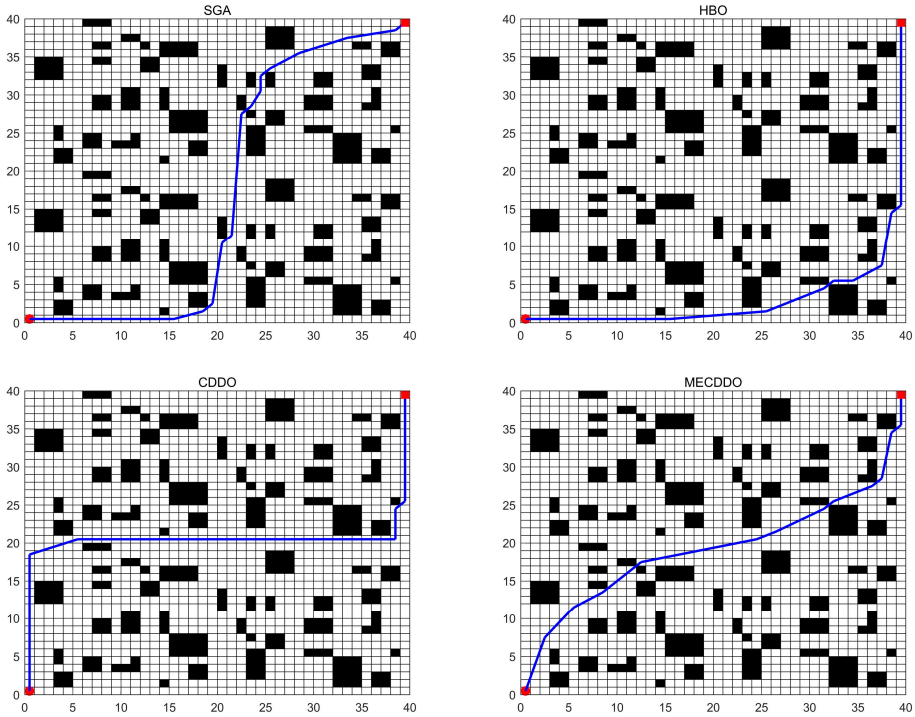


**Figure 14.** Planned trajectories in a 40×40 grid-based environment.

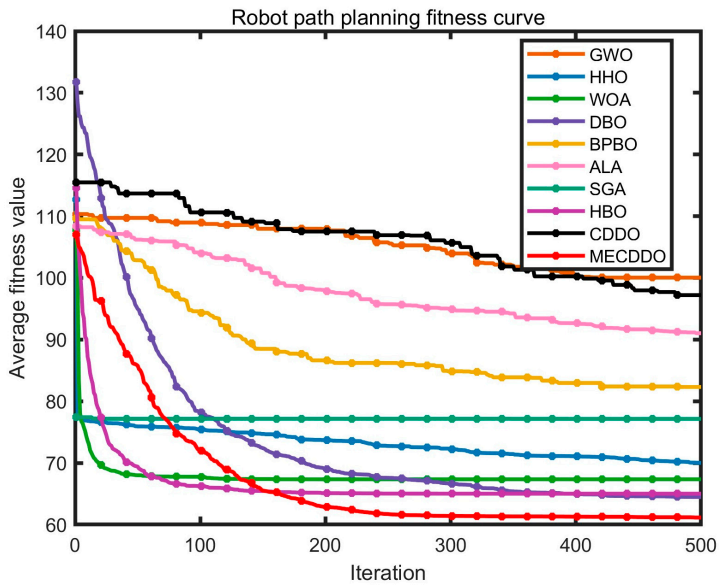


**Figure 15.** Fitness evolution across algorithms in the 40×40 grid environment.

Figure 14 presents the path planning results of ten algorithms in a 40×40 high-complexity grid environment, where obstacles are more densely distributed and the grid size is doubled, placing higher demands on global exploration and local obstacle-avoidance coordination. The trajectories generated by MECDDO demonstrate the core advantages of "short paths, low redundancy, and high safety." From the start (1,1) to the goal (40,40), only 3–4 necessary turns occur in dense obstacle regions (e.g., grids (15,15)–(25,25) and (30,30)–(35,35)), and the path quickly returns to the straight-line direction between start and goal without detours. Overall, the trajectory closely follows the optimal path trend, accurately navigating narrow passages (e.g., grids (20,20)–(25,25)) while maintaining a safe distance of more than one grid unit from surrounding obstacles, avoiding any collision risk. Comparative algorithms exhibit notable trajectory deficiencies. CDDO produces significant

