Contemporary Mathematics
https://ojs.wiserpub.com/index.php/CM/

Research Article

Empowered Cat and Mouse Optimizer for Exploring Quasi Optimal Solutions in Optimization Problems

Nursyiva Irsalinda[1,2], Danang A. Pratama[1], Maharani A. Bakar[1,3*], Fatimah Noor Harun[1,3], Sugiyato Surono[2]

[1] Faculty of Computer Science and Mathematics, Universiti Malaysia Terengganu, Terengganu, Malaysia
[2] Mathematics Study Program, Universitas Ahmad Dahlan, Bantul, Indonesia
[3] Special Interest Group Modelling and Data Analytics, Universiti Malaysia Terengganu, Terengganu, Malaysia
E-mail: maharani@umt.edu.my

Received: 25 November 2024; **Revised:** 26 December 2024; **Accepted:** 31 December 2024

Abstract: Population-based optimization algorithms are essential for solving a wide range of optimization problems encountered in various scientific fields and real-world applications. This study introduces a modified version of the Cat and Mouse Based Optimizer (CMBO) known as Empowered CMBO (E-CMBO), aimed at global optimization. By considering the population proportions of mice and cats and incorporating additional formulas for mouse movements to evade cat attacks, this adjustment significantly reduces the execution time of the CMBO algorithm. Experimental results demonstrate the effectiveness of E-CMBO in addressing diverse optimization problems, particularly those posed by unimodal and high-dimensional multimodal objective functions. E-CMBO consistently outperforms CMBO by producing fitness values closer to the minimum and achieving faster convergence rates across different scenarios. While it may exhibit slightly reduced efficacy in fixed-dimensional multimodal functions in one scenario compared to CMBO, overall, E-CMBO demonstrates robust performance across various problem landscapes. The advantages of E-CMBO lie in its simplicity, high robustness, fast convergence, and fewer parameters. This study highlights E-CMBO as a promising optimization tool, offering significant advantages in convergence speed and solution quality, particularly for complex optimization problems.

Keywords: cat and mouse based optimizer, global optimization, metaheuristics, quasi optimal solution, swarm optimization

MSC: 65K10, 68T20, 90C26, 90C59, 68W50

1. Introduction

Optimization is a crucial aspect of research that plays an essential role in achieving desired goals. The goals could be maximization of profit/efficiency or minimization of cost/resources utilized as the global optimal solution from the available resources within a set of constraints [1]. A general constrained optimization is formulated as follows [2]:

$$\text{Minimize} \quad F(x) = f_1(x),\, f_2(x),\, \ldots,\, f_m(x), \tag{1}$$

$$c_i(x) \leq 0, \quad i = 1, 2, \ldots, p, \tag{2}$$

$$c_j(x) = 0, \quad j = 1, 2, \ldots, q, \tag{3}$$

$$x_l^i \leq x^i \leq x_u^i, \quad i = 1, 2, \ldots, N. \tag{4}$$

where $x = (x_1, x_2, \ldots, x_n)^T$ represents the candidate solution comprising n decision variables, $F(x)$ denotes the objective function, while $c_i(x) \leq 0$ and $c_j(x) = 0$ are the constrained function. Equation (1) simplifies to a single-objective problems if $m = 1$ and escalates to multi-objective problems if $m \geq 2$. However, it is crucial to recognize that the optimal solutions are hardly attained by the optimization techniques. Consequently, solutions derived from the optimizations are termed as quasi-optimal, indicating their comparative or close to the global optimum [3]. Therefore, when assessing the efficacy of various optimization algorithms, the superiority of an algorithm is determined by its ability to generate a quasi-optimal solution.

From a broad perspective, optimization techniques are categorized into two main groups. There are deterministic methods, which rely on fixed data and precise models, and stochastic methods, which account for randomness and uncertainty in data [4]. Metaheuristics represent a diverse category of optimization methods, including both stochastic approaches and deterministic techniques. These methods are designed to efficiently explore large and complex solution spaces, often balancing exploration and exploitation to find near-optimal solutions for challenging optimization problems. Metaheuristics derived form term *meta* means "beyond" or "higher level" and *heuristic* denotes "to find" or "to discover by trial and error" [5]. Generally, these algorithms represent a class of methodologies that iterate through evaluations of the objective function [6]. These algorithms provide numerous advantages, such as conceptual simplicity, straightforward implementation, independence from specific problem types, effectiveness in addressing non-linear, non-convex, non-differentiable, discontinuous, high-dimensional, and NP-hard problems, and efficiency in exploring uncharted search spaces [7]. Notably, the success of an optimization algorithm in addressing one problem does not necessarily guarantee its adequacy for another. Hence, researchers have developed numerous optimization algorithms with the goal of achieving quasi-optimal solutions deemed more acceptable and closer to the global optimum [8].

Over last few decades, a diverse concepts have been employed in the development of metaheuristic algorithms that draw inspiration from diverse domains such as natural processes, swarm behavior, human-based principles, physics, and evolutionary approaches [9–12]. The first breakthrough of metaheuristic optimization occurred in the 1960s with the pioneering development of the evolutionary Genetic Algorithm (GA) by John Holland and his colleagues at the University of Michigan [13]. The evolution-based has continued to evolve, with researchers developing more sophisticated algorithms in this fields including Evolutionary Algorithm [14], Differential Evolution (DE) [15, 16], Evolution Strategies [17], Genetic Programming [18, 19], etc. The category of swarm-based metaheuristic algorithms, exemplified by Particle Swarm Optimization (PSO) [20], Artificial Bee Colony (ABC) [21], Chicken Swarm Optimization [22], and Whales Optimization Algorithm [23], demonstrates the implementation of these principles. Moreover, metaheuristics grounded in human-based principles include Teaching-Learning-Based Optimization (TLBO) [24], and Mother Optimization Algorithm [25]. Additionally, Electromagnetic Field Optimization [26] represents an instance of a physics-based metaheuristic. For a more detailed concept of the categorization of metaheuristic optimization concepts, the various existing methods are illustrated in Figure 1.

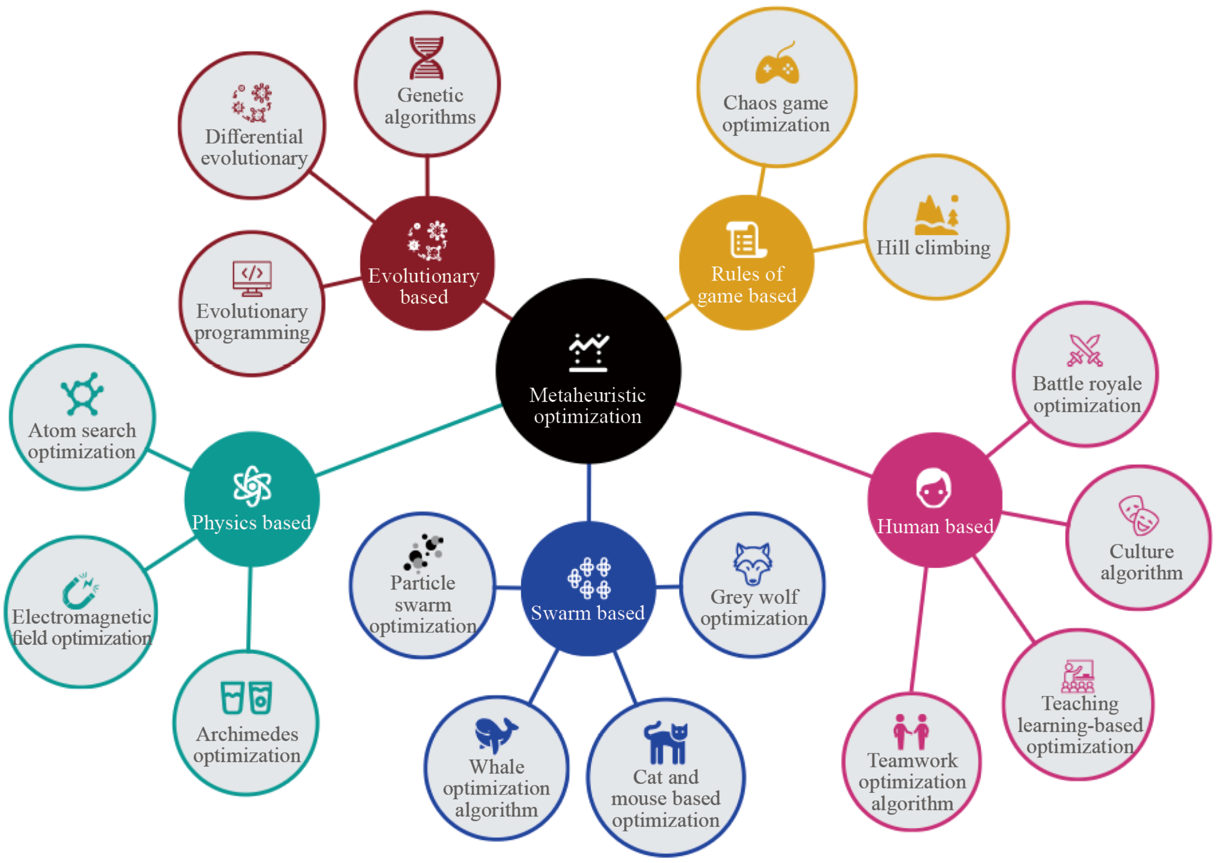

Metaheuristic optimization
Evolutionary based
Differential evolutionary
Genetic algorithms
Evolutionary programming
Rules of game based
Chaos game optimization
Hill climbing
Human based
Battle royale optimization
Culture algorithm
Teaching learning-based optimization
Teamwork optimization algorithm
Swarm based
Particle swarm optimization
Grey wolf optimization
Whale optimization algorithm
Cat and mouse based optimization
Physics based
Atom search optimization
Electromagnetic field optimization
Archimedes optimization

Figure 1. Metaheuristics optimization category

Furthermore, there is fundamental principles in metaheuristic algorithm that say no optimization algorithm is better than any other algorithm; each one is best-suited for a particular class of optimization problems. This principles also called *No Free Lunch Theorem* [27]. However, there are still numerous ongoing developments in the creation of new metaheuristic methods, tailored to address specific challenges encountered in various problem domains.

Metaheuristic algorithms, either in general or particularly within swarm intelligence, are based on two core search behaviors there are exploration and exploitation [28]. Exploration involves exploring uncharted areas within the feasible space, while exploitation focuses on searching the vicinity of a promising region [29, 30]. The classification of search operators and strategies within a metaheuristic method is often ambiguous, as these elements can contribute in various ways to the exploration or exploitation of search space [31]. Hence, achieving an optimal balance between exploration and exploitation is the key with which researchers are widely contending to improve the performance of metaheuristic search methods [28, 32–34].

Numerous efforts have been made to attain the optimal balance between exploration and exploitation. These include refining existing methods, hybridizing two or more available methods, or developing entirely new methodologies. Improving the existing method is as simple as introducing a new adaptive strategy in the adaptive human learning optimization algorithm [35]. Then, enhancing the ABC algorithm performance by using a Student's-t distribution sampling technique [36], modified chaotic dragonfly method [37], improved genetic algorithm [38], improved Particle Swarm Optimization [39], and many more modified algorithms are available in literature. Furthermore, the idea of hybridizing the existing metaheuristics is well-established with numerous hybrid of metaheuristics can be found in the literature, such as Hybrid Artificial Neural Network-Gravitational Search Algorithm [40], Hybrid TLBO (HTLBO) that integrates the TLBO algorithm with other methods such as Quadratic Approximation (QA) [41] or Harmony Search [42] to improve both global and local search capabilities, and hybridizes Cuckoo Search (CS) and Biogeography-based

Optimization (BBO) by employing two search strategies called Biogeography-based Heterogeneous Cuckoo Search algorithm [43].

In contrast, developing of entirely new methodologies is a challenging task since we have to ensure that the new methodology is genuinely novel or not just a minor variation of existing algorithms. Moreover, significant issues such as escaping from local optima, divergence probability, handling complex constraints, and computational difficulties in computing derivatives pose challenges to new methodologies [44].

In 2021, Dehghani et al. introduced a novel metaheuristic that successfully maintains a balance between the exploration and exploitation phases [45]. The algorithm they proposed adopts the hunting behavior of cats in pursuing mice called Cat and Mouse Based Optimization (CMBO). The algorithm's concept almost similar to the Cat Swarm Optimizer (CSO) introduced by Chu et al. [46]. Both CMBO and CSO is inspired by the observed behaviors of cats and their interactions with their environment or prey. However, the execution process of the two algorithms differs significantly. In CMBO, each population is assigned an equal number of individuals, representing exploration during the mouse phase and exploitation during the cat phase. The developed algorithm yields highly effective and superior results compared to another population-based metaheuristics. Despite its acceptable performance across various problems, CMBO has certain limitations that could be addressed for improvement. One of them is about the limited exploitation abilities that causing CMBO may face challenges in intensively searching around promising areas, which is crucial for fine-tuning solutions to reach the global optimum. Therefore, this paper purposes a modification of CMBO that has ability to empowering the exploitation phase called Empowered CMBO (E-CMBO). This modification incorporates a new formula that governs the movement of each cat towards another cat's position to explore alternative food sources beyond mice. This process will significantly exploit the wider potential of new solutions.

The key contributions of this study include the development of an enhanced E-CMBO algorithm to refine the exploitation phase of the original CMBO algorithm, the evaluation of its performance using recently developed benchmark functions commonly employed in optimization research, and a comparative analysis of the E-CMBO's capabilities against other well-known metaheuristics through experimental testing on these benchmark functions

The paper is structured as follows: Section 2 presents an overview of the CMBO. Section 3 details the E-CMBO, which serves as an enhancement to CMBO. Section 4 provides experimental results, highlighting the performance of E-CMBO in comparison to other metaheuristic algorithms. Section 5 discusses the findings, and Section 6 concludes the paper with final remarks.

2. Cat and mouse based optimizer

Different from Cat Swarm optimizer that mostly adopt the behavior of cats in the different hunting modes, CMBO is designed around the natural dynamics of a cat pursuing a mouse and the mouse's attempts to escape for safety. Within this algorithm, search agents are categorized into two groups: cats and mice, each navigating the problem search space through random movements. The algorithm updates population members in two distinct phases. The initial phase models the movements of cats toward mice, while the subsequent phase simulates the mice's escape to refuge for preserving their lives.

From mathematical point of view, each member of the population is a proposed solution of the problem. In fact, a member of the population gives values to the problem variables according to its position in the search space. Thus, each member of the population is a vector whose values determine the variables of the problem. The population of the algorithm is determined using a matrix which is called the population matrix as in equation (5).

