Cluster Computing
https://doi.org/10.1007/s10586-024-04448-w

Multiplayer battle game-inspired optimizer for complex optimization problems

Yuefeng Xu[1] · Rui Zhong[2] · Chao Zhang[3] · Jun Yu[4]

Received: 29 January 2024 / Revised: 11 March 2024 / Accepted: 19 March 2024

Abstract
Various popular multiplayer battle royale games share a lot of common elements. Drawing from our observations, we summarized these shared characteristics and subsequently proposed a novel heuristic algorithm named multiplayer battle game-inspired optimizer (MBGO). The proposed MBGO streamlines mainstream multiplayer battle royale games into two discrete phases: movement and battle. Specifically, the movement phase incorporates the principles of commonly encountered "safe zones" to incentivize participants to relocate to areas with a higher survival potential. The battle phase simulates a range of strategies players adopt in various situations to enhance the diversity of the population. To evaluate and analyze the performance of the proposed MBGO, we executed it alongside ten other algorithms, including three classics and five latest ones, across multiple diverse dimensions within the CEC2017 and CEC2020 benchmark functions. In addition, we employed several industrial design problems to evaluate the scalability and practicality of the proposed MBGO. The statistical analysis results reveal that the novel MBGO demonstrates significant competitiveness, excelling in convergence speed and achieving high levels of convergence accuracy across both benchmark functions and real-world problems.

Keywords Evolutionary computation · Meta-heuristic algorithm · Multiplayer battle game-inspired optimizer · Complex optimization

1 Introduction

Optimization problems are a frequently encountered but highly important class of computer science and math topics [1], aiming to find a feasible global optimum within acceptable costs under given constraints. Thanks to the continuous dedication and improvement of practitioners, some classic optimization algorithms, including Newton's method [2], linear programming [3], and hill-climbing methods [4], have been successfully applied to various industrial scenarios. However, as the problem's dimensionality increases and the complexity of mathematical modeling intensifies, these aforementioned algorithms also encounter challenges in addressing increasingly complex problems. These dilemmas force practitioners to find ways to improve the performance of optimization algorithms [5].

Evolutionary computation (EC), as a novel optimization approach, has rapidly garnered interest thanks to its outstanding characteristics, such as problem independence, high robustness, and parallel processing capabilities [6, 7]. As a pioneer member of the EC algorithms, genetic algorithm (GA) incrementally enhances the precision of candidate solutions by emulating the principle of "survival of the fittest" observed in biological evolution, ultimately converging towards the global optimum [8, 9]. This concept, borrowing population-based iterative optimization

✉ Jun Yu
yujun@ie.niigata-u.ac.jp

Yuefeng Xu
xyf20070623@gmail.com

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

Chao Zhang
zhang@u-fukui.ac.jp

[1] School of Engineering, University of Fukui, Fukui, Japan

[2] Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Japan

[3] Department of Engineering, University of Fukui, Fukui, Japan

[4] Institute of Science and Technology, Niigata University, Niigata, Japan

Published online: 10 April 2024

ideas, has provided fresh inspiration to many practitioners, catalyzing substantial growth in the field of EC since the 1990 s [10]. Up to the present, EC algorithms have been successfully solving various optimization problems featuring distinct characteristics, including multi-modal optimization [11, 12], multi-objective optimization [13, 14], dynamic optimization [15, 16], and others [17–19].

The widespread adoption of EC algorithms has led to the emergence and dissemination of numerous algorithms beyond the classic GA. For example, particle swarm optimization (PSO) treats each candidate solution as an individual bird and updates candidate solutions by simulating the foraging behavior of a flock of birds [20]. Differential evolution (DE) leverages differential information between individuals to generate new candidate solutions efficiently and converge rapidly toward the global optimum [21]. Compared with the above two algorithms, the whale optimization algorithm (WOA) primarily emulates the hunting behavior of humpback whales and does not necessitate additional parameter settings [22]. In addition to these well-established algorithms, many interesting new algorithms are developed every year, such as honey badger algorithm (HBA) [23], tunicate swarm algorithm (TSA) [24], the sailfish optimization algorithm (SFO) [25], seagull optimization algorithm (SOA) [26], and Aquila optimizer (AO) [27]. After decades of development, the EC community has seen the inclusion of hundreds of novel algorithms. While the motivations behind these algorithms may vary, they can still be broadly categorized into six major categories: bio-inspired, plant-based, swarm-based, human-inspired, physics-based, and math-based.

There is also a large proportion of researchers who prefer to optimize and improve existing algorithms rather than proposing new ones, e.g., the well-known algorithm WOA mentioned earlier has a total of more than 100 improved and hybrid new algorithms [28]. The well-known crow search algorithm (CSA) [29] also has a large number of improved proposals like an evolutionary crow search algorithm equipped with an interactive memory mechanism (ECSA) [30]. In addition, many researchers prefer to improve new algorithms that have been proposed recently. For example, an improved new algorithm called binary starling murmuration optimizer (BSMO) [31] has been proposed and works well in the feature selection problem by binary mapping of starling murmuration optimizer (SMO) [32]. To solve the problem of premature convergence that exists in Moth-flame optimization (MFO) [33, 34], the researchers proposed an enhanced Moth-flame Optimization Algorithm called MFO-SFR [35] to solve the global optimization problem by combining the stagnation finding and replacing (SFR) strategy. By combining the Hybrid Binary Operator (HBO) and the Distance-based Binary Search Strategy (DBSS), the researchers improved the quantum-based avian navigation optimizer algorithm (QANA) [36] and proposed an improved binary quantum-based avian navigation optimizer algorithm (IBQANA) [37]. In addition to these mentioned algorithms, there are also many other EC algorithms that are continuously improved. For example, in the scenario of solving engineering problems, an improved grey wolf optimizer (I-GWO) [38], gaze cues learning-based grey wolf optimizer (GGWO) [39], an effective multi-trial vector-based differential evolution algorithm (MTDE) [40] have recently received more attention. Besides, Multi-trial vector-based monkey king evolution algorithm (MMKE) [41] have also achieved good results. Binary Aquila optimizer (BAO) [42] is used in covid-19 case study. Discrete improved grey wolf optimizer (DI-GWOCD) [43] is used for community detection.

However, many proposed new and improved algorithms generally do not control the balance between exploration and exploitation well. Meanwhile, many algorithms focus on pre-exploitation and post-exploration, which cannot fully utilize the algorithm's performance. This also means that many algorithms cannot perform well when facing complex problems that require better performance. To solve this problem, we design an algorithm that solves this problem by alternating two-phase updates with different focuses. This dynamic updating mechanism allows the algorithm to balance exploration and exploitation dynamically. Also, it allows the algorithm to have high convergence and population diversity by alternating between two different searches when facing a complex problem, thus obtaining higher performance.

To the best of our knowledge, only a limited number of researchers draw inspiration from games and propose a handful of algorithms, even though games have been an integral part of human life since ancient times. In ancient civilizations, different cultures developed their games. For example, ancient Egyptians played games like Senet, a board game with religious significance. Ancient Greeks had various athletic and board games, while the Roman Empire had its games, including dice games and chariot races. Especially with the emergence of e-sports games, gaming allows individuals to relish the excitement of competition and kindles an interest in communication. Thus, we are not constrained to a particular game but aim to distill the common patterns shared by these games that captivate individuals and, subsequently, design a new optimization algorithm called multiplayer battle game-inspired optimizer (MBGO). The proposed MBGO refines the competitive process of battle games into two distinct phases, labeled the movement and battle phases. Through the repeated execution of these two stages, a balance between exploration and exploitation is achieved, resulting in the proposed MBGO displaying remarkable performance. Here, we summarize the contributions of the new MBGO algorithm as follows:

- A new game-inspired optimization algorithm is crafted to balance exploration and exploitation from a novel perspective. This design enables the new MBGO

algorithm to achieve swift convergence while maintaining a high level of population diversity.
- The movement phase uses the distribution information of the population to partition the entire search space into two different areas. Individuals in these separate areas employ different strategies to update their positions to converge toward potential areas quickly.
- The battle phase uses individual fitness information instead of Euclidean distance to simulate the battle strategies adopted by individuals under various psychological conditions when encountering opponents, which is focused on maintaining the diversity of the population.
- A series of analytical experiments are conducted to investigate the performance of the new MBGO. The experimental results demonstrate that the proposed MBGO exhibits robust competitiveness and significant potential, consistently ranking among the top performers in test function sets and real-world industrial problems.

The structure of the remaining sections is outlined as follows. Section 2 summarizes related studies and highlights the distinctions between our work and them. Section 3 provides the motivation and presents a detailed description of the mathematical model of the proposed MBGO. Many comparative experiments are conducted to evaluate the performance of the proposed MBGO in Sect. 4. We further analyze the principles of effectiveness and scalability of the proposed MBGO and conclude our work in Sects. 5 and 6.

2 Related work

We conducted a comprehensive search across multiple prominent databases. We found the fact that there is a clear lack of meta-heuristics algorithms rooted in the principles and mechanics of games. This discrepancy highlights a unique domain teeming with innovative potential and unexplored opportunities. Specifically, the utilization of game-inspired concepts holds the potential to yield pioneering and highly effective optimization techniques. Given the trend of interdisciplinary cross-pollination [44], it is foreseeable that an increasing number of researchers and developers will be drawn to the intriguing intersection of games and optimization [45]. Thus, it is time to broaden our perspectives, directing our efforts not only toward the creation of innovative and captivating games but also toward the exploration of novel avenues for the application of game-derived inspiration in diverse fields. This is also one of the driving forces that prompted the emergence of this work. In the subsequent sections, we will introduce two papers that fall within the same category as this one and delineate the distinctions between them.

As the first game-inspired EC algorithm, the battle royale optimization algorithm (BRO) [46] mainly draws inspiration from the "battle royale" genre within digital gaming. It gradually emulates the player's survival strategies to converge toward the optimal area. Since this type of game includes renowned titles such as Player Unknown's Battlegrounds (PUBG) [47] and Apex Legends, the foundational BRO selects the "deathmatch" of PUBG, simulating the battle behaviors of multiple players to address single-objective optimization problems. Shortly after introducing the basic BRO, several enhanced versions were also proposed. For example, Akan et al. introduced an additional movement operator to improve the search efficiency further [48], and they extended the MBGO algorithm to tackle discrete optimization problems [49]. The advent of these variants further underscores the potential for developing EC algorithms rooted in the gaming world.

The other related study within the game-inspired category, the squid game optimizer (SGO) [50], was recently introduced. The SGO takes inspiration from the fundamental rules of a traditional Korean game to design optimization operators. In this game, multiple players are divided into two distinct teams. One team serves as the offensive players, while the other takes on the role of defensive players, all within a playing field shaped like a squid. The primary objectives within this game involve the attackers completing their objectives or the teams eliminating each other. Drawing from these game principles, SGO employs a unique approach in which individuals are categorized into two groups, each with distinct evolutionary strategies. This innovative concept aligns with the game's dynamics and offers a fresh perspective for optimization algorithms. Since the SGO was proposed not long ago, there are not many improvements and applications yet.

These similar studies persist in their focus on a specific game, devising new optimization frameworks through the simplification of gameplay [51]. Achieving a lossless representation of a game is highly challenging. Thus, selecting the operations that need to be retained during the simplification process is crucial, which often requires a robust background in optimization design. Because of the above considerations, we sought to explore a distinctive pathway for algorithmic design. Consequently, instead of fixing a specific game, we distill the common features inherent in battle games to construct a brand-new optimization framework.

Furthermore, our approach extends beyond the game mechanics by incorporating player behavior [52], including the mentalities and strategies employed when facing different enemies, into the new algorithm for the first time. This integration aims to simulate a dynamic interaction between the game and multiple players, under the belief that a game becomes truly interesting and charming only when players actively engage in it. Finally, we present a new MBGO algorithm based on simulated battle games' characteristics and player behaviors.

3 Multiplayer battle game-inspired optimizer

3.1 Motivation

Battle royale games [53, 54], such as PUBG, Infinite Law, Warzone, and Call of Duty, typically integrate elements from both survival games, focusing on exploration and equipment collection, and competitive gaming, where the objective is to eliminate opponents until only one person remains. While the rules of these games may not be identical, there are discernible common patterns that characterize their gameplay. For example, a prevalent game mechanic known as "safe zones" is widely employed to compel players into more confined areas, fostering encounters and battles among them. Over time, the designated safe area progressively diminishes, and players outside this zone risk injury or elimination until only the last individual or team remains the victor. We acknowledge the crucial role played by the "safe zone" mechanic in these games, significantly shaping player movement and battle strategies. Recognizing this mechanic's widespread influence and impact, we have intentionally directed our simulation efforts toward generating and utilizing the safe zone.

Another factor that affects the outcome of a match is the players themselves, as different players tend to choose different strategies when faced with even the same situation. For example, cautious players typically engage in battle when they are more likely to succeed and avoid confrontation when the odds are less favorable. On the other hand, aggressive players consistently lean towards engaging in battle upon encountering a new opponent. Furthermore, the quality of equipment significantly influences players' decision-making. Generally, superior equipment correlates with an elevated probability of winning, instilling greater confidence in players when engaging in battle. When designing the new algorithm, we especially introduce several strategies to simulate players' behaviors under different psychological conditions to reflect the importance of players. Our intention is for these strategies to preserve individual diversity effectively.

Since our primary objective is the development of an effective algorithm rather than blindly pursuing simulation games, we are open to making trade-offs on the rules. Here, we consider each player as a candidate solution within the search space and map their actions to coordinate updates in the solution space. As players are gradually eliminated, the number of individuals in the game decreases. However, we have not adopted this rule to maintain a constant population size. Furthermore, we overlook players' equipment differences and treat each individual equally. This implies adopting different movement and battle strategies with equal probability for all individuals. In the subsequent subsection, we comprehensively overview the proposed MBGO algorithm.

3.2 Mathematical model

The newly proposed MBGO algorithm is a population-based heuristic algorithm that iteratively refines the population by simulating common mechanisms found in battle games and behaviors taken by players under various psychological states. Like typical EC algorithms, the MBGO randomly generates multiple individuals to form an initial population. Then, the Euclidean distance between the best and worst individuals in the current population defines the "safe zone", effectively partitioning the entire search space into safe and non-safe areas. Individuals in different areas use different movement strategies to generate new individuals. Importantly, individuals choose to update their current positions only if the new individual represents an improvement; otherwise, the current individual remains unchanged. After all individuals have gone through this movement phase, they transition into the battle phase. During this stage, each individual randomly chooses another individual as an opponent and determines the battle strategy to generate a new individual. The proposed MBGO still maintains elite selection in the battle phase, accepting the new individual only if it is superior; otherwise, the individual "loses" the battle and is reborn in the same place. Subsequently, individuals re-enter the movement and battle phases again, and this cycle continues iteratively until the termination condition is met. Finally, Fig. 1 provides a simple optimization process of the proposed MBGO algorithm: initialization, movement phase, and battle phase.

Before delving into the detailed introduction of three core operators, we first define some symbols that will be used in the following to avoid ambiguity. Here, we take the minimization problem as an example, assuming that the optimization problem contains D variables, that is, a D-dimensional problem. Thus, for any individual $\mathbf{x}_i$ (i.e., candidate solution), it comprises D variables, represented as $\mathbf{x}_i = (x_i^1, x_i^2, ..., x_i^D)$. For any dimension $k(k \in \{1, 2, ..., D\})$, its upper bound and lower bound are defined as UB^k and LB^k respectively. Additionally, $\mathbf{x}_{best}$ and $\mathbf{x}_{worst}$ represent the best individual and the worst individual in the current population, and $Fit()$ is the evaluation function and returns the fitness value of the passed-in individual.

Initialization is executed only once to generate the initial population. We employ the most common approach among various initialization methods: random initialization. For any individual $\mathbf{x}_i$, Eq. (1) uses the uniform distribution to generate the initial value in the k-th dimension:

$$x_i^k = LB_i^k + rand(0, 1) \times (UB_i^k - LB_i^k) \tag{1}$$

where $rand(0, 1)$ satisfies the uniform distribution and returns a random number between 0 and 1.