detours in grids (10,10)–(15,20), resulting in a substantially longer path. GWO's path near grids (25,25)–(30,30) approaches obstacles too closely, requiring 8–9 additional direction adjustments to avoid collisions. HHO and SGA generally avoid obstacles but show redundant turns in regions such as (5,5)–(10,10) and (35,35)–(40,40), resulting in much lower path smoothness than MECDDO. Additionally, MECDDO's trajectory nodes are more evenly distributed, with smaller directional changes between adjacent grids, providing better kinematic adaptability for the robot and reducing energy consumption caused by frequent directional switches.

Table 10 quantitatively validates MECDDO's superior performance in the 40 × 40 grid environment, with key metrics highlighting its advantages in high-complexity scenarios. For average path length (Ave), MECDDO achieves 61.15516, ranking first, which is 3.32758 shorter than the second-ranked DBO (64.48274) and 36.0482 shorter than traditional CDDO (97.20336), representing a 37.19% reduction. Compared with the 20 × 20 grid environment (8.83% reduction), this improvement demonstrates that MECDDO's optimization advantage becomes even more pronounced as environmental complexity increases. Regarding extremal performance, MECDDO attains the shortest minimum path length (58.24101) among all algorithms, while its maximum path length (65.30184) exceeds the minimum by only 7.06083—much smaller than comparative algorithms. For example, GWO's maximum (129.7612) exceeds its minimum (67.29792) by 62.46328, and CDDO's maximum (121.1821) exceeds its minimum (74.81734) by 46.36476, indicating that MECDDO maintains stable path-planning precision even in highly complex environments, with lower sensitivity to environmental disturbances.

The standard deviation (Std) further confirms MECDDO's stability. Its Std value (1.815313) is only 11.3% of CDDO (16.02579) and 12.1% of GWO (15.01177). Even compared with the second-ranked DBO (3.528144), MECDDO is lower by 1.712831, indicating minimal fluctuation in path length across 30 independent trials, far surpassing other algorithms. In overall ranking, MECDDO holds a clear first place, with a ranking gap of 3.32758 over DBO—much larger than the gap in the 20×20 environment (0.03109)—further demonstrating that its comprehensive performance advantage is more pronounced in high-complexity grid scenarios.

Figure 15 illustrates the fitness (path length) evolution of ten algorithms in a 40×40 grid environment, highlighting MECDDO's advantages in convergence speed, accuracy, and stability. Regarding convergence speed, MECDDO exhibits a rapid decline during the early iterations (0–150), with the fitness value dropping below 75 by iteration 50 and approaching the optimal value (~61.15) by iteration 150. In contrast, CDDO requires 400 iterations to drop below 90, GWO remains above 100 even after 500 iterations, and HHO and SGA take 300 iterations to approach 70. This demonstrates MECDDO's ability to quickly lock onto high-quality path regions in high-complexity environments, significantly reducing computational time. In terms of convergence accuracy, MECDDO's fitness curve stabilizes completely after 200 iterations, with the final value consistently around 61.15 and no observable fluctuations. Meanwhile, CDDO, GWO, and other algorithms continue to exhibit significant oscillations in later iterations (e.g., CDDO fluctuates between 97 and 95 during iterations 400–500), indicating susceptibility to local optima and difficulty in consistently achieving the global optimal path.

Moreover, MECDDO's fitness curve remains below all comparative algorithms throughout the process, with the gap widening over iterations. At iteration 500, MECDDO's fitness value is 36.0482 lower than CDDO, 38.8891 lower than GWO, and 8.8599 lower than HHO, further confirming its superiority in path planning accuracy and convergence stability in high-complexity grid environments. These results are fully consistent with the quantitative data in Table 7 and the trajectory visualizations in Figure 12, collectively

demonstrating that MECDDO can efficiently and reliably plan short, safe paths for mobile robots in large-scale, high-complexity grids, underscoring its strong practical applicability.

