Cluster Computing
https://doi.org/10.1007/s10586-024-04600-6

Gene-targeting multiplayer battle game optimizer for large-scale global optimization via cooperative coevolution

Rui Zhong[1] · Jun Yu[2]

Received: 18 March 2024 / Revised: 18 May 2024 / Accepted: 23 May 2024

Abstract
This paper proposes an efficient variant of the multiplayer battle game optimizer (MBGO) named gene-targeting MBGO (GTMBGO). We simplify the original MBGO and introduce a well-performed gene-targeting search operator to strengthen its optimization performance. Comprehensive numerical experiments on CEC2017 and CEC2022 benchmark functions confirm the efficiency and effectiveness of GTMBGO compared with state-of-the-art optimizers, and the ablation experiments are also implemented to investigate the contribution of proposed strategies independently. Additionally, we extend the proposed GTMBGO to solve large-scale optimization problems (LSOPs). Since the existence of the curse of dimensionality, LSGO challenges the performance of optimizers severely. Inspired by the divide-and-conquer, the cooperative coevolution (CC) framework decomposes the LSOP and optimizes them alternatively, which provides a potential avenue to solve LSOPs. Based on the efficient recursive differential grouping (ERDG) decomposition method, we propose an enhanced version named ERDG with maximum volume (ERDG$_k$) and incorporate it with GTMBGO to deal with LSOPs. The experimental results and statistical analyses on CEC2013 LSGO benchmark functions verify the competitiveness of GTMBGO-ERDG$_k$ when compared with state-of-the-art methodologies specifically designed for LSOPs.

Keywords Gene-targeting operator · Multiplayer battle game optimizer (MBGO) · Large-scale optimization problems (LSOPs) · Cooperative coevolution (CC) · Efficient recursive differential grouping with maximum volume (ERDG$_k$)

1 Introduction

In recent years, the field of optimization has undergone a remarkable surge of interest, particularly in the application of metaheuristic algorithms (MAs) across a diverse array of domains. These problems, often characterized by their non-convexity, multi-modality, non-differentiability, and intricate fitness landscapes, pose formidable challenges for conventional optimization techniques [1, 2]. MAs, inspired by the dynamics of natural processes such as biological evolution, natural selection, and swarm intelligence, have emerged as powerful tools adept at navigating these complex landscapes and overcoming the associated challenges [3–5].

MAs represent a paradigm shift in optimization methodologies, leveraging the principles of exploration and exploitation to iteratively pursue high-quality solutions. Renowned for their adaptability, robustness, and scalability, these algorithms prove highly effective in addressing real-world optimization challenges across a diverse array of domains, including engineering [6], finance [7], logistics [8], and healthcare [9]. However, the challenge of optimization intensifies as problem dimensions escalate, leading to an exponential expansion of the search space known as the curse of dimensionality [10]. This phenomenon significantly hampers the performance of conventional MAs, rendering their utilization in large-scale optimization problems (LSOPs) a formidable challenge that has garnered considerable attention from researchers and scholars [11–13].

Presently, the predominant methodologies for addressing evolutionary LSOPs are twofold: (1) employing the cooperative coevolution (CC) framework, and (2) devising more efficient search strategies. Inspired by the concept of

✉ Jun Yu
yujun@ie.niigata-u.ac.jp

Rui Zhong
rui.zhong.u5@elms.hokudai.ac.jp

1 Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Japan

2 Institute of Science and Technology, Niigata University, Niigata, Japan

divide-and-conquer, CC divides LSOPs into multiple small- and median-scale problems, optimizing them alternately to mitigate the adverse effects of the curse of dimensionality directly. The crux of the CC framework lies in the decomposition strategy, which entails effectively and efficiently identifying interactions between decision variables and suitably forming sub-components. This aspect is pivotal for the successful implementation of CC. The review for the rich history and development of decomposition techniques are not the concentration in this paper, and interested readers can refer to [13–16]. Concurrently, enhanced versions of genetic algorithm (GA) [17, 18], differential evolution (DE) [19, 20], particle swarm optimization (PSO) [21, 22], competitive swarm optimizer (CSO) [23–25], and other swarm intelligence approaches [26, 27] have emerged to tackle LSOPs.

With the rapid development of the MA community, many newly proposed MAs have demonstrated their efficiency and effectiveness in solving LSOPs [28, 29]. As a recently introduced MA, multiplayer battle game optimizer (MBGO) [30] demonstrates superior characteristics, including rapid optimization convergence, high convergence accuracy, and the ability to escape from local optima. Thanks to its advantages, we believe that MBGO holds great potential for addressing these challenges. Therefore, we propose an enhanced version named gene-targeting MBGO (GTMBGO) in this study. The gene-targeting operator was originally developed in gene-targeting differential evolution (GTDE) [31], which is specifically designed for solving LSOPs, and the effectiveness and efficiency of this operator have been demonstrated across various problem domains. Besides, a comprehensive investigation in [32] has revealed the unbalanced exploration and exploitation search ability in MBGO, which is due to the designed search operators in the movement phase. Therefore, we further simplify the MBGO by reducing the operators in the movement phase, and integrate the efficient gene targeting operator into MBGO to propose the GTMBGO. We begin by conducting a thorough evaluation experiment of GTMBGO's performance in solving low- and median-dimensional problems. Additionally, we observe a phenomenon within the CC framework where differential grouping (DG)-based approaches assign all decision variables into a single group for simultaneous optimization in some separable functions, most overlapping functions, and fully non-separable functions. While this decomposition may achieve high decomposition accuracy, it cannot significantly accelerate optimization in these instances since the curse of dimensionality still exists. To address this limitation, we propose an efficient recursive differential grouping with maximum volume (ERDG$_k$) approach. In ERDG$_k$, if the volume of a sub-component exceeds a predefined threshold, it is partitioned into smaller-scale sub-components. By integrating GTMBGO with ERDG$_k$, we present GTMBGO-ERDG$_k$, a novel approach tailored for LSOPs. Specifically, the main contributions of this paper can be summarized as follows:

- We propose a novel gene-targeting multiplayer battle game optimizer (GTMBGO) for numerical optimization. The efficient gene-targeting operator is integrated into GTMBGO to enhance the exploitative search capacity.
- We notice the shortage of ERDG in overlapping and fully non-separable functions and propose an ERDG with maximum volume (ERDG$_k$).
- We incorporate GTMBGO and ERDG$_k$ to solve LSOPs.
- The experimental results and statistical analysis confirm the efficiency and effectiveness of our proposed methodology.

The rest of this paper is organized as follows: Sect. 2 covers the related works, Sect. 3 introduces the proposed GTMBGO and ERDG$_k$ in detail, Sect. 4 presents numerical experiments and statistical analyses, and the performance of our proposal is discussed in Sect. 5. Finally, Sect. 6 concludes this paper.

2 Related works and preliminaries

2.1 Cooperative Coevolution (CC)

The cooperative coevolution (CC) framework was first proposed in [17] and has become a popular technique in solving LSOPs. A general CC framework involves three procedures: decomposition, optimization, and combination.

- **Decomposition**: The decision variables in the original LSOP are divided into multiple sub-components with a specific decomposition algorithm.
- **Optimization**: Optimization techniques such as DE and PSO are applied to optimize the sub-components independently and alternatively. Commonly, a sub-solution cannot form a complete solution, and the context vector [33] participates in the sub-solution evaluation.
- **Combination**: The sub-solutions derived from their respective sub-components are combined to form a complete solution, representing the optimal solution of the original problem identified through the CC framework.

2.2 Multiplayer battle game optimizer (MBGO)

Multiplayer battle game optimizer (MBGO) is a human-inspired MA designed with inspiration drawn from the dynamics of multiplayer battle games. MBGO divides the optimization process into two distinct phases: the movement phase and the battle phase. This approach mimics the strategic decision-making processes in games, providing a more nuanced and efficient optimization technique.

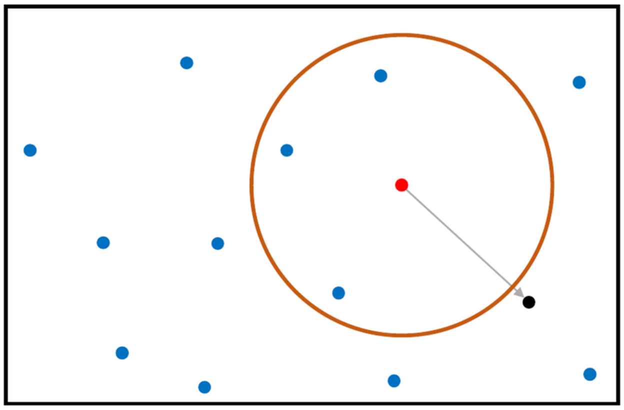


Fig. 1 A demonstration of the safe zone in MBGO

Movement phase: In the movement phase, MBGO mimics the strategic movements of players within the game environment, influenced by the concept of the safe zone-a fundamental mechanism observed in many multiplayer battle games. This safe zone serves as a pivotal factor to guide the movement of individuals in MBGO. Equation (1) is employed to determine the position of the safe zone.

$$R = (\|X_{best} - X_{worst} + \epsilon\|) \cdot \delta \tag{1}$$

MBGO utilizes the Euclidean distance between the current best solution X_{best} and the current worst solution X_{worst} to determine the safe zone. ϵ in Eq. (1) is a tiny value to avoid the radius of the safe zone being zero, and δ following a uniform distribution $U(0.8, 1.2)$ is a scaling factor to adjust the radius. A visual demonstration is shown in Fig. 1.

The blue, red, and black points denote solutions, the current best solution, and the current worst solution, respectively. The grey line with an arrow indicates the radius of the safe zone, and the brown circle represents the safe zone. The solutions within the safe zone adopt Eq. (2) to construct offspring.

$$X_{new} = X_i + X_{best} \cdot \sin(2\pi r) \tag{2}$$

where r is a random number within the range of (0, 1). For external individuals, Eq. (3) is applied to construct offspring.

$$X_{new,j} = \begin{cases} X_{i,j} + \theta, \ if \ r < 0.5 \\ X_{i,j} + (X_{best,j} - X_{i,j}) \cdot r, \ otherwise \end{cases} \tag{3}$$

where θ is a random value following the standard normal distribution $N(0, 1)$. Specifically, for each independent dimension j, each scheme in Eq. (3) has an equal probability of being selected for offspring individual construction. Notably, the movement phase incorporates an internal greedy selection mechanism. This means that following the construction of a new offspring individual, the selection operator is activated, allowing the superior offspring individual to survive.

Battle phase: The battle phase simulates various player behaviors when encountering random enemies in the game. While players' behavior may differ during the game, the ultimate objective remains survival until the end of the game and the elimination of enemies as much as possible. MBGO simplifies real-game observations and imposes idealized constraints. Each individual is assumed to face only one random enemy, and the fitness value represents the ability of the player. Eqs (4) and (5) demonstrate mathematical models when confronting stronger and weaker enemies, respectively

$$X_{new,j} = \begin{cases} X_{i,j} + \mathbf{dir}_j \cdot r, \ if \ r < 0.5 \\ X_{enemy,j} + \mathbf{dir}_j \cdot r, \ otherwise \end{cases} \tag{4}$$

$$X_{new} = X_i + \mathbf{dir} \cdot \cos(2\pi r) \tag{5}$$

where **dir** indicates the differential vector between the i^{th} individual X_i and the random enemy X_{enemy}, as formulated in Eq. (6).

$$\mathbf{dir} = \begin{cases} X_i - X_{enemy} \ if \ X_i \ has \ a \ better \ fitness \ value \\ X_{enemy} - X_i, \ otherwise \end{cases} \tag{6}$$

In summary, the pseudocode of MBGO is presented in Algorithm 1.

2.3 Recursive Differential Grouping (RDG) and Efficient RDG (ERDG)

Through the mathematical definition, if the second-order partial derivative between two decision variables x_i and x_j is 0, these two variables are additively separable, which is formulated in Eq. (7).

$$\begin{aligned} &if \ \frac{\partial f^2(x)}{\partial x_i \partial x_j} = 0 \\ &then \ x_i \ and \ x_j \ are \ separable \end{aligned} \tag{7}$$

However, most studies in the evolutionary computation (EC) community focus on black-box problems, which means that accurate gradient information is not accessible. Therefore, differential grouping (DG) [34] approximately simulates the computation of gradient using Eq. (8).

$$\begin{aligned} &\Delta_1 = f(s_{ij}) - f(s_j), \Delta_2 = f(s_i) - f(s) \\ &if \ |\Delta_1 - \Delta_2| < \varepsilon \\ &then \ x_i \ and \ x_j \ are \ separable \end{aligned} \tag{8}$$

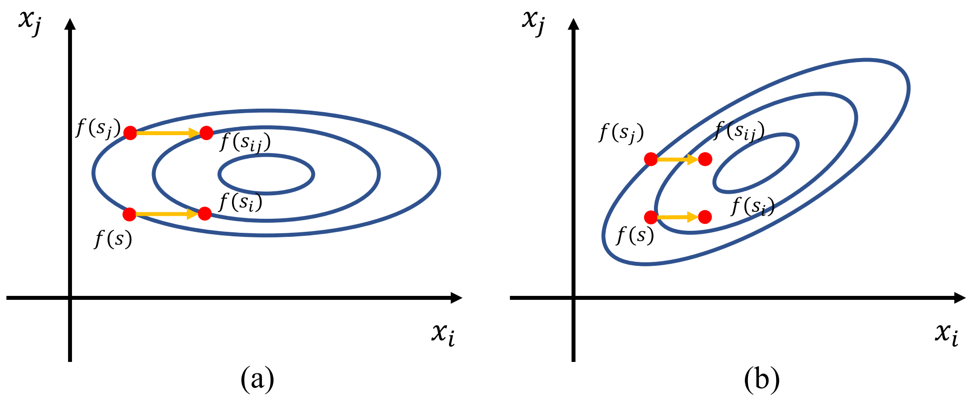


Fig. 2 A demonstration of DG. **a** DG in separable decision variables. **b** DG in non-separable decision variables

Algorithm 1 MBGO [30]

```
Require: Population size: $N$, Dimension: $D$, Maximum iteration: $T_{max}$
Ensure: Optimum: $X_{best}$
 1: Population initialization
 2: $X_{best} \leftarrow$ best($R$)
 3: $t \leftarrow 0$
 4: while $t < T_{max}$ do
 5:     • Movement phase
 6:     for $i = 0$ to $N$ do
 7:         Locate the safe zone using Eq. (1)
 8:         if $X_i$ is within the safe zone then
 9:             Construct $X_{new}$ using Eq. (2)
10:         else
11:             Construct $X_{new}$ using Eq. (3)
12:         end if
13:         Internal greedy selection
14:     end for
15:     • Battle phase
16:     for $i = 0$ to $N$ do
17:         Select a random enemy $X_{enemy}$ for $X_i$
18:         if $X_{enemy}$ has a better fitness value then
19:             Construct $X_{new}$ using Eq. (4)
20:         else
21:             Construct $X_{new}$ using Eq. (5)
22:         end if
23:         Internal greedy selection
24:     end for
25:     $X_{best} \leftarrow$ best($R$)
26:     $t \leftarrow t + 1$
27: end while
28: return $X_{best}$
```

where $s = \{x_1, ..., x_i, ..., x_j, ..., x_D\}$, $s_i = \{x_1, ..., x_i + \epsilon, ..., x_j, ..., x_D\}$, $s_j = \{x_1, ..., x_i, ..., x_j + \epsilon, ..., x_D\}$, and $s_{ij} = \{x_1, ..., x_i + \epsilon, ..., x_j + \epsilon, ..., x_D\}$ are four sample points in the fitness landscape. ϵ represents the perturbation, and ε is a tiny threshold (e.g. 10^{-3} is recommended in DG) to control the allowable error. Here, a visual demonstration is presented in Fig. 2.

As the pioneer in interaction identification, DG employs four fitness evaluations (FEs) to ascertain the interaction between a pair of decision variables. Despite adopting a mechanism for reusing fitness values, DG still suffers from a computational budget of $\frac{N}{2m}(m + N - 2)$, where N is the dimension size and m is the size of the sub-components. In an extreme instance when a 1000-D problem is fully separable, DG costs 1,001,000 FEs for the complete interaction identification. Fortunately, the appearance of Recursive DG (RDG) [35] alleviates this severe computational resource consumption. Rather than detecting the interaction variable to variable, RDG proposed a binary search based subset-to-subset interaction identification principle and significantly reduced the theoretical computational cost from $O(N^2)$ to $O(N \log N)$. Specifically, supposing that $X_1 \subset X$ and $X_2 \subset X$ are two mutually exclusive subsets of decision variables (i.e., $X_1 \cap X_2 = \emptyset$). If there are two unit vectors $\mathbf{u}_1 \in U_{X_1}$, $\mathbf{u}_2 \in U_{X_2}$, two arbitrary numbers $l_1, l_2 > 0$, and a candidate solution $\mathbf{x}^*$ in the search space to satisfy Eq. (9).

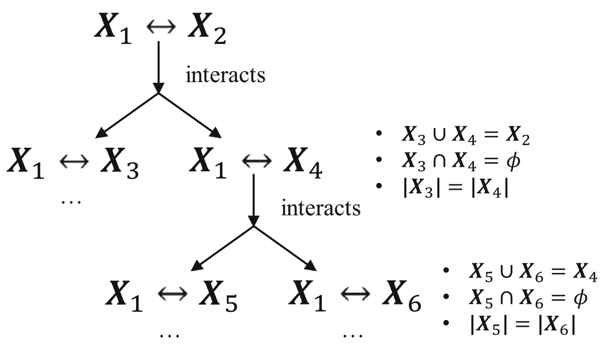


Fig. 3 A demonstration of procedures in RDG

$$f(\mathbf{x}^* + l_1\mathbf{u}_1 + l_2\mathbf{u}_2) - f(\mathbf{x}^* + l_2\mathbf{u}_2) \neq f(\mathbf{x}^* + l_1\mathbf{u}_1) - f(\mathbf{x}^*) \tag{9}$$

Then, interactions exist between decision variables in subsets X_1 and X_2. Therefore, RDG further decomposes X_2 into two mutually exclusive subsets X_3 and X_4 to determine the interaction until all decision variables are well-assigned. Figure 3 demonstrates the procedures of RDG.