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,m} \\ x_{2,1} & x_{2,2} & \cdots & x_{1,m} \\ \vdots & \cdots & \ddots & \cdots \\ x_{N,1} & x_{N,2} & \cdots & x_{N,m} \end{bmatrix}, \tag{5}$$

where X is the population matrix of CMBO, $x_{i,\,d}$ is the i-th search agent for the d-th problem variable, N is the number of population members, and m is the number of problem variables. As mentioned earlier, each member of the population determines the proposed values for the problem variables. Therefore, a value for the objective function is determined for each member of the population. The value of objective function are denoted by a vector as in equation (6).

$$F = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ \vdots \\ F_N \end{bmatrix}, \tag{6}$$

where F is the vector of the objective function values and F_i is the objective function value for the i-th search agent. Based on the values obtained for the objective functions, the members of the population are ordered from the best member with the lowest objective function value, to the worst member of the population with the highest objective function value. In the sorted population matrix, it is assumed that half of the population members which gave better values (minimum) form the mouse population and the other half of the population members form the cat population. To update the search factors, the first phase models the position change of cats based on natural behavior of cats and their movements towards mice. This phase of updating the proposed CMBO is modeled mathematically with equation (7).

$$c_{j,\,d}^{new} = c_{j,\,d} + r(m_{k,\,d} - Ic_{j,\,d}), \tag{7}$$

where c and m represent cat and mice, respectively, $j = 1,\, 2,\, \ldots,\, N_c$, $d = 1,\, 2,\, \ldots,\, c$, $k = 1,\, 2,\, \ldots,\, N_m$, and N_c and N_m denote the number of cats and mice. Moreover, r is a uniformly distributed random value, and $I = round(1 + r)$. The selection of the best cat candidates is using the equation (8).

$$c_j = \begin{cases} c_j^{new} & F(c_j^{new}) \le F(c_j^{old}) \\ c_j^{old} & F(c_j^{new}) \ge F(c_j^{old}) \end{cases}. \tag{8}$$

In the second phase of CMBO, the escape of mice into refuges is modeled. It is assumed that there is a random refuge for each mouse, and the mice escape to these refuges. The position of the refuges in the search space is created randomly by patterning the positions of the different members of the algorithm. This phase of updating the position of the mice is modeled mathematically by equation (9).

$$m_{i,\,d}^{new} = m_{i,\,d} + r(r_{i,\,d} - Im_{i,\,d})sign(F(m_{i,\,d}) - F(r_{i,\,d})), \tag{9}$$

with $r_{i,\,d}$ are the refuges in i-th particle and in d-th problem variable, $i = 1,\, 2,\, \ldots,\, N_m$ number of mice, and $d = 1,\, 2,\, \ldots,$ c. The selection of the best mice candidates to be included in the population is made using the equation (10).

$$m_i = \begin{cases} m_i^{new} & F(m_i^{new}) \le F(m_i^{old}) \\ m_i^{old} & F(m_j^{new}) \ge F(m_i^{old}) \end{cases}, \tag{10}$$

here $F(m_{i,d})$ is the haven for the i-th mouse, $F(r_{i,d})$ is its objective function, m_i^{new} is the new status of i-th mice, and $F(m_i^{new})$ its objective function value.

All members of the population are updated by using equation (5)-(10), the iterative process is repeated until the stopping condition is reached. The stopping condition for optimization algorithms can be a certain number of iterations, or definition of an acceptable error between the obtained solutions in successive iterations. The procedure above is summarized in Algorithm 1.

```
**Algorithm 1** The CMBO Algorithm
**Input:** The objective function and the constraints problem.
**Data:** Number of search agents (NS), number of variables (MS), and iterations (T).
Generate an initial population matrix with $((NS) \times (MS))$ dimension by using equation (5);
Evaluate the objective function and sort as in equation (6);
  **for** $t = 1$ **to** $T$ **do**
  Sort population matrix based on objective function value;
  Select population of mice and cats;
  **Phase 1:** Update positions of cats;
  **for** $j = 1$ **to** $N/2$ **do**
    Update the status of the $t$-th cat using equation (7) and equation (8);
  **Phase 2:** Update positions of mice;
  **for** $j = (N/2) + 1$ **to** $NS$ **do**
    Update the status of the $t$-th mouse using equation (9) and equation (10);
Output the best quasi-optimal solution obtained with the CMBO.
```

3. Empowered the cat and mouse based optimizer

Initially, we investigated the impact of mice and cats populations on CMBO performance. The proportion ratio is defined as *mice* : *cats*, where the mice proportion increased from 10% to 90%, while the cats proportion decreased from 90% to 10%. The completed proportion allocation can be seen on Table 1. We then implemented a unimodal function F_1 as a benchmark function in this investigation afterwards. The F_1 has the form as follows:

$$F_1(x) = \sum_{i=1}^{m} x_i^2. \tag{11}$$

The value of x is randomly generated and constrained within the range $[-100, 100]$, with a dimension $m = 30$. The simulation runs for 100 iterations and repeatedly 100 times. The experiments were conducted on a machine with an Intel Core i5-8250U CPU and running Windows 10. The statistical metrics, including minimum, maximum, standard deviation, and execution time values, obtained from the investigation are recorded and evaluated to observe the differences of each population ratio. In this section, the execution time values represent the total execution time for 100 repetitions, ensuring a comprehensive understanding of the computational performance. Table 1 displays the results of the investigation using benchmark F_1 function.

Table 1 shows that as the proportion of mice increases, the minimum values also increase. This trend highlights that a higher proportion of cats relative to mice yields more favorable results. Additionally, this also leads to consistency, as indicated by the standard deviation exhibiting the similar result. On the other hand, the algorithm's execution time demonstrates that a smaller proportion of mice leads to a shorter search time. This investigation led to the conclusion that a smaller proportion of mice generates better results and faster execution time. The visualization of the tested population variations results are clearly visualized in Figure 2.

Table 1. Population proportion variation for the unimodal test function

Variation	Proportion of mice : cat (%)	Minimum	Mean	Maximum	Standard deviation	Execution time (s)
1	10 : 90	2.93e-27	2.76e-26	1.91e-25	2.91e-26	30.49
2	20 : 80	2.67e-24	2.46e-23	9.80e-23	1.82e-23	41.86
3	30 : 70	2.05e-22	1.51e-21	5.42e-21	1.01e-21	54.49
4	40 : 60	5.30e-21	6.57e-20	4.63e-19	6.77e-20	66.05
5	50 : 50	2.24e-19	1.62e-18	6.18e-18	1.19e-18	77.19
6	60 : 40	4.11e-18	3.10e-17	1.26e-16	2.33e-17	93.61
7	70 : 30	3.64e-17	8.16e-16	1.14e-14	1.18e-15	101.89
8	80 : 20	2.84e-14	3.20e-15	2.01e-13	3.24e-14	113.12
9	90 : 10	1.18e-13	7.63e-13	2.80e-12	5.95e-13	125.45

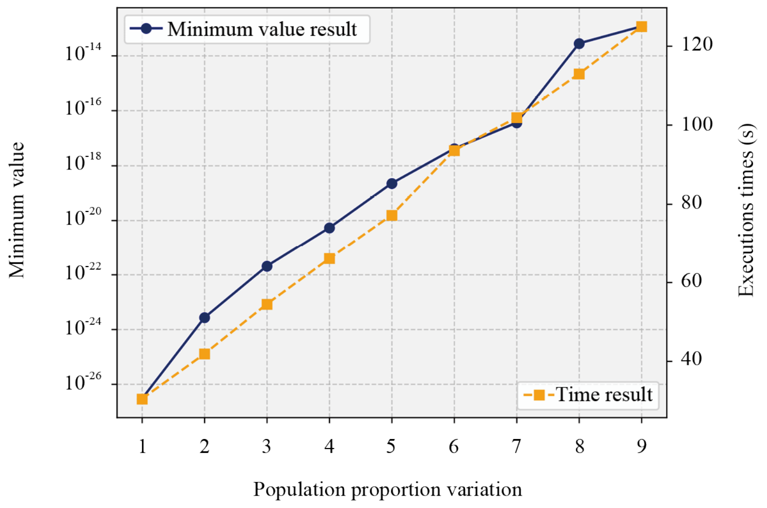


Figure 2. Comparison of the minimum value and execution time of each population proportion

In this section, we introduce an innovative method proposed with the aim of enhancing CMBO based on the analysis of its objective functions. Notably, balanced proportion ratio (50 : 50), as applied in the original CMBO, is not universally useful to all types of objective functions addressed. Based on the prior investigation, it was found that the ratio $mice$: $cats$ yields the optimum value if mice proportion is smaller than cats proportion. The idea is anticipated to optimize the CMBO strategy through an effective combination of exploration and exploitation, ultimately leading to the discovery of the optimal solutions.

In addition to the smaller proportion of mice in the entire population, the movement of cats incorporates an additional formula. This formula aligns with the reality on the ground, besides stray cats chasing some mice on the streets, they also engage an aggressive interactions to chase off the intruder to mark their territories in the vicinity. This new formula dictates how each cat moves towards another cat, demonstrating dominance as a higher-ranked individuals. The additional update movement by cats is formulated in equation (12).

$$c_{j,d}^{new*} = c_{j,d} + r(c_{j,d} - c_{k,d}), \tag{12}$$

where $j = 1, \ldots, N_c$, $k = 1, 2, \ldots, N_m$, that $j \neq k$, and r is a uniformly distributed random value. The selection of the best cat candidates to be included in the population is expressed in the equation (13).

$$c_j = argmin\left\{F(c_j^{new}),\, F(c_j^{new*}),\, F(c_j)\right\}. \tag{13}$$

Following the process in CMBO as normally, hence, the movement of mice is determined by equation (9) and equation (10). From the explained description, the E-CMBO proposed by the researcher can be streamlined through the following Algorithm 2.

```
Algorithm 2 E-CMBO Algorithm
Input: The objective function and the constraints problem.
Data: Number of search agents ($N$), number of variables ($M$), and iterations ($T$).
Generate an initial population matrix with (($N$) $\times$ ($M$)) dimension by using equation (5);
Evaluate the objective function and sort as in equation (6);
  for $t = 1$ to $T$ do
  Sort population matrix based on objective function value;
  Divide the population into 10% of mice and 90% of cats;
  Phase 1: Update the positions of cats;
  for $j = 1$ to $N/2$ do
    Update the $t$-th cat using equation (7), equation (12), and equation (13);
  Phase 2: Update the positions of mice;
  for $j = (N/2) + 1$ to $N$ do
    Update the $t$-th mouse using equation (9) and equation (10);
Output best quasi-optimal solution obtained.
```

4. Experimental results

In this section, the proposed E-CMBO, is applied to address 23 benchmark functions. These functions consist of 7 unimodal functions, 6 high-dimensional multimodal functions, and 10 fixed-dimensional multimodal functions, as outlined in Tables 2, 3, and 4. To evaluate the performance of E-CMBO, we compared its minimum function values with those obtained from CMBO and other prominent metaheuristic algorithms, including PSO, GWO, TLBO, and DE. PSO and GWO were selected due to their popularity and strong performance as population-based metaheuristics [47]. TLBO was chosen to represent effective human-based metaheuristics [48], while DE was included as a leading evolutionary-based metaheuristic [49]. This selection ensures a comprehensive comparison of E-CMBO against various established metaheuristic approaches.

All computations are conducted under uniform conditions, including the number of individuals in the population, iterations, and trial repetitions using Mealpy an open-source library meta-heuristic algorithms in Python [50].

Experimental conditions have been standardized, however certain comparative algorithms demand specific parameters sourced from prior research studies. These parameters serve as the main distinguishing factors between Dehgani's CMBO algorithm and other algorithms. The key advantage of CMBO lies in its simplicity, as it does not require parameter tuning. This characteristic enables its direct application to various problems without necessitating further research to identify optimal parameters. Similarly, E-CMBO, an extension of CMBO, also does not require parameter tuning. Consequently, both CMBO and E-CMBO provide ease of use and high flexibility in addressing diverse problems without the need for parameter tuning. The parameters used for the comparative algorithms based on [50] are provided in Table 5.

Table 2. Unimodal test functions

Objective function	Range	Dimensions	$F_{\min}$
$F_1(x) = \sum_{i=1}^{m} x_i^2$	$[-100, 100]$	30	0
$F_2(x) = \sum_{i=1}^{m} \sum_j x_{ij} + \prod_{i=1}^{m} \prod_j x_{ij}$	$[-10, 10]$	30	0
$F_3(x) = \sum_{i=1}^{m} \left(\sum_{j=1}^{i} x_i\right)^2$	$[-100, 100]$	30	0
$F_4(x) = \max \lvert x_i \rvert,\ 1 \le i \le m$	$[-100, 100]$	30	0
$F_5(x) = \sum_{i=1}^{m-1} \left(100(x_{i+1}^2 - x_i^2)^2 + (x_i - 1)^2\right)$	$[-30, 30]$	30	0
$F_6(x) = \sum_{i=1}^{m} [x_i + 0.5]^2$	$[-100, 100]$	30	0
$F_7(x) = \sum_{i=1}^{m} i \cdot x_i^4 + \text{random}(0, 1)$	$[-1.28, 1.28]$	30	0

Table 3. High-dimensional multimodal test functions

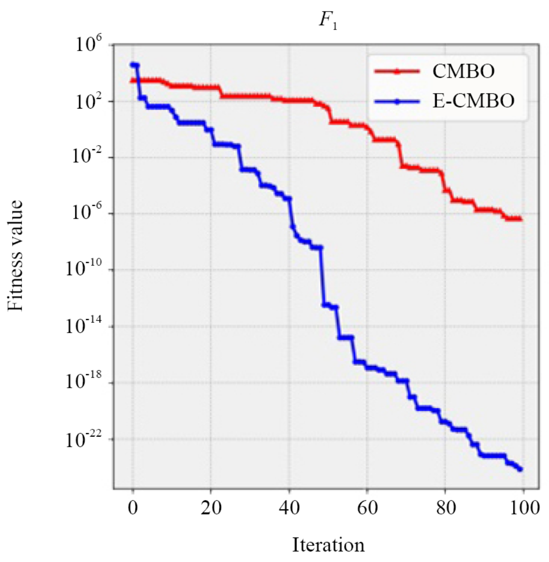
Objective function	Range	Dimensions	$F_{\min}$
$F_8(x) = \sum_{i=1}^{m} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	$[-500, 500]$	30	$-12{,}569$
$F_9(x) = \sum_{i=1}^{m} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12, 5.12]$	30	0
$F_{10}(x) = 20\exp\left(0.2\sqrt{\frac{1}{m}\sum_{i=1}^{m} x_i^2}\right) - \exp\left(\frac{1}{m}\sum_{i=1}^{m} \cos(2\pi x_i)\right) + 20 + e$	$[-32, 32]$	30	0
$F_{11}(x) = \frac{1}{4{,}000}\sum_{i=1}^{m} x_i^2 - \prod_{i=1}^{m} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]$	30	0
$F_{12}(x) = \frac{\pi}{m}\left\{10\sin(\pi y_1) + \sum_{i=1}^{m} (y_{i-1})^2 \left[1 + 10\sin^2(\pi y_i + 1)\right] + (y_m - 1)^2\right\} + \sum_{i=1}^{m} u(x_i, 10, 100, 4)$ $u(x_i, a, k, n) = \begin{cases} k(x_i - a)^n, & \text{if } x_i > a \\ 0, & \text{if } -a < x_i < a \\ k(-x_i - a)^n, & \text{if } x_i < -a \end{cases}$	$[-50, 50]$	30	0
$F_{13}(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{m} (x_i - 1)^2 \left[1 + \sin^2(3\pi x_i + 1)\right] + (x_m - 1)^2 \left[1 + \sin^2(2\pi x_m)\right]\right\} + \sum_{i=1}^{m} u(x_i, 5, 100, 4)$	$[-50, 50]$	30	0