Overall, MECDDO leverages the synergistic effects of adaptive cooperative search, multi-strategy dynamic selection, and global-optimum guided boundary repair to achieve multi-objective optimization of path "length–smoothness–safety," while simultaneously ensuring high convergence efficiency and stability. This provides a robust and efficient solution for mobile robot path planning in complex real-world environments.

It should be noted that the current path planning experiments are conducted in static grid environments. In dynamic or real-world environments where obstacles or targets may change, MECDDO would require integration with real-time sensing and replanning mechanisms. This limitation is acknowledged and will be addressed in future work.

Limitations of MECDDO in dynamic environments: The current path planning experiments are conducted in fixed grid environments. For dynamic scenarios (e.g., moving obstacles in fire-field rescue), MECDDO requires real-time path recalculation. The algorithm's fast convergence enables near-real-time adjustment, but the multi-strategy framework may increase computational latency in highly dynamic environments. Future improvements will focus on integrating a dynamic obstacle detection module and simplifying strategy selection in real-time scenarios to enhance adaptability.

Additionally, for multi-target path planning (sequencing of multiple waypoints), MECDDO can be extended by integrating a multi-objective optimization framework (e.g., non-dominated sorting) to balance path length, smoothness, and target visiting order, similar to the multi-objective sequencing method in mixed-model assembly lines.

## 5. Conclusions and Prospects

This study addresses the core limitations of the traditional Children's Drawing Development Optimization (CDDO) algorithm—namely, its tendency to fall into local optima, slow convergence, and insufficient information exchange—when applied to global optimization problems and mobile robot path planning. To overcome these issues, a Multi-Strategy Enhanced Children's Drawing Development Optimization (MECDDO) algorithm is proposed, and its superiority is validated through systematic numerical experiments and engineering applications. From the algorithmic improvement perspective, MECDDO achieves performance breakthroughs through three core strategies. First, an adaptive cooperative search strategy is introduced, integrating multi-source information from the global best, worst, and randomly selected individuals. Dynamic weighting coefficients guide individual updates, effectively expanding the exploration of the solution space and addressing the limitations of CDDO's traditional "encircling" search. Second, a multi-strategy adaptive selection mechanism comprising four differentiated strategies is constructed, dynamically adjusting selection probabilities based on the success rate of each strategy. This enables a dynamic balance between early-stage exploration and late-stage exploitation, overcoming the insufficient adaptability of a single strategy. Third, a global-optimum guided boundary repair strategy is designed, which intelligently returns out-of-bounds solutions to the feasible domain based on the global-best position, avoiding the loss of high-quality information caused by conventional truncation methods and improving local exploitation efficiency.

In numerical optimization tests based on the CEC2017 benchmark suite, MECDDO demonstrates outstanding precision and stability across 30-, 50-, and 100-dimensional spaces: in 30 dimensions, the mean fitness of F1 is reduced by seven orders of magnitude compared to CDDO, and in 100 dimensions, the performance degradation of F2 is only 1/200 of that of CDDO. Friedman tests further show that MECDDO ranks first in overall performance across all dimensions, with statistically significant differences from the second-best algorithm. In mobile robot path planning applications, MECDDO short-

ens the average path length by 8.83% in a 20 × 20 grid environment and by 37.19% in a 40 × 40 high-complexity grid. The planned trajectories involve no redundant turns while maintaining safe distances from obstacles, and the convergence speed is 2–3 times faster than comparative algorithms, fully demonstrating MECDDO's practical effectiveness in real-world engineering problems.

From a statistical perspective, the Friedman test results across all dimensional settings yielded $p$-values significantly lower than 0.05, confirming the existence of statistically significant differences among the compared algorithms. Furthermore, the Wilcoxon signed-rank test demonstrated that MECDDO achieved a dominant number of wins against all competing algorithms in 30D, 50D, 100D, and 20D experiments. These statistical findings rigorously validate that the observed performance improvements are not due to random fluctuations but stem from the effectiveness of the proposed multi-strategy enhancement framework. Therefore, MECDDO can be considered a statistically superior optimization algorithm for both numerical benchmark problems and real-world path planning applications.