Based on the mechanism of RDG, Yang et al. [36] introduced Efficient RDG (ERDG), which maintains an identical theoretical computational complexity while demonstrating reduced consumption of FEs in practice. Specifically, ERDG achieves this efficiency by using historical information to avoid redundant interaction examinations in RDG. From Fig. 3, variable subset X_3 and X_4 are mutually exclusive and $X_3 \cup X_4 = X_2$. Therefore, Eq. (9) can be transformed to Eq. (10)

$$f(\mathbf{x}^* + l_1\mathbf{u}_1 + l_2(\mathbf{u}_3 + \mathbf{u}_4)) - f(\mathbf{x}^* + l_2(\mathbf{u}_3 + \mathbf{u}_4)) \neq f(\mathbf{x}^* + l_1\mathbf{u}_1) - f(\mathbf{x}^*) \tag{10}$$

Here, ERDG separates the components in Eq. (10) and defines

$$\begin{aligned} \Delta_1 &= f(\mathbf{x}^* + l_1\mathbf{u}_1 + l_2(\mathbf{u}_3 + \mathbf{u}_4)) - f(\mathbf{x}^* + l_2(\mathbf{u}_3 + \mathbf{u}_4)) \\ \Delta_2 &= f(\mathbf{x}^* + l_1\mathbf{u}_1) - f(\mathbf{x}^*) \end{aligned} \tag{11}$$

Subsequently, if X_1 interacts with X_2, then X_2 will be divided into X_3 and X_4, and the interaction identification between X_1 and X_3 can be calculated using Eq. (12).

$$\begin{aligned} \Delta_1^{'} &= f(\mathbf{x}^* + l_1\mathbf{u}_1 + l_2\mathbf{u}_3) - f(\mathbf{x}^* + l_2\mathbf{u}_3) \\ \Delta_2^{'} &= f(\mathbf{x}^* + l_1\mathbf{u}_1) - f(\mathbf{x}^*) = \Delta_2 \end{aligned} \tag{12}$$

Similarly, the formulation of the interaction identification between X_1 and X_4 is presented in Eq. (13).

$$\begin{aligned} \Delta_1^{''} &= f(\mathbf{x}^* + l_1\mathbf{u}_1 + l_2\mathbf{u}_4) - f(\mathbf{x}^* + l_2\mathbf{u}_4) \\ \Delta_2^{''} &= \Delta_2 \end{aligned} \tag{13}$$

Reasonably, ERDG infers that if $(\Delta_1 - \Delta_2) = (\Delta_1^{'} - \Delta_2^{'})$, then X_1 and X_4 are separable; otherwise, X_1 interacts with X_4, and the computation of $\Delta_1^{''}$ in Eq. (13) is unnecessary. Through this intelligent computation technique, ERDG reduces the necessary FEs to an average of 7620 on CEC2013 benchmark functions.

3 Our proposal: GTMBGO-ERDG$_k$

This section contains two sub-sections: Sect. 3.1 details our proposed optimizer gene targeting multiplayer battle game optimizer (GTMBGO), and Sect. 3.2 introduces our proposed decomposition method efficient recursive differential grouping with maximum volume (ERDG$_k$).

3.1 Gene targeting multiplayer battle game optimizer (GTMBGO)

We begin by illustrating the operational framework of the proposed GTMBGO, as presented in Fig. 4. Following a structure akin to many swarm-based optimization approaches, GTMBGO commences by initializing the population. It then proceeds into the primary iteration loop, initializing the current iteration variable $t = 0$ and individual index $i = 0$. During each iteration, GTMBGO identifies whether the current individual X_i is consistent with the current best solution encountered so far, denoted as X_{best}. If so, the gene-targeting operator is invoked to produce offspring individuals. Conversely, if X_i is not the current best solution X_{best}, it constructs the offspring individual using the Eqs. (4), (5), and (6). Subsequently, an internal greedy selection mechanism is promptly activated to update the population. This mechanism ensures that any superior knowledge collected through the efficient search operators is swiftly integrated into the swarm. This iterative optimization continues until the optimal solution is found or the computational budget is exhausted. In the following context, we will introduce the design of GTMBGO in detail.

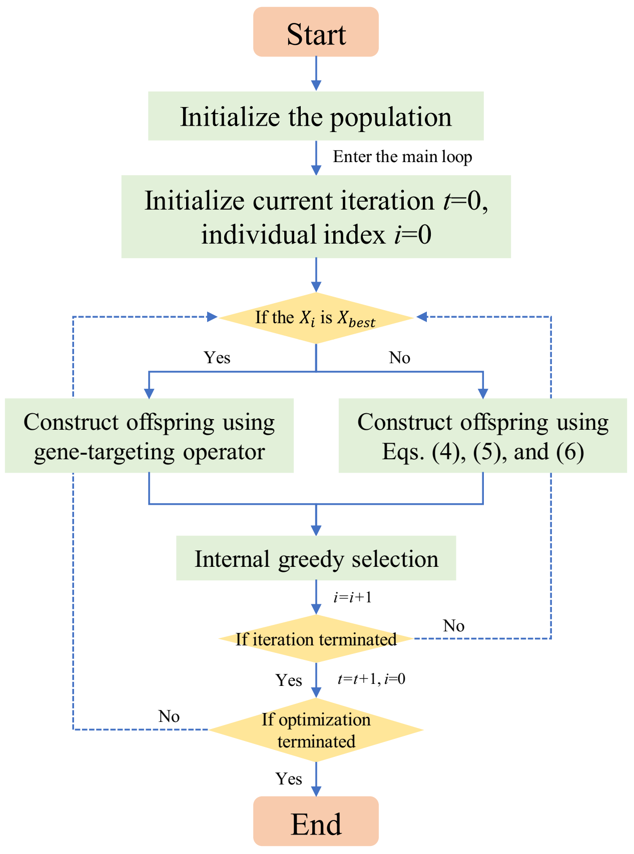

Start
Initialize the population
Enter the main loop
Initialize current iteration t=0, individual index i=0
If the X_i is X_{best}
Yes
No
Construct offspring using gene-targeting operator
Construct offspring using Eqs. (4), (5), and (6)
Internal greedy selection
i=i+1
If iteration terminated
No
Yes
t=t+1, i=0
No
If optimization terminated
Yes
End

Fig. 4 The flowchart of GTMBGO

As the comprehensive investigation in Zhong et al. [32], the performance of MBGO is severely deteriorated by the search operators designed in the movement phase through ablation analysis. Therefore, the motivation of our proposal is to simplify the search operators in the movement phase and integrate the efficient gene targeting operator into our proposal. Subsequently, the gene-targeting operator is introduced as follows.

The gene-targeting operator was originally proposed in gene-targeting DE (GTDE) [31] to simulate the genetic information interaction in living organisms. Specifically, after identifying the location of the causative gene, the gene-targeting technique adjusts it by constructing a homologous targeting vector and inserting it into the embryonic stem cell, resulting in a mutated embryonic stem cell. Targeting the causative gene for modification enables the mutated embryonic stem cell to express the desired trait more effectively. Figure 5 demonstrates an example of the gene-targeting operator.

Drawing inspiration from the gene-targeting technique, the gene-targeting operator probabilistically targets and modifies the bottleneck dimensions of individuals. This operator comprises three primary steps: targeting the causative genes, constructing the homologous targeting vector, and inserting the vector into the embryonic stem cell.

Targeting the causative genes: The gene-targeting operator utilizes the Monte Carlo method to sample the causative genes (i.e. the bottleneck dimensions), as presented in Algorithm 2.

Algorithm 2 Targeting the causative genes

```
Require: Individual: $X_i$, Dimension: $D$
1: for $j = 0$ to $D$ do
2:     $r \leftarrow$ rand()
3:     $P_j \leftarrow \mathbf{N}(0.01, 0.01)$
4:     if $r < P_j$ then
5:         $X_{i,j}$ is determined as the causative gene
6:     end if
7: end for
```

The function **rand**() generates a random number in the range of (0, 1) uniformly. The advantage of employing the Monte Carlo method lies in its capability to assign a probability to each dimension as the potential causative gene. This approach ensures a balanced evolution across different dimensions, thereby fostering global convergence on all dimensions.

Constructing the homologous targeting vector: The gene-targeting technique constructs the homologous targeting vector by combining two mutation operators in DE: DE/best/1 and DE/r-best/1, as formulated in Eqs. (14) and (15), respectively.

$$V_{i,j} = X_{best,j} + F \cdot (X_{r1,j} - X_{r2,j}) \tag{14}$$

$$V_{i,j} = X_{best,j} + F \cdot (X_{r1,j} - X_{rand,j}) \tag{15}$$

DE/r-best/1 is a variant of the DE/best/1 where an additional random individual X_{rand} participates in the mutation. F is the scaling factor sampled from a normal distribution with the expectation of 0.5 and the standard deviation of 0.1. Algorithm 3 demonstrates the pseudocode of the homologous targeting vector construction.

Algorithm 3 Constructing the homologous targeting vector

```
Require: Individual: $X_i$, Dimension: $D$
1: $F \leftarrow \mathbf{N}(0.5, 0.1)$
2: Randomly sample two distinct individuals $X_{r1}$ and $X_{r2}$
3: for $j = 0$ to $D$ do
4:     $r \leftarrow$ rand()
5:     if $r < P_m$ ($P_m = 0.01$) then
6:         Randomly sample an additional individual $X_{rand}$
7:         Construct $V_{i,j}$ using Eq. (15)
8:     else
9:         Construct $V_{i,j}$ using Eq. (14)
10:    end if
11:    Ensure $V_{i,j}$ is within the search space
12: end for
```

Inserting the vector into the embryonic stem cell: Since the gene-targeting operator is specifically designed for constructing the offspring for the current best individual X_{best}, this operator aims to insert the knowledge from the homologous targeting vector into the embryonic stem cell, as demonstrated in Algorithm 4.

Algorithm 4 Inserting the vector into the embryonic stem cell

```
Require: Best individual: $X_{best}$, Homologous targeting vector: $V_i$,
    Dimension: $D$
1: for $j = 0$ to $D$ do
2:     if the $j^{th}$ dimension is the causative gene then
3:         $X'_{best,j} = V_{i,j}$
4:     else
5:         $X'_{best,j} = X_{best,j}$
6:     end if
7: end for
```

In summary, the whole steps of the gene-targeting operator for one time are demonstrated in Algorithm 5. This operator locates and modifies the bottleneck dimensions of the best-performing individual in each iteration, offering informative guidance to the evolution and progressively promoting the population toward the global optimum.

Conclusively, the proposed GTMBGO is presented in Algorithm 6.

3.2 Efficient recursive differential grouping with maximum volume (ERDG$_k$)

The efficiency and effectiveness of ERDG have been well-documented across numerous LSOP tasks. However, it is worth noting that ERDG exhibits shortcomings when dealing with certain types of functions. Specifically, it struggles

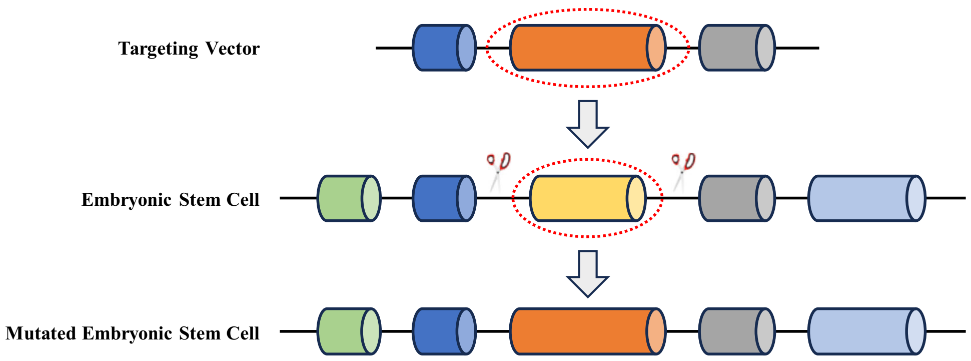


Fig. 5 A demonstration of the gene-targeting operator

Algorithm 5 Gene-targeting operator

```
Require: Best individual: $X_{best}$, Dimension: $D$
1: Locating the causative genes using Algorithm 2
2: Constructing the homologous targeting vector using Algorithm 3
3: Inserting the homologous targeting vector into $X_{best}$ using Algorithm 4
4: if $X'_{best,j}$ has a better fitness value then
5:     $X_{best,j} = X'_{best,j}$
6: end if
```

Algorithm 6 GTMBGO

```
Require: Population size: $N$, Dimension: $D$, Maximum iteration: $T_{max}$
Ensure: Optimum: $X_{best}$
1: Population initialization
2: $X_{best} \leftarrow$ best($R$)
3: $t \leftarrow 0$
4: while $t < T_{max}$ do
5:     for $i = 0$ to $N$ do
6:         if $X_i$ is current best individual $X_{best}$ then
7:             Construct $X_{new}$ by the gene-targeting operator in Algorithm 5
8:         else
9:             Select a random enemy $X_{enemy}$ for $X_i$
10:            if $X_{enemy}$ has a better fitness value then
11:                Construct $X_{new}$ by Eq. (4)
12:            else
13:                Construct $X_{new}$ by Eq. (5)
14:            end if
15:        end if
16:        Internal greedy selection
17:    end for
18:    $X_{best} \leftarrow$ best($R$)
19:    $t \leftarrow t + 1$
20: end while
21: return $X_{best}$
```

with some fully separable functions, some overlapping functions, and all fully non-separable functions. For instance, in the case of the CEC2013 LSGO function f_3, which is fully separable, ERDG fails to capture the interactions between decision variables, as highlighted in Sun et al. [36].

Furthermore, while ERDG demonstrates a satisfied performance by accurately assigning all decision variables to a subcomponent in some overlapping functions and fully non-separable functions (i.e., f_{12}, f_{13}, and f_{15} in CEC2013 LSGO), the integration of the CC framework cannot significantly accelerate optimization convergence in these scenarios due to the persistence of the curse of dimensionality. Given the unsettled challenges associated with ERDG, we propose an enhanced variant termed efficient recursive differential grouping with maximum volume (ERDG$_k$). This variant introduces a maximum volume constraint on subcomponents, aiming to alleviate the negative influence of dimensionality during optimization. Although the incorporation of the maximum volume mechanism may sacrifice the decomposition accuracy, it is anticipated to form subcomponents with acceptable scales and achieve superior optimization convergence. Algorithm 7 presents the pseudocode of ERDG$_k$.

In Algorithm 7, ERDG$_k$ first detects interactions between x_1 and the remaining decision variables (i.e., X_1 and X_2). If these two subsets are separable, ERDG$_k$ further identifies the length of the subset X_1. If X_1 contains more than one decision variable, ERDG$_k$ identifies X_1 as a non-separable sub-component; otherwise, it is considered separable. If interactions exist between X_1 and X_2, ERDG$_k$ segregates the interrelated decision variables in X_2 and assigns them to X_1. ERDG$_k$ repeats this process until all decision variables are appropriately placed. Additionally, the primary distinction between ERDG and our proposed ERDG$_k$ lies in the implementation from Algorithm 7, Line 26 to 32. Upon completion of ERDG procedures, if the scale of a subcomponent exceeds the designated maximum volume k, we employ the random decomposition strategy outlined in Yang et al. [37] for further decomposition. Moreover, following the recommendation in Yang et al. [38], the maximum volume k is set to 100. Especially for fully non-separable and overlapping functions, the accurate decomposition for these two types of problems requires that all decision variables be divided into sub-components. However, optimization based on this decomposition cannot be accelerated using the CC framework. Thus, ERDG$_k$ sacrifices the accuracy of decomposition and randomly decomposes the sub-component that exceeds the scale of S, leading to a degeneration in the accuracy of this decomposition, as presented in Eq. (16).

Algorithm 7 ERDG$_k$

```
Require: Lower and upper bound vector: $\mathbf{lb}$ and $\mathbf{ub}$, Maximum volume: $k$
Ensure: Separable and non-separable subcomponents: $sep$ and $nonsep$
1: $sep = \emptyset$, $nonsep = \emptyset$
2: $\mathbf{x}_{l,l} = \mathbf{lb}$, $y_{l,l} = f(\mathbf{x}_{l,l})$
3: $X_1 = \{x_1\}$, $X_2 = \{x_2, x_3, ..., x_D\}$
4: while $X_2 \neq \emptyset$ do
5:     $\mathbf{x}_{u,l} = \mathbf{x}_{l,l}$, $\mathbf{x}_{u,l}(X_1) = \mathbf{ub}(X_1)$, $y_{u,l} = f(\mathbf{x}_{u,l})$
6:     $F = \{y_{l,l}, y_{u,l}, nan, nan\}$ # $nan$ is a nonsensical value
7:     $(X_1^*, \hat{\beta}) =$ Interact$(X_1, X_2, \mathbf{x}_{l,l}, \mathbf{x}_{u,l}, \mathbf{lb}, \mathbf{ub}, F)$ # Ref to [36] Algorithm 2
8:     if $|X_1^*| = |X_1|$ then
9:         if $|X_1^*| > 1$ then
10:            $nonsep = \{nonsep, X_1^*\}$
11:        else
12:            $sep = sep + X_1^*$
13:        end if
14:        $X_1 = X_2[0]$, $X_2 = (X_2 - X_2[0])$
15:    else
16:        $X_1 = X_1^*$, $X_2 = (X_2 - X_1)$
17:    end if
18:    if $X_2 = \emptyset$ then
19:        if $|X_1| > 1$ then
20:            $nonsep = \{nonsep, X_1\}$
21:        else
22:            $sep = sep + X_1$
23:        end if
24:    end if
25: end while
26: for $group$ in $nonsep$ do
27:    if $|group| > k$ then
28:        $nonsep = nonsep - group$
29:        Random separate $group$ to $\{sub_1, sub_2, ..., sub_j\}$ with maximum volume $k$
30:        $nonsep = nonsep + \{sub_1, sub_2, ..., sub_j\}$
31:    end if
32: end for
33: return $sep$, $nonsep$
```

$$Acc = \frac{S}{N} \cdot 100\% \tag{16}$$

Given that N represents the dimension size of the original problem, with $S = 100$ and $N = 1000$, the decomposition of fully non-separable and overlapping functions would decrease significantly to only 10%. This reduction demonstrates how the process can alleviate the curse of dimensionality and accelerate optimization convergence, while at the expense of decomposition accuracy.