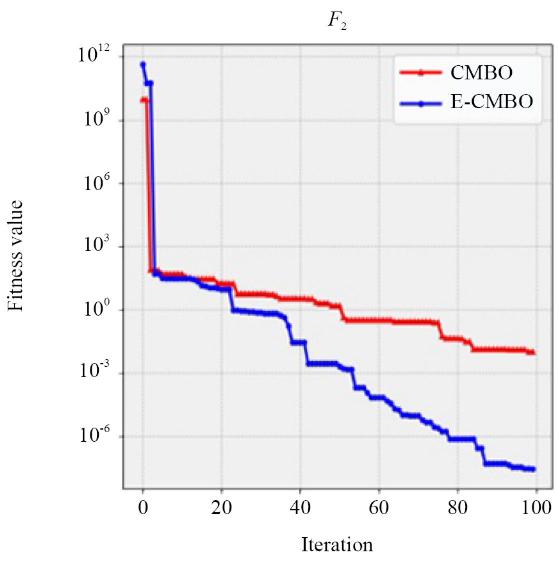
Table 4. Fixed-dimensional multimodal test functions

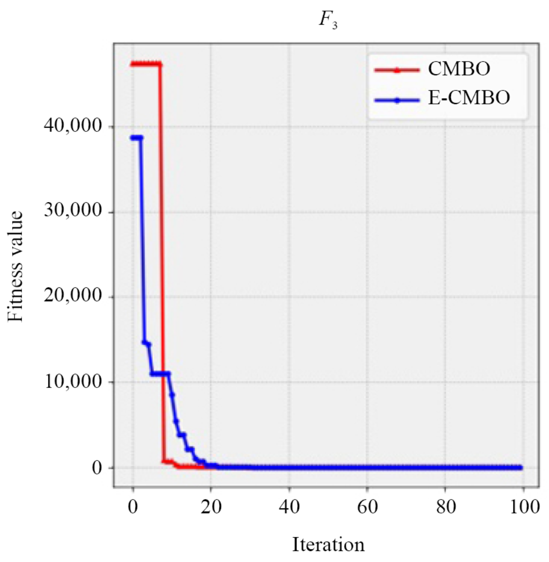
Objective function	Range	Dimensions	$F_{\min}$
$F_{14}(x)=\left(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})}\right)^{-1}$	$[-65.53, 65.53]$	2	0.998
$F_{15}(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right]^2$	$[-5, 5]$	4	0.00030
$F_{16}(x)=4x_1^2+2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	$[-5, 5]$	2	-1.0316
$F_{17}(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{\pi}\right)\cos x_1+10$	$[-5, 10]\times[0, 15]$	2	0.398
$F_{18}(x)=\left[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)\right]\times \left[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)\right]$	$[-5, 5]$	2	3
$F_{19}(x)=-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{3}a_{ij}(x_j-P_{ij})^2\right)$	$[0, 1]$	3	-3.68
$F_{20}(x)=-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{6}a_{ij}(x_j-P_{ij})^2\right)$	$[0, 1]$	6	-3.22
$F_{21}(x)=-\sum_{i=1}^{5}\left[(X-a_i)(X-a_i)^T+6c_i\right]^{-1}$	$[0, 10]$	4	-10.1532
$F_{22}(x)=-\sum_{i=1}^{7}\left[(X-a_i)(X-a_i)^T+6c_i\right]^{-1}$	$[0, 10]$	4	-10.4029
$F_{23}(x)=-\sum_{i=1}^{10}\left[(X-a_i)(X-a_i)^T+6c_i\right]^{-1}$	$[0, 10]$	4	-10.5364

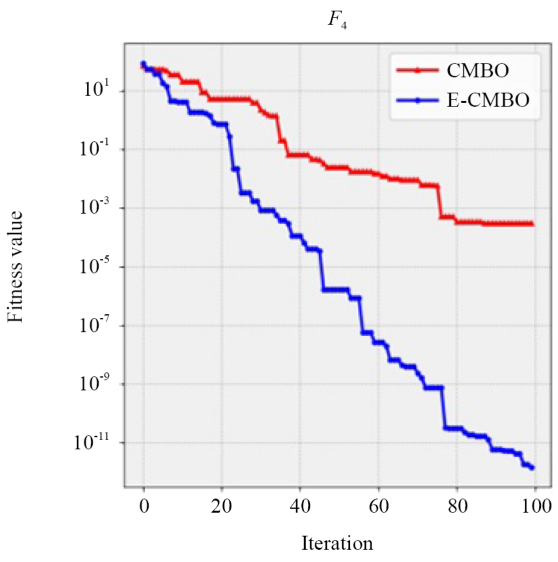
Table 5. Parameter values for various algorithms

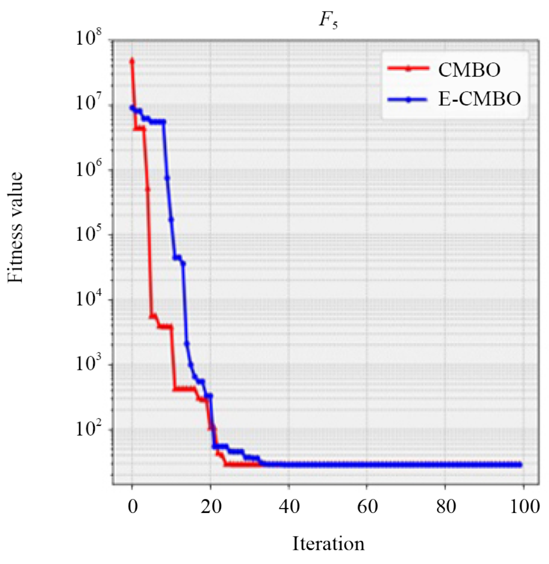
Algorithm	Parameter	Value
PSO	Cognitive coefficient (c_1)	2.05
	Social coefficient (c_2)	0.5
	Inertia weight (w)	0.4
GWO	Convergence parameter (a)	Decreases linearly from 2 to 0
TLBO	Teacher factor (TF)	[1, 2]
DE	Scaling factor (μ_f)	0.5
	Crossover probability (μ_{cr})	0.5
	Probability of updating trial vector components (p_t)	0.1
	Probability of updating scaling factor F (A_p)	0.1

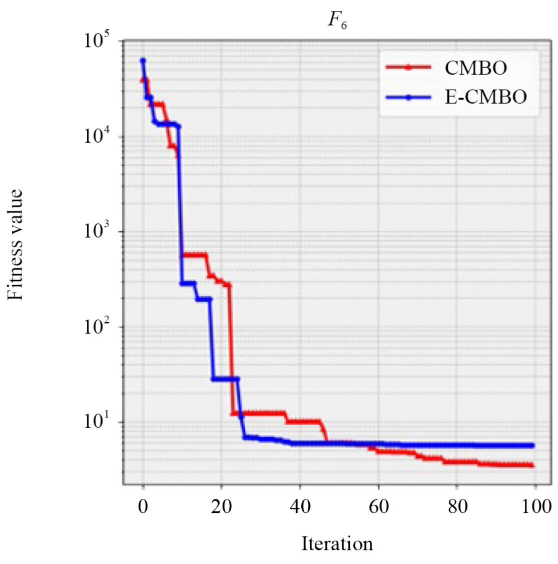

F_1
CMBO
E-CMBO
Fitness value
Iteration
10^6
10^2
10^{-2}
10^{-6}
10^{-10}
10^{-14}
10^{-18}
10^{-22}
0
20
40
60
80
100


F_2
CMBO
E-CMBO
Fitness value
Iteration
10^{12}
10^9
10^6
10^3
10^0
10^{-3}
10^{-6}
0
20
40
60
80
100


F_3
CMBO
E-CMBO
Fitness value
Iteration
40,000
30,000
20,000
10,000
0
0
20
40
60
80
100


F_4
CMBO
E-CMBO
Fitness value
Iteration
10^1
10^{-1}
10^{-3}
10^{-5}
10^{-7}
10^{-9}
10^{-11}
0
20
40
60
80
100


F_5
CMBO
E-CMBO
Fitness value
Iteration
10^8
10^7
10^6
10^5
10^4
10^3
10^2
0
20
40
60
80
100


F_6
CMBO
E-CMBO
Fitness value
Iteration
10^5
10^4
10^3
10^2
10^1
0
20
40
60
80
100

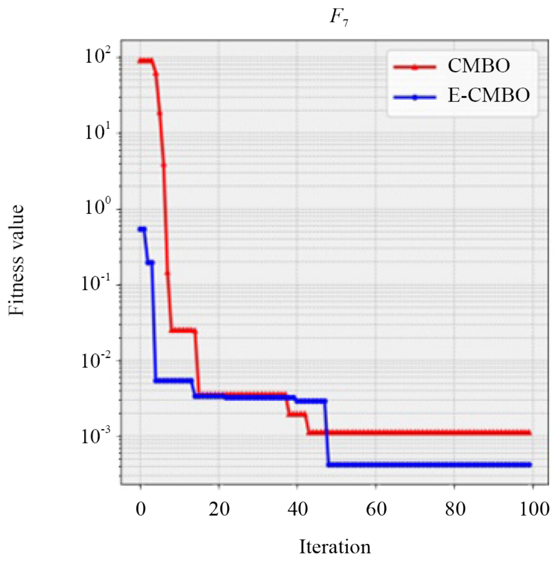


Figure 3. Comparison of fitness function values obtained by CMBO and E-CMBO on unimodal test functions

4.1 *Comparative analysis of empowered CMBO in relation to classical CMBO*

This subsection presents an analysis of the performance of E-CMBO across twenty-three distinct test functions, juxtaposed with CMBO for comparison. Performance assessment entails the examination of fitness function values obtained over one hundred iterations, utilizing a population size of thirty individuals. The comparative fitness function values of both methods are illustrated graphically in Figure 3 for unimodal test functions, Figure 4 for high-dimensional multimodal test functions, and Figure 5 for fixed-dimensional multimodal test functions.

As illustrated in Table 2, it is evident that the minimum objective function value is universally zero. Consequently, the graphs in Figure 3 depict that E-CMBO generates minimum values closer to the objective function minimum and exhibits superior convergence speed compared to CMBO across most unimodal test functions, except for F_6. This indicates that the E-CMBO algorithm significantly outperforms CMBO in handling 85.71% of unimodal test functions.

In the results of the simulation for solving high-dimensional multimodal function tests, it is clear from Figure 4 that the E-CMBO method reaches the minimum value faster than the CMBO. This indicates that E-CMBO has a higher convergence speed in finding optimal solutions for most of the functions tested. However, for the F_{13} function, E-CMBO has not been able to surpass CMBO in achieving the desired optimum value. This suggests that while E-CMBO is generally more efficient, there are certain cases where CMBO still outperforms in finding the optimal solution. Nevertheless, when considering the overall perspective, it can be observed that the convergence rate of the E-CMBO method is significantly better than CMBO. This indicates that E-CMBO is capable of achieving faster and more efficient convergence in seeking optimal solutions within complex search spaces. Overall, from the tests conducted, it can be concluded that E-CMBO demonstrates better capability in optimizing solutions for high-dimensional multimodal functions with fewer iterations compared to CMBO. This advantage is crucial in practical applications where time efficiency and computational resources are key factors for success.

In the context of fixed-dimensional multimodal optimization, the E-CMBO method demonstrates consistent performance in seeking minimum values, specifically fitness function values, across nearly all provided test functions. Notably, it exhibits comparable performance to CMBO in almost all test functions, indicating its effectiveness in traversing the solution space efficiently. However, there is a single test function, F_{15}, where E-CMBO does not outperform CMBO (see Figure 5).

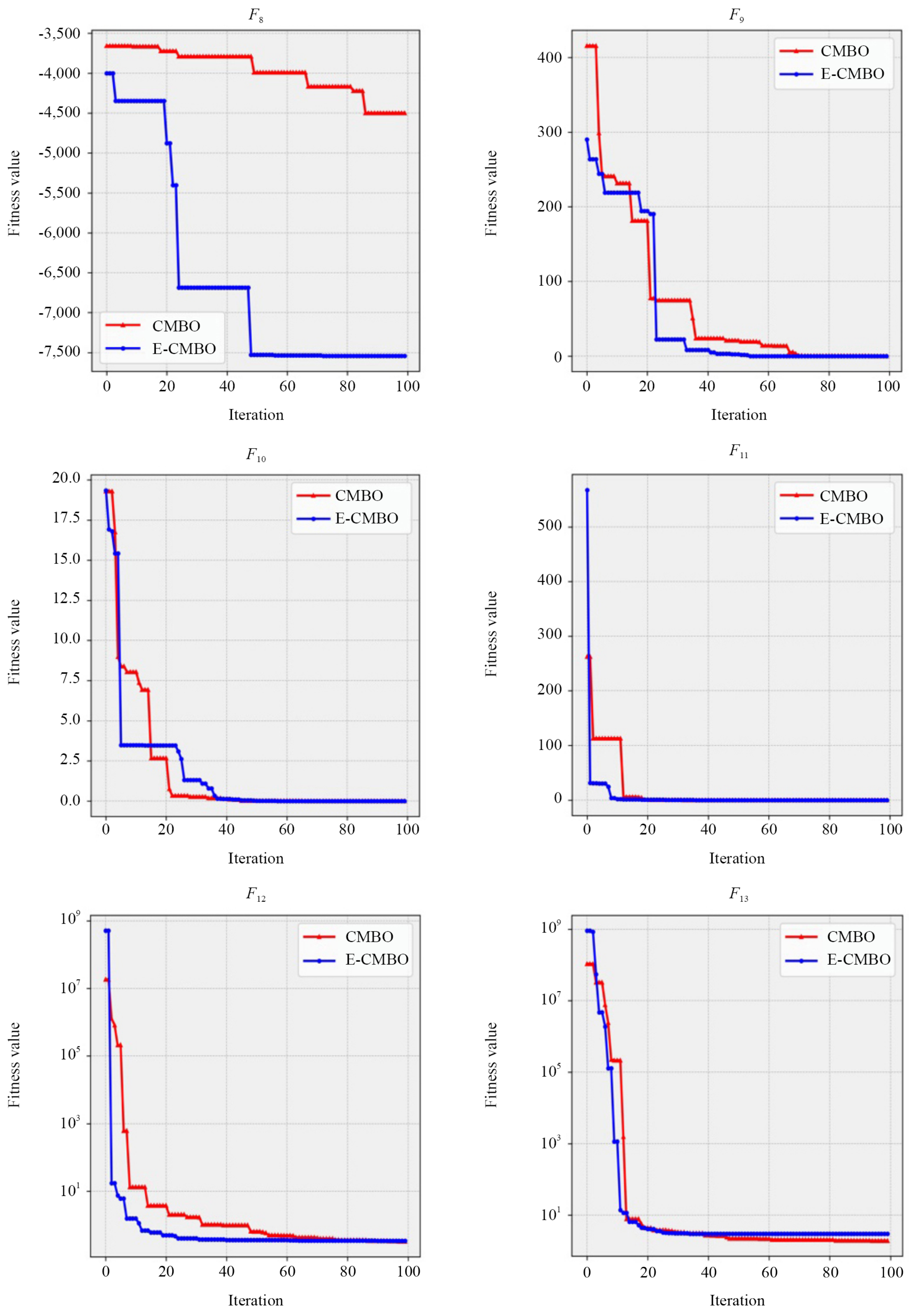
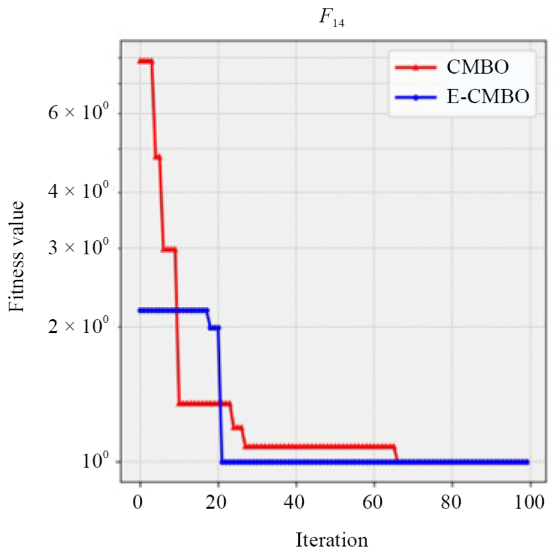
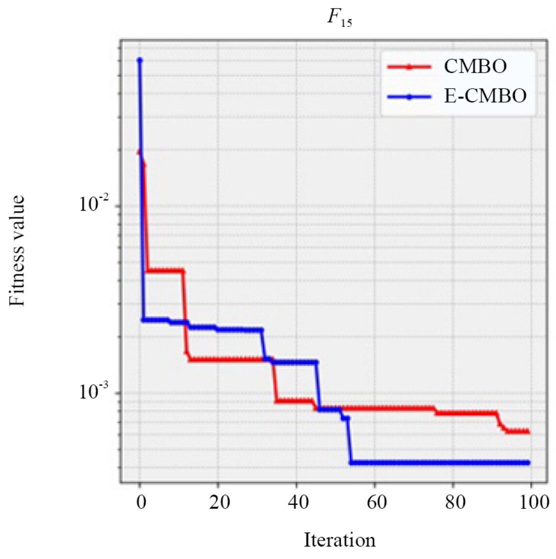