Despite its advantages, MECDDO still has areas for improvement. In high-dimensional or large-population scenarios, multi-strategy statistics and Euclidean distance computations increase computational overhead by 15–20% compared with traditional CDDO, limiting real-time applicability. When obstacle density exceeds 60% in grid environments, the number of trajectory turns rises by 30% and the path optimization rate drops to 22.5%, indicating insufficient adaptability to extreme environments. Moreover, key algorithm parameters currently rely on trial-and-error tuning, requiring repeated adjustments in non-continuous or strongly constrained problems, reflecting a lack of self-adaptive parameter regulation.

Future work will focus on targeted enhancements: dimensionality reduction via principal component analysis (PCA) and sliding-window statistics to reduce computational complexity and meet real-time requirements; an obstacle-density awareness module to dynamically adjust search weights and optimize trajectory smoothness, improving adaptability in extreme scenarios; integration with reinforcement learning to form a closed-loop, self-adaptive parameter tuning mechanism; and extension to other applications such as UAV trajectory planning and smart grid scheduling to further validate algorithm generality and provide solutions for a broader range of engineering optimization problems.

**Author Contributions:** Conceptualization, Z.W. and S.W.; methodology, Z.W. and S.W.; software, Z.W. and S.W.; validation, Z.W. and S.W.; formal analysis, Z.W. and S.W.; investigation, Z.W. and S.W.; resources, S.Y. and G.W.; data curation, S.Y. and G.W.; writing original draft preparation, S.Y. and G.W.; writing review and editing, S.Y. and G.W.; visualization, S.Y. and G.W.; supervision, S.Y. and G.W.; funding acquisition, S.Y. and G.W. All authors have read and agreed to the published version of the manuscript.

**Funding:** Taizhou Science and Technology Plan Project (25gyb46), Key Project of Zhejiang Provincial Philosophy and Social Sciences (25WH04ZD).

**Data Availability Statement:** All data in this paper are included in the manuscript.

**Acknowledgments:** The authors would like to express their sincere gratitude to all those who contributed to the completion of this work.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Ying, W.; Wang, Z.; Li, H.; Du, S.; Zhao, M. Intelligent Ship Scheduling and Path Planning Method for Maritime Emergency Rescue. *Algorithms* **2024**, *17*, 197. [CrossRef]
2. Wu, X.; Zhang, Y.; Shi, M.; Li, P.; Li, R.; Xiong, N.N. An adaptive federated learning scheme with differential privacy preserving. *Future Gener. Comput. Syst.* **2022**, *127*, 362–372. [CrossRef]