In summary, the pseudocode of the complete proposed method, GTMBGO-ERDG$_k$, is demonstrated in Algorithm 8.

Algorithm 8 GTMBGO-ERDG$_k$

```
Require: Problem: $P$, Maximum iteration: $T$
Ensure: Optimum: $X_{best}$
1: Initialize the context vector $V$
2: Decompose the problem $P$ by ERDG$_k$ (Algorithm 7) into multiple sub-components $\{sc_1, ..., sc_n\}$
3: while The computational budget is not exhausted do
4:     for $i = 0$ to $n$ do
5:         Optimize the sub-component $sc_i$ with the GTMBGO (Algorithm 6) and obtain sub-solution $ss_i$
6:         Update the context vector $V$
7:     end for
8: end while
9: Combine the sub-solutions $\{ss_1, ..., ss_n\}$ to form $X_{best}$
10: return $X_{best}$
```

4 Numerical experiments

This section provides comprehensive experiments to evaluate the performance of our proposal. Section 4.1 details the experimental settings, including benchmark functions and competitor algorithms. Section 4.2 presents the experimental results and statistical analyses to demonstrate the superiority of our proposal.

4.1 Experimental settings

4.1.1 Benchmark functions

We conduct thorough numerical experiments on the following benchmark functions.

- 30-D and 50-D CEC2017 benchmark functions.[1]
- 10-D and 20-D CEC2022 benchmark functions.[2]
- CEC2013 LSGO benchmark functions with the dimensions of 905 and 1000 [39].

4.1.2 Competitor algorithms and parameters

To fairly and comprehensively investigate the performance of our proposal, we adopt two distinct categories of competitor algorithms in median- and large-scale instances. Note that the parameters of all competitor algorithms are employed as the original paper recommended.

- In CEC2017 and CEC2022 benchmark functions, PSO [40], DE [41], Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [42], Comprehensive Learning PSO (CLPSO) [43], JADE [44], Success-History Adaptive DE (SHADE) [45], SHADE with Linear population size reduction (L-SHADE) [46], Improved L-SHADE (iL-

[1] https://github.com/P-N-Suganthan/CEC2017-BoundContrained.

[2] https://github.com/P-N-Suganthan/2022-SO-BO/tree/main.

Table 1 The parameters of competitor algorithms in CEC2017 and CEC2022 benchmark functions

Optimizer	Parameters	Value
PSO	Inertia factor w	1
	Coefficients c_1 and c_2	2.05
	Max. and min. speed	2 and -2
DE	Mutation strategy	DE/cur-to-rand/1/bin
	Scaling factor F	0.8
	Crossover rate Cr	0.9
CMA-ES	σ	1.3
CLPSO	Local coefficient c_{local}	1.2
	Max. and min. weight	0.9 and 0.4
JADE	μ_F and μ_{Cr}	0.5 and 0.5
SHADE	μ_F and μ_{Cr}	0.5 and 0.5
L-SHADE	μ_F and μ_{Cr}	0.5 and 0.5
iL-SHADE	μ_F and μ_{Cr}	0.5 and 0.8
	Max. and min. proportion of elite	0.2 and 0.1
AHPSO	Inertia factor w	1
	Coefficient c	1.49445
GTDE	μ and σ in causative gene targeting	0.1 and 0.1
	μ_F and σ_F in gene-targeting operator	0.5 and 0.1
	μ_F and σ_F	0.7 and 0.5
	μ_{Cr} and σ_{Cr}	0.5 and 0.5
MBGO	Ratio of radius	0.8 and 1.2
GTMBGO	μ and σ in causative gene targeting	0.1 and 0.1
	μ_F and σ_F in gene-targeting operator	0.5 and 0.1

Table 2 The parameters of competitor algorithms in CEC2013 LSGO benchmark functions

Method	Parameters	Value
CCPSO2	Inertia factor w	0.729
	Coefficients c_1 and c_2	2.05
	Potential subcomponent size S	{2, 5, 50, 100, 200}
DECC-DG	Threshold ε	10^{-3}
DECC-DG2	Parameter-free	
DECC-RDG	Parameter-free	
DECC-ERDG	Parameter-free	
DECC-MDG	Parameter-free	
μDSDE	N_{mc}	3
	N_{ls}	3000
	Scaling factor F	[0.3, 1]
DMS-L-PSO	Inertia factor w	0.729
	Pc_{mean}	0.5
	Coefficient c	1.49445
Hip-DE	μ_F and σ_F	0.6 and 0.1
	μ_{Cr} and σ_{Cr}	0.8 and 0.1
	F_{best} and Cr_{best}	0.5 and 0.9
eWOA	Parameter-free	
DSCA	Parameter-free	
FADE	F_1 and Cr_1	0.5 and 0.1
	F_2 and Cr_2	0.8 and 0.5
	F_3 and Cr_3	1.0 and 0.9
GTMBGO-ERDG$_k$	μ and σ in causative gene targeting	0.1 and 0.1
	μ_F and σ_F in gene-targeting operator	0.5 and 0.1
	Maximum volume	100

SHADE) [47], Adaptive Heterogeneous comprehensive learning PSO (AHPSO) [48], GTDE [31], and original MBGO [30] are employed as competitor algorithms. The specific parameter settings are summarized in Table 1.

- In CEC2013 LSGO benchmark functions, we adopt Cooperatively Coevolving PSO (CCPSO2) [49], DECC-DG [34], DECC-DG2 [50], DECC-RDG [35], DECC with Merged Differential Grouping (DECC-MDG) [51], micro DE with a Directional Local Search (μDSDE) [52], Dynamic Multi-Swarm PSO with Local Search (DMS-L-PSO) [53], Hip-DE [54], enhanced Whale Optimization Algorithm (eWOA) [55], Dynamic Sine Cosine Algorithm (DSCA) [56], and Fitness-based Adaptive DE (FADE) [57] as competitor algorithms. The specific parameter settings are summarized in Table 2.

The population size for involved algorithms remains constant at 100, with the maximum fitness evaluations (FEs) set differently across benchmark functions: 1000 $\times D$ (D = dimension size) for CEC2017 and CEC2022, and 3,000,000 for CEC2013 LSGO benchmark functions. Each optimizer undergoes 25 independent executions to mitigate the effects of randomness in the optimization process.

4.2 Experimental results and statistical analyses

This section presents the experimental results and statistical analyses obtained through numerical experiments. We commence by showcasing the results of comparison experiments conducted on both low- and median-dimensional CEC2017 and CEC2022 benchmark functions. Subsequently, ablation experiments on CEC2017 benchmark functions are conducted to investigate the individual contributions in the proposed GTMBGO. Following this, we integrate the proposed optimizer GTMBGO with the proposed decomposition method ERDG$_k$ and evaluate the performance of GTMBGO-ERDG$_k$ on CEC2013 LSGO benchmark func-

Table 3 Experimental results and statistical analyses on 30-D CEC2017 benchmark functions

Func.		PSO	DE	CMA-ES	CLPSO	JADE	SHADE	L-SHADE	iL-SHADE	AHPSO	GTDE	MBGO	GTMBGO
f_1	mean	4.350e+09 +	3.796e+10 +	3.202e+09 +	2.838e+07 +	4.795e+06 +	2.241e+05 +	2.215e+05 +	1.120e+04 +	4.343e+09 +	9.100e+03 ≈	3.445e+05 +	**5.604e+03**
	std	1.850e+09	4.492e+09	5.672e+08	7.300e+06	2.179e+06	1.187e+05	9.453e+04	6.837e+03	7.630e+08	8.124e+03	1.720e+05	3.269e+03
f_3	mean	1.091e+05 +	2.469e+05 +	9.489e+04 +	9.846e+04 +	8.969e+04 +	8.548e+04 +	9.008e+04 +	7.952e+04 +	1.010e+05 +	1.047e+05 +	3.611e+04 +	**2.463e+04**
	std	1.997e+04	3.098e+04	1.482e+04	1.698e+04	1.456e+04	1.775e+04	1.469e+04	1.594e+04	1.521e+04	2.378e+04	6.882e+03	4.768e+03
f_4	mean	2.386e+03 +	3.489e+03 +	7.201e+02 +	5.594e+02 +	5.402e+02 +	5.172e+02 +	5.181e+02 +	5.093e+02 ≈	1.205e+03 +	**4.958e+02** −	5.244e+02 +	5.126e+02
	std	1.499e+03	6.880e+02	4.690e+01	9.898e+00	9.711e+00	8.525e+00	7.535e+00	1.514e+01	1.452e+02	1.718e+01	3.098e+01	2.372e+01
f_5	mean	8.444e+02 +	8.907e+02 +	7.665e+02 +	6.969e+02 +	6.603e+02 +	6.695e+02 +	6.683e+02 +	6.577e+02 +	7.773e+02 +	6.759e+02 +	6.687e+02 +	**6.409e+02**
	std	4.159e+01	2.102e+01	1.596e+01	1.376e+01	1.150e+01	1.325e+01	1.242e+01	1.485e+01	1.663e+01	3.584e+01	1.132e+01	2.523e+01
f_6	mean	6.566e+02 +	6.751e+02 +	6.404e+02 +	6.131e+02 +	6.080e+02 +	6.051e+02 +	6.062e+02 +	6.015e+02 +	6.492e+02 +	6.023e+02 +	6.001e+02 +	**6.000e+02**
	std	1.318e+01	3.781e+00	3.518e+00	2.236e+00	9.694e−01	7.399e−01	8.869e−01	3.178e−01	4.946e+00	1.665e+00	1.231e−01	4.098e−03
f_7	mean	1.147e+03 +	2.557e+03 +	1.161e+03 +	9.475e+02 +	8.954e+02 ≈	9.016e+02 +	9.017e+02 +	**8.916e+02** ≈	1.092e+03 +	9.407e+02 +	9.054e+02 +	8.922e+02
	std	7.956e+01	1.261e+02	3.312e+01	1.417e+01	9.151e+00	1.107e+01	9.886e+00	1.109e+01	3.020e+01	2.740e+01	1.164e+01	1.414e+01
f_8	mean	1.101e+03 +	1.190e+03 +	1.067e+03 +	9.957e+02 +	9.549e+02 +	9.605e+02 +	9.617e+02 +	9.529e+02 +	1.069e+03 +	9.708e+02 +	9.596e+02 +	**9.334e+02**
	std	4.105e+01	2.078e+01	2.029e+01	1.303e+01	1.047e+01	1.023e+01	1.018e+01	9.270e+00	1.930e+01	3.446e+01	1.193e+01	2.185e+01
f_9	mean	9.754e+03 +	1.759e+04 +	5.216e+03 +	4.706e+03 +	2.873e+03 +	2.637e+03 +	2.793e+03 +	1.103e+03 +	9.080e+03 +	1.726e+03 +	9.182e+02 +	**9.127e+02**
	std	3.527e+03	2.550e+03	5.865e+02	7.000e+02	3.611e+02	4.632e+02	5.470e+02	8.924e+01	1.610e+03	8.719e+02	1.540e+01	2.801e+01
f_{10}	mean	8.728e+03 +	8.215e+03 +	8.558e+03 +	6.875e+03 −	**6.311e+03** −	6.321e+03 −	6.469e+03 −	6.612e+03 −	7.804e+03 ≈	7.481e+03 −	7.917e+03 ≈	7.896e+03
	std	3.105e+02	3.910e+02	4.201e+02	2.776e+02	2.725e+02	2.369e+02	3.159e+02	3.663e+02	4.635e+02	5.128e+02	3.128e+02	2.936e+02
f_{11}	mean	3.970e+03 +	3.066e+03 +	1.343e+03 +	1.714e+03 +	1.895e+03 +	1.384e+03 +	1.391e+03 +	1.301e+03 +	2.996e+03 +	1.339e+03 +	1.298e+03 +	**1.232e+03**
	std	1.710e+03	4.072e+02	2.471e+01	1.821e+02	2.308e+02	3.611e+01	3.660e+01	2.110e+01	5.584e+02	7.511e+01	3.701e+01	3.091e+01
f_{12}	mean	5.129e+08 +	1.673e+09 +	2.240e+07 +	8.407e+06 +	1.289e+07 +	3.769e+06 +	2.655e+06 +	1.214e+06 +	2.403e+08 +	6.276e+06 +	1.891e+06 +	**6.711e+05**
	std	6.838e+08	3.159e+08	6.159e+06	2.119e+06	2.852e+06	1.004e+06	9.516e+05	5.615e+05	6.860e+07	2.163e+07	1.598e+06	3.970e+05
f_{13}	mean	4.287e+07 +	1.116e+08 +	**7.161e+03** ≈	6.004e+05 +	5.434e+06 +	4.848e+05 +	3.182e+05 +	1.364e+05 +	7.525e+07 +	1.818e+05 +	1.577e+04 ≈	1.054e+04
	std	1.490e+08	2.652e+07	1.119e+03	3.027e+05	2.660e+06	2.421e+05	1.835e+05	7.296e+04	3.997e+07	7.920e+05	1.701e+04	6.330e+03
f_{14}	mean	5.616e+05 +	1.095e+04 +	**1.508e+03** −	2.641e+05 +	3.288e+05 +	4.095e+04 +	4.446e+04 +	1.301e+04 +	2.810e+05 +	2.204e+03 −	5.236e+03 +	2.301e+03
	std	5.429e+05	4.864e+03	5.797e+00	1.586e+05	1.580e+05	2.157e+04	2.929e+04	1.243e+04	1.351e+05	1.855e+03	2.320e+03	4.835e+02
f_{15}	mean	5.590e+06 +	4.303e+05 +	**1.839e+03** −	1.740e+05 +	1.085e+06 +	8.255e+04 +	6.646e+04 +	3.067e+04 +	6.772e+06 +	1.300e+04 +	1.328e+04 +	5.723e+03
	std	1.413e+07	2.027e+05	5.077e+01	1.259e+05	6.487e+05	4.997e+04	3.074e+04	1.764e+04	5.444e+06	1.422e+04	9.256e+03	2.755e+03
f_{16}	mean	4.288e+03 +	3.852e+03 +	3.585e+03 +	2.976e+03 +	2.985e+03 +	2.871e+03 ≈	2.871e+03 ≈	2.842e+03 ≈	3.571e+03 +	**2.675e+03** ≈	2.928e+03 +	2.783e+03
	std	2.993e+02	1.758e+02	1.832e+02	1.146e+02	1.464e+02	2.048e+02	1.426e+02	1.417e+02	1.568e+02	3.138e+02	2.214e+02	2.554e+02

Table 4 Experimental results and statistical analyses on 30-D CEC2017 benchmark functions (Continued)

Func.		PSO	DE	CMA-ES	CLPSO	JADE	SHADE	L-SHADE	iL-SHADE	AHPSO	GTDE	MBGO	GTMBGO
f_{17}	mean	2.966e+03 +	2.707e+03 +	2.572e+03 +	2.177e+03 +	2.207e+03 +	2.093e+03 +	2.043e+03 +	2.016e+03 +	2.525e+03 +	2.129e+03 +	2.002e+03 +	**1.952e+03**
	std	2.193e+02	1.831e+02	1.226e+02	1.108e+02	8.052e+01	8.796e+01	8.557e+01	9.749e+01	1.228e+02	1.671e+02	8.860e+01	8.179e+01
f_{18}	mean	1.188e+07 +	4.311e+06 +	**4.140e+03** −	8.372e+05 +	1.365e+06 +	5.522e+05 +	5.120e+05 +	4.359e+05 +	1.936e+06 +	3.410e+05 +	1.408e+05 +	5.224e+04
	std	1.089e+07	2.142e+06	8.229e+02	3.687e+05	6.199e+05	2.610e+05	1.954e+05	1.716e+05	8.723e+05	3.289e+05	6.872e+04	1.615e+04
f_{19}	mean	2.023e+07 +	9.737e+06 +	**2.140e+03** −	2.478e+05 +	7.552e+05 +	8.419e+04 +	5.974e+04 +	2.946e+04 +	8.281e+06 +	1.313e+04 +	1.344e+04 +	5.552e+03
	std	5.607e+07	4.227e+06	4.786e+01	1.714e+05	3.532e+05	5.242e+04	2.968e+04	1.450e+04	4.096e+06	1.602e+04	1.856e+04	2.266e+03
f_{20}	mean	2.952e+03 +	2.632e+03 +	2.880e+03 +	2.564e+03 +	2.565e+03 +	2.480e+03 ≈	2.440e+03 ≈	2.472e+03 ≈	2.843e+03 +	**2.395e+03** ≈	2.517e+03 +	2.432e+03
	std	9.516e+01	1.489e+02	1.114e+02	9.659e+01	1.138e+02	8.984e+01	9.339e+01	8.912e+01	1.063e+02	1.650e+02	8.756e+01	9.092e+01
f_{21}	mean	2.603e+03 +	2.655e+03 +	2.558e+03 +	2.487e+03 +	2.454e+03 +	2.467e+03 +	2.459e+03 +	2.455e+03 +	2.566e+03 +	2.473e+03 +	2.452e+03 +	**2.430e+03**
	std	3.805e+01	2.386e+01	1.233e+01	2.051e+01	1.260e+01	8.700e+00	1.020e+01	1.092e+01	1.647e+01	3.448e+01	1.334e+01	1.968e+01
f_{22}	mean	4.129e+03 +	9.924e+03 +	9.677e+03 +	3.286e+03 +	3.051e+03 +	2.790e+03 +	2.821e+03 +	2.423e+03 +	5.431e+03 +	7.333e+03 +	2.308e+03 +	**2.301e+03**
	std	1.767e+03	2.602e+02	1.245e+03	3.522e+02	1.999e+02	2.225e+02	2.696e+02	1.947e+02	7.578e+02	2.615e+03	3.189e+00	1.345e+00
f_{23}	mean	3.084e+03 +	2.991e+03 +	2.925e+03 +	2.857e+03 +	2.811e+03 +	2.822e+03 +	2.814e+03 +	2.802e+03 +	2.955e+03 +	2.784e+03 +	2.806e+03 +	**2.746e+03**
	std	7.154e+01	2.398e+01	1.584e+01	1.400e+01	1.135e+01	1.258e+01	8.643e+00	1.148e+01	2.006e+01	3.861e+01	1.855e+01	3.892e+01
f_{24}	mean	3.238e+03 +	3.126e+03 +	3.073e+03 +	3.032e+03 +	2.991e+03 +	3.001e+03 +	2.989e+03 +	2.977e+03 +	3.091e+03 +	2.976e+03 +	3.000e+03 +	**2.919e+03**
	std	6.260e+01	1.584e+01	1.923e+01	1.671e+01	1.629e+01	1.488e+01	1.084e+01	1.095e+01	1.827e+01	3.993e+01	1.764e+01	3.797e+01
f_{25}	mean	3.218e+03 +	6.398e+03 +	3.144e+03 +	2.947e+03 +	2.900e+03 ≈	**2.891e+03** −	2.896e+03 −	2.893e+03 −	3.233e+03 +	2.896e+03 −	2.920e+03 ≈	2.912e+03
	std	1.663e+02	6.530e+02	4.448e+01	8.490e+00	4.776e+00	6.326e+00	8.613e+00	1.045e+01	5.190e+01	1.637e+01	2.281e+01	1.976e+01
f_{26}	mean	8.205e+03 +	7.532e+03 +	6.366e+03 +	5.872e+03 +	5.232e+03 +	5.205e+03 +	5.054e+03 +	5.122e+03 +	7.159e+03 +	5.030e+03 +	5.278e+03 +	**4.489e+03**
	std	5.572e+02	2.781e+02	1.749e+02	4.188e+02	1.175e+02	4.829e+02	5.430e+02	2.723e+02	2.781e+02	6.897e+02	2.861e+02	4.616e+02
f_{27}	mean	3.616e+03 +	3.290e+03 +	3.241e+03 +	3.257e+03 +	3.234e+03 +	3.234e+03 +	3.237e+03 +	3.230e+03 +	3.302e+03 +	3.229e+03 ≈	3.232e+03 +	**3.222e+03**
	std	1.158e+02	1.580e+01	1.185e+01	5.347e+00	4.109e+00	4.002e+00	4.408e+00	4.117e+00	1.813e+01	2.038e+01	1.573e+01	1.202e+01
f_{28}	mean	4.129e+03 +	4.725e+03 +	3.448e+03 +	3.356e+03 +	3.296e+03 +	**3.251e+03** −	3.267e+03 −	3.258e+03 −	3.968e+03 +	3.451e+03 +	3.292e+03 ≈	3.283e+03
	std	7.542e+02	6.563e+02	4.272e+01	1.324e+01	1.841e+01	1.702e+01	2.086e+01	2.019e+01	9.565e+01	6.168e+02	3.129e+01	2.264e+01
f_{29}	mean	5.333e+03 +	4.295e+03 +	4.464e+03 +	4.039e+03 +	3.836e+03 +	3.848e+03 +	3.861e+03 +	3.868e+03 +	4.710e+03 +	3.842e+03 +	3.898e+03 +	**3.761e+03**
	std	3.582e+02	2.996e+02	1.468e+02	1.335e+02	9.511e+01	9.497e+01	1.004e+02	9.298e+01	1.552e+02	1.864e+02	2.037e+02	1.398e+02
f_{30}	mean	1.910e+07 +	1.168e+07 +	9.112e+04 +	1.238e+06 +	6.692e+05 +	4.398e+05 +	3.699e+05 +	2.685e+05 +	1.961e+07 +	8.638e+04 +	9.064e+04 +	**1.496e+04**
	std	2.906e+07	4.033e+06	3.264e+04	5.426e+05	2.806e+05	1.380e+05	1.375e+05	1.160e+05	8.648e+06	2.072e+05	1.646e+05	1.007e+04
+/≈/−:		29/0/0	29/0/0	24/1/4	28/0/1	26/2/1	24/2/3	24/2/3	22/4/3	28/1/0	21/4/4	25/4/0	–