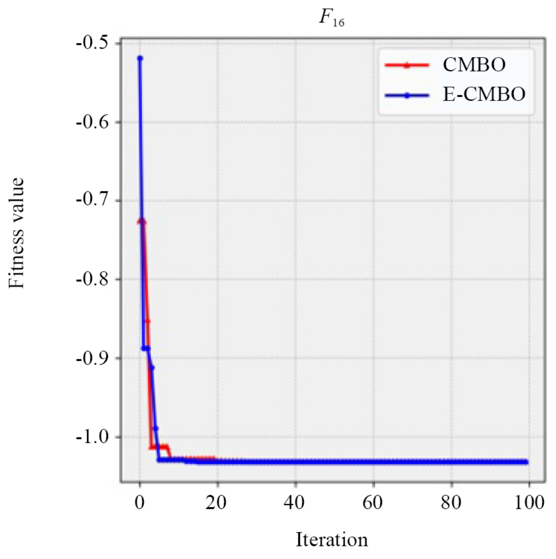
Figure 4. Comparison of fitness function values obtained by CMBO and E-CMBO on high-dimensional multimodal test functions



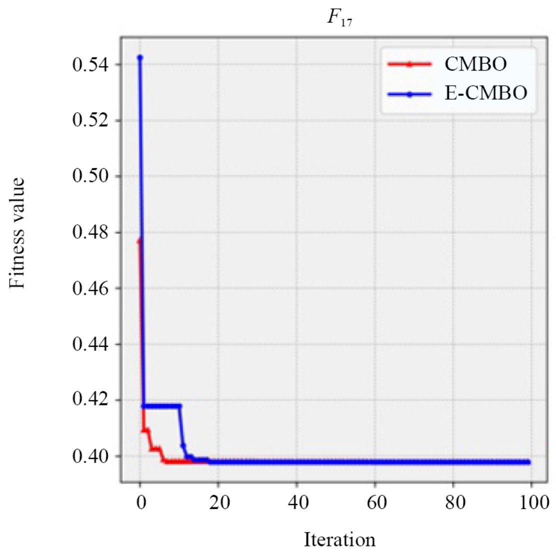
F_{14}
CMBO
E-CMBO
Fitness value
Iteration



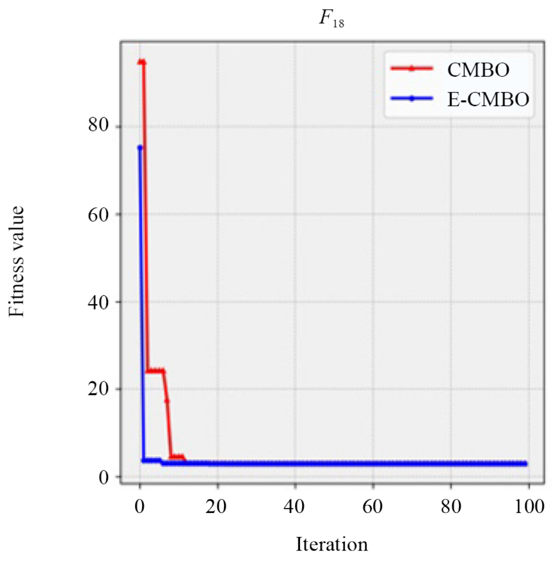
F_{15}
CMBO
E-CMBO
Fitness value
Iteration



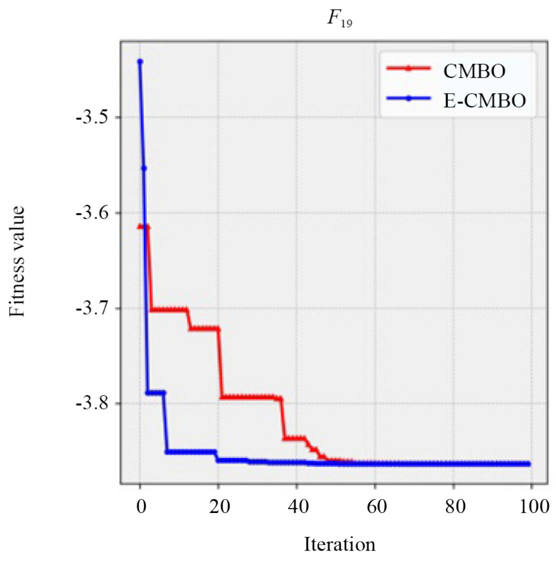
F_{16}
CMBO
E-CMBO
Fitness value
Iteration



F_{17}
CMBO
E-CMBO
Fitness value
Iteration



F_{18}
CMBO
E-CMBO
Fitness value
Iteration



F_{19}
CMBO
E-CMBO
Fitness value
Iteration

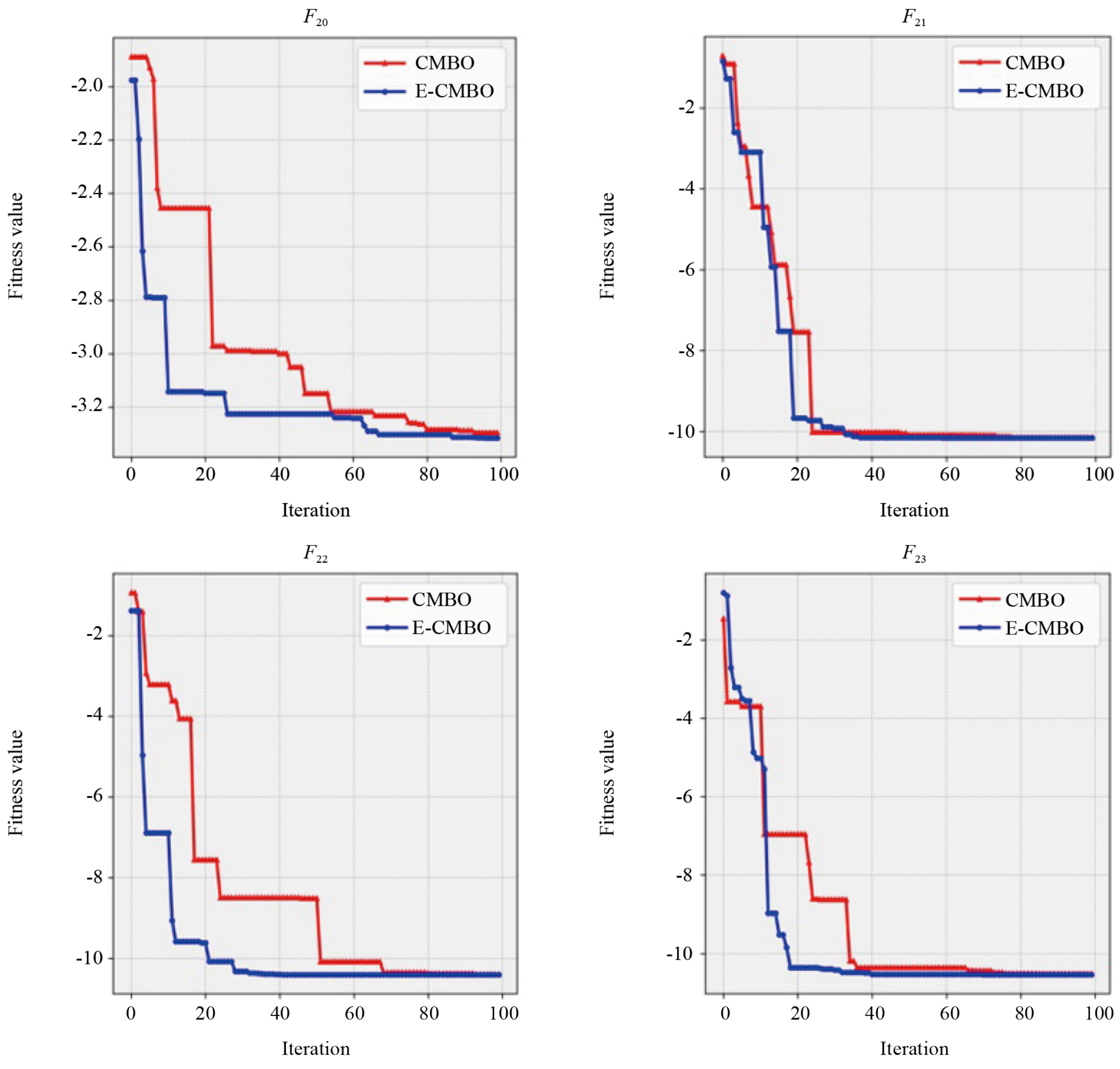


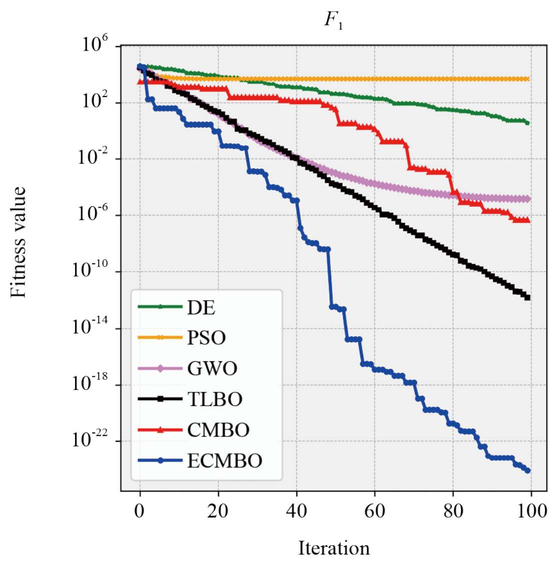
Figure 5. Comparison of fitness function values obtained by CMBO and E-CMBO on fixed-dimensional multimodal test functions

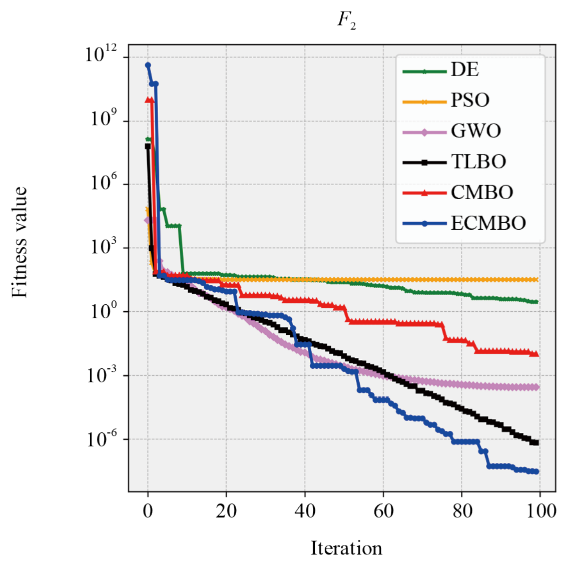
Based on the conducted analysis, E-CMBO appears to be a highly promising solution for addressing challenges in handling unimodal and high-dimensional multimodal objective functions. Besides being able to produce fitness function values close to the minimum and demonstrating superior convergence compared to CMBO, E-CMBO also exhibits more efficient execution times. However, when applied to fixed-dimensional multimodal objective functions, E-CMBO's performance is relatively lower compared to CMBO. Nevertheless, despite both methods yielding fitness function values that are almost similar, the execution time of E-CMBO tends to be longer than CMBO. This study has the potential to be expanded to delve into the distinctive characteristics of various types of objective functions, thus enhancing our understanding of optimization algorithm behaviors across diverse problem landscapes.

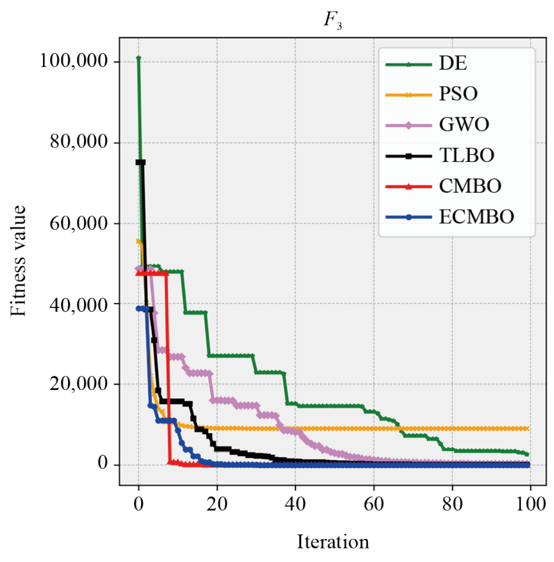
This observation underscores the reliability and competitiveness of the E-CMBO approach in addressing fixed multimodal optimization challenges. Although it may not always universally outperform CMBO in all contexts, the performance of E-CMBO remains prominent and robust across various test scenarios. Such insights contribute to the understanding and evaluation of optimization algorithm in tackling real-world optimization problems that tend to have diverse characteristics and complexities.