3. Cao, L.; Wei, Q. SZOA: An Improved Synergistic Zebra Optimization Algorithm for Microgrid Scheduling and Management. *Biomimetics* **2025**, *10*, 664. [CrossRef]
4. Hu, W.; Ma, X. Optimization Algorithm of UAVs Task Assignment and Path Planning Based on Dynamic Cluster Particle Swarm Optimization. *IEEE Trans. Intell. Transp. Syst.* **2025**, *26*, 18157–18169. [CrossRef]
5. Huang, S.; Fu, Y.; Liu, D.; Chen, J.; He, L.; Zhang, J. DLC-NGO: An enhanced northern goshawk optimization for 3D UAV path planning in complex environments with multiple threats. *Clust. Comput.* **2025**, *28*, 677. [CrossRef]
6. Chen, B.; Hai, M.; Di, G.; Zhou, B.; Zhang, Q.; Wang, M.; Guo, Y. Enhanced Dung Beetle Optimizer-Optimized KELM for Pile Bearing Capacity Prediction. *Buildings* **2025**, *15*, 2654. [CrossRef]
7. Chen, Z.; Fu, B.; Yang, Y. A Novel Elite-Guided Hybrid Metaheuristic Algorithm for Efficient Feature Selection. *Biomimetics* **2025**, *10*, 747. [CrossRef] [PubMed]
8. Wang, H.; Zhang, X.; Xia, Y.; Wu, X. An intelligent blockchain-based access control framework with federated learning for genome-wide association studies. *Comput. Stand. Interfaces* **2022**, *84*, 103694. [CrossRef]
9. Wu, X.; Dong, J.; Bao, W.; Zou, B.; Wang, L.; Wang, H. Augmented Intelligence of Things for Emergency Vehicle Secure Trajectory Prediction and Task Offloading. *IEEE Internet Things J.* **2024**, *11*, 36030–36043. [CrossRef]
10. Fan, D.; Zhu, X.; Xiang, Z.; Lu, Y.; Quan, L. Dimension-Reduction Many-Objective Optimization Design of Multi-Mode Double-Stator Permanent Magnet Motor. *IEEE Trans. Transp. Electrif.* **2024**, *11*, 1984–1994. [CrossRef]
11. Fan, D.; Miao, D.; Shan, W.; Xiang, Z.; Zhu, X. Short-Circuit Fault Demagnetization Assessment and Optimization of Double-Electrical-Port Vernier Permanent Magnet Motor. *IEEE Trans. Ind. Appl.* **2025**, 1–10. [CrossRef]
12. Qi, H. A Unified Metric Architecture for AI Infrastructure: A Cross-Layer Taxonomy Integrating Performance, Efficiency, and Cost. *arXiv* **2025**, arXiv:2511.21772. [CrossRef]
13. Guo, S.; Song, Y.; Wu, Y.; Hu, J.; Lu, J.; Wang, C.; Jia, Z.; Hua, L. Microscopic mechanism of enhanced strength-plasticity synergy in pre-damaged TC11 titanium alloys via novel electroshock treatment. *J. Alloys Compd.* **2026**, *1052*, 186139. [CrossRef]
14. Milin, V.; Stanivuk, T.; Skoko, I.; Bulić, T. Dijkstra and A* Algorithms for Algorithmic Optimization of Maritime Routes and Logistics of Offshore Wind Farms. *J. Mar. Sci. Eng.* **2025**, *13*, 1863. [CrossRef]
15. Qi, H. Location-Robust Cost-Preserving Blended Pricing for Multi-Campus AI Data Centers. *arXiv* **2025**, arXiv:2512.14197. [CrossRef]
16. Qi, H.; Chunyu, Q. Modular Landfill Remediation for AI Grid Resilience. *arXiv* **2025**, arXiv:2512.19202. [CrossRef]