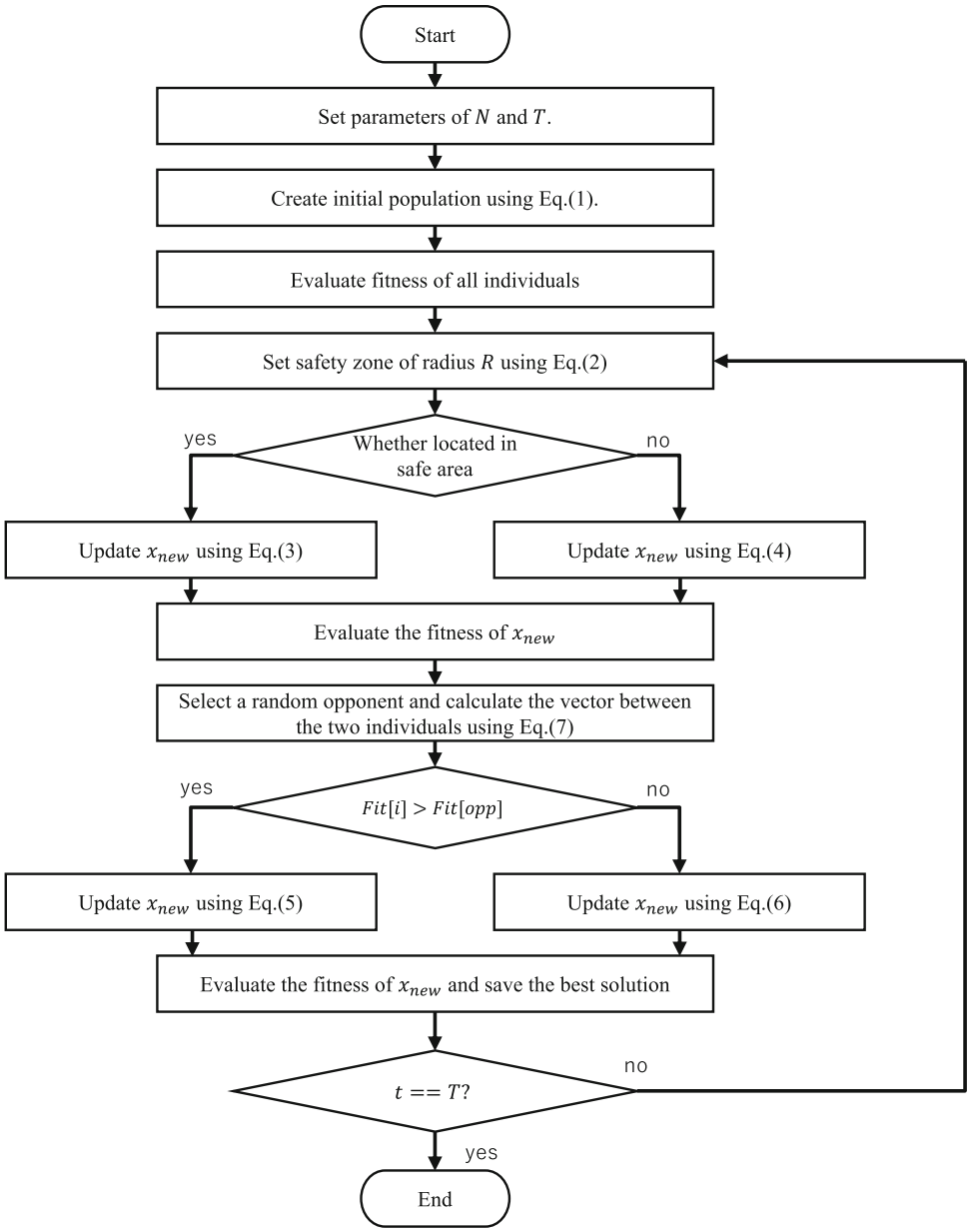

Fig. 1 The universal optimization framework of proposed MBGO algorithm

Suppose the population size, a hyperparameter that needs to be defined in advance, is set to N. The initial population can be represented in the form of a matrix as follows:

$$\text{The initial population} = \begin{bmatrix} x_1^1 & x_1^2 & x_1^3 & \cdots & x_1^D \\ x_2^1 & x_2^2 & x_2^3 & \cdots & x_2^D \\ \vdots & \vdots & \vdots & \vdots & \\ x_N^1 & x_N^2 & x_N^3 & \cdots & x_N^D \end{bmatrix}$$

where each row represents an individual (i.e., candidate solution), and each column represents a dimension (i.e., variable).

Movement Phase is mainly responsible for the exploitation capability and uses the "safe zone" concept to guide individuals to converge toward potential areas. Fig. 2 illustrates this mechanism used in two real battle royale games. Although the rules in each game may differ, the general idea is to select a part of the area as a safe area and continuously shrink the subsequent safe prefetching to encourage players to gather together as the game progresses. To simulate this mechanism, a crucial consideration is determining the boundaries of the safe area.

As an attempt, we leverage the current optimal individual, $\mathbf{x}_{best}$, and designate it as the center of the safe area. Regarding determining the radius covered by the safe area,

(a) The safe zone adopted by PUBG (b) The safe zone adopted by APEX

Fig. 2 Screenshots from two games, where the entire map is divided into two areas. (https://apexlegends-news.com/apex-aruaru-5/)

we use the Euclidean distance between the best and worst individuals in the current population as the baseline. Notably, the baseline distance may not necessarily equal the distance between the two furthest individuals in the current population. This baseline distance is multiplied by a random factor between 0.8 and 1.2 to increase randomness, and the resulting value is utilized as the final safety radius, as shown in Eq. (2). Although the random factor does not change the radius of the region very much, it is clear from the formulae for the area of a circle and the volume of a sphere that such a small change is a large change in the extent of the entire space. Thus, the entire search space dynamically divides into two distinct areas as the population converges:

$$R = (|\mathbf{x}_{best} - \mathbf{x}_{worst}| + eps) \times rand(0.8, 1.2) \tag{2}$$

where R is the safety radius, and eps is a tiny non-negative number to ensure a non-zero radius for the safe area.

For individuals within the safe area, where the distance from the current best individual is less than the safety radius, it suggests that the individual is likely situated in a potential area. Here, we introduce Eq. (3), which uses the current optimal individual (considered as the center-most position of the safety zone) to bootstrap using the sin function on top of the original individual, thereby performing local exploration. This process mimics the behavior of a real game where the player will only make small movements toward the center of the safe zone when in the safe zone. However, for individuals outside the safe area, meaning they are currently distant from the potential area, Eq. (4) is employed to accelerate movement toward the current optimal area. One of the things to note is that in real games, players may be attacked by opponents inside the safe zone as they move outside the safe zone to inside the safe zone. Therefore, the player must always stay hidden while moving and observe the surrounding situation. To simulate the behavior of the player stopping to scout, we use the individual dimension values plus a random number that conforms to a normal distribution to simulate the player stopping to scout the surroundings. To simulate the player accelerating into the safe zone, we use the individual's dimension value plus a vector of differences from the most available individual. Specifically, for each dimension k, one of the two methods is selected for update with equal probability, which better simulates the alternation between the player's scouting and accelerating into the safe zone. The last point that needs to be mentioned is that the elite selection is employed to update the population. This implies that an individual will only be updated if the fitness of the new individual, $\mathbf{x}_{new}$, surpasses that of the original individual, $\mathbf{x}_i$, otherwise, the current state will be maintained:

$$\mathbf{x}_{new} = \mathbf{x}_i + \mathbf{x}_{best} \times \sin(2 \times \pi \times rand(0, 1)) \tag{3}$$

$$x_{new}^k = \begin{cases} x_i^k + normal(), & \text{if } rand(0,1) < 0.5 \\ x_i^k + (x_{best}^k - x_i^k) \times rand(0,1), & \text{otherwise} \end{cases} \tag{4}$$

where $\mathbf{x}_{new}$ and x_{new}^k respectively represent the moved individual and its updated value in the k-th dimension, and $normal()$ returns a random number that adheres to the standard normal distribution.

Battle Phase is mainly responsible for the exploration capability and simulates diverse battle behaviors that arise when players encounter each other randomly in the game. While players may employ different strategies when facing opponents of varying skill levels, common objectives include avoiding opponent attacks to minimize damage received and inflicting maximum damage on the opponent to secure victory. To simulate the diverse behaviors of players under various psychological conditions, we have simplified the observations from real games and imposed some idealized restrictions. Here, we assume a player will not simultaneously confront multiple opponents and randomly designate another individual as the opponent in each battle phase. Additionally, fitness information is employed to gauge the strength of an individual, aligning with factors such as player level.

We have designed two strategies to simulate the player's behavior as outlined in Eqs. (5) and (6) to update the population. For the sake of simplicity, we have standardized the rules for selecting battle strategies for all individuals. Specifically, if the individual encounters a stronger opponent (i.e., an opponent with higher fitness), adopt Eq. (5); otherwise, adopt Eq. (6) to generate the new individual. Similarly, the elite selection is also applied during the battle phase:

$$x_{new}^{k} = \begin{cases} x_i^k + rand(0,1) \times dir^k, & \text{if } rand(0,1) < 0.5 \\ x_{opponent}^k + rand(0,1) \times dir^k, & \text{otherwise} \end{cases} \quad (5)$$

$$\mathbf{x}_{new} = \mathbf{x}_i + \mathbf{dir} \times \cos(2 \times \pi \times rand(0,1)) \quad (6)$$

where **dir** denotes the vector between the i-th individual and the randomly selected opponent individual $\mathbf{x}_{opponent}$. The intention is to establish the starting point of the vector at the positions of two individuals with higher fitness, thereby constructing an exclusive vector to hinder rapid convergence and prevent individuals from falling into local optimality. Additionally, the calculation of vectors is as follows:

$$\mathbf{dir} = \begin{cases} x_i^j - x_{opponent}^j, & \text{if } Fit(x_i) < Fit(x_{opponent}) \\ x_{opponent}^j - x_i^j, & \text{otherwise} \end{cases} \quad (7)$$

The introduction of these core components of the proposed MBGO algorithm has ended. It should be noted that the selection operation is designed into the movement and battle phases, and they both involve generation updates. Algorithm 1 finally gives a detailed overview of the proposed MBGO.

4 Evaluation experiments

We designed a series of comparative experiments to analyze the performance of the proposed MBGO algorithm. Specifically, we selected 29 functions from the CEC2017 test suite [55] and ten functions from the CEC2020 test suite [56]. These functions exhibit diverse characteristics, including multimodal, rotation, and non-separate traits, effectively covering a spectrum of optimization scenarios. Additionally, we incorporated ten constrained engineering design problems to evaluate the scalability and potential of the proposed MBGO algorithm. As competitors, we employed ten other EC algorithms, including four classic well-established algorithms (PSO, DE, CMAES, and WOA) and six contemporary promising algorithms (HBA, SFO, TSA, SOA, BRO, and AO). To ensure the fairness of comparison, all algorithms were implemented in Python, and the execution environment was identical for all experiments. In addition, each algorithm was independently executed 30 times, whether applied to a benchmark function or an engineering problem.

Algorithm 1 The standard optimization process of the MBGO Algorithms.

Set the hyperparameters, including population size: N, maximum number of fitness evaluations: t_{max}, current number of fitness evaluations: $t = 0$.
Initialize the population according to Eq.(1) and evaluate the fitness of all individuals.
while $t \leq t_{max}$ **do**
  //Start the movement phase:
  for $i = 1$ to N **do**
    Determine the safety radius according to Eq.(2).
    if the i-individual is located in a safe area **then**
      Generate $\mathbf{x}_{new}$ using Eq.(3).
    else
      Generate $\mathbf{x}_{new}$ using Eq.(4).
    end if
    Evaluate the individual $\mathbf{x}_{new}$ and compare it with the i-individual.
    The winner survives and updates the i-individual.
  end for
  //Start the battle phase:
  for $i = 1$ to N **do**
    Randomly select an opponent, $\mathbf{x}_{opponent}$, for the i-th individual.
    if the fitness of $\mathbf{x}_{opponent}$ is better than that of $\mathbf{x}_i$ **then**
      Generate $\mathbf{x}_{new}$ using Eq.(5).
    else
      Generate $\mathbf{x}_{new}$ using Eq.(6).
    end if
    Evaluate the individual $\mathbf{x}_{new}$ and update it by comparing it with the i-individual.
    The winner survives and updates the i-individual.
  end for
end while
Output the optimal solution found.

4.1 Performance evaluation based on two CEC suites

The benchmark functions are carefully designed to compare the performance of different algorithms, with the flexibility to set different dimensions based on specific requirements. In this context, we selected widely used dimensions, specifically 10-D and 30-D, for the CEC2017 functions. Additionally, for the CEC2020 functions, we opted for higher dimensions, namely 50-D and 100-D.

The parameter configurations for the involved algorithms are as follows. The common parameters across the eleven algorithms are uniformly set as follows: the population size is set to 100, and the maximum number of fitness evaluations is set to $1000 \times D$, where D represents the dimensionality of the optimization problem. The parameters unique to each algorithm are aligned with mainstream settings, and their specific settings are summarized in Table 1.

To investigate whether there are significant differences between the proposed algorithm and other algorithms, we conducted the U-test and Holm's multiple comparisons at the termination of the algorithm, that is, at the maximum number of fitness evaluations. Additionally, we present the average and standard deviation of the optimal solution found in the final 30 trial runs for all algorithms. Tables 3 and 4 summarize the results of 10-D and 30-D on CEC2017 test suites, and Tables 5 and 6 summarize the results of 50-D, and 100-D on CEC2020 test suites, respectively. Besides, we present the average convergence curves of all algorithms in both 50-D and 100-D for all CEC2020 functions, as illustrated in Figs. 3, 4, 5 and 6.

Table 1 The unique parameter settings for each algorithm

Algorithm	Parameters
HBA	β = 6 (Default)
	C = 2 (Default)
PSO	ω decreases linearly from 0.9 to 0.4 (Default)
	C_1 = 2.05 (Default) C_2 = 2.05(Default)
DE	Weighting Factor, F = 0.8 (Default)
	Crossover Rate, C_r = 0.9 (Default)
WOA	a variable decreases linearly from 2 to 0 (Default)
	$a2$ linearly decreases from 1 to 2 (Default)
TSA	Parameter P_{min} = 1 (Default)
	Parameter P_{max} = 4 (Default)
SFO	Parameter A = 4 (Default)
	Parameter ϵ = 0.001 (Default)
SOA	Parameter A is random from 2 to 0 (Default)
	f_c = 2 (Default)
AO	$G1 = 2 \times rand$ - 1 (Default)
	$G2$ decreasing values from 2 to 0 (Default)
BRO	$threshold = 3$ (Default)
CMAES	$\sigma = 1.3$ (Default)
MBGO	α is random from 0.8 to 1.2 (Default)

4.2 Performance evaluation based on real-world problems

Not limited to human-designed benchmark functions, we employed ten real industrial design problems to evaluate the performance of the proposed MBGO algorithm. These real-world problems are frequently characterized by robust constraints, posing challenges that enhance the complexity of the search for optimal solutions. Since none of the algorithms inherently addresses constraints, a unified penalty function is employed in this context, which imposes an infinite penalty on the fitness of individuals that violate constraints, reflecting the most unfavorable outcomes. The details of the ten engineering design problems are outlined in Table 2. Please refer to the Appendix for specific mathematical formulas.

The parameter configuration for all algorithms remains identical to the previous experiment, except for the maximum number of fitness evaluations. Since the dimensions of these real-world problems vary, we standardized the maximum number of fitness evaluations for all problems to 10,000. To analyze the performance between algorithms, we again employ the U-test and Holm's multiple comparisons to determine whether there is a significant difference at the end of all algorithms. Table 7 summarizes the average and standard deviation of optimal solutions discovered by these algorithms, along with the outcomes of statistical testing.

5 Discussion

5.1 Computational complexity of the proposed MBGO

For the sake of simplicity, we continue to use the symbols introduced in Sect. 3 and define the maximum number of fitness evaluations as T. Therefore, the complexity of the core operations of the MBGO algorithm is as follows:

- Initialization: $O(N \times D)$.
- The selection for the current best individual: $O(N)$.
- The selection for the current worst individual: $O(N)$.
- The movement phase: $O(N \times D)$.
- The battle phase: $O(N \times D)$.

In summary, the computational complexity of the MBGO algorithm can be calculated by Eq. (8).