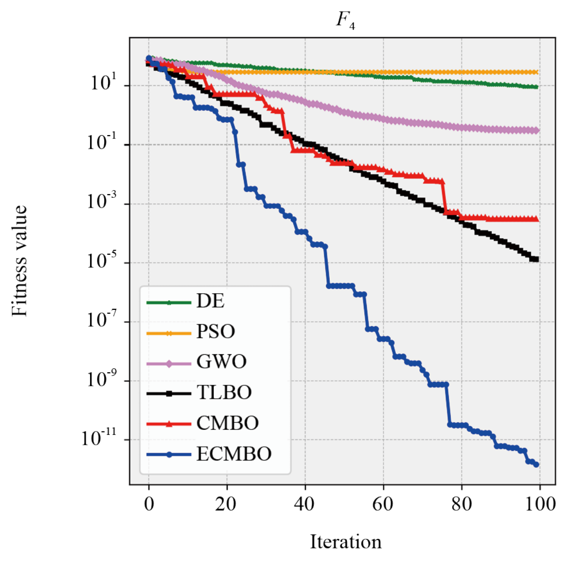
Table 6. Comparison of methods for unimodal functions

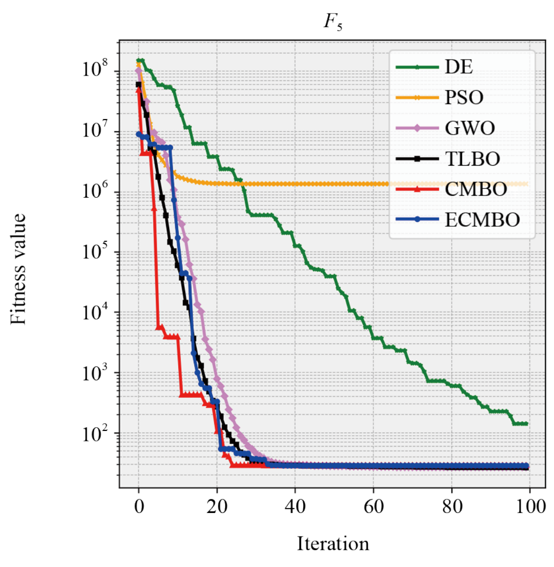
Function	F_{min}	Fitness values	E-CMBO	CMBO	PSO	GWO	TLBO	DE
F_1	0	Min	7.8532E-25	6.9119E-10	3.5240E + 03	1.8099E-06	6.5810E-13	2.2963E + 00
		Max	2.8601E-16	1.0219E-04	7.1443E + 03	3.8263E-05	1.9648E-12	4.8682E + 00
		Mean	4.6281E-17	1.0630E-05	5.3819E + 03	1.1559E-05	1.1294E-12	3.6433E + 00
		Std	9.5050E-17	3.0529E-05	1.0641E + 03	1.1392E-05	4.3945E-13	8.4647E-01
		Time/Iter (sec)	0.03	0.017	0.023	0.016	0.037	0.038
F_2	0	Min	1.6654E-11	5.2933E-07	2.8679E + 01	2.4466E-04	4.2341E-07	2.0490E + 00
		Max	2.8578E-08	1.7789E-02	4.6234E + 01	1.0171E-03	1.0135E-06	3.8155E + 00
		Mean	3.2166E-09	3.0903E-03	3.5320E + 01	5.0999E-04	7.2331E-07	2.8794E + 00
		Std	8.4922E-09	5.7238E-03	5.5545E + 00	2.2702E-04	1.7261E-07	5.6933E-01
		Time/Iter (sec)	0.032	0.016	0.026	0.017	0.038	0.037
F_3	0	Min	7.7823E-27	1.1649E-08	5.9016E + 03	1.0652E + 01	1.8572E + 00	1.4795E + 03
		Max	9.0940E-14	1.4789E-02	1.5722E + 04	6.7184E + 01	6.6660E + 00	3.8402E + 03
		Mean	1.4206E-14	1.5072E-03	1.1030E + 04	3.3419E + 01	3.3824E + 00	2.7050E + 03
		Std	2.7486E-14	4.4277E-03	3.3842E + 03	1.6763E + 01	1.6473E + 00	6.0657E + 02
		Time/Iter (sec)	0.068	0.041	0.031	0.031	0.067	0.053
F_4	0	Min	1.4621E-12	2.7243E-06	2.7314E + 01	2.1057E-01	1.2905E-05	4.5175E + 00
		Max	5.1303E-08	1.9569E-03	3.3791E + 01	1.2571E + 00	2.6512E-05	8.9696E + 00
		Mean	1.2615E-08	5.8731E-04	2.9954E + 01	4.9202E-01	1.6489E-05	6.3704E + 00
		Std	1.7855E-08	6.1297E-04	2.1709E + 00	2.8348E-01	3.7885E-06	1.2623E + 00
		Time/Iter (sec)	0.031	0.015	0.025	0.016	0.037	0.038
F_5	0	Min	2.8815E + 01	2.8807E + 01	1.0111E + 06	2.5855E + 01	2.6205E + 01	1.4228E + 02
		Max	2.8938E + 01	2.8932E + 01	3.2073E + 06	2.8918E + 01	2.7155E + 01	2.5778E + 02
		Mean	2.8880E + 01	2.8885E + 01	1.9233E + 06	2.7524E + 01	2.6778E + 01	1.9286E + 02
		Std	3.2670E-02	3.2670E-02	2.3622E + 05	3.2670E-02	2.3622E + 05	3.2670E-02
		Time/Iter (sec)	0.021	0.015	0.035	0.016	0.027	0.028
F_6	0	Min	3.8912E-16	2.1289E-09	4.9032E + 03	1.2093E-04	7.1213E-11	9.4128E + 00
		Max	1.2437E-14	1.5608E-02	1.2458E + 04	2.9631E + 02	1.4079E-05	7.9103E + 01
		Mean	3.4782E-15	2.5678E-03	7.6034E + 03	1.2049E + 01	5.4275E-07	2.3811E + 01
		Std	7.1310E-15	6.0345E-03	2.6703E + 03	5.6071E + 00	1.3295E-05	8.0237E + 00
		Time/Iter (sec)	0.042	0.038	0.045	0.032	0.051	0.056
F_7	0	Min	1.2976E-10	2.0334E-06	9.4251E + 01	1.4237E-02	2.5369E-07	3.2565E + 02
		Max	3.7462E-09	7.8421E-03	1.6319E + 02	6.2895E-01	1.4798E-04	8.1472E + 02
		Mean	1.0983E-09	3.2461E-04	1.0172E + 02	2.1457E-02	4.8713E-06	4.5238E + 02
		Std	2.5632E-09	8.1773E-04	5.3154E + 01	1.1274E-01	9.8743E-06	1.9759E + 02
		Time/Iter (sec)	0.058	0.033	0.065	0.027	0.049	0.072

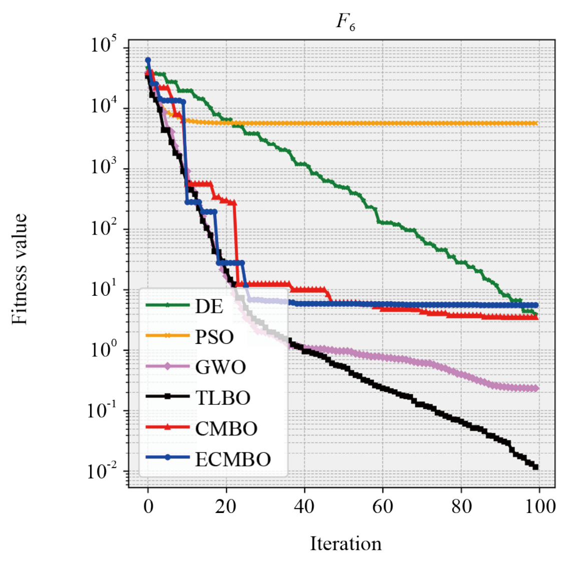

F_1
Fitness value
Iteration
DE
PSO
GWO
TLBO
CMBO
ECMBO


F_2
Fitness value
Iteration
DE
PSO
GWO
TLBO
CMBO
ECMBO


F_3
Fitness value
Iteration
DE
PSO
GWO
TLBO
CMBO
ECMBO


F_4
Fitness value
Iteration
DE
PSO
GWO
TLBO
CMBO
ECMBO


F_5
Fitness value
Iteration
DE
PSO
GWO
TLBO
CMBO
ECMBO


F_6
Fitness value
Iteration
DE
PSO
GWO
TLBO
CMBO
ECMBO

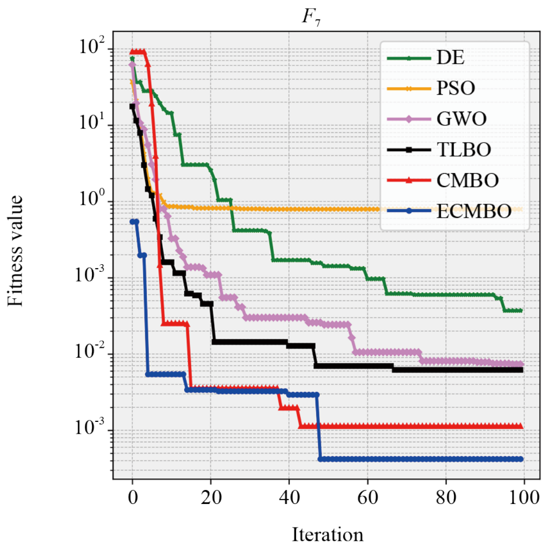


Figure 6. Comparison of fitness function for different algorithms on unimodal test function

4.2 *The analysis of E-CMBO compared to other metaheuristics*

After evaluating the performance of E-CMBO compared to the original CMBO, this subsection presents a comprehensive performance comparison among E-CMBO, CMBO, PSO, GWO, TLBO, and DE. The evaluations were conducted under uniform conditions, with each algorithm tested using 100 individuals per population and 100 iterations per execution. The parameters required for each comparative method are specified in Table 5. In this experimental setup, performance was assessed using several key metrics: the best fitness function value, denoted as "Min"; the worst fitness function value, denoted as "Max"; and the standard deviation, labeled as "Std", which indicates the convergence rate of each method. Additionally, the average time per iteration in seconds for each method, referred to as "Time/Iter (sec)", was recorded. These metrics provide a comprehensive overview of each algorithm's efficiency and effectiveness in the given scenario.

To facilitate analysis, the overall experimental results are organized and reviewed based on the type of function used in the testing. The results of experiments using unimodal functions are presented in Table 6 and Figure 6, providing detailed insights into the performance of the tested methods in the context of simple functions with a single global minimum.

Based on Table 2, it is known that functions F_1 to F_7 have a minimum function value of 0. Therefore, the best method is the one that achieves a fitness function value closest to zero. From the results presented in Table 6, it can be seen that for functions F_1, F_2, F_3, F_4, and F_7, the E-CMBO method performs exceptionally well by producing fitness function values closest to zero compared to other methods. This demonstrates E-CMBO's superiority in optimizing these unimodal functions. However, the scenario is different for functions F_5 and F_6, where E-CMBO does not show the best performance. For these two functions, the TLBO method manages to achieve fitness function values closer to zero than E-CMBO, indicating a superior performance for F_5 and F_6. Nonetheless, E-CMBO still performs better than other methods for these functions. Overall, E-CMBO can be considered a superior method for the majority of the tested functions, although there are specific cases where the TLBO method outperforms it. Figure 6 illustrates the reduction in loss function values for each method across all functions. It can be observed that the reduction in fitness function values is significantly achieved by E-CMBO within a shorter number of iterations.

The experimental results for high-dimensional multimodal functions, characterized by numerous local minima and requiring advanced search strategies, are presented in Table 7 and Figure 7. These tables and figures illustrate how each method handles the added complexity of higher dimensions.

Table 7. Comparison of methods for high dimensional multimodal functions

Function	F_{min}	Fitness values	E-CMBO	CMBO	PSO	GWO	TLBO	DE
F_8	-12,569	Min	-9,153.1	-8,401.4	-7,512.9	-8,571.1	-5,578.6	-6,516
		Max	-4,374.5	-4,311.9	-4,601.5	-3,375.9	-4,012.3	-5,575.7
		Mean	-6,946.5	-5,739.2	-5,364.1	-5,960.7	-4,718.5	-6,133.7
		Std	1,404.2	1,428.3	8,063.1	1,759.6	4,576.7	3,282.3
		Time/Iter (sec)	0.026	0.017	0.023	0.023	0.047	0.049
F_9	0	Min	0.0000E + 00	1.3895E-09	5.6300E + 01	1.9065E + 01	1.3203E + 01	1.4190E + 02
		Max	2.6785E-09	6.1934E-01	1.3887E + 02	1.3912E + 02	1.2983E + 02	1.6858E + 02
		Mean	3.3534E-10	2.6332E-02	9.3391E + 01	5.1385E + 01	7.1660E + 01	1.5855E + 02
		Std	8.0654E-13	5.2987E-02	2.8073E + 01	3.4499E + 01	3.9416E + 01	7.4288E + 00
		Time/Iter (sec)	0.028	0.016	0.029	0.021	0.045	0.048
F_{10}	0	Min	2.8821E-13	1.1622E-05	1.1590E + 01	3.8056E-04	2.3801E-07	1.0707E + 00
		Max	7.8984E-08	4.9970E-02	1.4712E + 01	1.1992E-03	3.9411E-07	1.8409E + 00
		Mean	9.2971E-09	1.0099E-02	1.3182E + 01	7.1130E-04	3.2571E-07	1.4838E + 00
		Std	2.3325E-11	1.7016E-02	9.6153E-01	2.1455E-04	5.1225E-08	2.2162E-01
		Time/Iter (sec)	0.053	0.027	0.029	0.024	0.049	0.053
F_{11}	0	Min	0	5.0071E-14	1.9746E + 01	2.2949E-06	1.1940E-12	1.0099E + 00
		Max	1.5784E-11	5.1991E-05	6.7608E + 01	4.8943E-02	1.5341E-11	1.0511E + 00
		Mean	1.5858E-12	9.7815E-06	4.4281E + 01	1.0506E-02	4.2078E-12	1.0336E + 00
		Std	4.7326E-15	1.6200E-05	1.5319E + 01	1.5393E-02	3.8338E-12	1.1337E-02
		Time/Iter (sec)	0.046	0.027	0.032	0.024	0.047	0.049
F_{12}	0	Min	1.5997E-01	2.2906E-01	6.1170E + 01	7.1937E-03	1.7335E-04	1.1948E-01
		Max	4.3336E-01	4.8794E-01	7.4590E + 05	1.5012E-01	6.7595E-04	4.8301E-01
		Mean	3.3084E-01	3.1698E-01	2.2735E + 05	4.0617E-02	3.5103E-04	2.5603E-01
		Std	9.1465E-05	7.6945E-02	2.7586E + 05	3.8609E-02	1.4751E-04	9.9190E-02
		Time/Iter (sec)	0.063	0.044	0.029	0.031	0.062	0.054
F_{13}	0	Min	8.9761E-01	1.6615E + 00	9.8495E + 04	1.2886E-01	2.1491E-02	3.2843E-01
		Max	2.9931E + 00	2.2918E + 00	1.0045E + 07	1.6365E + 00	2.0017E-01	9.0678E-01
		Mean	2.2438E + 00	2.0345E + 00	3.4128E + 06	6.4567E-01	6.6615E-02	6.0542E-01
		Std	5.7840E-04	1.7730E-01	3.3815E + 06	3.9904E-01	4.9293E-02	1.9145E-01
		Time/Iter (sec)	0.057	0.038	0.027	0.028	0.060	0.052