f_1: Unimodal function; $f_3 - f_9$: Simple multimodal functions; $f_{10} - f_{19}$: Hybrid functions; $f_{20} - f_{30}$: Composition functions

Table 5 Experimental results and statistical analyses on 50-D CEC2017 benchmark functions

Func.		PSO	DE	CMA-ES	CLPSO	JADE	SHADE	L-SHADE	iL-SHADE	AHPSO	GTDE	MBGO	GTMBGO
f_1	mean	1.907e+10 +	1.017e+11 +	1.106e+10 +	1.741e+07 +	2.518e+05 ≈	**2.338e+03** −	1.567e+04 −	2.436e+03 −	6.232e+09 +	1.607e+08 +	2.620e+06 +	2.057e+05
	std	3.757e+09	9.010e+09	3.086e+09	4.898e+06	2.671e+05	1.539e+03	8.883e+03	2.781e+03	9.835e+08	6.528e+08	1.035e+06	1.578e+05
f_3	mean	2.338e+05 +	4.635e+05 +	2.449e+05 +	2.039e+05 +	1.915e+05 +	1.781e+05 +	1.919e+05 +	1.440e+05 +	2.177e+05 +	2.673e+05 +	8.646e+04 +	**6.366e+04**
	std	3.344e+04	4.261e+04	3.240e+04	2.136e+04	2.685e+04	4.707e+04	2.481e+04	6.950e+04	2.176e+04	4.823e+04	1.479e+04	1.243e+04
f_4	mean	5.625e+03 +	1.591e+04 +	1.771e+03 +	6.419e+02 +	6.284e+02 +	**5.509e+02** −	5.661e+02 ≈	5.524e+02 −	1.728e+03 +	6.287e+02 +	6.425e+02 +	5.771e+02
	std	2.441e+03	2.748e+03	4.345e+02	2.521e+01	3.269e+01	4.517e+01	3.748e+01	3.798e+01	1.453e+02	1.165e+02	3.815e+01	4.194e+01
f_5	mean	1.080e+03 +	1.289e+03 +	1.021e+03 +	8.666e+02 +	8.095e+02 +	8.050e+02 +	8.206e+02 +	7.977e+02 +	1.023e+03 +	7.936e+02 +	8.227e+02 +	**7.472e+02**
	std	1.048e+02	3.421e+01	2.146e+01	2.381e+01	1.670e+01	1.655e+01	1.835e+01	2.057e+01	1.775e+01	8.412e+01	4.202e+01	6.196e+01
f_6	mean	6.795e+02 +	6.961e+02 +	6.550e+02 +	6.101e+02 +	6.099e+02 +	6.032e+02 +	6.034e+02 +	6.004e+02 +	6.524e+02 +	6.051e+02 +	6.008e+02 +	**6.002e+02**
	std	1.071e+01	5.583e+00	5.874e+00	1.262e+00	9.970e−01	5.236e−01	7.226e−01	1.479e−01	4.219e+00	2.982e+00	5.478e−01	2.505e−01
f_7	mean	1.593e+03 +	4.963e+03 +	1.510e+03 +	1.137e+03 +	1.070e+03 ≈	1.055e+03 ≈	1.063e+03 ≈	**1.040e+03** −	1.371e+03 +	1.185e+03 +	1.132e+03 +	1.072e+03
	std	1.308e+02	3.272e+02	4.491e+01	2.392e+01	1.981e+01	1.794e+01	1.987e+01	1.876e+01	3.172e+01	6.224e+01	2.496e+01	4.067e+01
f_8	mean	1.352e+03 +	1.568e+03 +	1.312e+03 +	1.168e+03 +	1.109e+03 +	1.105e+03 +	1.115e+03 +	1.095e+03 +	1.319e+03 +	1.135e+03 +	1.132e+03 +	**1.032e+03**
	std	9.590e+01	3.856e+01	2.503e+01	3.487e+01	1.630e+01	1.734e+01	1.915e+01	1.756e+01	2.474e+01	7.175e+01	2.357e+01	5.724e+01
f_9	mean	3.250e+04 +	4.838e+04 +	1.338e+04 +	1.703e+04 +	9.480e+03 +	8.600e+03 +	9.445e+03 +	1.206e+03 +	2.873e+04 +	6.392e+03 +	1.647e+03 +	**1.106e+03**
	std	8.482e+03	5.515e+03	3.243e+03	2.195e+03	1.201e+03	1.542e+03	1.531e+03	2.028e+02	3.652e+03	3.035e+03	6.101e+02	2.398e+02
f_{10}	mean	1.518e+04 +	1.488e+04 +	1.508e+04 +	1.152e+04 −	1.054e+04 −	**1.051e+04** −	1.063e+04 −	1.088e+04 −	1.323e+04 ≈	1.340e+04 ≈	1.352e+04 +	1.322e+04
	std	3.970e+02	3.638e+02	4.329e+02	4.178e+02	6.266e+02	3.157e+02	3.502e+02	3.974e+02	4.365e+02	9.631e+02	4.970e+02	4.835e+02
f_{11}	mean	1.317e+04 +	1.961e+04 +	1.642e+03 +	3.205e+03 +	7.275e+03 +	2.213e+03 +	2.217e+03 +	1.666e+03 +	7.559e+03 +	1.507e+03 +	1.665e+03 +	**1.335e+03**
	std	4.132e+03	3.310e+03	5.901e+01	6.493e+02	1.274e+03	2.468e+02	2.552e+02	1.056e+02	1.509e+03	1.013e+02	1.395e+02	4.749e+01
f_{12}	mean	6.037e+09 +	1.372e+10 +	2.571e+08 +	1.856e+07 +	6.305e+07 +	5.943e+06 +	5.325e+06 +	**2.672e+06** ≈	1.076e+09 +	5.394e+07 +	1.099e+07 +	2.769e+06
	std	6.980e+09	1.801e+09	5.059e+07	3.748e+06	1.917e+07	1.716e+06	1.672e+06	9.079e+05	2.174e+08	1.635e+08	4.819e+06	1.124e+06
f_{13}	mean	2.179e+08 +	1.935e+09 +	2.115e+05 +	1.054e+05 +	4.500e+06 +	1.517e+05 +	9.051e+04 +	3.603e+04 +	1.555e+08 +	2.531e+07 +	8.837e+03 +	**3.397e+03**
	std	6.023e+08	5.498e+08	5.091e+04	5.204e+04	2.712e+06	1.006e+05	4.508e+04	1.224e+04	6.407e+07	1.344e+08	3.667e+03	1.358e+03
f_{14}	mean	4.154e+06 +	1.404e+06 +	**1.622e+03** −	1.669e+06 +	1.665e+06 +	2.336e+05 +	2.743e+05 +	6.509e+04 +	2.346e+06 +	7.456e+04 +	5.552e+04 +	1.454e+04
	std	3.797e+06	5.760e+05	1.492e+01	4.984e+05	7.958e+05	1.793e+05	1.521e+05	7.342e+04	8.849e+05	1.105e+05	4.760e+04	7.973e+03
f_{15}	mean	1.223e+08 +	4.987e+07 +	**3.582e+03** −	3.616e+04 +	1.403e+06 +	5.840e+04 +	4.926e+04 +	2.703e+04 +	1.351e+07 +	2.357e+04 +	5.423e+03 ≈	7.419e+03
	std	4.066e+08	7.107e+07	2.946e+02	2.585e+04	9.235e+05	2.363e+04	1.863e+04	1.389e+04	6.918e+06	1.252e+04	3.770e+03	3.940e+03
f_{16}	mean	6.448e+03 +	6.287e+03 +	5.425e+03 +	3.731e+03 +	4.021e+03 +	3.813e+03 +	3.695e+03 +	3.582e+03 +	4.751e+03 +	3.419e+03 ≈	3.873e+03 +	**3.346e+03**
	std	4.489e+02	3.039e+02	2.474e+02	1.746e+02	2.259e+02	2.296e+02	1.881e+02	1.747e+02	2.425e+02	3.995e+02	3.886e+02	4.099e+02

Table 6 Experimental results and statistical analyses on 50-D CEC2017 benchmark functions (Continued)

Func.		PSO	DE	CMA-ES	CLPSO	JADE	SHADE	L-SHADE	iL-SHADE	AHPSO	GTDE	MBGO	GTMBGO
f_{17}	mean	4.755e+03 +	5.027e+03 +	4.226e+03 +	3.189e+03 ≈	3.358e+03 +	3.162e+03 ≈	**3.070e+03** ≈	3.105e+03 ≈	3.938e+03 +	3.321e+03 ≈	3.150e+03 ≈	3.124e+03
	std	5.542e+02	2.188e+02	1.912e+02	1.606e+02	1.544e+02	1.715e+02	1.432e+02	1.420e+02	1.746e+02	4.177e+02	2.630e+02	1.862e+02
f_{18}	mean	3.663e+07 +	2.361e+07 +	**6.284e+04** −	3.516e+06 +	6.046e+06 +	2.730e+06 +	2.502e+06 +	1.601e+06 +	7.448e+06 +	1.226e+06 +	7.149e+05 +	2.682e+05
	std	2.310e+07	6.953e+06	1.900e+04	1.234e+06	2.522e+06	1.023e+06	1.013e+06	9.166e+05	1.828e+06	1.184e+06	4.596e+05	1.402e+05
f_{19}	mean	1.750e+07 +	4.832e+07 +	**5.168e+03** −	3.311e+04 +	4.394e+05 +	4.008e+04 +	3.431e+04 +	2.643e+04 +	4.161e+06 +	1.787e+05 +	1.387e+04 ≈	1.691e+04
	std	8.651e+07	2.213e+07	1.137e+03	1.112e+04	2.640e+05	1.238e+04	1.168e+04	7.582e+03	2.059e+06	8.598e+05	7.928e+03	4.378e+03
f_{20}	mean	4.234e+03 +	4.375e+03 +	4.202e+03 +	3.352e+03 ≈	3.472e+03 +	3.321e+03 ≈	**3.265e+03** ≈	3.309e+03 ≈	3.778e+03 +	3.466e+03 +	3.414e+03 +	3.276e+03
	std	1.961e+02	1.968e+02	1.430e+02	1.412e+02	1.502e+02	1.616e+02	1.448e+02	1.375e+02	1.796e+02	3.595e+02	1.801e+02	1.705e+02
f_{21}	mean	2.924e+03 +	3.067e+03 +	2.817e+03 +	2.663e+03 +	2.609e+03 +	2.596e+03 +	2.602e+03 +	2.579e+03 +	2.791e+03 +	2.625e+03 +	2.606e+03 +	**2.500e+03**
	std	8.872e+01	2.994e+01	2.533e+01	2.010e+01	1.879e+01	1.639e+01	1.600e+01	1.315e+01	2.314e+01	8.952e+01	3.934e+01	7.152e+01
f_{22}	mean	1.659e+04 +	1.658e+04 +	1.667e+04 +	1.173e+04 +	1.159e+04 +	1.126e+04 +	1.150e+04 +	1.147e+04 +	1.448e+04 +	1.501e+04 +	1.058e+04 +	**9.202e+03**
	std	5.297e+02	3.559e+02	3.662e+02	2.596e+03	2.015e+03	2.574e+03	1.913e+03	2.196e+03	1.182e+03	8.060e+02	5.970e+03	6.018e+03
f_{23}	mean	3.637e+03 +	3.499e+03 +	3.321e+03 +	3.129e+03 +	3.049e+03 +	3.044e+03 +	3.048e+03 +	3.026e+03 +	3.320e+03 +	3.005e+03 +	3.053e+03 +	**2.873e+03**
	std	1.229e+02	2.881e+01	3.655e+01	2.434e+01	2.171e+01	1.672e+01	1.613e+01	1.858e+01	3.123e+01	6.625e+01	5.932e+01	6.050e+01
f_{24}	mean	3.917e+03 +	3.553e+03 +	3.466e+03 +	3.336e+03 +	3.226e+03 +	3.217e+03 +	3.211e+03 +	3.189e+03 +	3.437e+03 +	3.219e+03 +	3.274e+03 +	**3.058e+03**
	std	1.271e+02	3.613e+01	3.550e+01	2.793e+01	1.808e+01	2.248e+01	2.203e+01	2.416e+01	2.630e+01	6.963e+01	3.222e+01	6.626e+01
f_{25}	mean	5.270e+03 +	1.761e+04 +	4.023e+03 +	3.182e+03 +	3.104e+03 −	3.096e+03 −	3.114e+03 −	3.101e+03 −	4.057e+03 +	**3.069e+03** −	3.150e+03 ≈	3.151e+03
	std	6.263e+02	2.638e+03	2.585e+02	2.333e+01	2.506e+01	1.952e+01	2.358e+01	2.469e+01	1.366e+02	4.621e+01	3.149e+01	2.487e+01
f_{26}	mean	1.309e+04 +	1.192e+04 +	9.398e+03 +	7.696e+03 +	6.873e+03 +	6.747e+03 +	6.815e+03 +	6.772e+03 +	9.764e+03 +	6.637e+03 +	6.042e+03 +	**5.574e+03**
	std	1.339e+03	5.473e+02	3.086e+02	3.292e+02	1.441e+02	1.919e+02	2.019e+02	2.109e+02	2.307e+02	5.841e+02	8.505e+02	4.364e+02
f_{27}	mean	4.937e+03 +	3.650e+03 +	3.495e+03 +	3.563e+03 +	3.483e+03 +	3.435e+03 ≈	3.463e+03 +	**3.372e+03** −	3.828e+03 +	3.504e+03 +	3.514e+03 +	3.413e+03
	std	3.623e+02	7.167e+01	9.017e+01	2.634e+01	2.923e+01	3.685e+01	3.788e+01	4.176e+01	5.453e+01	1.378e+02	7.162e+01	5.474e+01
f_{28}	mean	6.329e+03 +	8.914e+03 +	3.905e+03 +	3.599e+03 +	3.391e+03 −	**3.355e+03** −	3.387e+03 −	3.377e+03 −	5.179e+03 +	4.247e+03 +	3.477e+03 ≈	3.451e+03
	std	1.058e+03	2.344e+02	1.935e+02	4.109e+01	3.483e+01	2.669e+01	3.118e+01	2.745e+01	2.118e+02	1.620e+03	5.855e+01	4.260e+01
f_{29}	mean	8.236e+03 +	6.772e+03 +	5.997e+03 +	4.532e+03 +	4.373e+03 +	4.311e+03 +	4.314e+03 +	4.341e+03 +	6.107e+03 +	4.469e+03 +	4.235e+03 +	**4.006e+03**
	std	1.203e+03	2.840e+02	2.026e+02	1.653e+02	1.594e+02	1.321e+02	1.358e+02	1.385e+02	2.294e+02	3.944e+02	3.130e+02	2.860e+02
f_{30}	mean	2.662e+08 +	3.364e+08 +	9.529e+06 +	6.084e+06 +	7.179e+06 +	6.440e+06 +	7.171e+06 +	5.802e+06 +	1.584e+08 +	3.168e+06 +	1.591e+06 +	**8.806e+05**
	std	3.282e+08	1.123e+08	2.602e+06	1.429e+06	1.795e+06	1.180e+06	9.903e+05	1.501e+06	4.620e+07	2.027e+06	3.578e+05	8.964e+04
+/≈/−:		29/0/0	29/0/0	25/0/4	28/0/1	25/3/1	22/2/5	24/2/3	22/4/3	28/1/0	21/4/4	24/5/0	–