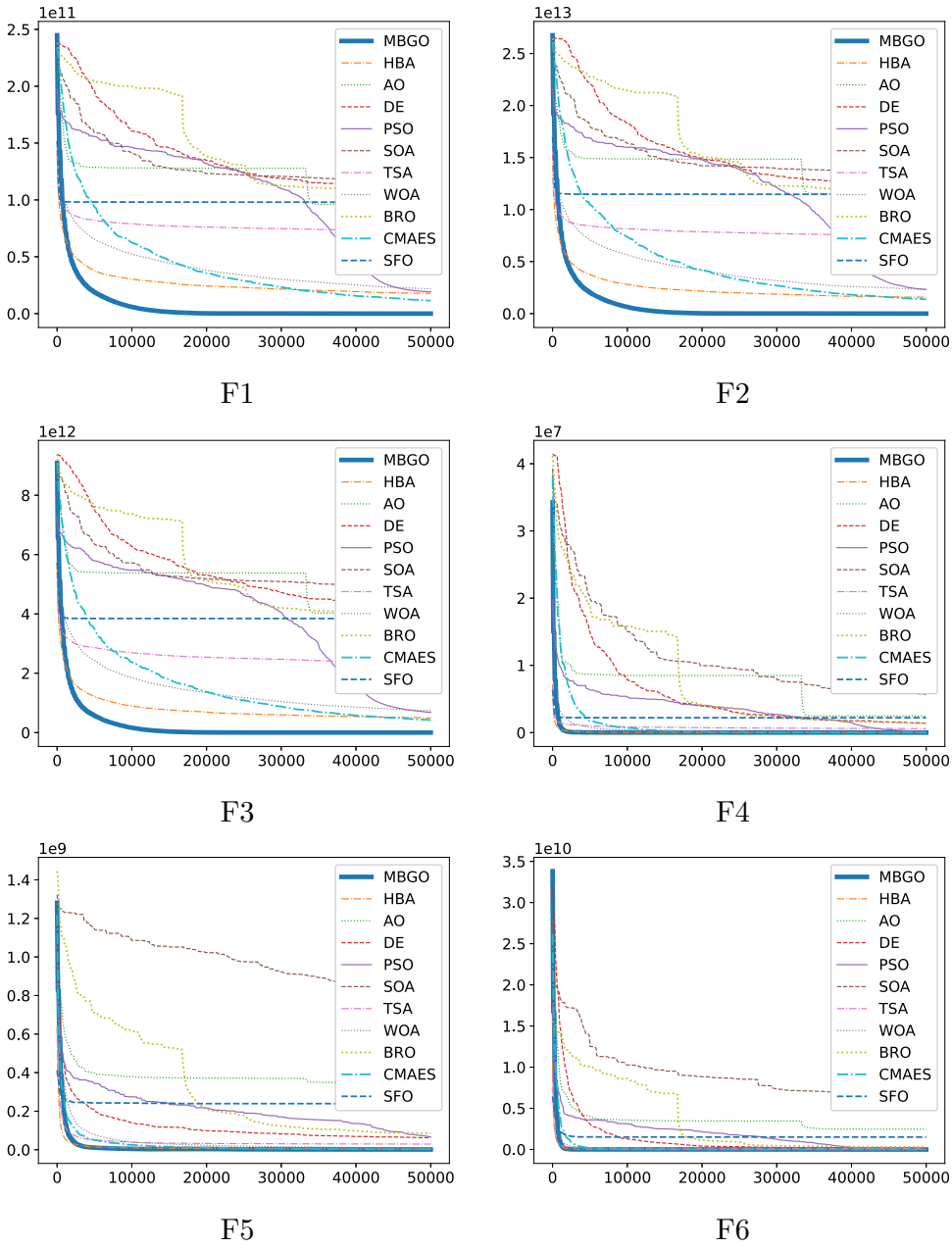

Fig. 3 The average convergence curves of all competitor algorithms on 50-D CEC2020 functions (F1-F6). The horizontal coordinate represents the number of evaluations, and the vertical coordinate represents the fitness value

$$\begin{aligned} &O(N \times D + T \times (N + N + N \times D + N \times D)) \\ =&O(N \times D + T \times N(2 + D)) \\ =&O(N \times D + T \times N \times D)) \\ =&O(T \times N \times D) \end{aligned} \tag{8}$$

5.2 Benefits from the core components

The proposed MBGO primarily consists of three operations: initialization, the movement phase, and the battle phase. Excluding the initialization operation applied in all comparison algorithms, the remaining two operations significantly improve performance. This fundamental distinction is the key reason for the proposed MBGO algorithm's superior performance compared to others. Here, we conduct an in-depth analysis of the two operations and enumerate their benefits.

The movement phase uses the Euclidean distance between the best and worst individuals to divide the entire search space into potential (safe) and non-potential (unsafe) areas. Individuals in different areas employ varied strategies to update their current positions, effectively reducing the risk of rapid convergence that may confine the population to local areas, thereby preventing premature maturation. Moreover, as the population converges,

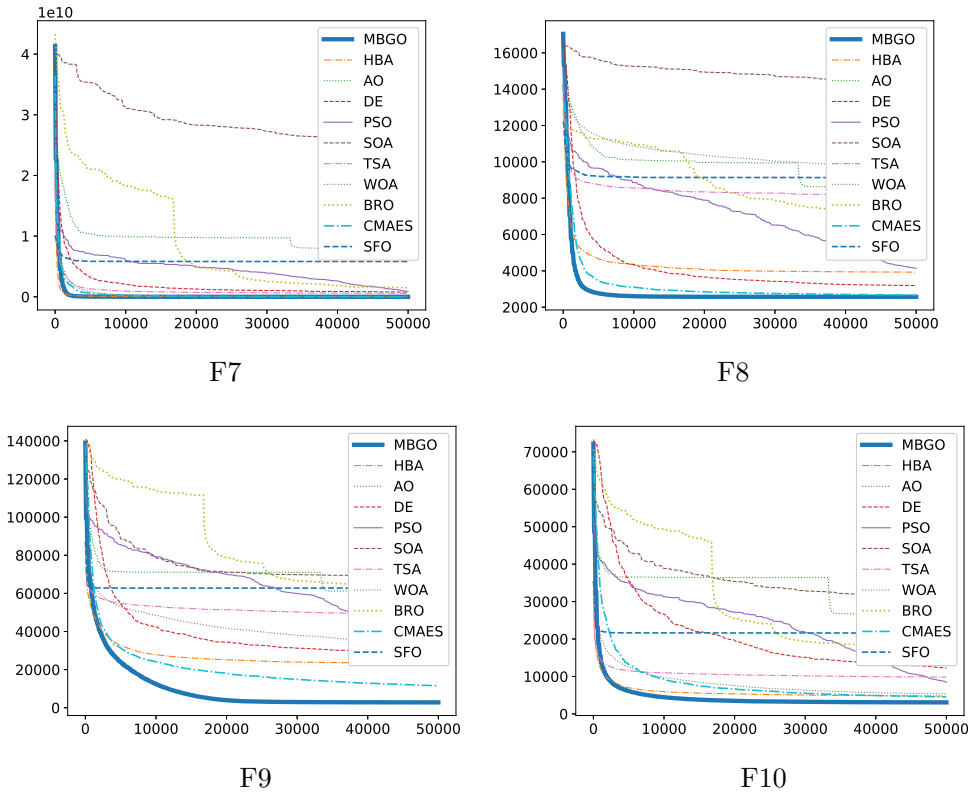

Fig. 4 The average convergence curves of all competitor algorithms on 50-D CEC2020 functions (F7-F10). The horizontal coordinate represents the number of evaluations, and the vertical coordinate represents the fitness value

individuals gradually converge toward the global area, gradually reducing the distance between the best individuals and the worst individuals. This effectively simulates the widespread adoption of "safe zones" in the game, which gradually shrinks to a smaller area. Furthermore, the optimal individual of each generation is designated as the center of the safe area, offering effective guidance to all other individuals to reduce random searches and accelerate overall processes. Thus, we can say that the movement phase guarantees the convergence speed of the proposed MBGO algorithm.

The battle phase simulates various player behaviors in response to opponents, aiming to increase the diversity of the population. Each individual randomly selects an opponent and employs different update strategies based on the opponent's fitness, fostering information exchange among individuals. Furthermore, the adversaries individuals encounter in different generations undergo constant changes, fundamentally heightening the diversity of the population and facilitating the generation of diverse offspring individuals. Additionally, we intentionally use vector information from better individuals to worse individuals to encourage the exploration of unknown potential areas and prevent premature convergence. The concept of simulated annealing inspires this approach. However, it is essential to clarify that pursuing diversity does not imply the unconditional acceptance of inferior individuals. In this context, the elite selection is implemented to ensure that population diversity is maintained without weakening the population.

Excessive emphasis on either convergence speed or diversity can pose challenges for the population to converge to the global optimum. Striking the right balance between these factors is crucial for effectively optimizing the algorithm. The collaborative synergy between the two stages enables the proposed MBGO algorithm to achieve a harmonious balance between exploration and exploitation, distinguishing itself prominently among all competing algorithms when facing various optimization problems. In addition, the rational use of individual distribution and fitness landscape information also helps the MBGO algorithm select appropriate search strategies, ensuring sustained high search performance.

5.3 Analysis of exploration and exploitation

Our proposed algorithm MBGO is divided into two phases: "movement phase" and "battle phase". In the "movement phase", the best individuals of the current generation are used to guide the individuals to update. Therefore, the "movement phase" is more focused on exploitation, and in the "battle phase", we use random individuals to engage in battles so that individuals can exchange dimensional information with each other. This allows individuals to "mutate", thus increasing the diversity of the population. Therefore, the "battle phase" focuses more on exploration,

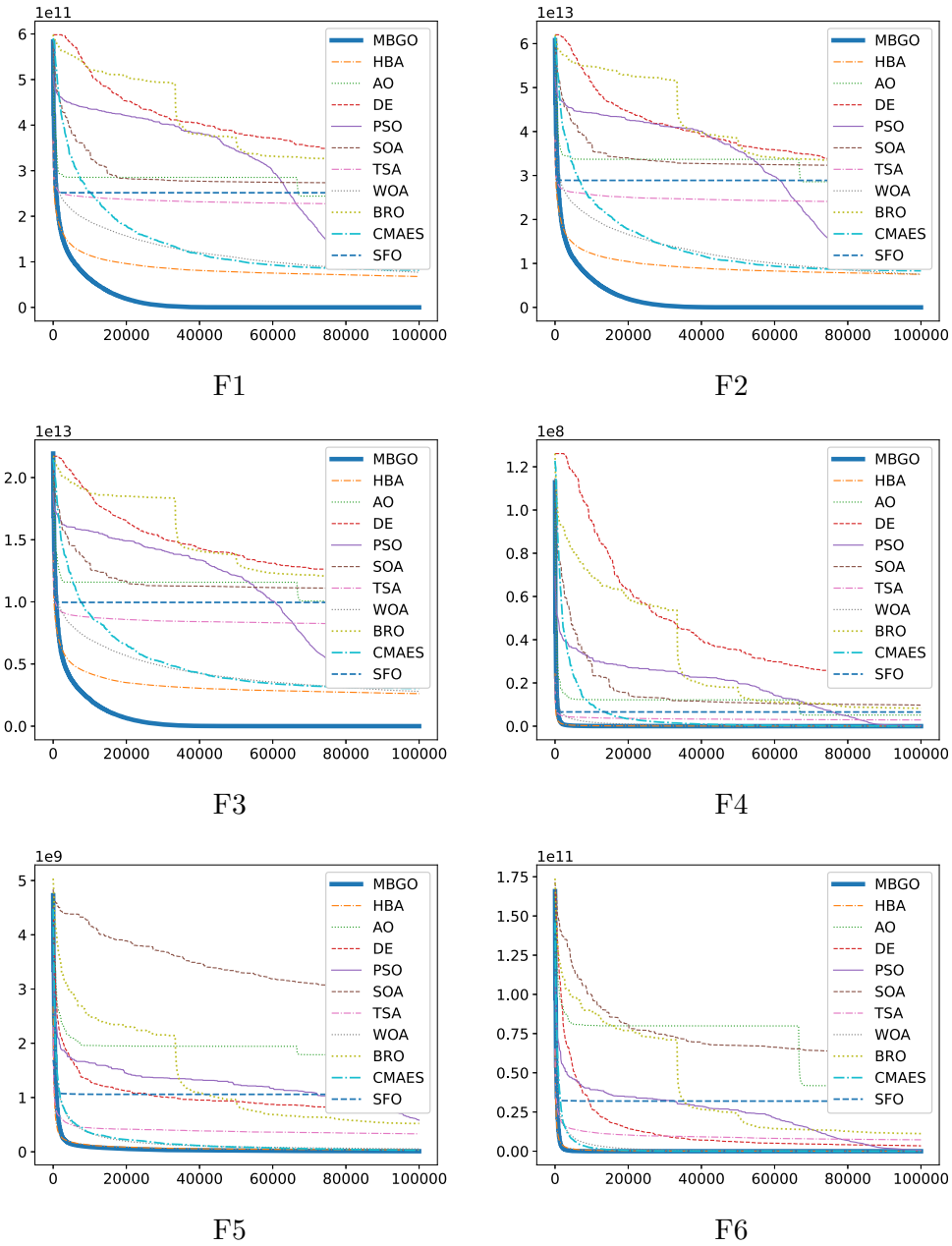

Fig. 5 the average convergence curves of all competitor algorithms on 100-D CEC2020 functions (F1-F6). The horizontal coordinate represents the number of evaluations, and the vertical coordinate represents the fitness value

and by alternating between these two phases, the algorithm can dynamically balance exploration and exploitation, which allows the algorithm to face difficult optimization problems with higher dimensionality. This allows the algorithm to perform well when facing difficult optimization problems with higher dimensions.

5.4 Analysis of experimental results

Based on the data presented in Tables 3 and 4, our algorithm demonstrates superior performance compared to the comparison algorithms across the majority of test functions within the CEC2017 dataset, particularly in low and medium dimensions. Specifically, our algorithm outperforms PSO, AO, SOA, SFO, TSA, BRO, and CMAES on more than 25 functions at 10-D and 30-D. Furthermore, our algorithm performs better on more than half of the functions compared to DE, WOA, and HBA. Moreover, as indicated in Tables 5 and 6, our algorithm maintains its superiority even as the dimensionality increases to 50D and beyond 100D within the CEC2020 dataset. Notably, our algorithm outperforms all other algorithms across all functions in CEC2020. Additionally, visual analysis of Figs. 3, 4, 5, and 6 reveals that our algorithm exhibits rapid convergence in the early stages and maintains this convergence throughout, while other algorithms often become

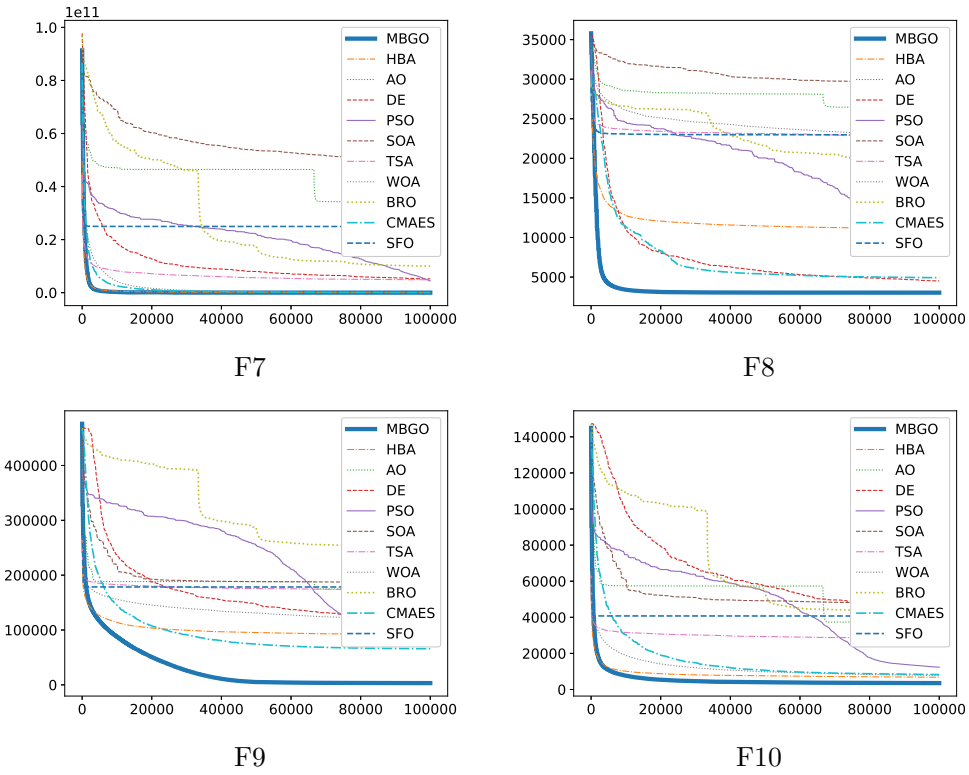

Fig. 6 The average convergence curves of all competitor algorithms on 100-D CEC2020 functions (F7-F10). The horizontal coordinate represents the number of evaluations, and the vertical coordinate represents the fitness value

Table 2 The unique parameter settings for each algorithm

Pro.	Description
WBP	Design cost minimization optimization problem for welded beams
PVP	The problem of minimizing the total cost of a cylindrical vessel capped by a hemispherical head
TBTD	Minimize three-bar structure weight subject to supporting a total load Pacting vertically downwards
GTD	Unconstrained discrete design optimization problem
CBD	Minimize a cantilever beam's weight
IBD	Minimizes the vertical deflection of a beam
TCD	Optimize the column's cost regarding resisting torsion, bending and shear.
PLD	Piston rod cost optimization problem
CBHD	Corrugated Groove Head Cost Optimization Problem
RCB	Cost minimization problem for reinforced concrete beams

trapped in local optima due to inferior diversity. Furthermore, according to Table 2, our algorithm consistently outperforms comparison algorithms in most engineering problems, especially those with constraints, demonstrating its effectiveness in practical applications.