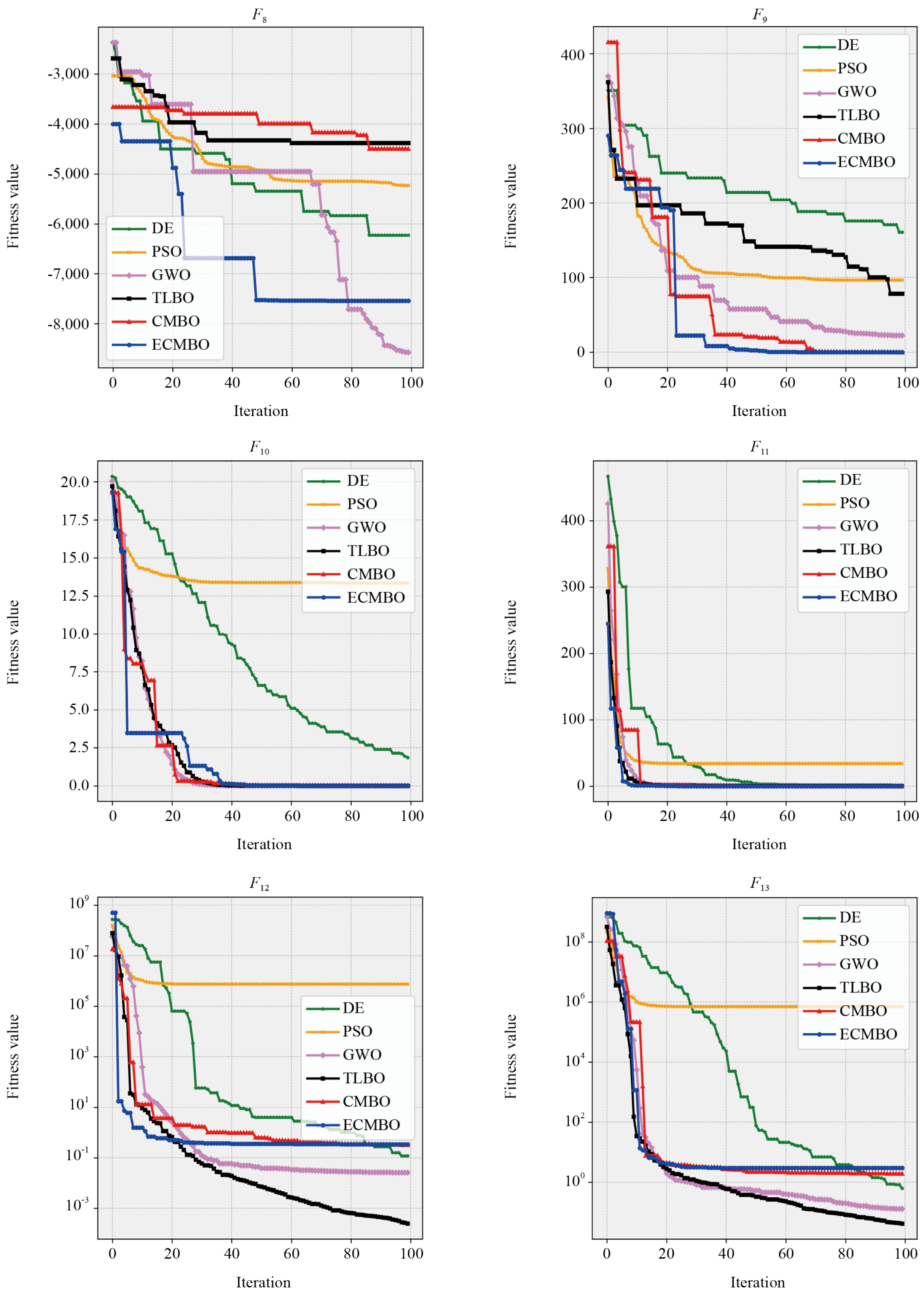


Figure 7. Comparison of fitness function values for different algorithms on high-dimensional multimodal test functions

Table 8. Comparison of methods for fixed-dimensional multimodal functions

Function	F_{min}	Fitness values	E-CMBO	CMBO	PSO	GWO	TLBO	DE
F_{14}	0.998	Min	0.998	0.998	0.998	0.998	0.998	0.998
		Max	0.998	1.99	11.71	2.98	0.998	0.998
		Mean	0.998	1.09	3.26	1.39	0.998	0.998
		Std	0	0.297	2.92	0.793	2.8938E-15	3.5459E-14
		Time/Iter (sec)	0.113	0.063	0.051	0.054	0.103	0.074
F_{15}	3.0E-4	Min	3.0844E-04	3.1891E-04	3.8240E-04	4.5317E-04	3.6861E-04	4.6319E-04
		Max	1.2232E-03	7.9586E-04	1.6756E-03	2.0363E-02	7.1437E-04	6.8647E-04
		Mean	5.0614E-04	5.3241E-04	7.0060E-04	2.5247E-03	5.1262E-04	6.0488E-04
		Std	2.6174E-08	1.7557E-04	3.8430E-04	5.9464E-03	1.2418E-04	7.3204E-05
		Time/Iter (sec)	0.047	0.026	0.032	0.023	0.049	0.049
F_{16}	-1.0316	Min	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316
		Max	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316
		Mean	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316
		Std	9.6300E-17	6.0700E-07	7.0200E-17	8.4400E-08	1.8600E-16	3.6200E-09
		Time/Iter (sec)	0.021	0.012	0.017	0.022	0.044	0.045
F_{17}	0.398	Min	0.398	0.398	0.398	0.398	0.398	0.398
		Max	0.398	0.3978	0.398	0.3979	0.398	0.3978
		Mean	0.398	0.398	0.398	0.398	0.398	0.398
		Std	0	4.6500E-09	0	4.1700E-06	0	3.2200E-13
		Time/Iter (sec)	0.032	0.019	0.028	0.033	0.076	0.093
F_{18}	3	Min	3	3	3	3	3	3
		Max	3	3	3	3	3	3
		Mean	3	3	3	3	3	3
		Std	5.7800E-18	3.1400E-05	1.0400E-15	5.4000E-06	1.1700E-15	6.5000E-14
		Time/Iter (sec)	0.024	0.013	0.024	0.026	0.054	0.057
F_{19}	-3.68	Min	-3.86	-3.86	-3.86	-3.86	-3.86	-3.86
		Max	-3.86	-3.86	-3.86	-3.86	-3.86	-3.86
		Mean	-3.86	-3.86	-3.86	-3.86	-3.86	-3.86
		Std	3.0500E-08	1.0100E-04	7.0900E-04	3.4900E-04	8.8800E-16	1.0000E-13
		Time/Iter (sec)	0.045	0.022	0.031	0.031	0.073	0.077
F_{20}	-3.22	Min	-3.32	-3.32	-3.32	-3.32	-3.32	-3.32
		Max	-3.13	-3.19	-3.18	-3.19	-3.32	-3.20
		Mean	-3.25	-3.29	-3.26	-3.22	-3.32	-3.30
		Std	6.7004E-06	3.6898E-02	6.0416E-02	4.8082E-02	1.6410E-03	3.5539E-02
		Time/Iter (sec)	0.060	0.027	0.034	0.031	0.064	0.061

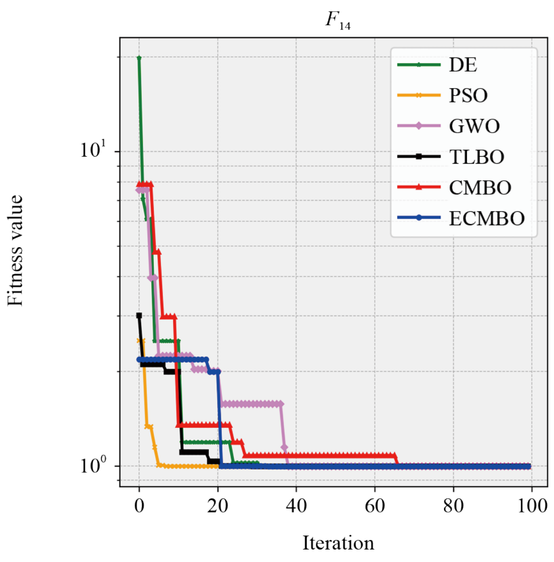
Table 8. (cont.)

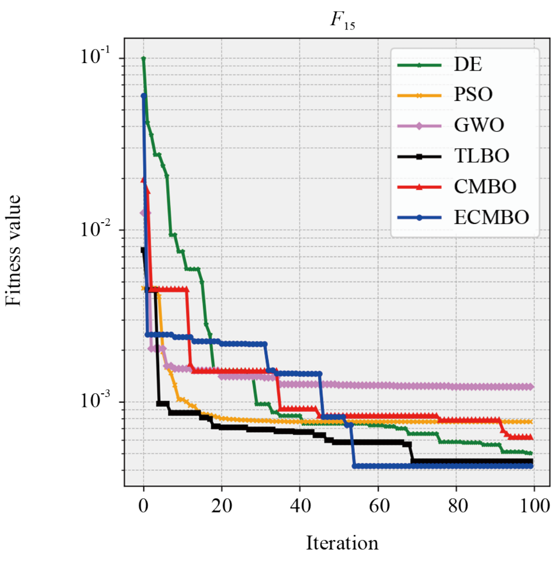
Function	F_{min}	Fitness values	E-CMBO	CMBO	PSO	GWO	TLBO	DE
F_{21}	-10.1532	Min	-10.1532	-10.1510	-10.1532	-10.1512	-10.1532	-10.1532
		Max	-5.0552	-5.0551	-5.0552	-5.0551	-10.1532	-10.1412
		Mean	-9.6086	-8.6126	-8.1231	-6.1001	-10.1532	-10.1512
		Std	1.5184E-04	1.8351	2.4864	2.0248	7.0200E-07	3.4400E-03
		Time/Iter (sec)	0.036	0.020	0.028	0.032	0.087	0.067
F_{22}	-10.4029	Min	-10.4029	-10.4010	-10.4029	-10.4020	-10.4029	-10.4029
		Max	-9.6054	-7.8919	-2.7659	-2.7516	-10.4029	-10.4029
		Mean	-10.3121	-9.5667	-8.9714	-9.6345	-10.4029	-10.4029
		Std	2.3637E-05	9.3907E-01	2.8711	2.2943	1.8800E-08	1.0800E-04
		Time/Iter (sec)	0.034	0.018	0.024	0.025	0.053	0.065
F_{23}	-10.5364	Min	-10.5364	-10.5364	-10.5364	-10.5364	-10.5364	-10.5364
		Max	-5.1284	-5.1256	-2.4273	-10.5294	-10.5211	-10.5364
		Mean	-9.3593	-9.7889	-9.0554	-10.5331	-10.5352	-10.5364
		Std	2.1208E-04	1.5808	2.9787	2.3210E-03	4.5260E-03	1.1000E-04
		Time/Iter (sec)	0.042	0.021	0.027	0.033	0.078	0.054

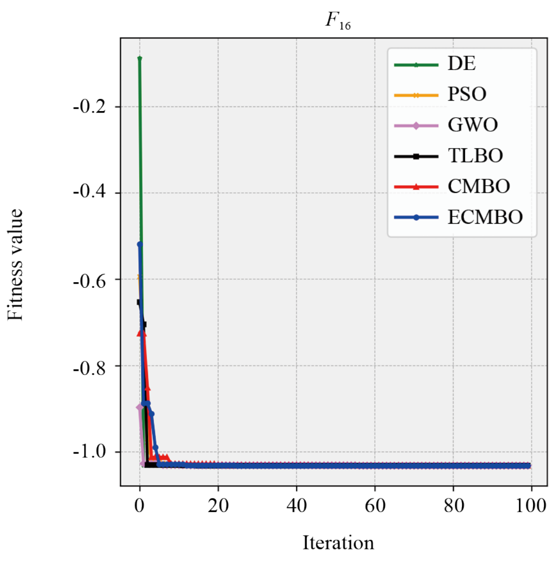
Based on the experimental results presented in Table 3, it is observed that the minimum function value for F_8 is -12.569, while for other functions (F_9 to F_{13}), the minimum value is 0. The experimental outcomes displayed in Table 7 indicate that the E-CMBO method yields the best fitness function values, which are closer to the minimum value compared to other methods, such as CMBO and PSO, for functions F_8 to F_{11}. Remarkably, E-CMBO achieves the exact minimum value for F_{11}, demonstrating its superiority in optimizing this function. However, for functions F_{12} and F_{13}, E-CMBO only outperforms CMBO and PSO, without reaching a lower minimum value than other methods. This suggests that while E-CMBO excels in optimizing certain functions, it has limitations in optimizing all the tested functions of high-dimensional multimodal. The reduction in loss function values for each method across all high dimensional multimodal functions is illustrated in Figure 7.

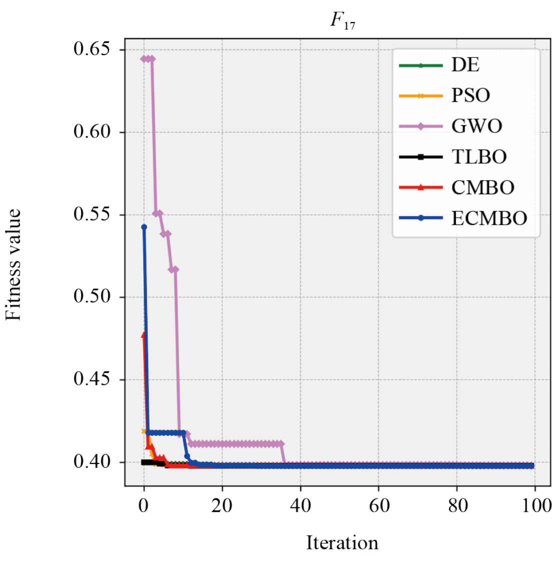
Meanwhile, the results of the experiments conducted to solve fixed-dimensional multimodal functions are presented in detail. These outcomes are systematically documented in Table 8, which highlights the performance of various optimization methods on functions from F_{14} to F_{23}. To better understand the data presented in Table 8, it is essential to refer to Table 4. This reference table provides the minimum values for each function, offering a benchmark against which the performance of different methods can be measured. By comparing the results shown in Table 8 with the benchmark minimum values in Table 4, it becomes possible to thoroughly evaluate the effectiveness of each tested method.