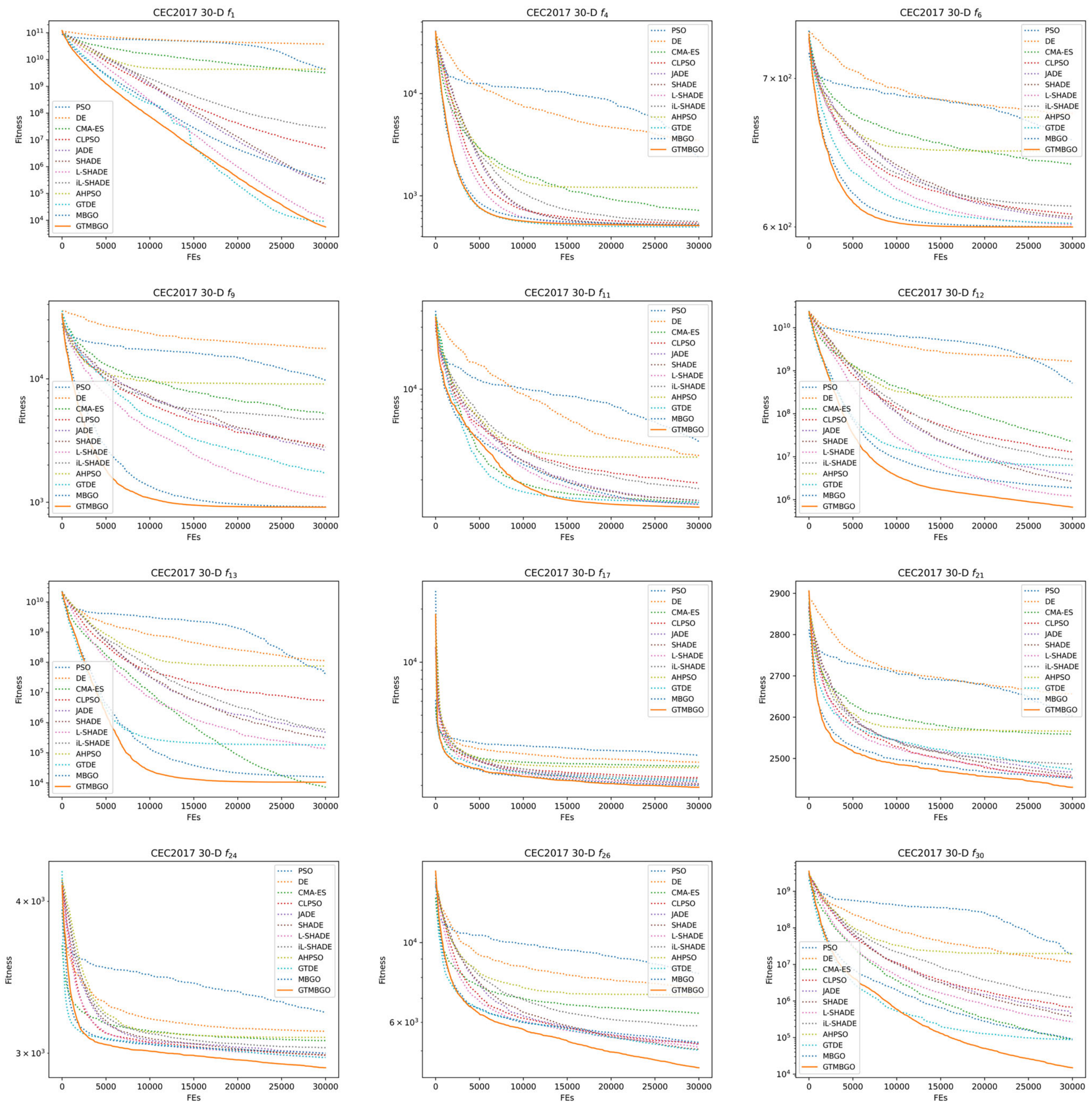


Fig. 6 Convergence curves of competitor optimizers on 30-D CEC2017 representative benchmark functions

tions. To identify the significance of differences between our proposal and other competitor algorithms, we employ the Mann-Whitney U test for each pair of algorithms. We then utilize the Holm multiple comparison test to adjust the p-values obtained from the Mann–Whitney U test. The symbols $+$, $\approx$, and $-$ indicate that our proposal is significantly better, shows no significance, or is significantly worse compared to the method under consideration, respectively. Additionally, the best-performing fitness value is denoted in bold.

4.2.1 Comparison experiments on CEC2017

We summarize the mean and standard deviation (std) with 25 trial runs on CEC2017 benchmark functions in Tables 3, 4, 5, and 6. Additionally, the convergence curves on representative benchmark functions (i.e., f_1: Unimodal function; f_4, f_6, and f_9: Simple multimodal functions; f_{11}, f_{12}, f_{13}, and f_{17}: Hybrid functions; f_{21}, f_{24}, f_{26}, and f_{30}: Composition functions) are presented in Figs. 6 and 7.

Table 7 Experimental results and statistical analyses on 10-D CEC2022 benchmark functions

Func.		PSO	DE	CMA-ES	CLPSO	JADE	SHADE	L-SHADE	iL-SHADE	AHPSO	GTDE	MBGO	GTMBGO
f_1	mean	1.906e+03 +	6.936e+03 +	4.382e+02 +	2.432e+03 +	1.089e+03 +	8.036e+02 +	7.807e+02 +	4.387e+02 +	5.128e+03 +	**3.013e+02** ≈	3.670e+02 +	3.023e+02
	std	1.437e+03	1.749e+03	5.239e+01	8.344e+02	3.233e+02	2.236e+02	1.974e+02	6.253e+01	1.306e+03	1.820e+00	5.525e+01	4.139e+00
f_2	mean	4.965e+02 +	4.576e+02 +	4.154e+02 +	4.491e+02 +	4.151e+02 +	4.157e+02 +	4.183e+02 +	4.097e+02 +	4.977e+02 +	4.099e+02 +	4.158e+02 +	**4.044e+02**
	std	3.685e+01	1.232e+01	2.620e+00	1.519e+01	3.893e+00	8.598e+00	7.970e+00	4.254e+00	2.736e+01	1.313e+01	1.690e+01	6.216e+00
f_3	mean	6.000e+02 +	6.001e+02 +	6.000e+02 +	6.000e+02 +	6.000e+02 +	6.000e+02 +	6.000e+02 +	6.000e+02 +	6.000e+02 +	6.000e+02 ≈	6.000e+02 +	**6.000e+02**
	std	5.018e−03	1.233e−02	6.888e−04	7.456e−04	2.554e−05	3.672e−05	4.925e−05	5.006e−06	1.604e−02	1.414e−08	1.209e−06	2.101e−09
f_4	mean	8.010e+02 +	8.011e+02 +	8.011e+02 +	8.007e+02 ≈	8.005e+02 ≈	8.004e+02 −	**8.004e+02** −	8.006e+02 ≈	8.009e+02 +	8.005e+02 ≈	8.007e+02 ≈	8.006e+02
	std	2.001e−01	2.034e−01	1.757e−01	1.470e−01	1.280e−01	9.964e−02	6.716e−02	1.376e−01	2.046e−01	2.236e−01	1.756e−01	1.543e−01
f_5	mean	9.009e+02 +	9.022e+02 +	9.004e+02 +	9.005e+02 +	9.003e+02 +	9.003e+02 +	9.003e+02 +	9.001e+02 +	9.014e+02 +	9.001e+02 +	9.000e+02 +	**9.000e+02**
	std	6.853e−01	5.628e−01	7.714e−02	1.542e−01	7.719e−02	8.072e−02	8.803e−02	3.828e−02	4.830e−01	2.865e−01	3.055e−03	4.329e−04
f_6	mean	1.261e+06 +	1.045e+05 +	**4.123e+03** −	1.551e+05 +	5.419e+04 +	3.966e+04 +	3.293e+04 +	2.657e+04 +	2.121e+06 +	2.388e+04 +	2.231e+04 +	1.375e+04
	std	2.146e+06	3.778e+04	6.533e+02	9.079e+04	3.349e+04	1.466e+04	1.348e+04	8.120e+03	1.087e+06	1.346e+04	9.698e+03	4.157e+03
f_7	mean	2.132e+03 +	2.072e+03 +	2.077e+03 +	2.062e+03 +	2.037e+03 +	2.038e+03 +	2.039e+03 +	2.037e+03 +	2.153e+03 +	2.039e+03 +	2.037e+03 +	**2.032e+03**
	std	6.052e+01	1.172e+01	1.623e+01	1.095e+01	3.869e+00	5.161e+00	5.373e+00	3.304e+00	6.249e+01	2.366e+01	4.515e+00	3.531e+00
f_8	mean	3.743e+03 +	2.255e+03 +	2.228e+03 ≈	2.306e+03 +	2.245e+03 +	2.239e+03 +	2.238e+03 +	2.232e+03 +	2.931e+03 +	2.243e+03 +	2.230e+03 +	**2.228e+03**
	std	4.228e+03	7.304e+00	3.227e+00	4.656e+01	9.608e+00	5.330e+00	6.710e+00	3.867e+00	6.855e+02	7.461e+01	3.043e+00	1.643e+00
f_9	mean	2.657e+03 +	2.665e+03 +	2.582e+03 +	2.514e+03 +	2.387e+03 +	2.334e+03 +	2.335e+03 +	**2.315e+03** −	2.700e+03 +	2.659e+03 +	2.351e+03 +	2.328e+03
	std	1.633e+02	4.502e+01	1.306e+02	1.077e+02	6.536e+01	1.807e+01	2.029e+01	9.960e+00	6.596e+01	6.941e+01	1.098e+02	9.309e+01
f_{10}	mean	2.618e+03 +	2.623e+03 +	2.606e+03 +	2.624e+03 +	2.611e+03 +	2.609e+03 +	2.610e+03 +	2.607e+03 +	2.647e+03 +	2.641e+03 +	2.601e+03 +	**2.600e+03**
	std	9.079e+00	4.233e+00	8.057e−01	9.644e+00	2.006e+00	2.449e+00	3.375e+00	1.670e+00	2.050e+01	1.520e+02	1.136e+00	6.390e−01
f_{11}	mean	2.623e+03 +	2.661e+03 +	2.652e+03 +	2.616e+03 +	2.603e+03 +	2.603e+03 +	2.603e+03 +	2.601e+03 +	2.657e+03 +	2.692e+03 +	2.601e+03 ≈	**2.600e+03**
	std	1.356e+01	1.513e+02	1.747e+02	3.755e+00	8.060e−01	7.118e−01	9.712e−01	1.770e−01	1.273e+01	2.614e+02	3.579e−01	1.373e−02
f_{12}	mean	2.905e+03 +	2.868e+03 +	2.867e+03 +	2.879e+03 +	2.868e+03 +	2.868e+03 +	2.868e+03 +	2.867e+03 +	2.874e+03 +	**2.867e+03** ≈	2.867e+03 ≈	2.867e+03
	std	1.978e+01	7.310e−01	5.121e−01	3.213e+00	6.639e−01	6.616e−01	8.566e−01	4.828e−01	1.756e+00	1.096e+00	3.865e−01	3.289e−01
+/≈/−:		12/0/0	12/0/0	10/1/1	11/1/0	11/1/0	11/0/1	11/0/1	10/1/1	12/0/0	8/4/0	9/3/0	–

Table 8 Experimental results and statistical analyses on 20-D CEC2022 benchmark functions

Func.		PSO	DE	CMA-ES	CLPSO	JADE	SHADE	L-SHADE	iL-SHADE	AHPSO	GTDE	MBGO	GTMBGO
f_1	mean	1.298e+04 +	3.428e+04 +	2.593e+03 +	5.348e+03 +	5.009e+03 +	1.243e+03 +	1.330e+03 +	4.825e+02 +	1.552e+04 +	3.553e+02 ≈	8.437e+02 +	**3.223e+02**
	std	5.865e+03	4.837e+03	4.391e+02	1.074e+03	1.227e+03	3.950e+02	3.691e+02	5.974e+01	2.739e+03	1.238e+02	3.666e+02	2.029e+01
f_2	mean	7.208e+02 +	1.047e+03 +	5.167e+02 +	5.115e+02 +	4.568e+02 −	4.574e+02 −	4.623e+02 ≈	**4.539e+02** −	6.686e+02 +	4.626e+02 ≈	4.754e+02 ≈	4.669e+02
	std	1.628e+02	1.361e+02	1.523e+01	1.336e+01	5.183e+00	8.878e+00	1.082e+01	8.321e+00	4.631e+01	3.434e+01	3.277e+01	8.821e+00
f_3	mean	6.001e+02 +	6.004e+02 +	6.000e+02 +	6.000e+02 +	6.000e+02 +	6.000e+02 +	6.000e+02 +	6.000e+02 +	6.001e+02 +	6.000e+02 +	6.000e+02 +	**6.000e+02**
	std	1.075e−01	6.400e−02	5.566e−03	3.662e−04	2.952e−05	4.529e−06	5.666e−06	1.698e−07	3.239e−02	8.585e−03	1.251e−06	2.713e−09
f_4	mean	8.037e+02 +	8.041e+02 +	8.039e+02 +	8.020e+02 −	8.019e+02 −	8.017e+02 −	**8.016e+02** −	8.020e+02 −	8.028e+02 ≈	8.016e+02 −	8.028e+02 ≈	8.027e+02
	std	4.074e−01	3.267e−01	4.267e−01	3.014e−01	2.638e−01	1.985e−01	2.593e−01	2.927e−01	4.094e−01	6.611e−01	3.165e−01	4.433e−01
f_5	mean	9.049e+02 +	9.150e+02 +	9.032e+02 +	9.023e+02 +	9.015e+02 +	9.012e+02 +	9.012e+02 +	9.004e+02 +	9.053e+02 +	9.018e+02 +	9.000e+02 +	**9.000e+02**
	std	3.275e+00	2.456e+00	5.982e−01	6.401e−01	3.791e−01	4.006e−01	3.582e−01	1.113e−01	1.755e+00	1.778e+00	5.632e−02	4.430e−02
f_6	mean	2.626e+08 +	2.185e+08 +	4.948e+06 +	9.175e+06 +	7.318e+06 +	4.739e+06 +	2.232e+06 +	2.553e+06 +	8.583e+07 +	3.849e+05 +	1.177e+05 +	**3.136e+04**
	std	3.635e+08	7.114e+07	1.895e+06	4.246e+06	3.170e+06	1.902e+06	8.215e+05	1.117e+06	3.823e+07	6.999e+05	5.366e+04	5.940e+03
f_7	mean	2.938e+03 +	2.819e+03 +	2.306e+03 +	2.221e+03 +	2.141e+03 +	2.138e+03 +	2.118e+03 +	2.096e+03 +	2.628e+03 +	2.124e+03 +	2.058e+03 +	**2.043e+03**
	std	5.545e+02	2.297e+02	6.084e+01	6.063e+01	2.877e+01	3.168e+01	2.596e+01	1.656e+01	2.024e+02	7.661e+01	1.380e+01	3.538e+00
f_8	mean	4.229e+07 +	1.113e+05 +	**2.415e+03** ≈	3.254e+03 +	3.310e+03 +	2.923e+03 +	2.869e+03 +	2.929e+03 +	1.591e+04 +	3.982e+03 +	3.228e+03 +	2.487e+03
	std	1.097e+08	2.665e+05	7.471e+01	4.877e+02	5.496e+02	3.360e+02	2.464e+02	2.527e+02	2.515e+04	1.492e+03	6.699e+02	3.352e+02
f_9	mean	3.207e+03 +	2.782e+03 +	2.653e+03 ≈	2.720e+03 +	2.669e+03 +	2.667e+03 +	2.673e+03 +	2.654e+03 ≈	2.827e+03 +	**2.639e+03** −	2.663e+03 ≈	2.658e+03
	std	2.208e+02	5.008e+01	6.486e+00	1.157e+01	9.833e+00	8.192e+00	9.534e+00	6.016e+00	3.391e+01	6.194e+00	1.416e+01	5.972e+00
f_{10}	mean	2.832e+03 +	3.594e+03 +	3.148e+03 +	2.875e+03 +	2.844e+03 +	2.829e+03 +	2.819e+03 +	2.807e+03 +	3.107e+03 +	3.580e+03 +	**2.766e+03** ≈	2.777e+03
	std	3.482e+01	1.362e+03	8.562e+02	1.165e+02	5.928e+01	7.664e+01	2.898e+01	1.928e+01	5.609e+02	1.351e+03	8.156e+00	3.804e+01
f_{11}	mean	2.801e+03 +	2.732e+03 +	2.621e+03 +	2.619e+03 +	2.602e+03 ≈	2.601e+03 ≈	2.601e+03 ≈	**2.600e+03** ≈	2.729e+03 +	2.938e+03 +	2.614e+03 +	2.601e+03
	std	3.397e+02	2.868e+01	2.187e+00	3.117e+00	5.352e−01	2.056e−01	3.386e−01	8.156e−02	4.049e+01	5.723e+02	3.278e+01	3.573e+00
f_{12}	mean	3.238e+03 +	2.956e+03 +	2.954e+03 +	2.988e+03 +	2.950e+03 +	2.952e+03 +	2.952e+03 +	2.949e+03 +	2.991e+03 +	2.954e+03 +	2.959e+03 +	**2.945e+03**
	std	9.552e+01	5.356e+00	6.520e+00	9.050e+00	3.209e+00	2.510e+00	2.711e+00	2.064e+00	1.234e+01	1.117e+01	1.462e+01	5.070e+00
+/≈/−:		12/0/0	12/0/0	10/2/0	11/0/1	9/1/2	9/1/2	9/2/1	8/2/2	11/1/0	8/2/2	8/4/0	–