The results of statistical tests demonstrate that the proposed MBGO algorithm consistently outperforms others, showcasing superior performance across manually designed functions and real engineering problems. In the case of the CEC2017 test suite, the MBGO algorithm demonstrates superiority over all but HBA in low-dimensional problems. However, the MBGO algorithm is excellent when considering the overall performance. Notably, it exhibits a growing advantage as dimensionality increases, leading to achieving the best performance on almost all problems. Moreover, in the evaluation across all functions of CEC2020, the MBGO algorithm overwhelmingly outperforms all comparison algorithms. The convergence curves depicted in Figs. 3, 4, 5, 6 further show that the proposed MBGO algorithm achieves high convergence accuracy and exhibits the characteristic of rapid convergence speed. The analysis and summary of all function characteristics show that the MBGO algorithm performs better on complex problems with high dimensions and multiple peaks.

Despite not meticulously designing the constraint processing module and opting for the widely used penalty

function, experimental results indicate that the proposed MBGO algorithm consistently discovers satisfactory feasible solutions compared to other classic EC algorithms. These strong constraints result in a substantial number of infeasible solutions and complicate the effectiveness of the search process. Despite the challenges posed by strong constraints, the MBGO algorithm remains competitive and ranks among the best. This observation underscores the potential applicability of the MBGO algorithm to constrained optimization problems.

5.5 Potential analysis and open topics

While we have highlighted the strengths of the MBGO algorithm, there is undeniable considerable room for improvement. In this context, we present some open topics to spark new inspiration for those who follow in our footsteps.

1. The interactive switching between the two stages is a key factor in the MBGO algorithm's success. Exploring how to make informed and context-specific switches based on the characteristics of the optimization problem is a worthwhile research topic. Instead of treating the two stages equally, tailoring the switching mechanism to the specific nuances of the problem could enhance the MBGO algorithm's adaptability and performance.
2. While elite selection ensures that the population converges toward the optimal area, it introduces a new challenge: directly discarding inferior offspring individuals without any utilization. This operation can lead to a waste of evaluation resources, particularly for expensive problems. Thus, a valuable research topic involves reusing these discarded individuals, aiming to enhance search performance further.
3. Real games often feature intricate and engaging gameplay mechanics. We simplify these complex mechanisms to design optimized game mechanics in this context. Thus, introducing these diverse strategies into subsequent improved versions could be an interesting research topic. Additionally, modeling the behavior of game players, aiming to replicate battle behavior more realistically, is a subject worthy of further study.
4. The hyperparameters of optimization algorithms frequently influence performance significantly. Here, we simplify the process by unifying and fixing these parameters. However, it's important to note that this strategy may somewhat constrain performance , especially when applied to problems with diverse characteristics. Thus, exploring adaptive or problem-specific tuning of hyperparameters could be a valuable avenue for improvement in future algorithm iterations.

Certainly, the scope of research is not limited to the topics listed. Numerous areas are worthy of study, such as extending the MBGO algorithm to different optimization problems, including multi-objective optimization, dynamic optimization, and multi-task optimization. Exploring these avenues can contribute to the MBGO algorithm's versatility and applicability across a broader range of problem domains.

6 Conclusion

We present a novel population-based heuristic algorithm named the multiplayer battle game-inspired optimizer (MBGO), inspired by the common mechanics of multiplayer battle royale and player behaviors in various scenarios. The proposed MBGO algorithm strategically balances exploration and exploitation by alternating between the movement and battle phases, enhancing its adaptability to optimization problems with diverse characteristics. The experimental results confirm the effectiveness and competitiveness of the new MBGO algorithm, whether applied to solving human-designed functions or real industrial problems.

In our forthcoming work, we intend to introduce more efficient search strategies further to enhance the performance of the proposed MBGO algorithm. We also aim to broaden the scope to include various optimization problems, such as multi-objective optimization, dynamic optimization, and constrained optimization. Additionally, we plan to conduct a theoretical analysis of the effective mechanism of the MBGO algorithm and work towards establishing feedback between the optimization problem and the algorithm to reduce overall optimization costs.

Appendix A

WBP Coello [57] proposed this benchmark problem, and the objective of the problem was to find the minimum manufacturing cost of a welded beam. The variables are

weld thickness $h(x_1)$, height $l(x_2)$, length $t(x_3)$ and bar thickness $b(x_4)$. The mathematical expression of the objective function is:

minimize:

$$f(X) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2),$$

subject to:

$$g_1(X) = \tau(X) - \tau_{max} \leq 0,$$
$$g_2(X) = \sigma(X) - \sigma_{max} \leq 0,$$
$$g_3(X) = \delta(X) - \delta_{max} \leq 0,$$
$$g_4(X) = x_1 - x_4 \leq 0,$$
$$g_5(X) = P - P_c(X) \leq 0,$$
$$g_6(X) = 0.125 - x_1 \leq 0,$$
$$g_7(X) = 1.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \leq 0,$$
$$\tau(X) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2},$$
$$\tau'(X) = \frac{P}{\sqrt{2}x_1x_2},$$
$$\tau''(X) = \frac{MR}{J},$$
$$M = P(L + \frac{x_2}{2}),$$
$$R = \sqrt{\frac{x_2^2}{4} + (\frac{x_1 + x_3}{2})^3},$$
$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{4} + (\frac{x_1 + x_3}{2})^2\right]\right\},$$
$$\sigma(\mathbf{X}) = \frac{6PL}{x_4x_3^2},$$
$$\delta(\mathbf{X}) = \frac{6PL^3}{Ex_3^2x_4},$$
$$P_c(\mathbf{X}) = \frac{4.013E\sqrt{x_3^2x_4^6/36}}{L^2}(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}),$$
$$P = 6000 \text{ lb},$$
$$L = 14 \text{ in},$$
$$\delta_{max} = 0.25 \text{ in},$$
$$E = 30 \times 10^6 \text{ psi},$$
$$G = 12 \times 10^6 \text{ psi},$$
$$\tau_{max} = 13,600 \text{ psi},$$
$$\sigma_{max} = 30,000 \text{ psi},$$

variable range:

$$x_1 \geq 0.1,$$
$$x_2 \geq 0.1,$$
$$x_3 \leq 10,$$
$$x_4 \leq 2.$$

PVP The objective is to minimize the total cost, including material, forming, and welding costs. The problem has four variables, including shell thickness $Ts(x1)$, head thickness $Th(x2)$, inner radius $R(x3)$, and the length of the cylindrical portion of the vessel (excluding the head) $L(x4)$. In addition, $x1$ and $x2$ are integer multiples of 0.0625 inches, while the other variables are continuous. The optimization problem can be expressed as follows:

minimize:

$$f(X) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3,$$

subject to:

$$g_1(X) = -x_1 + 0.0193x_3 \leq 0,$$
$$g_2(X) = -x_2 + 0.00954x_3 \leq 0,$$
$$g_3(X) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1,296,000 \leq 0,$$
$$g_4(X) = x_4 - 240 \leq 0,$$

variable range:

$$x_1, x_2 \in \{1 \times 0.0625, 2 \times 0.0625, 3 \times 0.0625, \ldots 1600 \times 0.0625\},$$
$$x_3 \geq 10,$$
$$x_4 \leq 200.$$

TBTD This problem is an optimization problem for a three-bar planar truss structure. The volume of a statically loaded 3-rod truss is to be minimized with stress (σ) constraints on each truss member. The objective is to evaluate the optimum cross-sectional areas $A_1(x_1)$ and $A_2(x_2)$. The optimization problem can be expressed as follows:

minimize:

$$f(X) = (2\sqrt{2}x_1 + x_2) \times l,$$

subject to:

$$g_1(X) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2}P - \sigma \leq 0,$$
$$g_2(X) = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2}P - \sigma \leq 0,$$
$$g_3(X) = \frac{1}{\sqrt{2}x_2 + x_1}P - \sigma \leq 0,$$
$$l = 100 \text{ cm},$$
$$P = 2 \text{ kN/cm}^3,$$
$$\sigma = 2 \text{ kN/cm}^3,$$

variable range:

$$x_1 \geq 0,$$
$$x_2 \leq 1.$$

GTD GTD is an unconstrained discrete design problem proposed by Sandgren [58]. This benchmark task minimizes the gear ratio, defined as the ratio of the output shaft's angular velocity to the input shaft's angular velocity. Ratio to input shaft angular velocity. The number of teeth of the gears $n_A(x_1)$, $n_B(x_2)$, $n_C(x_3)$ and $n_D(x_4)$ are

considered as the design variables for the problem. The mathematical formulation is as follows:

minimize:

$$f(X) = (\frac{1}{6.931} - \frac{x_3 x_2}{x_1 x_4})^2,$$

variable range:

$$x_1, x_2, x_3, x_4 \in \{12, 13, 14 \ldots, 60\}.$$

CBD This problem is a good benchmark for validating the ability of optimization methods to solve continuous, discrete, and mixed-variable structural design problems. The objective of the problem is to minimize the volume of the beam. The design variables include segment widths ($x1$, $x2$, $x3$, $x4$, $x5$) and segment heights ($x6$, $x7$, $x8$, $x9$, $x10$). In addition to the bending stress limitations, a specific aspect ratio is specified, i.e., the height-to-width ratio of the beam segments must be less than 20. The problem is formulated as follows:

minimize:

$$f(X) = D(x_1 x_6 l_1 + x_2 x_7 l_2 + x_3 x_8 l_3 + x_4 x_9 l_4 + x_5 x_{10} l_5),$$

subject to:

$$g_1(X) = \frac{6Pl_5}{x_5 x_{10}^2} - \sigma_d \le 0,$$

$$g_2(X) = \frac{6P(l_5 + l_4)}{x_4 x_9^2} - \sigma_d \le 0,$$

$$g_3(X) = \frac{6P(l_5 + l_4 + l_3)}{x_3 x_8^2} - \sigma_d \le 0,$$

$$g_4(X) = \frac{6P(l_5 + l_4 + l_3 + l_2)}{x_2 x_7^2} - \sigma_d \le 0,$$

$$g_5(X) = \frac{6P(l_5 + l_4 + l_3 + l_2 + l_1)}{x_1 x_6^2} - \sigma_d \le 0,$$

$$g_6(X) = \frac{Pl^3}{3E}(\frac{1}{I_5} + \frac{1}{I_4} + \frac{1}{I_3} + \frac{1}{I_2} + \frac{1}{I_1}) - \Delta_{max} \le 0,$$

$$g_7(X) = \frac{x_{10}}{x_5} - 20 \le 0,$$

$$g_8(X) = \frac{x_9}{x_4} - 20 \le 0,$$

$$g_9(X) = \frac{x_8}{x_3} - 20 \le 0,$$

$$g_{10}(X) = \frac{x_7}{x_2} - 20 \le 0,$$

$$g_{11}(X) = \frac{x_6}{x_1} - 20 \le 0,$$

$$P = 50000N,$$

$$\sigma_d = 14,000 \ \text{N/cm}^2,$$

$$E = 2 \times 10^7 \ \text{N/cm}^2,$$

$$\Delta_{max} = 2.7\text{cm},$$

$$D = 1.0,$$

variable range:

$$x_1 \in \{1, 2, 3, 4, 5\},$$

$$x_2, x_3 \in \{2.4, 2.6, 2.8, 3.1\},$$

$$x_4 \ge 1,$$

$$x_5 \le 5,$$

$$x_6 \in \{30, 31, 32, \ldots, 65\},$$

$$x_7, x_8 \in \{45, 50, 55, 60, 65\},$$

$$x_9 \ge 30,$$

$$x_{10} \le 65.$$

IBD The objective of this problem is to minimize the vertical deflection of the I-beam. The variables of this problem include flange width $b(x_1)$, section height $h(x_2)$, web thickness $t_w(x_3)$ and flange thickness $t_f(x_4)$. The maximum vertical deflection of the beam is $f(x) = PL^3/48EI$ when its length (L) and modulus of elasticity (E) are 5200 cm and 523.104kN/cm^2, respectively. The objective function of the problem The objective function and constraints of the problem are formulated as follows:

minimize:

$$f(X) = \frac{5000}{x_3(x_2 - 2x_4)^3/12 + (x_1 x_4^3/6) + 2bx_4(x_2 - x_4/2)^2},$$

subject to:

$$g_1(X) = 2x_1 x_3 + x_3(x_2 - 2x_4) \le 300,$$

$$g_2(X) = \frac{18x_2 \times 10^4}{x_3(x_2 - 2x_4)^3 + 2x_1 x_3(4x_4^2 + 3x_2(x_2 - 2x_4))} + \frac{15x_1 \times 10^3}{(x_2 - 2x_4)x_3^2 + 2x_3 x_1^3} \le 56,$$

variable range:

$$10 \le x_1 \le 50,$$

$$10 \le x_2 \le 80,$$

$$0.9 \le x_3 \le 5,$$

$$0.9 \le x_4 \le 5.$$

TCD This problem is an example of designing a uniform column of tubular cross-section to carry compressive loads at minimum cost. The problem has two design variables: the mean diameter of column $d(x_1)$ and tube t's thickness (x_2). The material yield stress of the tube column is $\sigma_y = 500\text{kgf/cm}^2$, and the modulus of elasticity is $E = 0.85 \times 10^6\text{kgf/cm}^2$ The optimization model for this problem is as follows:

minimize:

$$f(X) = 9.8x_1 x_2 + 2x_1,$$

subject to:

$$g_1(X) = \frac{P}{\pi x_1 x_2 \sigma_y} - 1 \le 0,$$

$$g_2(X) = \frac{8PL^2}{\pi^3 E x_1 x_2 (x_1^2 + x_2^2)} - 1 \le 0,$$

$$g_3(X) = \frac{2.0}{x_1} - 1 \le 0,$$

$$g_4(X) = \frac{x_1}{14} - 1 \le 0,$$

$$g_5(X) = \frac{0.2}{x_2} - 1 \le 0,$$

$$g_6(X) = \frac{x_2}{8} - 1 \le 0,$$

variable range:

$$2 \le x_1 \le 14,$$

$$0.2 \le x_2 \le 0.8.$$

PLD The main objective of the problem is to locate the piston parts $H(x_1)$, $B(x_2)$, $D(x_3)$ and $X(x_4)$ by minimizing

the amount of oil as the piston rod rises from 0° to 45°. The formula for this problem is as follows:

minimize:

$$f(X) = \frac{1}{4}\pi x_3^2(L_2 - L_1),$$

subject to:

$$g_1(X) = QL\cos\theta - R \times F \leq 0,$$
$$g_2(X) = Q(L - x_4) - M_{max} \leq 0,$$
$$g_3(X) = 1.2(L_2 - L_1) - L_1 \leq 0,$$
$$g_4(X) = \frac{x_3}{2} - x_2 \leq 0,$$

where:

$$R = \frac{|-x_4(x_4\sin\theta + x_1) + x_1(x_2 - x_4\cos\theta)|}{\sqrt{(x_4 - x_2)^2 + x_1^2}},$$
$$\theta = 45^\circ,$$
$$Q = 10,000 \text{ lbs},$$
$$L = 240 \text{ in},$$
$$M_{max} = 1.8 \times 10^6 \text{ lbs in},$$
$$P = 1500\text{psi},$$

variable range:

$$0.05 \leq x_1, x_2, x_4 \leq 500,$$
$$0.05 \leq x_2 \leq 120.$$

CBHD The problem aims to minimize the weight of the corrugated bulkheads of a chemical tanker [59], where the design variables include width (x_1), depth (x_2), length $(x3)$ and plate thickness (x_4). The mathematical model for this optimization problem is as follows:

minimize:

$$f(X) = \frac{5.885x_4(x_1 + x_3)}{x_1 + \sqrt{|x_3^2 - x_2^2|}},$$

subject to:

$$g_1(X) = -x_4x_2(0.4x_1 + \frac{x_3}{6}) + 8.94(x_1 + \sqrt{|x_3^2 - x_2^2|}) \leq 0,$$
$$g_2(X) = -x_4x_2^2(0.2x_1 + \frac{x_3}{12}) + 2.2(8.94(x_1 + \sqrt{|x_3^2 - x_2^2|}))^{4/3} \leq 0,$$
$$g_3(X) = -x_4 + 0.0156x_1 + 0.15 \leq 0,$$
$$g_4(X) = -x_4 + 0.0156x_3 + 0.15 \leq 0,$$
$$g_5(X) = -x_4 + 1.05 \leq 0,$$
$$g_6(X) = -x_3 + x_2 \leq 0,$$

variable range:

$$0 \leq x_1, x_2, x_3 \leq 100,$$
$$0 \leq x_4 \leq 5.$$

RCB Amir and Hasegawa [60] presented a simplified design optimization problem for a reinforced concrete beam. The beam was assumed to be supported with a span of 30 ft. It was subjected to a live load of 2000 lbs. and a dead load of 1000 lbs., including the weight of the beam. The compressive strength of the concrete (σ_x) is 5 ksi, and the yield stress of the steel reinforcement (σ_y) is 50 ksi. The cost of the concrete is 0.02 dollars/sq.ft., and the cost of the steel is 1.0 dollars/sq.ft. The cost of the steel is 1.0 dollars/sq.ft. The cost of the concrete is 1.0 dollars/sq.ft. To minimize the total structural cost, the area of reinforcement A $s(x_1)$, beam width $b(x_2)$ and beam depth $h(x_3)$ must be determined. According to ACI Building Code 318-77, the structure should be scaled to achieve the required strength as follows:

$$M_u = 0.9A_s\sigma_y(0.8h)(1.0 - 0.59\frac{A_s\sigma_y}{0.8bh\sigma_c} \geq 1.4M_d + 1.7M_l),$$

where M_u, M_d and M_l are the beam's flexural strength, dead $weight, andliveweight$

bending moment, respectively. In this example, M_d is 1350 kip and M_l is 2700 kip,

and the depth-to-width ratio of the beam must be less than $orequalto4$.

minimize:

$$f(X) = 2.9x_1 + 0.6x_2x_3,$$

subject to:

$$g_1(X) = \frac{x_2}{x_3} - 4 \leq 0,$$
$$g_2(X) = 180 + 7.375\frac{x_1^2}{x_3} - x_1x_2 \leq 0,$$

variable range:

$$x_1 \in \{6, 6.16, 6.32, 6.6, 7, 7.11, 7.2, 7.8, 7.9, 8, 8.4\},$$
$$x_2 \in \{28, 29, 30, \ldots, 40\},$$
$$5 \leq x_3 \leq 100.$$

See Tables 3, 4, 5, 6, 7.