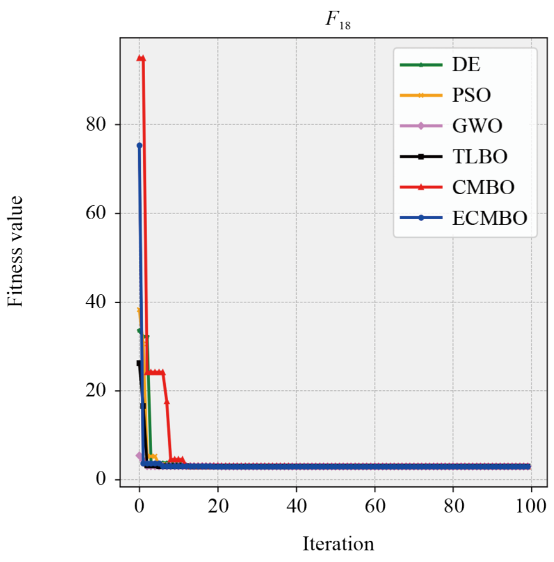
On a broader scale, it is observed that the best fitness function values achieved by all methods are identical, with each method successfully reaching the minimum value. However, a closer examination of the average values (mean) reveals slight variations. Specifically, for functions F_{22} and F_{23}, the methods TLBO and DE demonstrate superior performance compared to CMBO. This observation suggests that while E-CMBO is highly competitive and effective in solving fixed-dimensional multimodal functions, there are instances where TLBO and DE offer marginally better results. Figure 8 further illustrates the capabilities of E-CMBO, clearly demonstrating its efficiency in reaching optimal values. This figure visually reinforces the data presented in the tables, highlighting E-CMBO's ability to consistently achieve near-optimal solutions in a rapid manner. The combination of data from Tables 4 and 8, along with the graphical representation in Figure 8, provides a robust validation of E-CMBO's performance and potential as a powerful optimization tool in fixed-dimensional multimodal functions.

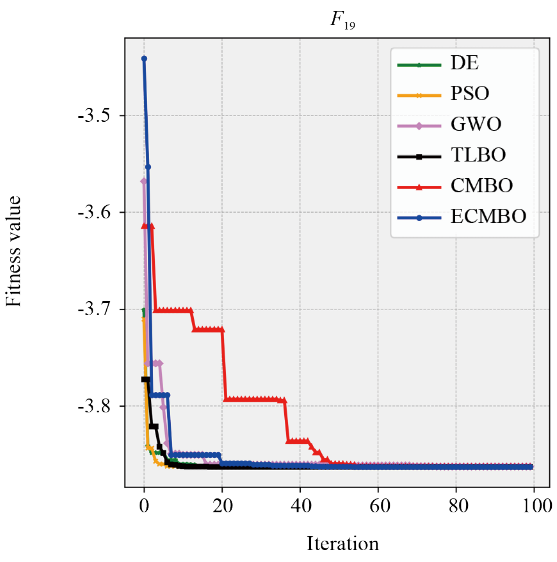

F_{14}
Fitness value
Iteration
10^1
10^0
0
20
40
60
80
100
DE
PSO
GWO
TLBO
CMBO
ECMBO


F_{15}
Fitness value
Iteration
10^{-1}
10^{-2}
10^{-3}
0
20
40
60
80
100
DE
PSO
GWO
TLBO
CMBO
ECMBO


F_{16}
Fitness value
Iteration
-0.2
-0.4
-0.6
-0.8
-1.0
0
20
40
60
80
100
DE
PSO
GWO
TLBO
CMBO
ECMBO


F_{17}
Fitness value
Iteration
0.65
0.60
0.55
0.50
0.45
0.40
0
20
40
60
80
100
DE
PSO
GWO
TLBO
CMBO
ECMBO


F_{18}
Fitness value
Iteration
80
60
40
20
0
0
20
40
60
80
100
DE
PSO
GWO
TLBO
CMBO
ECMBO


F_{19}
Fitness value
Iteration
-3.5
-3.6
-3.7
-3.8
0
20
40
60
80
100
DE
PSO
GWO
TLBO
CMBO
ECMBO

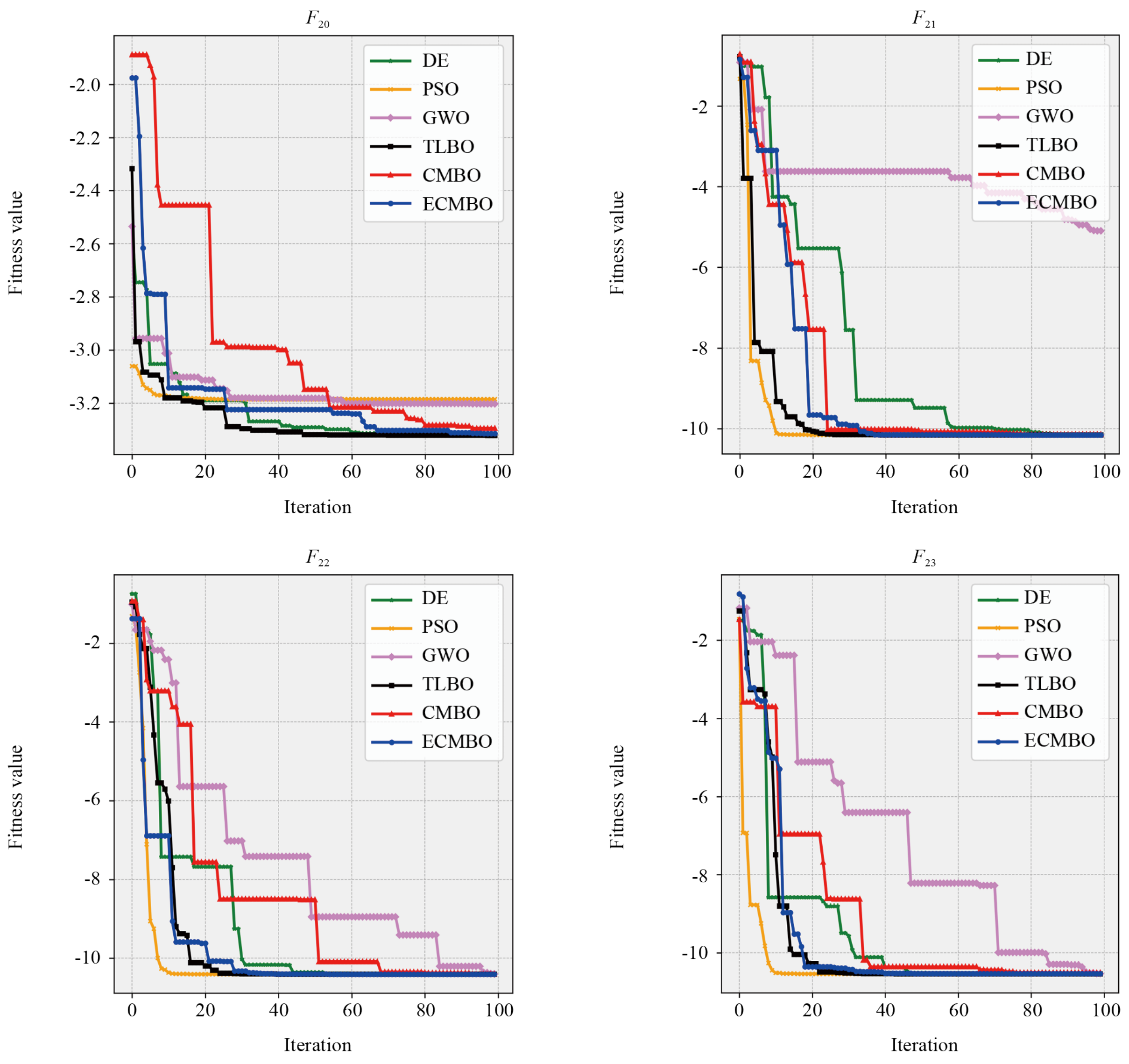
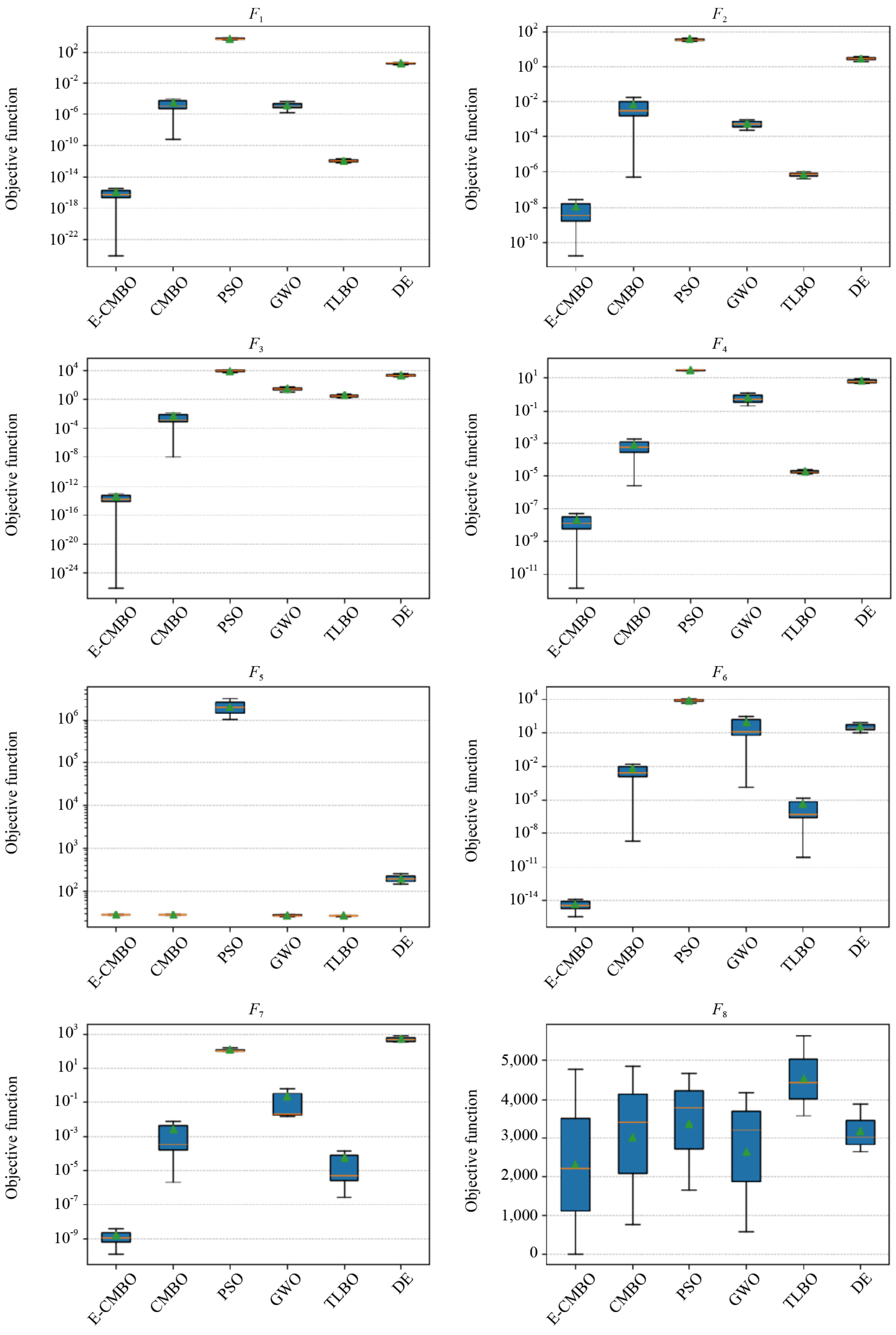
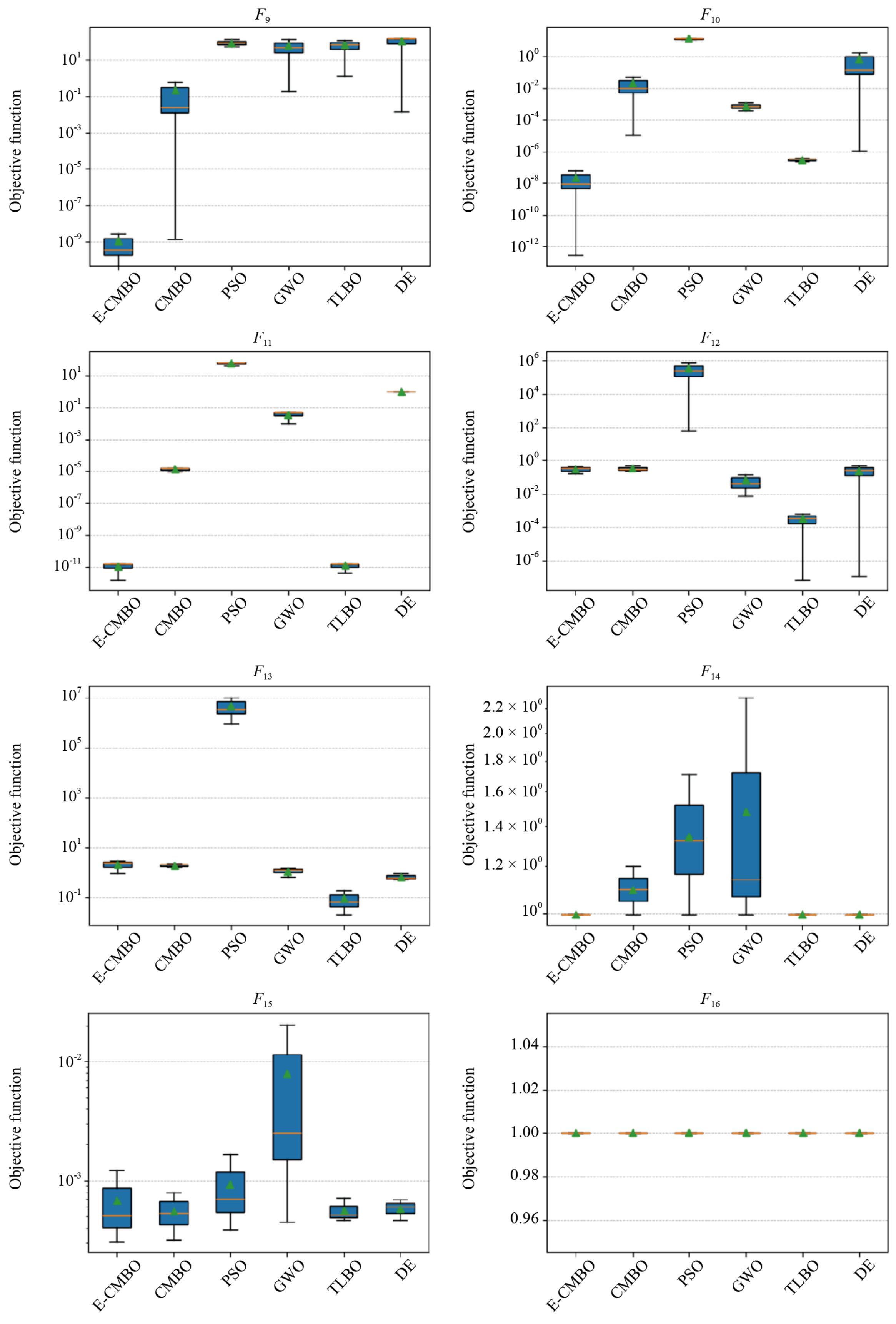


Figure 8. Comparison of fitness function values obtained by other metaheuristics on fixed-dimensional multimodal test functions

To provide a deeper analysis and a visual comparison of the optimization algorithms' performance, the boxplot results for each algorithm and objective function are presented in Figure 9. Based on the results, the E-CMBO algorithm consistently demonstrates superior performance in optimizing the specified objective functions.