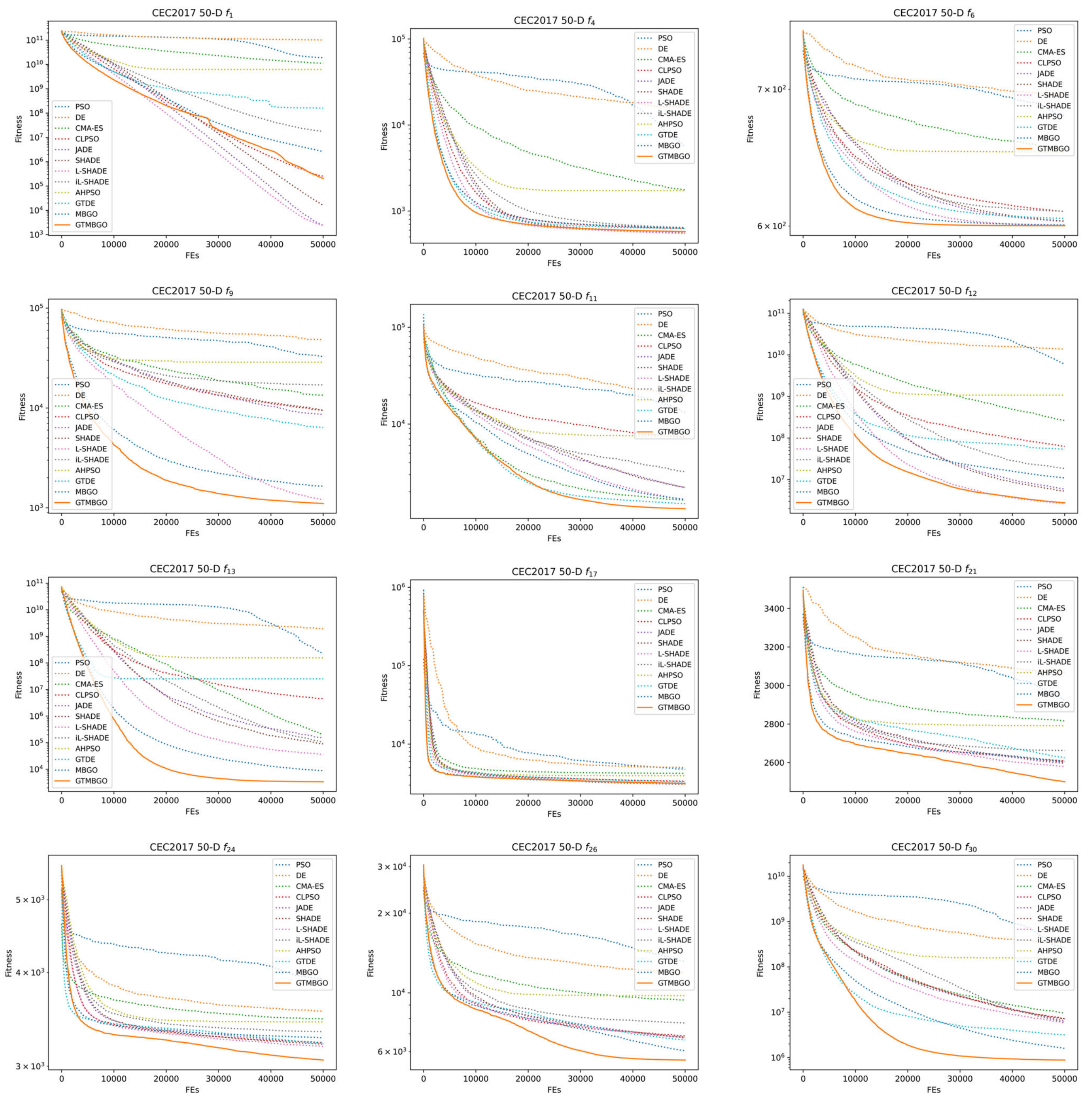


Fig. 7 Convergence curves of competitor optimizers on 50-D CEC2017 representative benchmark functions

4.2.2 Comparison experiments on CEC2022

Comparison experiments on CEC2022 benchmark functions are conducted to investigate the performance of involved optimizers in low-dimensional search spaces. Tables 7 and 8 summarize the experimental results and statistical analyses, while the convergence curves are demonstrated in Figs. 8 and 9.

4.2.3 Ablation experiments on CEC2017

Ablation experiments on CEC2017 benchmark functions are conducted to investigate the contribution of modifications in the proposed GTMBGO independently. MBGO: The original MBGO algorithm, MBGO1: MBGO without the movement phase, MBGO2: MBGO with gene-targeting operator, GTMBGO: our complete proposal. Tables 9 and 10 summarize the experimental results and statistical analyses on CEC2017 benchmark functions.

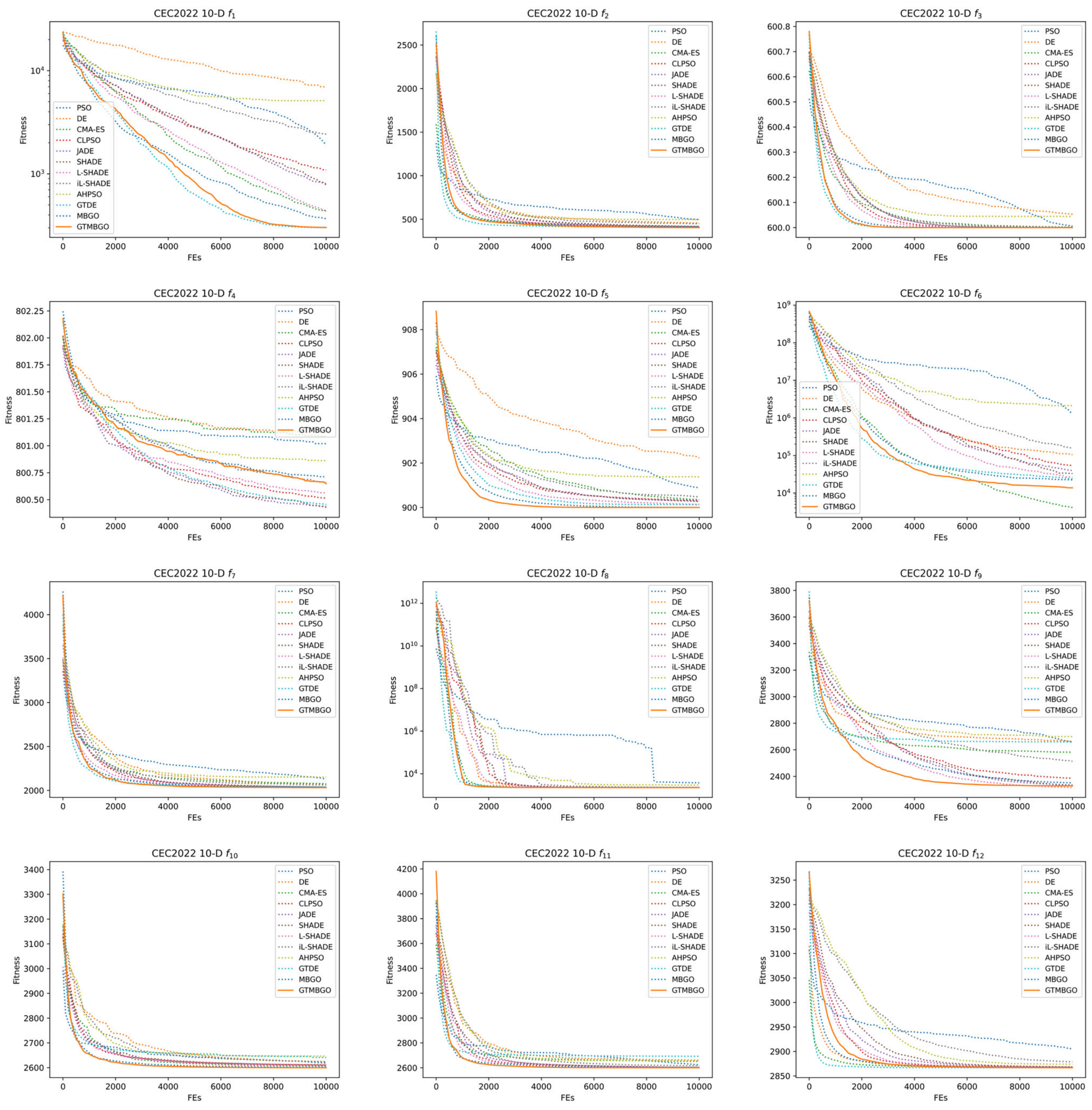


Fig. 8 Convergence curves of competitor optimizers on 10-D CEC2022 benchmark functions

4.2.4 Comparison experiments on CEC2013 LSGO

Since the scales of benchmark functions in CEC2013 LSGO are 905 and 1000, the optimization on these functions can reflect the performance of tested optimizers in large-scale search domains, and the optimization results are summarized in Table 11.

5 Discussion

This section analyzes the performance of our proposal. We present the theoretical computational analysis for ERDG$_k$ and GTMBGO at first, then, the performance analyses on CEC2017, CEC2022, ablation experiments, and CEC2013 LSGO benchmark functions are demonstrated subsequently.

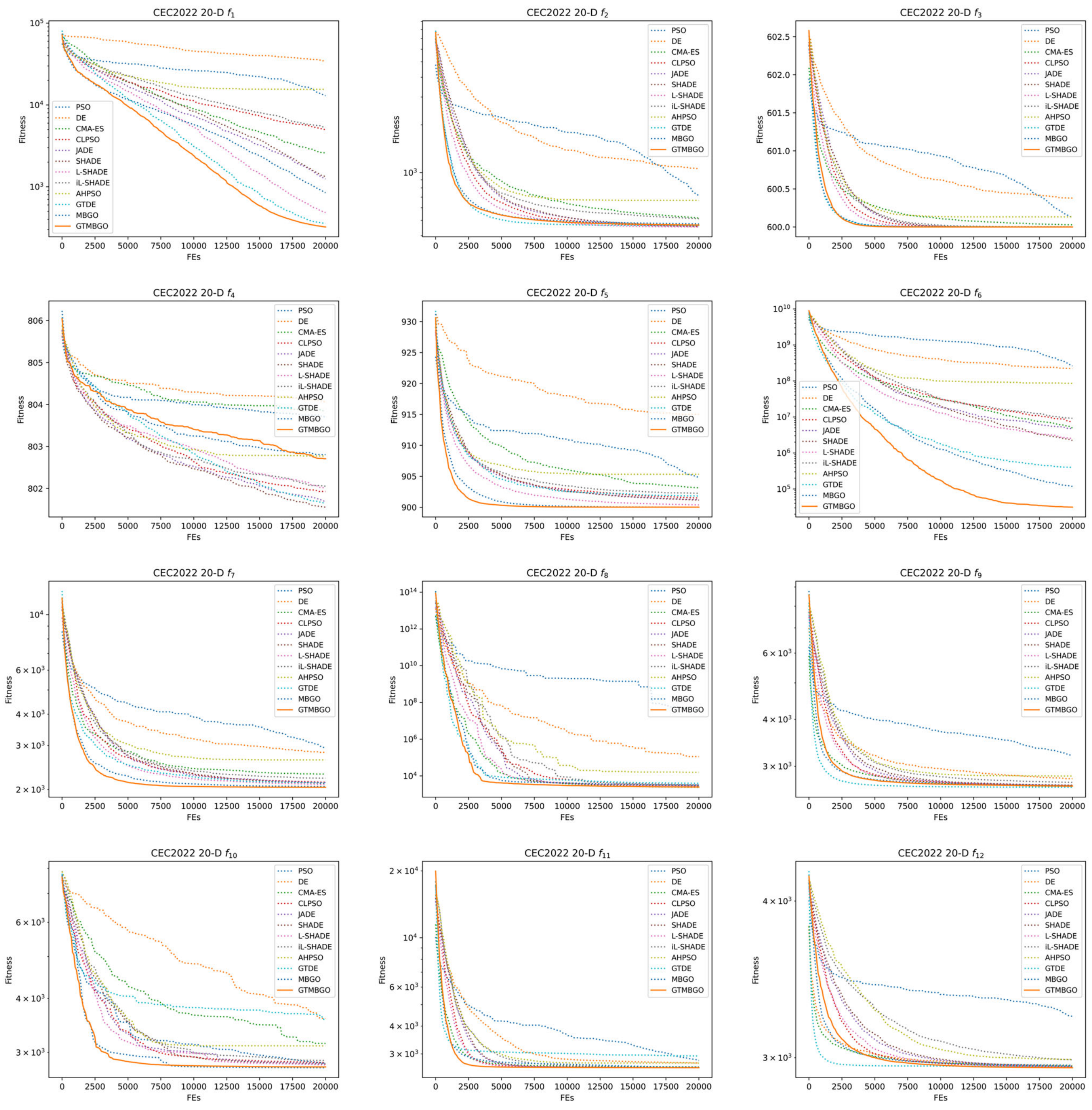


Fig. 9 Convergence curves of competitor optimizers on 20-D CEC2022 benchmark functions

5.1 Computational complexity analysis

Supposing the population size is N, the dimension size is D, and the maximum iteration is T. In the following context, we analyze the computational complexity of the decomposition technique ERDG$_k$ and the optimizer GTMBGO separately.

Computational complexity of ERDG$_k$: ERDG$_k$ inherits the main architecture of ERDG and integrates with an external random decomposition. Therefore, the theoretical computational complexity of ERDG$_k$ is $O(D \times \log D)$, which is identical to the original ERDG.

Computational complexity of GTMBGO: GTMBGO mainly consists of three components: population initialization, gene-targeting operator, and battle phase.

- Population initialization: $O(N \times D)$.
- Gene-targeting operator: As described in Algorithm 5, the gene-targeting operator is specified for the current best individual. Since the first step is locating the

Table 9 Ablation experiments on CEC2017 benchmark functions

Func.		30-D				50-D			
		MBGO	MBGO1	MBGO2	GTMBGO	MBGO	MBGO1	MBGO2	GTMBGO
f_1	Mean	3.445e+05	1.228e+05 −	1.266e+05 −	**5.604e+03** −	2.620e+06	1.148e+07 +	2.106e+06 −	2.057e+05 −
	Std	1.720e+05	1.077e+05	1.158e+05	3.269e+03	1.035e+06	2.698e+07	2.512e+06	1.578e+05
f_3	Mean	3.611e+04	3.123e+04 −	2.516e+04 −	**2.463e+04** −	8.646e+04	7.896e+04 −	6.831e+04 −	**6.366e+04** −
	Std	6.882e+03	6.517e+03	6.559e+03	4.768e+03	1.479e+04	1.537e+04	1.274e+04	1.243e+04
f_4	Mean	5.244e+02	**5.124e+02** −	5.275e+02 ≈	5.126e+02 −	6.425e+02	6.170e+02 −	6.393e+02 ≈	**5.771e+02** −
	Std	3.098e+01	1.675e+01	2.461e+01	2.372e+01	3.815e+01	4.404e+01	5.417e+01	4.194e+01
f_5	Mean	6.687e+02	6.569e+02 −	6.598e+02 −	**6.409e+02** −	8.227e+02	7.639e+02 −	8.103e+02 −	**7.472e+02** −
	Std	1.132e+01	1.384e+01	1.117e+01	2.523e+01	4.202e+01	5.729e+01	2.779e+01	6.196e+01
f_6	Mean	6.001e+02	6.000e+02 −	6.000e+02 −	**6.000e+02** −	6.008e+02	6.003e+02 −	6.004e+02 −	**6.002e+02** −
	Std	1.231e−01	6.586e−02	5.360e−02	4.098e−03	5.478e−01	3.504e−01	3.595e−01	2.505e−01
f_7	Mean	9.054e+02	8.985e+02 ≈	8.990e+02 ≈	**8.922e+02** −	1.132e+03	1.088e+03 −	1.101e+03 −	**1.072e+03** −
	Std	1.164e+01	1.803e+01	1.767e+01	1.414e+01	2.496e+01	2.570e+01	3.551e+01	4.067e+01
f_8	Mean	9.596e+02	9.474e+02 −	9.545e+02 ≈	**9.334e+02** −	1.132e+03	1.075e+03 −	1.099e+03 −	**1.032e+03** −
	Std	1.193e+01	1.838e+01	1.433e+01	2.185e+01	2.357e+01	5.551e+01	4.114e+01	5.724e+01
f_9	Mean	9.182e+02	9.114e+02 −	**9.102e+02** −	9.127e+02 −	1.647e+03	1.363e+03 −	1.274e+03 −	**1.106e+03** −
	Std	1.540e+01	1.623e+01	6.382e+00	2.801e+01	6.101e+02	3.413e+02	4.220e+02	2.398e+02
f_{10}	Mean	7.917e+03	7.911e+03 ≈	**7.698e+03** −	7.896e+03 ≈	1.352e+04	1.355e+04 ≈	**1.322e+04** −	1.322e+04 −
	Std	3.128e+02	3.407e+02	3.076e+02	2.936e+02	4.970e+02	5.676e+02	5.239e+02	4.835e+02
f_{11}	Mean	1.298e+03	1.259e+03 −	1.250e+03 −	**1.232e+03** −	1.665e+03	1.511e+03 −	1.397e+03 −	**1.335e+03** −
	Std	3.701e+01	2.530e+01	2.888e+01	3.091e+01	1.395e+02	1.044e+02	6.620e+01	4.749e+01
f_{12}	Mean	1.891e+06	9.465e+05 −	1.399e+06 ≈	**6.711e+05** −	1.099e+07	3.993e+06 −	9.171e+06 ≈	**2.769e+06** −
	Std	1.598e+06	5.709e+05	1.061e+06	3.970e+05	4.819e+06	2.651e+06	5.100e+06	1.124e+06
f_{13}	Mean	1.577e+04	1.206e+04 ≈	1.292e+04 ≈	**1.054e+04** ≈	8.837e+03	4.013e+03 −	5.479e+03 −	**3.397e+03** −
	Std	1.701e+04	6.961e+03	1.493e+04	6.330e+03	3.667e+03	2.079e+03	2.563e+03	1.358e+03
f_{14}	Mean	5.236e+03	2.612e+03 −	2.354e+03 −	**2.301e+03** −	5.552e+04	2.565e+04 −	2.147e+04 −	**1.454e+04** −
	Std	2.320e+03	7.018e+02	6.402e+02	4.835e+02	4.760e+04	1.536e+04	1.482e+04	7.973e+03
f_{15}	Mean	1.328e+04	9.342e+03 ≈	1.018e+04 ≈	**5.723e+03** −	**5.423e+03**	6.599e+03 ≈	5.571e+03 ≈	7.419e+03 ≈
	Std	9.256e+03	4.188e+03	5.480e+03	2.755e+03	3.770e+03	3.023e+03	5.723e+03	3.940e+03
f_{16}	Mean	2.928e+03	2.911e+03 ≈	2.826e+03 −	**2.783e+03** −	3.873e+03	**3.345e+03** −	3.810e+03 ≈	3.346e+03 −
	Std	2.214e+02	2.095e+02	1.820e+02	2.554e+02	3.886e+02	4.288e+02	4.507e+02	4.099e+02

causative genes, the computational complexity of this process is $O(D)$, then, the construction of the homologous targeting vector consumes $O(D)$ computational resource, and finally, the gene-targeting operator inserts the vector into the embryonic stem cell, where the computational complexity is also $O(D)$. Therefore, the computational complexity of the gene-targeting operator is $O(3D)$.