17. Sedeño-noda, A.; Colebrook, M. A Biobjective Dijkstra Algorithm. *Eur. J. Oper. Res.* **2019**, *276*, 106–118. [CrossRef]
18. Maristany de las Casas, P.; Kraus, L.; Sedeño-Noda, A.; Borndörfer, R. Targeted multiobjective Dijkstra algorithm. *Networks* **2023**, *82*, 277–298. [CrossRef]
19. Zhou, T.; Wei, W. Mobile robot path planning based on an improved ACO algorithm and path optimization. *Multimed. Tools Appl.* **2024**, *84*, 10899–10922. [CrossRef]
20. Duan, Z.; Zhang, Y.; Shao, Z.; Xu, Z.; Xiang, Z. Advanced robot path planning on rough terrain: A Q-learning-based multi-objective PSO algorithm. *Appl. Soft Comput.* **2025**, *184*, 113798. [CrossRef]
21. Qi, H.; Chunyu, Q. Waste-to-Energy-Coupled AI Data Centers: Cooling Efficiency and Grid Resilience. *arXiv* **2025**, arXiv:2512.24683. [CrossRef]
22. Yan, J.; Cheng, Y.; Zhang, F.; Li, M.; Zhou, N.; Jin, B.; Wang, H.; Yang, H.; Zhang, W. Research on multimodal techniques for arc detection in railway systems with limited data. *Struct. Health Monit.* **2025**. [CrossRef]
23. Wang, H.; Song, Y.; Yang, H.; Liu, Z. Generalized Koopman Neural Operator for Data-driven Modelling of Electric Railway Pantograph-catenary Systems. *IEEE Trans. Transp. Electrif.* **2025**, *11*, 14100–14112. [CrossRef]
24. Fu, Y.; Liu, D.; Chen, J.; He, L. Secretary bird optimization algorithm: A new metaheuristic for solving global optimization problems. *Artif. Intell. Rev.* **2024**, *57*, 123. [CrossRef]
25. Wang, T.-L.; Gu, S.-W.; Liu, R.-J.; Chen, L.-Q.; Wang, Z.; Zeng, Z.-Q. Cuckoo catfish optimizer: A new meta-heuristic optimization algorithm. *Artif. Intell. Rev.* **2025**, *58*, 326. [CrossRef]
26. Hasan, D.O.; Mohammed, H.M.; Abdul, Z.K. Griffon vultures optimization algorithm for solving optimization problems. *Expert Syst. Appl.* **2025**, *276*, 127206. [CrossRef]
27. Wang, L.; Cao, Q.; Zhang, Z.; Mirjalili, S.; Zhao, W. Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems. *Eng. Appl. Artif. Intell.* **2022**, *114*, 105082. [CrossRef]
28. Khishe, M.; Mosavi, M.R. Chimp optimization algorithm. *Expert Syst. Appl.* **2020**, *149*, 113338. [CrossRef]
29. Hassan, M.H.; Kamel, S. Supercell thunderstorm algorithm (STA): A nature-inspired metaheuristic algorithm for engineering optimization. *Neural Comput. Appl.* **2025**, *37*, 7207–7260. [CrossRef]
30. Sowmya, R.; Premkumar, M.; Jangir, P. Newton-Raphson-based optimizer: A new population-based metaheuristic algorithm for continuous optimization problems. *Eng. Appl. Artif. Intell.* **2024**, *128*, 107532. [CrossRef]
31. Zhang, R.; Li, S.; Ding, Y.; Qin, X.; Xia, Q. UAV Path Planning Algorithm Based on Improved Harris Hawks Optimization. *Sensors* **2022**, *22*, 5232. [CrossRef]