Table 3 The mean and standard deviation of the optimal solutions obtained from 30 trial runs for all algorithms are reported for the 10-D CEC2017 function functions

FunNum.		DE	PSO	AO	SOA	SFO	WOA	HBA	TSA	BRO	CMA-ES	MBGO
F1	Mean	7.64466E + 08 +	3.68993E + 08 +	4.24047E + 09 +	1.25309E + 10 +	2.40426E + 09 +	6.14546E + 07 +	1.47737E + 06 +	6.74260E + 08 +	6.74260E + 08 +	6.74260E + 08 +	**1.23427E + 05**
	Std	1.61470E + 08	5.84118E + 08	2.37566E + 09	3.67427E + 09	8.83315E + 08	9.46224E + 07	9.36622E + 05	8.98773E + 08	8.98773E + 08	8.98773E + 08	9.96621E + 04
F3	Mean	2.38647E + 04 +	9.62204E + 03 +	8.24022E + 03 +	5.63362E + 04 +	9.60599E + 03 +	3.97144E + 03 +	**4.12650E + 02** −	9.15844E + 03 +	9.15844E + 03 +	9.15844E + 03 +	2.41701E + 03
	Std	5.82857E + 03	4.12874E + 03	1.97750E + 03	3.68734E + 04	3.70132E + 03	2.34796E + 03	1.98065E + 02	3.89578E + 03	3.89578E + 03	3.89578E + 03	8.12163E + 02
F4	Mean	5.03108E + 02 +	4.68191E + 02 +	6.72252E + 02 +	1.32787E + 03 +	5.70530E + 02 +	4.39072E + 02 =	4.12297E + 02 =	4.39715E + 02 +	4.39715E + 02 +	4.39715E + 02 +	**4.10328E + 02**
	Std	2.58040E + 01	5.02897E + 01	1.14916E + 02	4.87052E + 02	8.03686E + 01	3.75341E + 01	1.94224E + 01	5.22347E + 01	5.22347E + 01	5.22347E + 01	1.32905E + 01
F5	Mean	5.56964E + 02 +	5.64241E + 02 +	5.70473E + 02 +	5.99794E + 02 +	5.60376E + 02 +	5.49968E + 02 +	5.30301E + 02 =	5.46232E + 02 +	5.46232E + 02 +	5.46232E + 02 +	**5.26292E + 02**
	Std	7.16766E + 00	8.14321E + 00	1.43013E + 01	2.00553E + 01	1.46486E + 01	1.58343E + 01	1.01718E + 01	1.23846E + 01	1.23846E + 01	1.23846E + 01	5.42200E + 00
F6	Mean	6.30678E + 02 +	6.25098E + 02 +	6.40934E + 02 +	6.58487E + 02 +	6.32815E + 02 +	6.35962E + 02 +	6.08496E + 02 +	6.25018E + 02 +	6.25018E + 02 +	6.25018E + 02 +	**6.00356E + 02**
	Std	4.86731E + 00	9.40723E + 00	1.03541E + 01	1.54637E + 01	9.47478E + 00	1.00427E + 01	6.40122E + 00	9.95576E + 00	9.95576E + 00	9.95576E + 00	1.06416E − 01
F7	Mean	8.68418E + 02 +	7.87146E + 02 +	8.30876E + 02 +	8.69088E + 02 +	7.98153E + 02 +	8.04610E + 02 +	7.48736E + 02 =	7.65938E + 02 +	7.65938E + 02 +	7.65938E + 02 +	**7.44233E + 02**
	Std	2.55129E + 01	2.37557E + 01	1.80645E + 01	4.19110E + 01	1.67873E + 01	4.38470E + 01	1.12907E + 01	1.35376E + 01	1.35376E + 01	1.35376E + 01	5.63262E + 00
F8	Mean	8.70355E + 02 +	8.66387E + 02 +	8.60948E + 02 +	8.86013E + 02 +	8.52620E + 02 +	8.40698E + 02 +	**8.23013E + 02** =	8.35398E + 02 +	8.35398E + 02 +	8.35398E + 02 +	8.27021E + 02
	Std	8.26465E + 00	1.02721E + 01	8.51332E + 00	1.47476E + 01	1.10650E + 01	1.39569E + 01	7.03877E + 00	1.16408E + 01	1.16408E + 01	1.16408E + 01	5.26578E + 00
F9	Mean	2.15082E + 03 +	1.28248E + 03 +	1.66262E + 03 +	2.24364E + 03 +	1.39451E + 03 +	1.60714E + 03 +	1.00343E + 03 +	1.14272E + 03 +	1.14272E + 03 +	1.14272E + 03 +	**9.00708E + 02**
	Std	2.60924E + 02	2.95529E + 02	2.89719E + 02	6.08839E + 02	2.54113E + 02	3.04448E + 02	1.20540E + 02	1.59222E + 02	1.59222E + 02	1.59222E + 02	4.96108E − 01
F10	Mean	**1.88908E + 03** −	2.73067E + 03 +	2.41646E + 03 =	3.12247E + 03 +	2.72096E + 03 +	2.12038E + 03 −	1.93844E + 03 −	2.33668E + 03 =	2.33668E + 03 +	2.33668E + 03 +	2.31465E + 03
	Std	2.01769E + 02	1.49715E + 02	2.38967E + 02	2.53377E + 02	2.26312E + 02	3.78668E + 02	3.01016E + 02	2.77813E + 02	2.77813E + 02	2.77813E + 02	1.76539E + 02
F11	Mean	1.19980E + 03 +	1.26296E + 03 +	3.23246E + 03 +	6.35805E + 03 +	1.41464E + 03 +	1.21216E + 03 +	1.13501E + 03 +	1.38067E + 03 +	1.38067E + 03 +	1.38067E + 03 +	**1.11422E + 03**
	Std	1.78088E + 01	1.03482E + 02	2.85908E + 03	6.29986E + 03	2.22539E + 02	7.54091E + 01	3.81377E + 01	7.99927E + 02	7.99927E + 02	7.99927E + 02	4.81034E + 00
F12	Mean	1.15837E + 07 +	3.88810E + 07 +	1.42977E + 08 +	6.79024E + 08 +	3.23156E + 07 +	7.55178E + 05 −	**3.05485E + 05** −	4.48287E + 06 +	4.48287E + 06 +	4.48287E + 06 +	8.58320E + 05
	Std	4.69301E + 06	5.31216E + 07	1.55445E + 08	4.49169E + 08	3.70923E + 07	1.29301E + 06	9.85968E + 05	2.19117E + 06	2.19117E + 06	2.19117E + 06	7.14588E + 05
F13	Mean	**1.97057E + 03** −	5.05614E + 04 +	4.27704E + 05 +	1.55918E + 07 +	5.60036E + 05 +	1.31381E + 04 +	1.30165E + 04 +	1.42373E + 04 +	1.42373E + 04 +	1.42373E + 04 +	3.30320E + 03
	Std	2.08420E + 02	8.37433E + 04	6.52991E + 05	2.34420E + 07	6.89500E + 05	9.54927E + 03	1.02598E + 04	7.85212E + 03	7.85212E + 03	7.85212E + 03	1.27851E + 03
F14	Mean	**1.43833E + 03** −	2.35281E + 03 +	1.87863E + 03 +	5.92395E + 03 +	3.22355E + 03 +	1.52021E + 03 +	1.62682E + 03 +	3.98344E + 03 +	3.98344E + 03 +	3.98344E + 03 +	1.46719E + 03
	Std	4.01878E + 00	1.24992E + 03	5.44713E + 02	1.28369E + 04	3.07387E + 03	3.34916E + 01	3.58388E + 02	2.00286E + 03	2.00286E + 03	2.00286E + 03	1.40046E + 01
F15	Mean	**1.59350E + 03** −	9.68597E + 03 +	7.44833E + 03 +	2.23714E + 04 +	1.21706E + 04 +	1.72585E + 03 =	3.93405E + 03 +	8.96332E + 03 +	8.96332E + 03 +	8.96332E + 03 +	1.85022E + 03
	Std	1.84308E + 01	1.02890E + 04	4.71682E + 03	1.38291E + 04	7.95718E + 03	1.29618E + 02	3.22394E + 03	7.09161E + 03	7.09161E + 03	7.09161E + 03	2.11487E + 02
F16	Mean	1.72863E + 03 +	1.92639E + 03 +	1.89830E + 03 +	2.24125E + 03 +	1.94525E + 03 +	1.87967E + 03 +	1.80061E + 03 +	1.90576E + 03 +	1.90576E + 03 +	1.90576E + 03 +	**1.65271E + 03**
	Std	5.67472E + 01	1.12011E + 02	1.04058E + 02	1.97419E + 02	1.17318E + 02	1.72267E + 02	1.42828E + 02	1.33435E + 02	1.33435E + 02	1.33435E + 02	3.17356E + 01
F17	Mean	1.79398E + 03 +	1.85232E + 03 +	1.84739E + 03 +	2.01478E + 03 +	1.81619E + 03 +	1.82033E + 03 +	**1.75134E + 03** =	1.81686E + 03 +	1.81686E + 03 +	1.81686E + 03 +	1.75615E + 03
	Std	1.32430E + 01	4.44707E + 01	4.66424E + 01	1.26226E + 02	2.81354E + 01	6.43461E + 01	2.07884E + 01	5.28438E + 01	5.28438E + 01	5.28438E + 01	1.10809E + 01
F18	Mean	5.56295E + 03 =	1.07571E + 06 +	1.65560E + 06 +	1.61697E + 07 +	4.83872E + 05 +	1.40718E + 04 =	2.00710E + 04 +	3.91751E + 04 +	3.91751E + 04 +	3.91751E + 04 +	**4.39943E + 03**
	Std	1.81156E + 03	1.81075E + 06	2.16421E + 06	2.41473E + 07	1.08961E + 06	1.39772E + 04	1.69808E + 04	2.32854E + 04	2.32854E + 04	2.32854E + 04	1.56176E + 03

Table 3 (continued)

FunNum.		DE	PSO	AO	SOA	SFO	WOA	HBA	TSA	BRO	CMA-ES	MBGO
F19	Mean	**1.94224E + 03** −	1.67011E + 04 +	2.55648E + 04 +	2.57652E + 06 +	1.28077E + 04 +	1.36289E + 04 +	8.97834E + 03 +	1.76280E + 04 +	1.76280E + 04 +	1.76280E + 04 +	2.01311E + 03
	Std	1.03476E + 01	1.36183E + 04	2.55707E + 04	3.87228E + 06	1.63252E + 04	2.02891E + 04	9.23597E + 03	5.42162E + 04	5.42162E + 04	5.42162E + 04	1.36988E + 02
F20	Mean	2.08342E + 03 +	2.15867E + 03 +	2.16317E + 03 +	2.34993E + 03 +	2.17053E + 03 +	2.19537E + 03 +	2.05277E + 03 =	2.17777E + 03 +	2.17777E + 03 +	2.17777E + 03 +	**2.04868E + 03**
	Std	1.47888E + 01	2.82106E + 01	5.75749E + 01	7.38924E + 01	5.24406E + 01	7.21145E + 01	4.17937E + 01	7.83246E + 01	7.83246E + 01	7.83246E + 01	1.36234E + 01
F21	Mean	2.33506E + 03 +	**2.24190E + 03** −	2.30841E + 03 =	2.38863E + 03 +	2.27704E + 03 =	2.32176E + 03 +	2.29016E + 03 =	2.31048E + 03 +	2.31048E + 03 =	2.31048E + 03 +	2.28119E + 03
	Std	4.82582E + 01	2.73547E + 01	5.36150E + 01	2.74926E + 01	5.25710E + 01	4.21343E + 01	5.72811E + 01	5.01527E + 01	5.01527E + 01	5.01527E + 01	4.37854E + 01
F22	Mean	2.43755E + 03 +	2.34170E + 03 +	2.59600E + 03 +	3.10213E + 03 +	2.48606E + 03 +	2.35311E + 03 +	2.30522E + 03 +	2.33834E + 03 +	2.33834E + 03 +	2.33834E + 03 +	**2.30509E + 03**
	Std	4.77959E + 01	4.74167E + 01	1.88941E + 02	4.40763E + 02	1.28593E + 02	4.11238E + 01	1.78814E + 01	4.78548E + 01	4.78548E + 01	4.78548E + 01	9.87647E + 00
F23	Mean	2.64975E + 03 +	2.66501E + 03 +	2.66126E + 03 +	2.72595E + 03 +	2.66904E + 03 +	2.66505E + 03 +	2.64819E + 03 +	2.72066E + 03 +	2.72066E + 03 +	2.72066E + 03 +	**2.62511E + 03**
	Std	6.87108E + 00	1.16566E + 01	2.39106E + 01	4.00173E + 01	1.87726E + 01	2.90997E + 01	1.80866E + 01	4.45878E + 01	4.45878E + 01	4.45878E + 01	4.50712E + 00
F24	Mean	2.78386E + 03 +	2.75783E + 03 +	2.76442E + 03 +	2.87626E + 03 +	2.75231E + 03 +	2.77731E + 03 +	2.74216E + 03 +	2.78566E + 03 +	2.78566E + 03 +	2.78566E + 03 +	**2.72220E + 03**
	Std	7.25681E + 00	5.83719E + 01	6.24661E + 01	4.86604E + 01	8.06207E + 01	7.96993E + 01	9.59917E + 01	7.93578E + 01	7.93578E + 01	7.93578E + 01	6.30112E + 01
F25	Mean	2.99627E + 03 +	2.97172E + 03 +	3.13613E + 03 +	3.66334E + 03 +	3.07981E + 03 +	2.96487E + 03 +	**2.93468E + 03** =	2.96627E + 03 +	2.96627E + 03 +	2.96627E + 03 +	2.93943E + 03
	Std	1.37705E + 01	3.06785E + 01	1.31515E + 02	3.98784E + 02	8.13017E + 01	4.91619E + 01	2.94351E + 01	5.56844E + 01	5.56844E + 01	5.56844E + 01	1.60294E + 01
F26	Mean	3.10245E + 03 +	3.15131E + 03 +	3.41132E + 03 +	4.22725E + 03 +	3.29186E + 03 +	3.47008E + 03 +	3.11029E + 03 +	3.40310E + 03 +	3.40310E + 03 +	3.40310E + 03 =	**2.93222E + 03**
	Std	2.99213E + 01	1.22436E + 02	2.02071E + 02	4.82428E + 02	1.19665E + 02	3.83458E + 02	2.49653E + 02	3.24124E + 02	3.24124E + 02	3.24124E + 02	1.48421E + 01
F27	Mean	3.09643E + 03 =	3.12402E + 03 +	3.15078E + 03 +	3.26996E + 03 +	3.14154E + 03 +	3.18056E + 03 +	3.10643E + 03 +	3.17376E + 03 +	3.17376E + 03 +	3.17376E + 03 =	**3.09602E + 03**
	Std	1.79361E + 00	1.04715E + 01	3.71359E + 01	6.51917E + 01	2.72829E + 01	4.70251E + 01	2.18016E + 01	3.55270E + 01	3.55270E + 01	3.55270E + 01	2.48369E + 00
F28	Mean	**3.22433E + 03** =	3.33044E + 03 =	3.65342E + 03 +	3.81320E + 03 +	3.38967E + 03 +	3.38925E + 03 =	3.29664E + 03 =	3.42355E + 03 +	3.42355E + 03 +	3.42355E + 03 =	3.28066E + 03
	Std	3.54469E + 01	7.32005E + 01	1.78459E + 02	1.53040E + 02	1.11614E + 02	1.79449E + 02	1.18874E + 02	1.39914E + 02	1.39914E + 02	1.39914E + 02	9.87999E + 01
F29	Mean	3.25273E + 03 +	3.31791E + 03 +	3.29423E + 03 +	3.58989E + 03 +	3.34508E + 03 +	3.37314E + 03 +	3.24660E + 03 =	3.29636E + 03 +	3.29636E + 03 +	3.29636E + 03 =	**3.21375E + 03**
	Std	3.40476E + 01	4.07843E + 01	6.35334E + 01	1.46898E + 02	6.65584E + 01	1.19969E + 02	6.46957E + 01	6.91539E + 01	6.91539E + 01	6.91539E + 01	3.11328E + 01
F30	Mean	6.48074E + 05 =	4.32629E + 06 +	4.40270E + 06 +	2.21409E + 07 +	4.18369E + 06 +	9.66807E + 05 =	**5.70735E + 05** =	8.52638E + 05 =	8.52638E + 05 +	8.52638E + 05 +	7.29942E + 05
	Std	3.21731E + 05	2.29846E + 06	4.12051E + 06	1.58854E + 07	3.81037E + 06	1.09513E + 06	6.54958E + 05	1.32075E + 06	1.32075E + 06	1.32075E + 06	8.07634E + 05
+/=/−		20/4/5	27/1/1	27/2/0	29/0/0	28/1/0	22/5/2	15/11/3	27/2/0	28/1/0	25/4/0	−