- Battle phase: $O((N-1)\times D)$.

In summary, the computational complexity of GTMBGO is $O(N\times D+T\times(3D+(N-1)\times D)) := O(T\times D\times N)$.

5.2 Performance analyses on CEC2017 and CEC2022

We first emphasize the remarkable effectiveness and efficiency demonstrated by GTMBGO in tackling the challenging benchmark functions of CEC2017 and CEC2022. Our findings reveal that GTMBGO exhibits performance levels that are exceptionally competitive when compared to both state-of-the-art optimizers and advanced variants of DE and PSO. This underscores the significant contribution made by the integration of the gene-targeting operator and the streamlined movement phase within GTMBGO. These enhancements accelerate the optimization process and yield notable improvements in convergence accuracy.

Nevertheless, it is imperative to acknowledge certain degenerative scenarios within GTMBGO when compared

Table 10 Ablation experiments on CEC2017 benchmark functions (Continued)

Func.		30-D				50-D			
		MBGO	MBGO1	MBGO2	GTMBGO	MBGO	MBGO1	MBGO2	GTMBGO
f_{17}	Mean	2.002e+03	1.987e+03 ≈	1.979e+03 ≈	**1.952e+03** –	3.150e+03	3.194e+03 ≈	**3.107e+03** ≈	3.124e+03 ≈
	Std	8.860e+01	1.106e+02	8.788e+01	8.179e+01	2.630e+02	1.627e+02	2.216e+02	1.862e+02
f_{18}	Mean	1.408e+05	8.250e+04 –	6.646e+04 –	**5.224e+04** –	7.149e+05	4.260e+05 –	2.965e+05 –	**2.682e+05** –
	Std	6.872e+04	4.996e+04	3.269e+04	1.615e+04	4.596e+05	2.263e+05	1.836e+05	1.402e+05
f_{19}	Mean	1.344e+04	8.306e+03 ≈	6.621e+03 ≈	**5.552e+03** –	1.387e+04	1.657e+04 ≈	**1.347e+04** ≈	1.691e+04 ≈
	Std	1.856e+04	4.023e+03	4.244e+03	2.266e+03	7.928e+03	4.160e+03	9.608e+03	4.378e+03
f_{20}	Mean	2.517e+03	2.439e+03 –	**2.421e+03** –	2.432e+03 –	3.414e+03	3.399e+03 ≈	3.319e+03 –	**3.276e+03** –
	Std	8.756e+01	1.298e+02	8.727e+01	9.092e+01	1.801e+02	1.547e+02	1.534e+02	1.705e+02
f_{21}	Mean	2.452e+03	2.441e+03 –	2.450e+03 ≈	**2.430e+03** –	2.606e+03	2.544e+03 –	2.593e+03 ≈	**2.500e+03** –
	Std	1.334e+01	1.699e+01	1.600e+01	1.968e+01	3.934e+01	4.972e+01	6.380e+01	7.152e+01
f_{22}	Mean	2.308e+03	2.303e+03 –	2.304e+03 –	**2.301e+03** –	1.058e+04	9.789e+03 ≈	9.857e+03 ≈	**9.202e+03** –
	Std	3.189e+00	1.699e+00	2.050e+00	1.345e+00	5.970e+03	6.025e+03	6.130e+03	6.018e+03
f_{23}	Mean	2.806e+03	2.780e+03 –	2.797e+03 ≈	**2.746e+03** –	3.053e+03	2.919e+03 –	3.019e+03 ≈	**2.873e+03** –
	Std	1.855e+01	3.214e+01	2.171e+01	3.892e+01	5.932e+01	7.876e+01	7.173e+01	6.050e+01
f_{24}	Mean	3.000e+03	2.936e+03 –	2.982e+03 –	**2.919e+03** –	3.274e+03	3.061e+03 –	3.222e+03 –	**3.058e+03** –
	Std	1.764e+01	2.932e+01	1.293e+01	3.797e+01	3.222e+01	7.646e+01	5.381e+01	6.626e+01
f_{25}	Mean	2.920e+03	2.929e+03 ≈	2.917e+03 ≈	**2.912e+03** ≈	3.150e+03	3.167e+03 ≈	**3.142e+03** ≈	3.151e+03 ≈
	Std	2.281e+01	2.605e+01	2.262e+01	1.976e+01	3.149e+01	3.696e+01	4.360e+01	2.487e+01
f_{26}	Mean	5.278e+03	4.806e+03 –	4.946e+03 ≈	**4.489e+03** –	6.042e+03	5.717e+03 ≈	6.126e+03 ≈	**5.574e+03** –
	Std	2.861e+02	5.297e+02	7.104e+02	4.616e+02	8.505e+02	5.322e+02	8.550e+02	4.364e+02
f_{27}	Mean	3.232e+03	3.224e+03 –	3.229e+03 ≈	**3.222e+03** –	3.514e+03	3.426e+03 –	3.465e+03 –	**3.413e+03** –
	Std	1.573e+01	1.270e+01	1.324e+01	1.202e+01	7.162e+01	3.836e+01	5.808e+01	5.474e+01
f_{28}	Mean	3.292e+03	3.300e+03 ≈	3.303e+03 ≈	**3.283e+03** ≈	3.477e+03	3.529e+03 ≈	3.485e+03 ≈	**3.451e+03** ≈
	Std	3.129e+01	1.815e+01	3.862e+01	2.264e+01	5.855e+01	5.459e+01	5.758e+01	4.260e+01
f_{29}	Mean	3.898e+03	3.796e+03 ≈	3.852e+03 ≈	**3.761e+03** –	4.235e+03	4.146e+03 ≈	4.405e+03 ≈	**4.006e+03** –
	Std	2.037e+02	1.677e+02	1.666e+02	1.398e+02	3.130e+02	2.994e+02	3.480e+02	2.860e+02
f_{30}	Mean	9.064e+04	2.871e+04 –	3.342e+04 –	**1.496e+04** –	1.591e+06	9.614e+05 –	1.118e+06 –	**8.806e+05** –
	Std	1.646e+05	2.578e+04	2.449e+04	1.007e+04	3.578e+05	9.594e+04	2.236e+05	8.964e+04
+/≈/–:		–	0/10/19	0/15/14	0/4/25	-	1/10/18	0/13/16	0/5/24

with renowned optimizers such as JADE, SHADE, L-SHADE, iL-SHADE, and GTDE, particularly evident in the optimization of composite functions such as CEC2017 f_{25}, f_{27}, and f_{28}. These functions present intricate fitness landscapes characterized by multiple local optima, posing formidable challenges for optimizers to balance exploration and exploitation while escaping local optima traps effectively.

To explain the inferiority in these instances, we first emphasize the competitiveness of JADE, SHADE, L-SHADE, iL-SHADE, and GTDE. These algorithms are well-established and state-of-the-art optimizers in the CEC benchmark testing. Therefore, it is reasonable for these optimizers to achieve remarkable performance and be significantly better than GTMBGO in some instances of the composite functions. Additionally, we turn to the No Free Lunch Theorem (NFLT) [58] to explain the possible deterioration of our proposed GTMBGO in specific instances. NFLT posits that any pair of optimization algorithms will exhibit identical average performance across all conceivable problems. If an algorithm demonstrates superior performance on a specific category of problems, it inevitably falters on others to maintain this equilibrium. Consequently, the focal point of research within the EC community lies in the development of problem-specific optimizers. In summary, the improvement between the original MBGO and GTMBGO is observable and obvious, which showcases superior optimization performance on the CEC2017 and CEC2022 benchmark functions.

Table 11 Experimental results and statistical analyses on CEC2013 LSGO benchmark functions

Func.		CCPSO2	DECC-DG	DECC-DG2	DECC-RDG	DECC-MDG	μDSDE	DMS-L-PSO	Hip-DE	FADE	eWOA	DSCA	GTMBGO-ERDG$_k$
f_1	Mean	3.53e+01 +	7.49e+03 +	1.19e+02 +	1.07e+00 +	1.08e+00 +	1.73e+07 +	5.70e+09 +	5.43e+10 +	3.93e+09 +	9.00e+09 +	2.89e+10 +	**7.87e-06**
	Std	2.33e+01	3.00e+04	3.14e+02	2.68e-01	1.45e+00	7.75e+05	1.44e+08	1.21e+10	9.00e+08	5.29e+08	8.79e+09	1.02e-06
f_2	Mean	3.57e+01 +	1.29e+04 +	1.26e+04 +	1.29e+04 +	1.21e+04 +	3.95e+04 +	1.24e+04 +	4.68e+04 +	2.64e+04 +	7.88e+04 +	8.61e+04 +	**1.05e-04**
	Std	5.22e+00	5.78e+02	6.78e+02	5.98e+02	4.54e+02	9.65e+02	2.69e+02	3.27e+03	1.67e+03	2.95e+02	9.56e+02	6.74e-05
f_3	Mean	**2.00e+01** −	2.14e+01 +	2.12e+01 ≈	2.14e+01 +	2.14e+01 +	2.15e+01 +	2.14e+01 +	2.14e+01 +	2.06e+01 −	2.16e+01 +	2.16e+01 +	2.12e+01
	Std	9.00e-05	1.58e-02	8.10e-03	1.74e-02	1.89e-02	2.56e-02	1.92e-02	1.28e-01	3.60e-02	5.42e-02	6.27e-02	1.72e-02
f_4	Mean	3.41e+10 −	8.20e+10 +	6.10e+10 +	8.53e+10 +	3.68e+10 −	2.79e+10 −	9.00e+11 +	5.87e+11 +	**2.37e+10** −	4.15e+12 +	1.11e+12 +	5.59e+10
	Std	1.97e+10	3.57e+10	1.84e+10	3.01e+10	1.31e+10	1.62e+09	3.78e+10	1.22e+11	6.98e+09	1.87e+11	1.71e+11	1.29e+10
f_5	Mean	1.64e+07 +	5.61e+06 +	5.40e+06 +	5.92e+06 +	4.81e+06 ≈	3.58e+07 +	5.49e+06 +	1.88e+07 +	6.51e+06 +	4.96e+07 +	4.91e+07 +	**4.39e+06**
	Std	4.22e+06	3.82e+05	4.45e+05	3.48e+05	6.85e+05	4.97e+06	4.19e+05	2.74e+06	6.22e+05	8.85e+05	1.77e+06	2.77e+05
f_6	Mean	1.05e+06 ≈	1.06e+06 +	1.06e+06 +	1.06e+06 +	1.06e+06 +	1.04e+06 −	**1.03e+06** −	1.06e+06 +	1.06e+06 +	1.06e+06 +	1.06e+06 +	1.05e+06
	Std	7.68e+03	1.68e+03	1.12e+03	1.12e+03	2.08e+03	5.89e+03	3.99e+03	3.36e+03	1.44e+03	3.18e+03	5.84e+03	1.07e+03
f_7	Mean	3.04e+08 +	4.72e+08 +	4.79e+07 −	9.94e+07 ≈	6.57e+07 −	**6.08e+06** −	3.55e+09 +	3.45e+10 +	2.44e+08 +	5.62e+11 +	5.18e+11 +	9.59e+07
	Std	4.42e+08	2.41e+08	2.11e+07	2.57e+07	2.82e+07	3.20e+05	2.61e+08	3.90e+10	8.37e+07	6.01e+09	2.25e+09	3.87e+07
f_8	Mean	1.02e+15 ≈	4.05e+15 +	5.88e+15 +	4.01e+15 +	3.68e+15 +	2.14e+14 −	6.79e+15 +	2.29e+15 ≈	**5.60e+13** −	5.08e+16 +	2.04e+16 +	1.89e+15
	Std	5.48e+14	2.11e+15	1.97e+15	1.53e+15	9.68e+15	8.34e+13	1.38e+15	1.32e+15	6.89e+12	8.35e+14	4.41e+15	2.23e+14
f_9	Mean	3.72e+09 +	4.82e+08 +	5.06e+08 +	4.78e+08 +	4.95e+08 +	5.21e+09 +	5.05e+08 +	1.33e+09 +	4.83e+08 +	7.15e+09 +	4.97e+09 +	**3.33e+08**
	Std	1.01e+09	2.38e+07	3.23e+07	2.88e+07	5.13e+07	6.02e+08	2.66e+07	1.71e+08	8.23e+07	6.43e+07	8.11e+07	5.69e+07
f_{10}	Mean	9.32e+07 ≈	9.45e+07 +	9.45e+07 +	9.44e+07 +	9.41e+07 +	**9.22e+07** −	9.31e+07 ≈	9.48e+07 +	9.42e+07 +	9.46e+07 +	9.48e+07 +	9.30e+07
	Std	4.92e+05	2.74e+05	3.06e+05	2.42e+05	1.77e+05	6.38e+05	3.11e+05	2.90e+05	2.31e+05	9.84e+05	3.77e+06	2.60e+04
f_{11}	Mean	9.32e+11 +	4.09e+10 +	4.98e+09 −	3.58e+09 −	**2.16e+09** −	1.36e+13 +	4.96e+11 +	1.23e+13 +	3.69e+09 −	4.56e+14 +	1.20e+14 +	1.20e+10
	Std	1.08e+10	5.06e+10	5.18e+09	8.87e+09	6.99e+08	3.10e+12	4.01e+10	9.05e+12	1.27e+09	8.21e+12	9.66e+12	3.23e+09
f_{12}	Mean	2.15e+03 ≈	1.68e+11 +	2.13e+05 +	**1.34e+03** −	3.66e+03 ≈	1.55e+04 +	4.42e+09 +	1.57e+12 +	2.49e+11 +	2.95e+11 +	2.51e+11 +	4.23e+03
	Std	2.12e+02	1.61e+10	7.59e+05	9.28e+01	4.02e+02	2.20e+03	8.34e+08	2.38e+11	2.90e+10	8.51e+09	6.45e+09	3.60e+02
f_{13}	Mean	4.13e+09 ≈	1.98e+10 +	1.49e+09 −	9.96e+09 +	**1.09e+09** −	1.22e+16 +	1.16e+11 +	1.01e+13 +	7.34e+09 ≈	7.58e+16 +	4.19e+16 +	6.33e+09
	Std	1.70e+09	4.84e+09	3.81e+08	2.41e+09	3.29e+08	8.39e+15	1.05e+10	1.14e+13	1.60e+09	3.53e+14	7.97e+14	3.78e+08
f_{14}	Mean	9.37e+10 ≈	2.02e+10 −	5.86e+09 −	3.69e+10 −	4.23e+09 −	**4.35e+08** −	1.25e+12 +	1.64e+13 +	5.88e+10 −	5.19e+13 +	8.16e+12 +	1.19e+11
	Std	1.02e+11	1.20e+10	3.58e+09	1.33e+10	4.72e+09	3.71e+07	1.59e+11	1.34e+13	2.24e+10	3.80e+12	4.17e+11	1.29e+10
f_{15}	Mean	**6.95e+06** −	9.44e+06 −	1.13e+07 ≈	9.85e+06 −	1.00e+07 −	8.01e+06 −	1.60e+09 +	1.67e+14 +	2.25e+08 +	5.02e+14 +	5.51e+14 +	6.29e+07
	Std	5.80e+06	1.62e+06	2.34e+06	1.56e+06	1.61e+06	7.96e+05	6.18e+08	8.99e+13	2.61e+08	6.44e+12	5.14e+13	6.64e+06
+/≈/−:		6/6/3	13/0/2	9/2/4	10/1/4	7/2/6	8/0/7	13/1/1	14/1/0	8/1/6	15/0/0	15/0/0	–

5.3 Performance analyses on ablation experiments

Our investigation involves conducting ablation experiments on the CEC2017 benchmark functions, aiming to assess the individual contributions of the proposed strategies within GTMBGO. The experimental outcomes and subsequent statistical analyses, as delineated in Tables 9 and 10, underscore the substantial impact of both the streamlined movement phase and the integration of the efficient gene-targeting operator. These strategies collectively play a pivotal role in significantly accelerating convergence rates and improving optimization accuracy.

Based on the results from the designed ablation experiments, we advocate for the simultaneous incorporation of both proposed strategies within GTMBGO. Such an approach ensures optimal performance enhancements and underscores the efficiency between these complementary components.

5.4 Performance analyses on CEC2013 LSGO

Given the formidable challenge presented by the curse of dimensionality, our approach entails the fusion of the CC framework with GTMBGO to tackle LSOPs effectively. Among the latest and most advanced decomposition methods, ERDG emerges as an outstanding technique, renowned for its efficient computational budget and remarkable precision in decomposition tasks. However, we notice a notable limitation of ERDG in specific scenarios, most prominently illustrated by its handling of f_3 within the CEC2013 LSGO benchmark functions. In this context, ERDG's failure to identify the separability among decision variables results in the aggregation of all variables into a single group, despite the function's fully separable characteristic.

In response to this observed challenge, we leverage the foundational principles of ERDG and propose a strengthened variant named ERDG with maximum volume (ERDG$_k$). This novel variant is designed to impose constraints on the maximum scale of subcomponents, thereby enhancing optimization efficiency within such functions. The experimental results and statistical analyses as presented in Table 11 provide compelling evidence of the efficacy of GTMBGO-ERDG$_k$ in addressing LSOPs.