32. Zhang, C.; Yang, Y.; Chen, W. Multi-strategy ensemble wind driven optimization algorithm for robot path planning. *Math. Comput. Simul.* **2024**, *231*, 144–159. [CrossRef]
33. Dai, Y.; Yu, J.; Zhang, C.; Zhan, B.; Zheng, X. A novel whale optimization algorithm of path planning strategy for mobile robots. *Appl. Intell.* **2022**, *53*, 10843–10857. [CrossRef]
34. Wang, R.-B.; Wang, W.-F.; Xu, L.; Pan, J.-S.; Chu, S.-C. An Adaptive Parallel Arithmetic Optimization Algorithm for Robot Path Planning. *J. Adv. Transp.* **2021**, *2021*, 3606895. [CrossRef]
35. Wang, Y.; Tong, K.; Fu, C.; Wang, Y.; Li, Q.; Wang, X.; He, Y.; Xu, L. Hybrid path planning algorithm for robots based on modified golden jackal optimization method and dynamic window method. *Expert Syst. Appl.* **2025**, *282*, 127808. [CrossRef]
36. Deng, X.; He, D.; Qu, L. A Multi-strategy Enhanced Arithmetic Optimization Algorithm and Its Application in Path Planning of Mobile Robots. *Neural Process. Lett.* **2024**, *56*, 18. [CrossRef]
37. Wu, D.; Du, K. A hybrid optimization algorithm of ACO and RRT for solving mobile robot path planning. *Meas. Sci. Technol.* **2025**, *36*, 076215. [CrossRef]
38. Abdulhameed, S.; Rashid, T.A. Child Drawing Development Optimization Algorithm Based on Child's Cognitive Development. *Arab. J. Sci. Eng.* **2022**, *47*, 1337–1351. [CrossRef]
39. Ameen, A.A.; Rashid, T.A.; Askar, S. CDDO-HS:Child Drawing Development Optimization Harmony Search Algorithm. *arXiv* **2023**, arXiv:2305.09615. [CrossRef]
40. Kari Balakrishnan, A.; Chellaperumal, A.; Lakshmanan, S.; Vijayakumar, S. A novel efficient data storage and data auditing in cloud environment using enhanced child drawing development optimization strategy. *Netw. Comput. Neural Syst.* **2025**, 1–41. [CrossRef] [PubMed]
41. Chen, Y.; Zhang, J.; Rauf, M.; Mumtaz, J.; Huang, S. Dynamic scheduling of hybrid flow shop problem with uncertain process time and flexible maintenance using NeuroEvolution of Augmenting Topologies. *IET Collab. Intell. Manuf.* **2024**, *6*, e12119. [CrossRef]
42. Rauf, M.; Guan, Z.; Sarfraz, S.; Mumtaz, J.; Shehab, E.; Jahanzaib, M. Multi-Criteria Inventory Classification Based on Multi-Criteria Decision-Making (MCDM) Technique. In *Advances in Manufacturing Technology XXXII: Proceedings of the 16th International Conference on Manufacturing Research, incorporating the 33rd National Conference on Manufacturing Research, Skövde, Sweden, 11–13 September 2018*; SAGE Publications Limited: Thousand Oaks, CA, USA, 2018.
43. Chen, M.; Feng, C.; Cheng, R. MetaDE: Evolving Differential Evolution by Differential Evolution. *IEEE Trans. Evol. Comput.* **2025**, *30*, 141–155. [CrossRef]
44. Zhang, S.X.; Liu, Y.H.; Hu, X.R.; Luo, J.T.; Zheng, L.M.; Zheng, S.Y. Differential evolution with dimensionally adaptive inheritance. *Eng. Appl. Artif. Intell.* **2025**, *163*, 112587. [CrossRef]
45. Zhu, Y.; Zhang, M.; Huang, Q.; Wu, X.; Wan, L.; Huang, J. Secretary bird optimization algorithm based on quantum computing and multiple strategies improvement for KELM diabetes classification. *Sci. Rep.* **2025**, *15*, 3774. [CrossRef]
46. Zhang, Y.; Wang, J.; Zhang, X.; Wang, B. ACPOA: An Adaptive Cooperative Pelican Optimization Algorithm for Global Optimization and Multilevel Thresholding Image Segmentation. *Biomimetics* **2025**, *10*, 596. [CrossRef] [PubMed]
47. Hou, X.; Li, Y.; Liu, Z. An Enhanced Polar Lights Optimization Algorithm with Symmetry Mechanisms for Global Optimization and Its Application to Wind Power Forecasting. *Symmetry* **2025**, *18*, 61. [CrossRef]
48. Wu, G.; Mallipeddi, R.; Suganthan, P. Problem Definitions and Evaluation Criteria for the CEC 2017 Competition and Special Session on Constrained Single Objective Real-Parameter Optimization. 2016. Available online: http://www.kiphub.com/paper/666152c71a6f2cfc2305a5b0 (accessed on 28 February 2026).
49. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
50. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris hawks optimization: Algorithm and applications. *Future Gener. Comput. Syst.* **2019**, *97*, 849–872. [CrossRef]
51. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
52. Zhu, F.; Li, G.; Tang, H.; Li, Y.; Lv, X.; Wang, X. Dung beetle optimization algorithm based on quantum computing and multi-strategy fusion for solving engineering problems. *Expert Syst. Appl.* **2024**, *236*, 121219. [CrossRef]
53. Ghasemi, M.; Akbari, M.A.; Zare, M.; Mirjalili, S.; Deriche, M.; Abualigah, L.; Khodadadi, N. Birds of prey-based optimization (BPBO): A metaheuristic algorithm for optimization. *Evol. Intell.* **2025**, *18*, 88. [CrossRef]
54. Xiao, Y.; Cui, H.; Khurma, R.A.; Castillo, P.A. Artificial lemming algorithm: A novel bionic meta-heuristic technique for solving real-world engineering optimization problems. *Artif. Intell. Rev.* **2025**, *58*, 84. [CrossRef]
55. Tian, A.-Q.; Liu, F.-F.; Lv, H.-X. Snow Geese Algorithm: A novel migration-inspired meta-heuristic algorithm for constrained engineering optimization problems. *Appl. Math. Model.* **2024**, *126*, 327–347. [CrossRef]
56. Ouertani, M.W.; Manita, G.; Korbaa, O. Hannibal Barca optimizer: The power of the pincer movement for global optimization and multilevel image thresholding. *Clust. Comput.* **2025**, *28*, 482. [CrossRef]
57. Mac, T.T.; Copot, C.; Tran, D.T.; De Keyser, R. Heuristic approaches in robot path planning: A survey. *Robot. Auton. Syst.* **2016**, *86*, 13–28. [CrossRef]

58. Felsner, K.; Schlachter, K.; Zambal, S. Robotic Coverage Path Planning for Ultrasonic Inspection. *Appl. Sci.* **2021**, *11*, 10512. [CrossRef]
59. Zhang, Z.; Xu, X.; Yue, F.; Ba, Y. Robot path planning based on concept lattice. *Int. J. Approx. Reason.* **2022**, *153*, 87–103. [CrossRef]
60. Gao, Y.; Jiang, Q.; Wang, M.; Dong, X. Advances in Path-Planning Algorithms for Agricultural Robots. *J. Field Robot.* **2025**, *43*, 89–145. [CrossRef]

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.