In this context, symbols + and − signify that the proposed algorithm outperforms or underperforms the comparison algorithm. The symbol = denotes no significant difference between the two. Furthermore, the optimal solutions identified among all algorithms are highlighted in bold for each function. The last row of the table represents the "number of functions that outperform the comparison algorithm/number of functions that are significantly different/number of functions that are lower than the comparison algorithm"

Table 4 The mean and standard deviation of the optimal solutions obtained from 30 trial runs for all algorithms are reported for CEC2017-30D

FunNum.		DE	PSO	AO	SOA	SFO	WOA	HBA	TSA	BRO	CMA-ES	MBGO
F1	Mean	3.87282E + 10 +	4.35017E + 09 +	4.99145E + 10 +	6.00640E + 10 +	4.30648E + 10 +	8.01078E + 09 +	4.42217E + 09 +	2.14780E + 10 +	2.14780E + 10 +	2.14780E + 10 +	**3.99992E + 05**
	Std	5.06087E + 09	1.85017E + 09	5.61867E + 09	8.10381E + 09	5.72354E + 09	3.36609E + 09	2.93876E + 09	5.22941E + 09	5.22941E + 09	5.22941E + 09	2.13384E + 05
F3	Mean	2.50686E + 05 +	1.09058E + 05 +	6.99651E + 04 +	2.04185E + 06 +	1.08376E + 05 +	1.52083E + 05 +	**2.64181E + 04** −	1.20910E + 05 +	1.20910E + 05 +	1.20910E + 05 +	3.70252E + 04
	Std	3.19897E + 04	1.99724E + 04	1.03126E + 04	4.35444E + 06	1.74360E + 04	7.65871E + 04	7.15016E + 03	2.89512E + 04	2.89512E + 04	2.89512E + 04	7.16807E + 03
F4	Mean	3.56580E + 03 +	2.38551E + 03 +	1.19041E + 04 +	1.86999E + 04 +	9.45542E + 03 +	1.08162E + 03 +	7.85472E + 02 +	2.53495E + 03 +	2.53495E + 03 +	2.53495E + 03 +	**5.19672E + 02**
	Std	6.36343E + 02	1.49944E + 03	2.91912E + 03	2.57001E + 03	2.55329E + 03	4.66143E + 02	3.81885E + 02	1.00287E + 03	1.00287E + 03	1.00287E + 03	2.62196E + 01
F5	Mean	8.98858E + 02 +	8.44372E + 02 +	9.39070E + 02 +	9.75277E + 02 +	8.73324E + 02 +	8.02708E + 02 +	7.02321E + 02 +	8.04491E + 02 +	8.04491E + 02 +	8.04491E + 02 +	**6.66385E + 02**
	Std	2.27770E + 01	4.15892E + 01	4.70816E + 01	5.58407E + 01	3.11155E + 01	5.95407E + 01	2.99687E + 01	4.40297E + 01	4.40297E + 01	4.40297E + 01	1.23010E + 01
F6	Mean	6.74197E + 02 +	6.56551E + 02 +	6.87605E + 02 +	6.99607E + 02 +	6.74702E + 02 +	6.66641E + 02 +	6.50016E + 02 +	6.69651E + 02 +	6.69651E + 02 +	6.69651E + 02 +	**6.00188E + 02**
	Std	5.12049E + 00	1.31809E + 01	8.90427E + 00	1.07954E + 01	9.95076E + 00	1.24397E + 01	7.84410E + 00	1.10059E + 01	1.10059E + 01	1.10059E + 01	1.65036E − 01
F7	Mean	2.42984E + 03 +	1.14668E + 03 +	1.46259E + 03 +	1.50815E + 03 +	1.39704E + 03 +	1.27800E + 03 +	1.10040E + 03 +	1.34196E + 03 +	1.34196E + 03 +	1.34196E + 03 +	**9.15313E + 02**
	Std	1.89555E + 02	7.95551E + 01	6.20837E + 01	6.29945E + 01	5.46806E + 01	8.80760E + 01	5.97573E + 01	9.18243E + 01	9.18243E + 01	9.18243E + 01	1.38793E + 01
F8	Mean	1.18980E + 03 +	1.10099E + 03 +	1.16686E + 03 +	1.18677E + 03 +	1.11999E + 03 +	1.04483E + 03 +	9.62411E + 02 =	1.07925E + 03 +	1.07925E + 03 +	1.07925E + 03 +	**9.61973E + 02**
	Std	1.54084E + 01	4.10527E + 01	3.03720E + 01	3.70237E + 01	2.47304E + 01	6.97442E + 01	2.63984E + 01	3.68188E + 01	3.68188E + 01	3.68188E + 01	1.41081E + 01
F9	Mean	1.73323E + 04 +	9.75352E + 03 +	1.20697E + 04 +	1.64439E + 04 +	9.97464E + 03 +	8.64269E + 03 +	6.30077E + 03 +	1.02326E + 04 +	1.02326E + 04 +	1.02326E + 04 +	**9.15182E + 02**
	Std	1.50910E + 03	3.52710E + 03	1.70080E + 03	3.89220E + 03	1.82430E + 03	2.84585E + 03	1.10683E + 03	2.38115E + 03	2.38115E + 03	2.38115E + 03	1.39264E + 01
F10	Mean	8.26985E + 03 +	8.72799E + 03 +	8.23496E + 03 +	9.58142E + 03 +	8.89105E + 03 +	6.69124E + 03 −	**5.92990E + 03** −	7.96282E + 03 =	7.96282E + 03 +	7.96282E + 03 +	7.85694E + 03
	Std	3.25130E + 02	3.10489E + 02	4.72787E + 02	3.63808E + 02	4.08562E + 02	1.08254E + 03	7.99841E + 02	5.93276E + 02	5.93276E + 02	5.93276E + 02	3.67659E + 02
F11	Mean	3.13158E + 03 +	3.97015E + 03 +	9.43675E + 03 +	3.42447E + 04 +	1.00525E + 04 +	3.51489E + 03 +	1.34643E + 03 =	5.01659E + 03 +	5.01659E + 03 +	5.01659E + 03 +	**1.31313E + 03**
	Std	3.71489E + 02	1.70996E + 03	2.05295E + 03	1.45265E + 04	2.83641E + 03	1.75156E + 03	7.66939E + 01	1.71376E + 03	1.71376E + 03	1.71376E + 03	4.15287E + 01
F12	Mean	1.72113E + 09 +	5.12897E + 08 +	8.54769E + 09 +	1.42996E + 10 +	6.29840E + 09 +	1.34365E + 08 +	2.71321E + 07 +	1.25743E + 09 +	1.25743E + 09 +	1.25743E + 09 +	**1.73083E + 06**
	Std	3.08981E + 08	6.83773E + 08	3.12903E + 09	3.41071E + 09	2.55685E + 09	2.68206E + 08	2.51188E + 07	8.88951E + 08	8.88951E + 08	8.88951E + 08	1.02862E + 06
F13	Mean	1.10304E + 08 +	4.28655E + 07 +	2.20775E + 09 +	8.21703E + 09 +	1.40420E + 09 +	5.92204E + 05 +	1.21188E + 05 +	8.77981E + 07 +	8.77981E + 07 +	8.77981E + 07 =	**2.74757E + 04**
	Std	3.79655E + 07	1.48982E + 08	9.94234E + 08	4.67575E + 09	9.40965E + 08	2.02880E + 06	1.09741E + 05	1.25054E + 08	1.25054E + 08	1.25054E + 08	7.09992E + 04
F14	Mean	1.05319E + 04 +	5.61622E + 05 +	1.36870E + 06 +	1.72536E + 07 +	1.86535E + 06 +	4.24898E + 05 +	6.46041E + 04 +	8.62651E + 05 +	8.62651E + 05 +	8.62651E + 05 +	**5.21534E + 03**
	Std	4.74610E + 03	5.42878E + 05	1.98431E + 06	1.31614E + 07	1.49491E + 06	4.87604E + 05	4.64506E + 04	7.88728E + 05	7.88728E + 05	7.88728E + 05	2.87811E + 03
F15	Mean	4.13731E + 05 +	5.59049E + 06 +	4.11784E + 08 +	1.99204E + 09 +	1.98906E + 08 +	4.27483E + 04 +	**1.25203E + 04** =	3.59207E + 06 +	3.59207E + 06 +	3.59207E + 06 +	1.48796E + 04
	Std	1.57540E + 05	1.41288E + 07	2.84855E + 08	1.20891E + 09	1.21907E + 08	2.97084E + 04	1.14892E + 04	4.54004E + 06	4.54004E + 06	4.54004E + 06	1.02040E + 04
F16	Mean	3.91419E + 03 +	4.28797E + 03 +	4.47455E + 03 +	6.17877E + 03 +	4.43404E + 03 +	3.79895E + 03 +	**2.95422E + 03** =	3.70504E + 03 +	3.70504E + 03 +	3.70504E + 03 +	2.97144E + 03
	Std	2.15279E + 02	2.99300E + 02	4.93265E + 02	1.16181E + 03	4.71613E + 02	8.40468E + 02	3.87190E + 02	5.07267E + 02	5.07267E + 02	5.07267E + 02	2.43760E + 02
F17	Mean	2.67231E + 03 +	2.96637E + 03 +	3.37363E + 03 +	5.97787E + 03 +	2.99510E + 03 +	2.75143E + 03 +	2.38882E + 03 +	2.48578E + 03 +	2.48578E + 03 +	2.48578E + 03 +	**2.05094E + 03**
	Std	1.61530E + 02	2.19267E + 02	2.92921E + 02	4.22431E + 03	2.21034E + 02	3.09104E + 02	2.82634E + 02	2.34403E + 02	2.34403E + 02	2.34403E + 02	9.23526E + 01
F18	Mean	4.78661E + 06 +	1.18832E + 07 +	2.55793E + 07 +	2.20798E + 08 +	2.07456E + 07 +	1.55077E + 06 +	8.65527E + 05 +	5.53992E + 06 +	5.53992E + 06 +	5.53992E + 06 +	**1.31889E + 05**
	Std	2.81910E + 06	1.08943E + 07	2.27461E + 07	2.39977E + 08	1.77008E + 07	2.17125E + 06	8.82569E + 05	6.60241E + 06	6.60241E + 06	6.60241E + 06	6.16111E + 04

Table 4 (continued)

FunNum.		DE	PSO	AO	SOA	SFO	WOA	HBA	TSA	BRO	CMA-ES	MBGO
F19	Mean	8.32283E + 06 +	2.02271E + 07 +	5.39230E + 08 +	1.83285E + 09 +	2.30778E + 08 +	1.00949E + 05 +	**1.37018E + 04** =	6.01732E + 06 +	6.01732E + 06 +	6.01732E + 06 +	1.79621E + 04
	Std	3.11815E + 06	5.60699E + 07	2.88322E + 08	9.27884E + 08	1.77311E + 08	1.45669E + 05	1.39950E + 04	5.00154E + 06	5.00154E + 06	5.00154E + 06	1.67414E + 04
F20	Mean	2.61740E + 03 =	2.95155E + 03 +	2.85540E + 03 +	3.36090E + 03 +	3.01291E + 03 +	2.90470E + 03 +	2.55964E + 03 =	2.79345E + 03 +	2.79345E + 03 +	2.79345E + 03 +	**2.53380E + 03**
	Std	1.70285E + 02	9.51565E + 01	1.51773E + 02	2.23367E + 02	1.56273E + 02	1.78102E + 02	1.71953E + 02	1.86044E + 02	1.86044E + 02	1.86044E + 02	9.83940E + 01
F21	Mean	2.66487E + 03 +	2.60274E + 03 +	2.68989E + 03 +	2.78076E + 03 +	2.64500E + 03 +	2.57963E + 03 +	2.50213E + 03 +	2.60121E + 03 +	2.60121E + 03 +	2.60121E + 03 +	**2.46112E + 03**
	Std	1.77809E + 01	3.80543E + 01	4.47733E + 01	5.73054E + 01	3.50833E + 01	5.57246E + 01	3.63302E + 01	3.31329E + 01	3.31329E + 01	3.31329E + 01	1.51753E + 01
F22	Mean	1.00281E + 04 +	4.12884E + 03 +	9.34369E + 03 +	1.05967E + 04 +	7.24908E + 03 +	7.71887E + 03 +	5.97139E + 03 +	8.01859E + 03 +	8.01859E + 03 +	8.01859E + 03 +	**2.30684E + 03**
	Std	2.80898E + 02	1.76727E + 03	1.06070E + 03	9.34504E + 02	1.35969E + 03	1.11373E + 03	1.98979E + 03	2.29258E + 03	2.29258E + 03	2.29258E + 03	2.91016E + 00
F23	Mean	2.98538E + 03 +	3.08350E + 03 +	3.24949E + 03 +	3.52536E + 03 +	3.28863E + 03 +	3.20753E + 03 +	3.04960E + 03 +	3.38270E + 03 +	3.38270E + 03 +	3.38270E + 03 +	**2.81677E + 03**
	Std	2.15172E + 01	7.15392E + 01	1.17433E + 02	1.18982E + 02	9.97961E + 01	1.09235E + 02	7.42184E + 01	1.32190E + 02	1.32190E + 02	1.32190E + 02	2.30261E + 01
F24	Mean	3.12810E + 03 +	3.23774E + 03 +	3.36079E + 03 +	3.75252E + 03 +	3.42898E + 03 +	3.30915E + 03 +	3.22909E + 03 +	3.50349E + 03 +	3.50349E + 03 +	3.50349E + 03 +	**2.99553E + 03**
	Std	1.63601E + 01	6.25966E + 01	1.08223E + 02	3.05978E + 02	9.64201E + 01	1.49691E + 02	1.02101E + 02	1.09534E + 02	1.09534E + 02	1.09534E + 02	1.59109E + 01
F25	Mean	6.44956E + 03 +	3.21801E + 03 +	4.58853E + 03 +	5.81978E + 03 +	4.68373E + 03 +	3.13932E + 03 +	3.01435E + 03 +	3.42294E + 03 +	3.42294E + 03 +	3.42294E + 03 +	**2.92060E + 03**
	Std	7.02498E + 02	1.66317E + 02	3.97635E + 02	5.98211E + 02	4.49232E + 02	8.38895E + 01	6.55820E + 01	2.57612E + 02	2.57612E + 02	2.57612E + 02	1.98504E + 01
F26	Mean	7.59704E + 03 +	8.20497E + 03 +	1.04627E + 04 +	1.20115E + 04 +	9.88676E + 03 +	8.34756E + 03 +	7.08719E + 03 +	9.13721E + 03 +	9.13721E + 03 +	9.13721E + 03 +	**5.05942E + 03**
	Std	2.66632E + 02	5.57170E + 02	1.15810E + 03	1.38149E + 03	9.96125E + 02	1.05051E + 03	1.06798E + 03	1.33842E + 03	1.33842E + 03	1.33842E + 03	6.92903E + 02
F27	Mean	3.28127E + 03 +	3.61604E + 03 +	3.75350E + 03 +	4.43601E + 03 +	3.81356E + 03 +	3.61448E + 03 +	3.35437E + 03 +	3.60172E + 03 +	3.60172E + 03 +	3.60172E + 03 +	**3.22909E + 03**
	Std	1.73580E + 01	1.15832E + 02	2.13584E + 02	3.79886E + 02	1.97007E + 02	2.38617E + 02	8.63762E + 01	1.26730E + 02	1.26730E + 02	1.26730E + 02	1.16740E + 01
F28	Mean	4.79002E + 03 +	4.12941E + 03 +	7.22525E + 03 +	7.62376E + 03 +	6.24416E + 03 +	3.81798E + 03 +	3.39604E + 03 +	4.46045E + 03 +	4.46045E + 03 +	4.46045E + 03 +	**3.29622E + 03**
	Std	7.33323E + 02	7.54155E + 02	6.57580E + 02	8.13174E + 02	6.09101E + 02	2.80336E + 02	7.56762E + 01	3.93627E + 02	3.93627E + 02	3.93627E + 02	3.31875E + 01
F29	Mean	4.35259E + 03 +	5.33292E + 03 +	5.84934E + 03 +	9.07165E + 03 +	5.70236E + 03 +	5.48655E + 03 +	4.44468E + 03 +	5.11798E + 03 +	5.11798E + 03 +	5.11798E + 03 +	**3.90077E + 03**
	Std	3.12285E + 02	3.58245E + 02	7.63446E + 02	2.36368E + 03	5.35102E + 02	5.09398E + 02	3.30588E + 02	3.86912E + 02	3.86912E + 02	3.86912E + 02	1.63996E + 02
F30	Mean	1.06039E + 07 +	1.90957E + 07 +	5.54288E + 08 +	1.76107E + 09 +	3.13723E + 08 +	2.05716E + 06 +	5.72918E + 05 +	3.69307E + 07 +	3.69307E + 07 +	3.69307E + 07 +	**5.38198E + 04**
	Std	3.97650E + 06	2.90568E + 07	2.36122E + 08	1.17306E + 09	1.90244E + 08	3.29324E + 06	5.08913E + 05	3.65753E + 07	3.65753E + 07	3.65753E + 07	2.90378E + 04
+/=/−		28/1/0	29/0/0	29/0/0	29/0/0	29/0/0	29/0/0	21/6/2	28/1/0	29/0/0	28/1/0	−