Meanwhile, we observe shortcomings in GTMBGO-ERDG$_k$, particularly in f_{15} when compared to CCPSO2, DECC-DG, DECC-RDG, DECC-MDG, and μDSDE. Given that f_{15} represents a fully non-separable function, interactions are abundant among the decision variables. Our proposed GTMBGO-ERDG$_k$ prefers to decompose this problem based on the pre-defined threshold, while the rest of the CC-based algorithms abstain from partitioning decision variables into sub-components. Although the decomposition of ERDG$_k$ alleviates the negative effect from the curse of dimensionality, it also neglects a large number of interactions and may misdirect the optimization process. Thus, striking the right balance between decomposition and optimization becomes pivotal in algorithm design for LSOPs.

6 Conclusion

This paper presents GTMBGO, a novel and efficient variant of MGBO designed to enhance optimization performance. Drawing from prior research, we introduce two key strategies in the development of GTMBGO: simplification of the movement phase and integration of the gene-targeting operator. Empirical evaluations on CEC2017 and CEC2022 benchmark functions underscore the competitiveness of GTMBGO against state-of-the-art optimizers and DE variants, with ablation experiments confirming the efficacy of the proposed strategies.

Additionally, leveraging the architecture of ERDG, we introduce an enhanced iteration named ERDG with maximum volume (ERDG$_k$). The fusion of GTMBGO and ERDG$_k$ yields a potent optimization methodology adept at addressing LSOPs.

Looking ahead, our future research endeavors will focus on further advancing algorithms and methodologies to solve large-scale real-world applications such as neural combinatorial optimization [59], reinforcement learning tasks [60], and feature selection challenges [61].

Acknowledgements This work was supported by JST SPRING Grant Number JPMJSP2119.

Author Contributions Rui Zhong: Conceptualization, methodology, investigation, writing–original draft, writing–review and editing, and funding acquisition. Jun Yu: investigation, methodology, formal analysis, writing–review and editing, and project administration.

Data Availability The source code of this research code can be downloaded from https://github.com/RuiZhong961230/GTMBGO.

Declarations

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

References

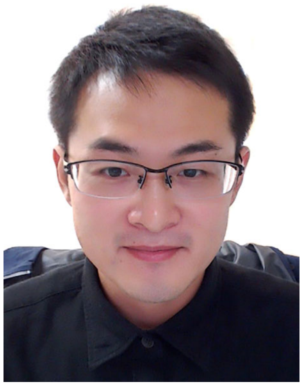
1. Zhang, Y., Li, S., Wang, Y., Yan, Y., Zhao, J., Gao, Z.: Self-adaptive enhanced learning differential evolution with surprisingly efficient decomposition approach for parameter identification of photovoltaic models. Energy Conver. Manag. **308**, 118387 (2024). https://doi.org/10.1016/j.enconman.2024.118387
2. Zhong, R., Peng, F., Yu, J., Munetomo, M.: Q-learning based vegetation evolution for numerical optimization and wireless sen-

sor network coverage optimization. Alexand. Eng. J. **87**, 148–163 (2024). https://doi.org/10.1016/j.aej.2023.12.028

3. Zhang, Y.-J., Wang, Y.-F., Yan, Y.-X., Zhao, J., Gao, Z.-M.: Lmraoa: an improved arithmetic optimization algorithm with multi-leader and high-speed jumping based on opposition-based learning solving engineering and numerical problems. Alexand. Eng. J. **61**, 12367–12403 (2022). https://doi.org/10.1016/j.aej.2022.06.017
4. Alorf, A.: A survey of recently developed metaheuristics and their comparative analysis. Eng. Appl. Artif. Intell. **117**, 105622 (2023). https://doi.org/10.1016/j.engappai.2022.105622
5. Zhong, R., Fan, Q., Zhang, C., Yu, J.: Hybrid remora crayfish optimization for engineering and wireless sensor network coverage optimization. Clust. Comput. 1–28 (2024). https://doi.org/10.1007/s10586-024-04508-1
6. Zhang, Y.-J., Wang, Y.-F., Yan, Y.-X., Zhao, J., Gao, Z.-M.: Self-adaptive hybrid mutation slime mould algorithm: case studies on uav path planning, engineering problems, photovoltaic models and infinite impulse response. Alexand. Eng. J. **98**, 364–389 (2024). https://doi.org/10.1016/j.aej.2024.04.075
7. Mousapour Mamoudan, M., Ostadi, A., Pourkhodabakhsh, N., Fathollahi-Fard, A.M., Soleimani, F.: Hybrid neural network-based metaheuristics for prediction of financial markets: a case study on global gold market. J. Comput. Design Eng. **10**(3), 1110–1125 (2023). https://doi.org/10.1093/jcde/qwad039
8. Gholizadeh, H., Goh, M., Fazlollahtabar, H., Mamashli, Z.: Modelling uncertainty in sustainable-green integrated reverse logistics network using metaheuristics optimization. Comput. Ind. Eng. **163**, 107828 (2022). https://doi.org/10.1016/j.cie.2021.107828
9. Saif, S., Das, P., Biswas, S., Khari, M., Shanmuganathan, V.: Hiids: Hybrid intelligent intrusion detection system empowered with machine learning and metaheuristic algorithms for application in iot based healthcare. Microprocess. Microsyst. 104622 (2022). https://doi.org/10.1016/j.micpro.2022.104622
10. Köppen, M.: The curse of dimensionality. In: 5th Online World Conference on Soft Computing in Industrial Applications (WSC5), vol. 1, pp. 4–8 (2000)
11. Tian, Y., Si, L., Zhang, X., Cheng, R., He, C., Tan, K.C., Jin, Y.: Evolutionary large-scale multi-objective optimization: A survey. ACM Comput. Surv. **54**(8) (2021). https://doi.org/10.1145/3470971
12. Hong, W., Yang, P., Tang, K.: Evolutionary computation for large-scale multi-objective optimization: A decade of progresses. Int. J. Autom. Comput. **18** (2021). https://doi.org/10.1007/s11633-020-1253-0
13. Liu, J., Sarker, R., Elsayed, S., Essam, D., Siswanto, N.: Large-scale evolutionary optimization: a review and comparative study. Swarm Evolut. Comput. **85**, 101466 (2024). https://doi.org/10.1016/j.swevo.2023.101466
14. Ma, X., Li, X., Zhang, Q., Tang, K., Liang, Z., Xie, W., Zhu, Z.: A survey on cooperative co-evolutionary algorithms. IEEE Trans. Evolut. Comput. **23**(3), 421–441 (2019). https://doi.org/10.1109/TEVC.2018.2868770
15. Omidvar, M.N., Li, X., Yao, X.: A review of population-based metaheuristics for large-scale black-box global optimization-part i. IEEE Trans. Evolut. Comput. **26**(5), 802–822 (2022). https://doi.org/10.1109/TEVC.2021.3130838
16. Omidvar, M.N., Li, X., Yao, X.: A review of population-based metaheuristics for large-scale black-box global optimization-part ii. IEEE Trans. Evolut. Comput. **26**(5), 823–843 (2022). https://doi.org/10.1109/TEVC.2021.3130835
17. Potter, M.A., De Jong, K.A.: A cooperative coevolutionary approach to function optimization. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 866 LNCS, 249–257 (1994)
18. Yu, Y., Xinjie, Y.: Cooperative coevolutionary genetic algorithm for digital iir filter design. IEEE Trans. Ind. Electron. **54**(3), 1311–1318 (2007). https://doi.org/10.1109/TIE.2007.893063
19. Yildiz, Y.E., Topal, A.O.: Large scale continuous global optimization based on micro differential evolution with local directional search. Inform. Sci. **477**, 533–544 (2019). https://doi.org/10.1016/j.ins.2018.10.046
20. Rui, Z., Zhang, E., Munetomo, M.: Cooperative coevolutionary surrogate ensemble-assisted differential evolution with efficient dual differential grouping for large-scale expensive optimization problems. Complex Intell. Syst. **10**, 2129–2149 (2023). https://doi.org/10.1007/s40747-023-01262-6
21. Yang, Q., Chen, W.-N., Deng, J.D., Li, Y., Gu, T., Zhang, J.: A level-based learning swarm optimizer for large-scale optimization. IEEE Trans. Evolut. Comput. **22**(4), 578–594 (2018). https://doi.org/10.1109/TEVC.2017.2743016
22. Qi, S., Zou, J., Yang, S., Zheng, J.: A level-based multi-strategy learning swarm optimizer for large-scale multi-objective optimization. Swarm Evolut. Comput. **73**, 101100 (2022). https://doi.org/10.1016/j.swevo.2022.101100
23. Cheng, R., Jin, Y.: A competitive swarm optimizer for large scale optimization. IEEE Trans. Cybern. **45**(2), 191–204 (2015). https://doi.org/10.1109/TCYB.2014.2322602
24. Li, L., Li, Y., Lin, Q., Liu, S., Zhou, J., Ming, Z., Coello, C.A.C.: Neural net-enhanced competitive swarm optimizer for large-scale multiobjective optimization. IEEE Trans. Cybern. 1–14 (2023). https://doi.org/10.1109/TCYB.2023.3287596
25. Li, Y., Li, L., Tang, H., Lin, Q., Ming, Z., Leung, V.C.M.: Redefined decision variable analysis method for large-scale optimization and its application to feature selection. Swarm Evolut. Comput. **82**, 101360 (2023). https://doi.org/10.1016/j.swevo.2023.101360
26. Molina, D., Lozano, M., Herrera, F.: Ma-sw-chains: memetic algorithm based on local search chains for large scale continuous global optimization. In: IEEE Congress on Evolutionary Computation, pp. 1–8 (2010). https://doi.org/10.1109/CEC.2010.5586034
27. Baş, E., Ülker, E.: Improved social spider algorithm for large scale optimization. Artif. Intell. Rev. **54**, 1–36 (2021). https://doi.org/10.1007/s10462-020-09931-5
28. Long, W., Jiao, J., Liang, X., Tang, M.: Inspired grey wolf optimizer for solving large-scale function optimization problems. Appl. Math. Model. **60**, 112–126 (2018). https://doi.org/10.1016/j.apm.2018.03.005
29. Long, W., Cai, S., Jiao, J., Tang, M.: An efficient and robust grey wolf optimizer algorithm for large-scale numerical optimization. Soft Comput. **24**(2), 997–1026 (2020). https://doi.org/10.1007/s00500-019-03939-y
30. Xu, Y., Zhong, R., Zhang, C., Yu, J.: Multiplayer battle game-inspired optimizer for complex optimization problems. Clust. Comput. 1–25 (2024). https://doi.org/10.1007/s10586-024-04448-w
31. Wang, Z.-J., Jian, J.-R., Zhan, Z.-H., Li, Y., Kwong, S., Zhang, J.: Gene targeting differential evolution: a simple and efficient method for large-scale optimization. IEEE Trans. Evolut. Comput. **27**(4), 964–979 (2023). https://doi.org/10.1109/TEVC.2022.3185665
32. Zhong, R., Xu, Y., Zhang, C., Yu, J.: Efficient multiplayer battle game optimizer for adversarial robust neural architecture search (2024)
33. van den Bergh, F., Engelbrecht, A.P.: A cooperative approach to particle swarm optimization. IEEE Trans. Evolut. Comput. **8**(3), 225–239 (2004). https://doi.org/10.1109/TEVC.2004.826069
34. Omidvar, M.N., Li, X., Mei, Y., Yao, X.: Cooperative co-evolution with differential grouping for large scale optimization. IEEE Trans. Evolut. Comput. **18**(3), 378–393 (2014). https://doi.org/10.1109/TEVC.2013.2281543
35. Sun, Y., Kirley, M., Halgamuge, S.K.: A recursive decomposition method for large scale continuous optimization. IEEE Trans.

Evolut. Comput. **22**(5), 647–661 (2018). https://doi.org/10.1109/TEVC.2017.2778089
36. Yang, M., Zhou, A., Li, C., Yao, X.: An efficient recursive differential grouping for large-scale continuous problems. IEEE Trans. Evolut. Comput. **25**(1), 159–171 (2021). https://doi.org/10.1109/TEVC.2020.3009390
37. Yang, Z., Tang, K., Yao, X.: Differential evolution for high-dimensional function optimization. In: 2007 IEEE Congress on Evolutionary Computation, pp. 3523–3530 (2007). https://doi.org/10.1109/CEC.2007.4424929
38. Yang, Z., Tang, K., Yao, X.: Large scale evolutionary optimization using cooperative coevolution. Inform. Sci. **178**(15), 2985–2999 (2008). https://doi.org/10.1016/j.ins.2008.02.017
39. Li, X., Tang, K., Omidvar, M.N., Yang, Z., Qin, K.: Benchmark Functions for the CEC'2013 Special Session and Competition on Large-Scale Global Optimization (2013)
40. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95 - International Conference on Neural Networks, vol. 4, pp. 1942–19484 (1995). https://doi.org/10.1109/ICNN.1995.488968
41. Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**, 341–359 (1997). https://doi.org/10.1023/A:1008202821328
42. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolut. Comput. **9**(2), 159–195 (2001). https://doi.org/10.1162/106365601750190398
43. Liang, J.J., Qin, A.K., Suganthan, P.N., Baskar, S.: Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans. Evolut. Comput. **10**(3), 281–295 (2006). https://doi.org/10.1109/TEVC.2005.857610
44. Zhang, J., Sanderson, A.C.: Jade: adaptive differential evolution with optional external archive. IEEE Trans. Evolut. Comput. **13**(5), 945–958 (2009). https://doi.org/10.1109/TEVC.2009.2014613
45. Tanabe, R., Fukunaga, A.: Success-history based parameter adaptation for differential evolution. In: 2013 IEEE Congress on Evolutionary Computation, pp. 71–78 (2013). https://doi.org/10.1109/CEC.2013.6557555
46. Tanabe, R., Fukunaga, A.S.: Improving the search performance of shade using linear population size reduction. In: 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 1658–1665 (2014). https://doi.org/10.1109/CEC.2014.6900380
47. Brest, J., Maučec, M.S., Bošković, B.: il-shade: improved l-shade algorithm for single objective real-parameter optimization. In: 2016 IEEE Congress on Evolutionary Computation (CEC), pp. 1188–1195 (2016)
48. Yang, X., Li, H., Yu, X.: Adaptive heterogeneous comprehensive learning particle swarm optimization with history information and dimensional mutation. Multimed. Tools Appl. **82**(7), 9785–9817 (2022). https://doi.org/10.1007/s11042-022-13044-2
49. Li, X., Yao, X.: Cooperatively coevolving particle swarms for large scale optimization. IEEE Trans. Evolut. Comput. **16**(2), 210–224 (2012). https://doi.org/10.1109/TEVC.2011.2112662
50. Omidvar, M.N., Yang, M., Mei, Y., Li, X., Yao, X.: Dg2: a faster and more accurate differential grouping for large-scale black-box optimization. IEEE Trans. Evolut. Comput. **21**(6), 929–942 (2017). https://doi.org/10.1109/TEVC.2017.2694221
51. Ma, X., Huang, Z., Li, X., Wang, L., Qi, Y., Zhu, Z.: Merged differential grouping for large-scale global optimization. IEEE Trans. Evolut. Comput. **26**(6), 1439–1451 (2022). https://doi.org/10.1109/TEVC.2022.3144684
52. Yildiz, Y.E., Topal, A.O.: Large scale continuous global optimization based on micro differential evolution with local directional search. Inform. Sci. **477**, 533–544 (2019). https://doi.org/10.1016/j.ins.2018.10.046
53. Zhao, S.Z., Liang, J.J., Suganthan, P.N., Tasgetiren, M.F.: Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization. In: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), pp. 3845–3852 (2008). https://doi.org/10.1109/CEC.2008.4631320
54. Meng, Z., Yang, C.: Hip-de: historical population based mutation strategy in differential evolution with parameter adaptive mechanism. Inform. Sci. **562**, 44–77 (2021). https://doi.org/10.1016/j.ins.2021.01.031
55. Chakraborty, S., Saha, A.K., Chakraborty, R., Saha, M.: An enhanced whale optimization algorithm for large scale optimization problems. Knowl. Based Syst. **233**, 107543 (2021). https://doi.org/10.1016/j.knosys.2021.107543
56. Li, Y., Zhao, Y., Liu, J.: Dynamic sine cosine algorithm for large-scale global optimization problems. Expert Syst. Appl. **177**, 114950 (2021). https://doi.org/10.1016/j.eswa.2021.114950
57. Xia, X., Gui, L., Zhang, Y., Xu, X., Yu, F., Wu, H., Wei, B., He, G., Li, Y., Li, K.: A fitness-based adaptive differential evolution algorithm. Inform. Sci. **549**, 116–141 (2021). https://doi.org/10.1016/j.ins.2020.11.015
58. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evolut. Comput. **1**(1), 67–82 (1997). https://doi.org/10.1109/4235.585893
59. Luo, F., Lin, X., Liu, F., Zhang, Q., Wang, Z.: Neural combinatorial optimization with heavy decoder: Toward large scale generalization. In: Oh, A., Neumann, T., Globerson, A., Saenko, K., Hardt, M., Levine, S. (eds.) Advances in Neural Information Processing Systems, vol. 36, pp. 8845–8864. Curran Associates, Inc. (2023)
60. Chen, M., Tan, Y.: Sf-fwa: a self-adaptive fast fireworks algorithm for effective large-scale optimization. Swarm Evolut. Comput. **80**, 101314 (2023). https://doi.org/10.1016/j.swevo.2023.101314
61. Ahadzadeh, B., Abdar, M., Safara, F., Khosravi, A., Menhaj, M.B., Suganthan, P.N.: Sfe: a simple, fast, and efficient feature selection algorithm for high-dimensional data. IEEE Trans. Evolut. Comput. **27**(6), 1896–1911 (2023). https://doi.org/10.1109/TEVC.2023.3238420

Rui Zhong received a Bachelor degree from Huazhong Agricultural University, China in 2019, and a Master degree from Kyushu University, Japan in 2022. He is currently pursuing a Ph.D. at Hokkaido University, Japan. His research interests include evolutionary computation, large-scale global optimization, and meta/hyperheuristics.



Jun Yu received a Bachelor degree from Northeastern University, China in 2014, and a Master degree and a doctorate from Kyushu University, Japan in 2017 and 2019, respectively. He is currently an Assistant Professor at Niigata University, Japan. His research interests include evolutionary computation, artificial neural networks, and machine learning.