The meaning of all symbols is the same as Table 3

Table 5 The mean and standard deviation of the optimal solutions obtained from 30 trial runs for all algorithms are reported for the 50-D CEC2020 function functions

FunNum.		DE	PSO	AO	SOA	SFO	WOA	HBA	TSA	BRO	CMA-ES	MBGO
F1	Mean	1.07350E + 11 +	1.90739E + 10 +	9.61449E + 10 +	1.16963E + 11 +	9.80235E + 10 +	2.18213E + 10 +	1.77895E + 10 +	7.25648E + 10 +	7.25648E + 10 +	7.25648E + 10 +	**9.82467E + 05**
	Std	1.04236E + 10	3.75682E + 09	7.32297E + 09	7.34650E + 09	8.54817E + 09	6.85941E + 09	7.13276E + 09	8.91004E + 09	8.91004E + 09	8.91004E + 09	4.11708E + 05
F2	Mean	1.15065E + 13 +	2.32240E + 12 +	1.15292E + 13 +	1.36595E + 13 +	1.14625E + 13 +	2.32273E + 12 +	1.57176E + 12 +	7.44001E + 12 +	7.44001E + 12 +	7.44001E + 12 +	**1.74640E + 08**
	Std	1.38538E + 12	5.45350E + 11	8.32625E + 11	1.08479E + 12	1.20986E + 12	7.67962E + 11	6.09539E + 11	1.00847E + 12	1.00847E + 12	1.00847E + 12	7.33230E + 07
F3	Mean	4.12510E + 12 +	6.79261E + 11 +	4.02543E + 12 +	4.90515E + 12 +	3.84432E + 12 +	7.40475E + 11 +	4.94638E + 11 +	2.33686E + 12 +	2.33686E + 12 +	2.33686E + 12 +	**3.78167E + 07**
	Std	4.60348E + 11	1.98483E + 11	2.87027E + 11	5.21381E + 11	3.35073E + 11	2.57092E + 11	2.13173E + 11	2.58212E + 11	2.58212E + 11	2.58212E + 11	1.56532E + 07
F4	Mean	1.40322E + 06 +	8.91865E + 04 +	2.47406E + 06 +	5.69613E + 06 +	2.20502E + 06 +	3.73527E + 04 +	6.30547E + 03 +	5.89739E + 05 +	5.89739E + 05 +	5.89739E + 05 +	**1.94500E + 03**
	Std	6.32042E + 05	1.30691E + 05	7.57476E + 05	2.26651E + 06	9.67004E + 05	3.36823E + 04	6.95360E + 03	4.25536E + 05	4.25536E + 05	4.25536E + 05	6.19139E + 00
F5	Mean	6.22937E + 07 +	6.63214E + 07 +	3.48756E + 08 +	8.26690E + 08 +	2.39129E + 08 +	1.03318E + 07 +	2.98381E + 06 +	2.90634E + 07 +	2.90634E + 07 +	2.90634E + 07 +	**1.16103E + 06**
	Std	2.17711E + 07	4.06429E + 07	1.59924E + 08	3.97604E + 08	1.40976E + 08	6.98698E + 06	1.30497E + 06	2.12708E + 07	2.12708E + 07	2.12708E + 07	5.29988E + 05
F6	Mean	9.22133E + 07 +	1.03390E + 07 +	2.47489E + 09 +	6.57062E + 09 +	1.50945E + 09 +	8.49051E + 05 +	1.23895E + 05 +	1.12850E + 08 +	1.12850E + 08 +	1.12850E + 08 +	**1.00630E + 04**
	Std	3.22256E + 07	1.46435E + 07	1.73896E + 09	3.01016E + 09	1.02279E + 09	1.00478E + 06	2.30093E + 05	2.17377E + 08	2.17377E + 08	2.17377E + 08	4.75932E + 03
F7	Mean	7.52935E + 08 +	9.15271E + 08 +	7.99620E + 09 +	2.47555E + 10 +	5.82460E + 09 +	3.58485E + 07 +	9.16991E + 06 +	5.94035E + 08 +	5.94035E + 08 +	5.94035E + 08 +	**1.73933E + 06**
	Std	1.80510E + 08	7.64537E + 08	6.41260E + 09	1.29434E + 10	3.21046E + 09	3.21093E + 07	5.64974E + 06	3.52671E + 08	3.52671E + 08	3.52671E + 08	1.12572E + 06
F8	Mean	3.18864E + 03 +	4.13660E + 03 +	8.58746E + 03 +	1.41820E + 04 +	9.13006E + 03 +	9.68433E + 03 +	3.92879E + 03 +	8.19779E + 03 +	8.19779E + 03 +	8.19779E + 03 +	**2.57033E + 03**
	Std	1.06441E + 02	9.35104E + 02	2.55592E + 03	1.79782E + 03	1.59116E + 03	2.30906E + 03	8.93599E + 02	2.42612E + 03	2.42612E + 03	2.42612E + 03	4.37785E + 01
F9	Mean	2.74360E + 04 +	2.30720E + 04 +	6.12287E + 04 +	6.90899E + 04 +	6.27666E + 04 +	3.34944E + 04 +	2.29052E + 04 +	4.89710E + 04 +	4.89710E + 04 +	4.89710E + 04 +	**2.80667E + 03**
	Std	1.90504E + 03	3.36777E + 03	2.50164E + 03	1.89179E + 03	4.11682E + 03	1.23974E + 04	7.12125E + 03	6.11759E + 03	6.11759E + 03	6.11759E + 03	2.16708E + 02
F10	Mean	1.22740E + 04 +	8.43579E + 03 +	2.67242E + 04 +	3.10816E + 04 +	2.16093E + 04 +	5.35655E + 03 +	4.74917E + 03 +	9.82008E + 03 +	9.82008E + 03 +	9.82008E + 03 +	**3.07567E + 03**
	Std	1.78141E + 03	1.87960E + 03	3.56724E + 03	5.67578E + 03	3.30428E + 03	7.47361E + 02	5.60489E + 02	1.40971E + 03	1.40971E + 03	1.40971E + 03	7.75321E + 01
+/=/−		10/0/0	10/0/0	10/0/0	10/0/0	10/0/0	10/0/0	10/0/0	10/0/0	10/0/0	10/0/0	−

The meaning of all symbols is the same as Table 3

Table 6 The mean and standard deviation of the optimal solutions obtained from 30 trial runs for all algorithms are reported for the 100-D CEC2020 function functions

FunNum.		DE	PSO	AO	SOA	SFO	WOA	HBA	TSA	BRO	CMA-ES	MBGO
F1	Mean	3.32895E + 11 +	1.00813E + 11 +	2.44129E + 11 +	2.72268E + 11 +	2.51562E + 11 +	7.69681E + 10 +	6.77815E + 10 +	2.25021E + 11 +	2.25021E + 11 +	2.25021E + 11 +	**8.09800E + 06**
	Std	2.41136E + 10	1.32987E + 10	1.07149E + 10	1.06213E + 10	1.41814E + 10	1.52543E + 10	1.43569E + 10	1.60604E + 10	1.60604E + 10	1.60604E + 10	2.32484E + 06
F2	Mean	3.08224E + 13 +	1.09518E + 13 +	2.85726E + 13 +	3.21282E + 13 +	2.88825E + 13 +	7.57222E + 12 +	7.51861E + 12 +	2.38215E + 13 +	2.38215E + 13 +	2.38215E + 13 +	**8.54600E + 08**
	Std	2.17079E + 12	1.37668E + 12	1.29418E + 12	1.39861E + 12	1.43580E + 12	1.75575E + 12	1.61157E + 12	2.06918E + 12	2.06918E + 12	2.06918E + 12	3.17158E + 08
F3	Mean	1.16587E + 13 +	3.60257E + 12 +	1.00593E + 13 +	1.10461E + 13 +	9.95537E + 12 +	2.80129E + 12 +	2.61868E + 12 +	8.15816E + 12 +	8.15816E + 12 +	8.15816E + 12 +	**2.95853E + 08**
	Std	8.57228E + 11	4.66009E + 11	4.71908E + 11	4.26565E + 11	5.43026E + 11	4.92179E + 11	4.52790E + 11	6.91203E + 11	6.91203E + 11	6.91203E + 11	1.44539E + 08
F4	Mean	2.09535E + 07 +	2.50963E + 05 +	5.13981E + 06 +	9.71884E + 06 +	6.53656E + 06 +	9.67447E + 04 +	3.29336E + 04 +	2.87210E + 06 +	2.87210E + 06 +	2.87210E + 06 +	**2.01267E + 03**
	Std	6.63832E + 06	1.09880E + 05	1.31971E + 06	2.58673E + 06	1.84696E + 06	5.38989E + 04	1.88802E + 04	1.05060E + 06	1.05060E + 06	1.05060E + 06	8.13770E + 00
F5	Mean	7.69953E + 08 +	5.81974E + 08 +	1.78897E + 09 +	2.88891E + 09 +	1.05794E + 09 +	5.29763E + 07 +	3.38563E + 07 +	3.31865E + 08 +	3.31865E + 08 +	3.31865E + 08 +	**1.06397E + 07**
	Std	1.03940E + 08	3.39194E + 08	5.24088E + 08	8.90014E + 08	3.20790E + 08	1.94398E + 07	1.38060E + 07	1.25241E + 08	1.25241E + 08	1.25241E + 08	2.53680E + 06
F6	Mean	3.42971E + 09 +	5.94161E + 08 +	4.17778E + 10 +	6.12336E + 10 +	3.19225E + 10 +	4.06326E + 07 +	3.75763E + 07 +	7.27454E + 09 +	7.27454E + 09 +	7.27454E + 09 +	**1.28503E + 04**
	Std	1.23933E + 09	7.67763E + 08	1.03444E + 10	1.68413E + 10	8.81122E + 09	8.49459E + 07	8.65412E + 07	3.14524E + 09	3.14524E + 09	3.14524E + 09	9.11173E + 03
F7	Mean	5.20767E + 09 +	4.41541E + 09 +	3.43142E + 10 +	4.95521E + 10 +	2.49416E + 10 +	1.47988E + 08 +	5.65228E + 07 +	4.85495E + 09 +	4.85495E + 09 +	4.85495E + 09 +	**6.24767E + 06**
	Std	1.47773E + 09	2.84759E + 09	9.64037E + 09	8.70950E + 09	7.46440E + 09	7.70449E + 07	3.30445E + 07	2.18854E + 09	2.18854E + 09	2.18854E + 09	2.52746E + 06
F8	Mean	4.50899E + 03 +	8.73679E + 03 +	2.64617E + 04 +	2.93162E + 04 +	2.29759E + 04 +	2.29226E + 04 +	1.09807E + 04 +	2.27743E + 04 +	2.27743E + 04 +	2.27743E + 04 +	**3.02900E + 03**
	Std	3.77492E + 02	1.73406E + 03	3.92285E + 03	1.44430E + 03	2.60088E + 03	3.68603E + 03	4.14858E + 03	4.59746E + 03	4.59746E + 03	4.59746E + 03	1.31994E + 02
F9	Mean	1.15764E + 05 +	1.02171E + 05 +	1.74864E + 05 +	1.87164E + 05 +	1.78447E + 05 +	1.17400E + 05 +	9.19353E + 04 +	1.72094E + 05 +	1.72094E + 05 +	1.72094E + 05 +	**3.21067E + 03**
	Std	1.26830E + 04	1.01582E + 04	3.34666E + 03	3.67246E + 03	6.61470E + 03	1.82121E + 04	2.36428E + 04	9.82435E + 03	9.82435E + 03	9.82435E + 03	3.24057E + 02
F10	Mean	4.19285E + 04 +	1.23574E + 04 +	3.73202E + 04 +	4.72404E + 04 +	4.07071E + 04 +	7.79847E + 03 +	6.78342E + 03 +	2.82329E + 04 +	2.82329E + 04 +	2.82329E + 04 +	**3.54133E + 03**
	Std	7.86933E + 03	1.59640E + 03	3.45469E + 03	5.16190E + 03	4.74770E + 03	9.91480E + 02	8.54992E + 02	4.14138E + 03	4.14138E + 03	4.14138E + 03	8.28948E + 01
+/=/−		10/0/0	10/0/0	10/0/0	10/0/0	10/0/0	10/0/0	10/0/0	10/0/0	10/0/0	10/0/0	−

The meaning of all symbols is the same as Table 3

Table 7 The mean and standard deviation of the optimal solutions obtained from 30 trial runs for all algorithms are reported for 10 real-world problems

Pro.		DE	PSO	AO	SOA	SFO	WOA	HBA	TSA	BRO	CMA-ES	MBGO
WBP	Mean	3.01499E + 01 +	3.03531E + 01 +	3.15855E + 01 +	3.22562E + 01 +	3.06198E + 01 +	3.12352E + 01 +	3.01497E + 01 −	**3.01977E + 01** +	3.05453E + 01 +	3.01498E + 01 +	3.01497E + 01
	Std	7.35328E − 05	1.12555E − 01	1.00782E + 00	1.33369E + 00	2.89865E − 01	1.05805E + 00	1.05380E − 06	2.05862E − 02	2.11848E − 01	2.86573E − 05	5.70830E − 06
PVP	Mean	1.74583E − 04 +	1.86911E − 04 +	2.63500E − 04 +	2.07405E − 04 +	1.76123E − 04 +	1.88391E − 04 +	1.74582E − 04 +	1.74583E − 04 +	1.75076E − 04 +	1.74582E − 04 +	**1.74582E − 04**
	Std	2.75355E − 09	3.87818E − 06	5.64297E − 05	5.98014E − 05	4.69676E − 06	3.11705E − 05	9.72542E − 14	6.19153E − 10	2.65755E − 06	2.13985E − 14	7.94876E − 20
TBTD	Mean	6.93877E + 00 +	7.44057E + 00 +	8.83658E + 00 +	9.77919E + 00 +	7.55386E + 00 +	7.53964E + 00 +	6.85269E + 00 +	6.92584E + 00 +	7.71074E + 00 +	6.87159E + 00 +	**6.84308E + 00**
	Std	3.00388E − 02	2.39434E − 01	1.09222E + 00	1.50828E + 00	2.42618E − 01	5.51761E − 01	1.45256E − 02	3.22929E − 02	3.20261E − 01	1.07002E − 02	7.66477E − 05
GTD	Mean	6.19691E + 03 +	1.69210E + 04 +	1.52851E + 04 +	1.18766E + 05 +	1.38133E + 04 +	4.24111E + 04 +	7.06406E + 03 +	7.59020E + 03 +	3.23347E + 04 +	8.33618E + 03 +	**3.11579E + 03**
	Std	2.56268E + 03	2.59300E + 03	3.57643E + 04	1.34297E + 05	3.56148E + 03	6.77324E + 04	2.30139E + 03	1.70092E + 03	2.40874E + 04	1.31885E + 00	8.15415E + 02
CBD	Mean	2.01884E + 00 +	2.14759E + 00 +	1.50628E + 00 +	3.30132E + 00 +	1.45516E + 00 +	3.98741E + 00 +	**1.34066E + 00** −	1.34438E + 00 +	3.17482E + 00 +	1.58063E + 00 +	1.34093E + 00
	Std	2.49656E − 01	2.61245E − 01	5.64065E − 02	1.34402E + 00	4.03176E − 02	1.49161E + 00	5.07010E − 04	2.41575E − 03	7.87785E − 01	9.96286E − 02	4.31190E − 04
IBD	Mean	1.66624E + 02 +	1.60439E + 02 −	1.71192E + 02 +	1.74644E + 02 +	1.65321E + 02 +	1.68256E + 02 +	1.60123E + 02 −	**1.60082E + 02** −	1.65429E + 02 +	1.66624E + 02 +	1.61999E + 02
	Std	6.05475E − 01	1.09560E + 00	4.06967E + 00	7.73054E + 00	2.79042E + 00	2.65397E + 00	2.23849E + 00	9.04569E − 01	2.15546E + 00	6.05480E − 01	2.01311E + 00
TCD	Mean	2.63896E + 02 −	2.64141E + 02 +	2.65614E + 02 +	2.68161E + 02 +	2.64454E + 02 +	2.66980E + 02 +	2.63899E + 02 +	2.63931E + 02 +	2.64332E + 02 +	**2.63896E + 02** −	2.63897E + 02
	Std	1.00980E − 05	1.63423E − 01	2.01084E + 00	3.46406E + 00	5.03549E − 01	3.32020E + 00	3.86866E − 03	2.55176E − 02	2.45614E − 01	1.28869E − 06	6.43328E − 04
PLD	Mean	1.16820E − 10 +	6.58197E − 09 +	3.33230E − 05 +	1.34055E − 07 +	7.24154E − 08 +	0.00000E + 00 −	1.54336E − 12 +	6.37268E − 11 +	3.15766E − 08 +	1.19851E − 10 +	2.07135E − 13
	Std	1.63449E − 10	1.16585E − 08	1.67324E − 04	3.49846E − 07	2.61998E − 07	**0.00000E + 00**	3.52739E − 12	1.15994E − 10	9.32177E − 08	2.15601E − 10	3.90296E − 13
CBHD	Mean	1.75886E + 00 +	2.04007E + 00 +	3.42213E + 00 +	3.73338E + 00 +	2.57787E + 00 +	3.24772E + 00 +	1.74168E + 00 +	1.73101E + 00 +	2.38144E + 00 +	1.71108E + 00 +	**1.70541E + 00**
	Std	1.98198E − 02	1.11802E − 01	7.39125E − 01	9.96940E − 01	2.68773E − 01	7.79000E − 01	9.85994E − 02	2.67926E − 02	3.11510E − 01	8.80543E − 03	1.52208E − 02
RCB	Mean	1.08842E + 00 +	1.03599E + 02 +	2.50336E + 02 +	8.13935E + 02 +	1.37626E + 02 +	3.40021E + 02 +	4.54358E + 01 +	6.99860E + 01 +	2.80743E + 02 +	1.32544E + 00 +	**1.05746E + 00**
	Std	1.96116E − 02	5.10214E + 01	2.41677E + 02	9.40748E + 02	1.29869E + 02	3.07743E + 02	7.35932E + 01	8.13330E + 01	1.01717E + 02	2.16974E − 01	7.08093E − 05
+/≈/−		9/0/1	9/0/1	10/0/0	10/0/0	10/0/0	9/0/1	7/0/3	9/0/1	10/0/0	9/0/1	−

The meaning of all symbols is the same as Table 3

Author contributions YX implemented all the code and completed the first draft. RZ realized the visualization of experimental results and corrected the paper for the first time. CZ and JY designed the optimization framework together. All authors reviewed the manuscript again.

Funding No funding was received for conducting this study.

Data availability No datasets were generated or analysed during the current study.

Declarations

Competing interests The authors declare no competing interests.

References

1. Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M.: Complexity and approximation: combinatorial optimization problems and their approximability properties. Springer, Cham (2012)
2. Ypma, T.J.: Historical development of the newton-raphson method. SIAM Rev. **37**, 531–551 (1995)
3. Chaimovich, M.: New structural approach to integer programming: a survey. Asterisque **258**, 341–362 (1999)
4. Hernando, L., Mendiburu, A., Lozano, J.A.: Hill-climbing algorithm: letś go for a walk before finding the optimum. In: 2018 IEEE Congress on Evolutionary Computation, pp. 1–7. IEEE (2018)
5. Simon, D.: Evolutionary Optimization Algorithms. Wiley, Hoboken (2013)
6. Yao, X., Xu, Y.: Recent advances in evolutionary computation. J. Comput. Sci. Technol. **21**(1), 1–18 (2006)
7. Schoenauer, M., Michalewicz, Z.: Evolutionary computation. Control Cybern. **26**(3), 307–338 (1997)
8. Whitley, D.: A genetic algorithm tutorial. Stat. Comput. **4**(2), 65–85 (1994)
9. Schmitt, L.M.: Theory of genetic algorithms. Theor. Comput. Sci. **259**(1–2), 1–61 (2001)
10. Back, T., Hammel, U., Schwefel, H.-P.: Evolutionary computation: comments on the history and current state. IEEE Trans. Evol. Comput. **1**(1), 3–17 (1997)
11. Li, Y.K., Chen, Y.L., Zhong, J.H., Huang, Z.X.: Niching particle swarm optimization with equilibrium factor for multi-modal optimization. Info. Sci. **494**, 233–246 (2019)
12. Yu, J., Takagi, H., Tan, Y.: Fireworks algorithm for multimodal optimization using a distance-based exclusive strategy. In: 2019 IEEE Congress on Evolutionary Computation, pp. 2215–2220. IEEE (2019)
13. Nedjah, N., Mourelle, L.D.: Evolutionary multi-objective optimisation: a survey. Int. J. Bio-Inspired Comput. **7**(1), 1–25 (2015)
14. Tian, Y., Si, L.C., Zhang, X.Y., Cheng, R., He, C., Tan, K.C., Jin, Y.C.: Evolutionary large-scale multi-objective optimization: a survey. ACM Comput. Surv. **54**(8), 174 (2021)
15. Rohlfshagen, P., Yao, X.: Dynamic combinatorial optimisation problems: an analysis of the subset sum problem. Soft Comput. **2011**, 1723–1734 (2011)
16. Nguyen, T.T., Yang, S.X., Branke, J.: Evolutionary dynamic optimization: a survey of the state of the art. Swarm Evol. Comput. **6**, 1–24 (2012)
17. Jin, Y.C.: Surrogate-assisted evolutionary computation: Recent advances and future challenges. Swarm Evol. Comput. **1**(2), 61–70 (2011)
18. Yu, J., Li, Y., Pei, Y., Takagi, H.: Accelerating evolutionary computation using a convergence point estimated by weighted moving vectors. Complex Intell. Syst. **6**, 55–65 (2019)
19. Zhan, Z.H., Shi, L., Tan, K.C., Zhang, J.: A survey on evolutionary computation for complex continuous optimization. Artif. Intell. Rev. **55**(1), 59–110 (2022)
20. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: 1995 IEEE International Conference on Neural Networks, 4th edn., pp. 1942–1948. IEEE (1995)
21. Storn, R., Price, K.: Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces. J. Global Opt. **11**, 341–359 (1997)
22. Mirjalili, S., Lewis, A.: The whale optimization algorithm. Adv. Eng. Softw. **95**, 51–67 (2016)
23. Hashim, F.A., Houssein, E.H., Hussain, K., Mabrouk, M.S., Al-Atabany, W.: Honey badger algorithm: new metaheuristic algorithm for solving optimization problems. Math. Comput. Simul. **192**, 84–110 (2022)
24. Kaur, S., Awasthi, L.K., Sangal, A.L., Dhiman, G.: Tunicate swarm algorithm: a new bio-inspired based metaheuristic paradigm for global optimization. Eng. Appl. Artif. Intell. **90**, 103541 (2020)
25. Shadravan, S., Naji, H.R., Bardsiri, V.K.: The sailfish optimizer: a novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems. Eng. Appl. Artif. Intell. **80**, 20–34 (2019)
26. Dhiman, G., Kumar, V.: Seagull optimization algorithm: theory and its applications for large-scale industrial engineering problems. Knowl. Based Syst. **165**, 169–196 (2019)
27. Abualigah, L., Yousri, D., Abd Elaziz, M., Ewees, A.A., Al-Qaness, M.A.A., Gandomi, A.H.: Aquila optimizer: a novel meta-heuristic optimization algorithm. Comput. Ind. Eng. **157**, 107252 (2021)
28. Nadimi-Shahraki, M.H., Zamani, H., Zahra, A.V., Mirjalili, S.: A systematic review of the whale optimization algorithm: theoretical foundation, improvements, and hybridizations. Arch. Comput. Methods Eng. **27**, 1–47 (2023)
29. Askarzadeh, Alireza: A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. Comput. Struct. **169**, 1–12 (2016)
30. Zamani, H., Nadimi-Shahraki, M.H.: An evolutionary crow search algorithm equipped with interactive memory mechanism to optimize artificial neural network for disease diagnosis. Biomed. Signal Process. Control **90**, 105879 (2024)
31. Nadimi-Shahraki, M.H., Varzaneh, A.Z., Zamani, H., Mirjalili, S.: Binary starling murmuration optimizer algorithm to select effective features from medical data. Appl. Sci. **13**(1), 564 (2022)
32. Zamani, H., Nadimi-Shahraki, M.H., Gandomi, A.H.: Starling murmuration optimizer: a novel bio-inspired algorithm for global and engineering optimization. Comput. Methods Appl. Mech. Eng. **392**, 114616 (2022)
33. Mirjalili, Seyedali: Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl. Based Syst. **89**, 228–249 (2015)
34. Zamani, H., Nadimi-Shahraki, M.H., Mirjalili, S., Soleimanian, G.F., Oliva, D.: A critical review of moth-flame optimization algorithm and its variants: structural reviewing, performance evaluation, and statistical analysis. Arch. Comput. Methods Eng. **2024**, 1–49 (2024)
35. Nadimi-Shahraki, M.H., Zamani, H., Fatahi, A., Mirjalili, S.: Mfo-sfr: an enhanced moth-flame optimization algorithm using an effective stagnation finding and replacing strategy. Mathematics **11**(4), 862 (2023)
36. Zamani, H., Nadimi-Shahraki, M.H., Gandomi, A.H.: Qana: quantum-based avian navigation optimizer algorithm. Eng. Appl. Artif. Intell. **104**, 104314 (2021)

37. Fatahi, A., Nadimi-Shahraki, M.H., Zamani, H.: An improved binary quantum-based avian navigation optimizer algorithm to select effective feature subset from medical data: a covid-19 case study. J. Bionic Eng. **2023**, 1–21 (2023)
38. Nadimi-Shahraki, M.H., Taghian, S., Mirjalili, S.: An improved grey wolf optimizer for solving engineering problems. Expert Syst. Appl. **166**, 113917 (2021)
39. Nadimi-Shahraki, M.H., Taghian, S., Mirjalili, S., Zamani, H., Bahreininejad, A.: Ggwo: Gaze cues learning-based grey wolf optimizer and its applications for solving engineering problems. J. Comput. Sci. **61**, 101636 (2022)
40. Nadimi-Shahraki, M.H., Taghian, S., Mirjalili, S., Hossam, F.: Mtde: an effective multi-trial vector-based differential evolution algorithm and its applications for engineering design problems. Appl. Soft Comput. **97**, 106761 (2020)
41. Nadimi-Shahraki, M.H., Taghian, S., Zamani, H., Mirjalili, S., Elaziz, M.A.: Mmke: multi-trial vector-based monkey king evolution algorithm and its applications for engineering optimization problems. Plos one **18**(1), e0280006 (2023)
42. Nadimi-Shahraki, M.H., Taghian, S., Mirjalili, S., Abualigah, L.: Binary aquila optimizer for selecting effective features from medical data: a covid-19 case study. Mathematics **10**(11), 1929 (2022)
43. Nadimi-Shahraki, M.H., Moeini, E., Taghian, S., Mirjalili, S.: Discrete improved grey wolf optimizer for community detection. J. Bionic Eng. **20**, 2331–2358 (2023)
44. Grodal, S., Thoma, G.: Cross-pollination in science and technology: concept mobility in the nanobiotechnology field. Ann. Econ. Stat. (2008). https://doi.org/10.15609/annaeconstat2009.115-116.57
45. Bowling, M., Fürnkranz, J., Graepel, T., Musick, R.: Machine learning and games. Mach. Learn. **63**(3), 211–215 (2006)
46. Farshi, T.R.: Battle royale optimization algorithm. Neural Comput. Appl. **33**, 1139–1157 (2020)
47. Ding, Y.: Research on operational model of pubg. In: MATEC Web of Conferences, p. 03062. EDP Sciences (2018)
48. Akan, S., Akan, T.: Battle Royale Optimizer with a New Movement Strategy, pp. 265–279. Springer, New York (2022)
49. Akan, T., Agahian, S., Dehkharghani, R.: Binbro: binary battle royale optimizer algorithm. Expert Syst. Appl. **195**, 116599 (2022)
50. Azizi, M., Shishehgarkhaneh, M.B., Moehle, R.C.: Squid game optimizer (sgo): a novel metaheuristic algorithm. Sci. Rep. **13**, 5373 (2023)
51. Fabricatore, C.: Gameplay and game mechanics: a key to quality in videogames. (2007)
52. Pierce, W.D., Cheney, C.D.: Behavior Analysis and Learning: A Biobehavioral Approach. Routledge, London (2017)
53. Choi, G., Kim, M.: Gameplay of battle royale game by rules and actions of play. In: 2018 IEEE 7th Global Conference on Consumer Electronics (GCCE), pp. 599–600. IEEE, New York (2018)
54. Choi, G., Kim, M.: Battle royale game: in search of a new game genre. Int. J. Cult. Technol. (IJCT) **2**(2), 5 (2018)
55. Awad, N.H., Ali, M.Z., Suganthan, P.N., Liang, J.J., Qu, B.Y.: Problem definitions and evaluation criteria for the cec 2017 special session and competition on single objective real-parameter numerical optimization. (2017)
56. Yue, C.T., Price, K. V., Suganthan, P. N., Liang, J. J., Ali, M. Z., Qu, B. Y., Awad, N. H., Biswas, Partha P.: Problem definitions and evaluation criteria for the cec 2020 special session and competition on single objective bound constrained numerical optimization. (2020)
57. Coello, C.A.: Use of a self-adaptive penalty approach for engineering optimization problems. Comput. Ind. **41**(2), 113–127 (2000)
58. Sandgren, E.: Nonlinear integer and discrete programming in mechanical design optimization. (1990)
59. Ravindran, A., Ragsdell, K.M., Reklaitis, G.V.: Methods and Applications. Wiley, Hoboken (2006)
60. Amir, H.M., Hasegawa, T.: Nonlinear mixed-discrete structural optimization. J. Struct. Eng. **115**(3), 626–646 (1989)

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH (“Springer Nature”).
Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users (“Users”), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use (“Terms”). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.
These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.
We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.
While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.
These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.
To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.
Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.
If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com