F_1
F_2
F_3
F_4
F_5
F_6
F_7
F_8
Objective function
E-CMBO
CMBO
PSO
GWO
TLBO
DE


F_9
F_{10}
F_{11}
F_{12}
F_{13}
F_{14}
F_{15}
F_{16}
Objective function
E-CMBO
CMBO
PSO
GWO
TLBO
DE

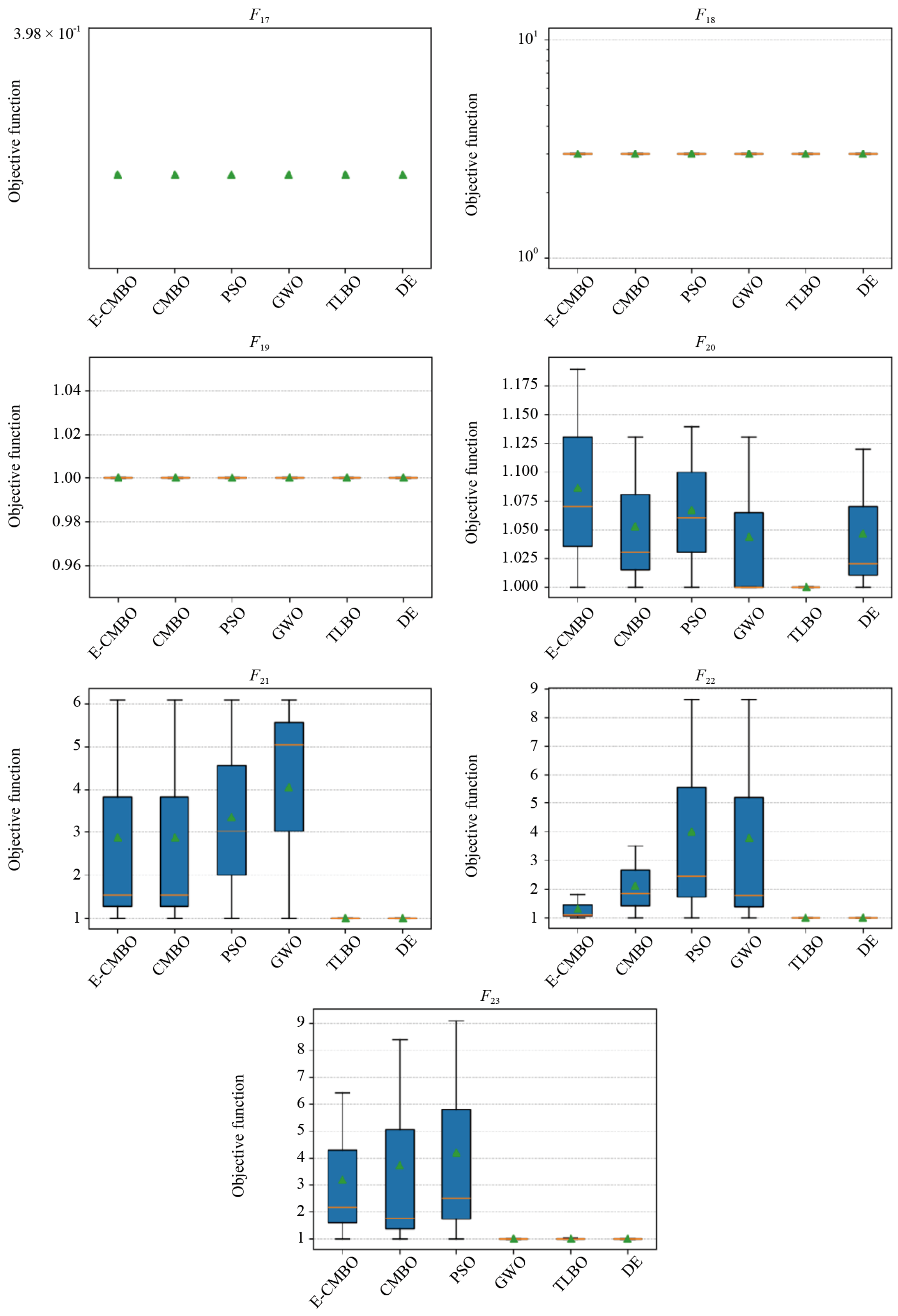


Figure 9. Boxplots showcasing the mean results of composite objective functions for various optimization algorithms

4.3 *Statistical analysis of E-CMBO*

In this subsection, the Wilcoxon rank-sum test, a statistical method, is introduced to further evaluate and analyze the performance of the optimization algorithms, including the proposed E-CMBO. This test is a widely used nonparametric statistical approach for performance comparisons. The Wilcoxon test relies on the p-value to determine the statistical significance of the evaluated algorithms, where a p-value below 0.05 indicates a statistically significant result. The outcomes of the statistical analysis using the Wilcoxon rank-sum test are summarized in Table 9.

Table 9. The results of the statistical analysis from the Wilcoxon test indicate significance at $p \leq 0.05$

The algorithms compared with E-CMBO	Unimodal functions	High-dimensional multi modal	Fixed-dimensional multi moda
CMBO	0.0215	0.0782	0.0040
PSO	0.0018	0.0065	0.0342
GWO	0.0032	0.0374	0.0040
TLBO	0.0127	0.2002	0.7054
DE	0.0021	0.0163	0.2899

The values presented in Table 9 demonstrate that the proposed E-CMBO exhibits substantial superiority over the compared algorithms when the p-value is less than 0.05. Specifically, the p-value serves as an indicator of whether E-CMBO significantly outperforms the competing algorithms. Based on the simulation results, E-CMBO consistently outperforms all other algorithms in optimizing the F_1 to F_7 unimodal function group. For the second group of objective functions (F_8 to F_{13}), E-CMBO achieves significant superiority over PSO, GWO, and DE. Additionally, in the third group of objective functions (F_{14} to F_{23}), E-CMBO significantly surpasses CMBO, PSO, and GWO. These results collectively highlight the effectiveness of E-CMBO in optimizing various types of objective functions, establishing it as a superior algorithm in this evaluation context.

4.4 *Sensitivity analysis of E-CMBO*

This subsection undertakes a comprehensive sensitivity analysis of the E-CMBO algorithm, focusing specifically on two pivotal parameters: the population size and the maximum number of iterations. By examining these parameters, we aim to understand how variations in their values influence the performance and effectiveness of the E-CMBO in finding optimal solutions.

To assess the E-CMBO's performance sensitivity to variations in population size, the algorithm was meticulously applied across all twenty-three objective functions, each representing a unique challenge and optimization problem. Different population sizes were tested, specifically 100, 300, 500, and 1,000 members, to observe how the number of members in the population affects the algorithm's convergence behavior and overall performance.

Table 10. Population size comparison of the algorithm concerning the unimodal functions

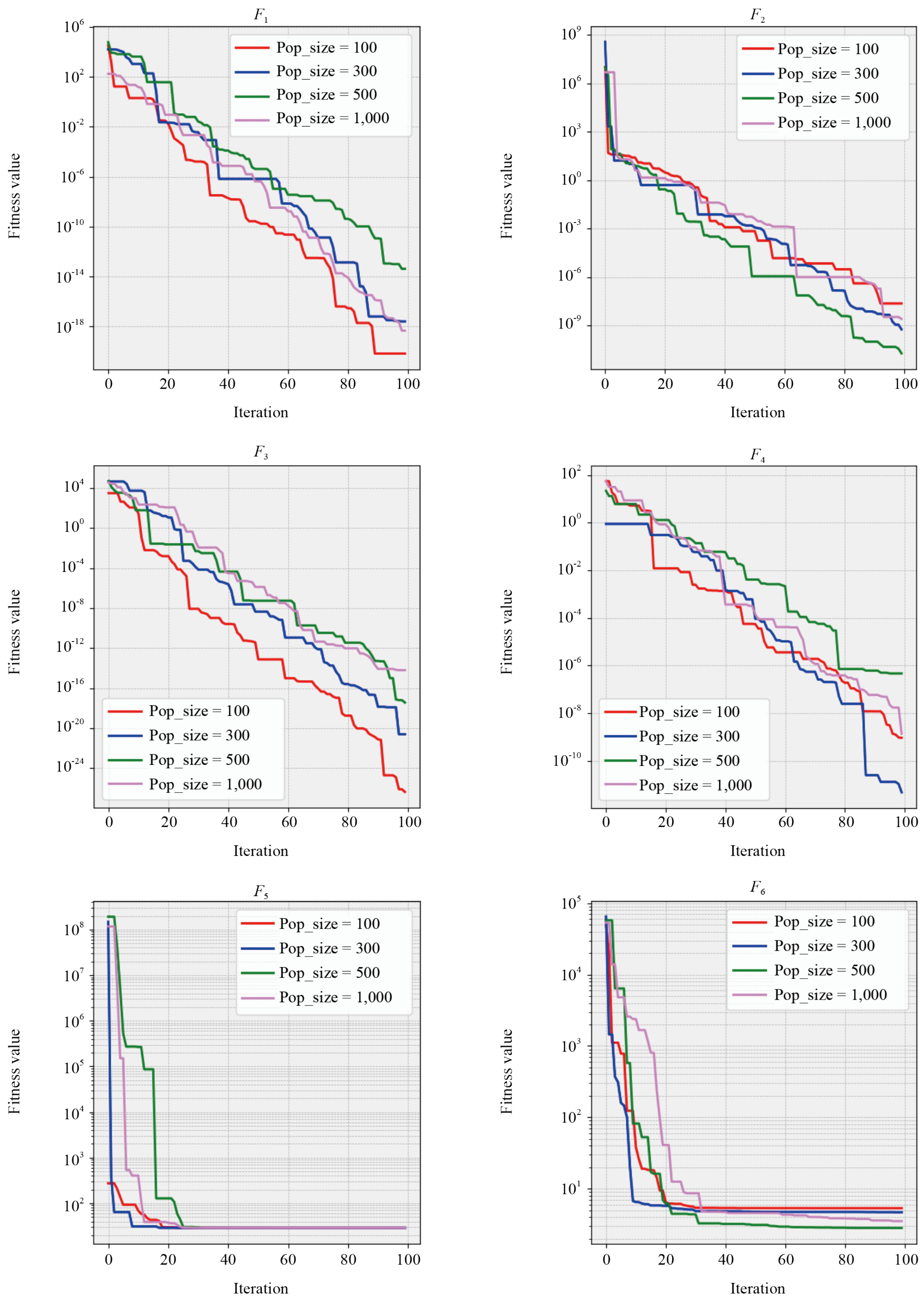
Function	F_{min}	Values	Pop size			
			100	300	500	1,000
F_1	0	Best fitness	6.8655E-21	2.6009E-18	4.1737E-14	4.7251E-19
		Time/Iter (sec)	0.028	0.067	0.124	0.216
F_2	0	Best fitness	2.3047E-08	5.7658E-10	1.8676E-11	2.5492E-09
		Time/Iter (sec)	0.040	0.083	0.131	0.258
F_3	0	Best fitness	4.2449E-27	2.4736E-21	3.6582E-18	6.3218E-15
		Time/Iter (sec)	0.116	0.167	0.238	0.456
F_4	0	Best fitness	9.1699E-10	4.7095E-12	4.6202E-07	1.3493E-09
		Time/Iter (sec)	0.030	0.055	0.091	0.128
F_5	0	Best fitness	28.88	28.81	28.87	28.79
		Time/Iter (sec)	0.042	0.052	0.084	0.162
F_6	0	Best fitness	5.34	4.68	2.81	3.51
		Time/Iter (sec)	0.027	0.046	0.074	0.186
F_7	0	Best fitness	8.2232E-04	6.6568E-04	6.4480E-04	2.2621E-04
		Time/Iter (sec)	0.037	0.083	0.095	0.173

The results of this comprehensive study are summarized in Table 10, offering a clear and concise overview of how each population size performs on unimodal functions. These functions, labeled F_1 to F_7, are well-established with a minimum value of zero, making them ideal for evaluating optimization algorithms. The table highlights that while population size influences the algorithm's performance, a larger population does not necessarily result in a closer approximation to the minimum value. Instead, it primarily contributes to longer iteration times due to the increased computational load from managing more individuals within the population.

Further insights are provided in Figure 10, which visually depicts the convergence dynamics across different population sizes. This figure offers a detailed representation of how the algorithm progresses toward optimal solutions over time, illustrating that while a larger population may improve solution diversity, it does not guarantee superior performance in approaching the minimum value. These findings underline the importance of balancing population size to achieve both computational efficiency and effective optimization.

The analysis presented in both the table and figure underscores a critical trade-off in algorithm design: while increasing population size can enhance exploration capabilities, it also demands greater computational resources, which may diminish returns in terms of convergence to optimal solutions. These observations provide valuable insights for tuning algorithm parameters to optimize performance on unimodal functions.

In addition, the experimental outcomes for high-dimensional multimodal functions, as presented in Table 11 and illustrated in Figure 11, provide valuable insights into the performance dynamics across various scenarios.


F_1
F_2
F_3
F_4
F_5
F_6
Pop_size = 100
Pop_size = 300
Pop_size = 500
Pop_size = 1,000
Fitness value
Iteration

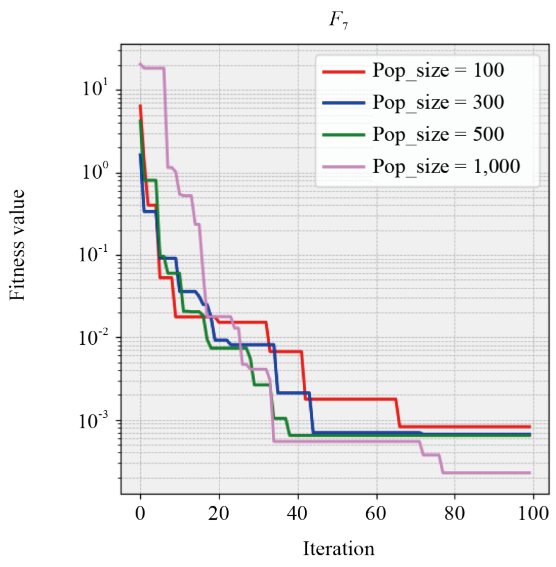


Figure 10. Population size analysis of E-CMBO on the unimodal function

Table 11. Population size comparison of the algorithm concerning the high-dimensional multimodal functions

Function	F_{min}	Values	Pop size			
			100	300	500	1,000
F_8	-2,569	Best fitness	-6,359.47	-7,395.92	-9,129.91	-10,467.81
		Time/Iter (sec)	0.028	0.023	0.052	0.161
F_9	0	Best fitness	1.9492E-10	0	0	0
		Time/Iter (sec)	0.028	0.059	0.103	0.174
F_{10}	0	Best fitness	2.2613E-08	3.7504E-09	3.4610E-10	7.8256E-11
		Time/Iter (sec)	0.032	0.050	0.083	0.155
F_{11}	0	Best fitness	0	0	0	0
		Time/Iter (sec)	0.035	0.090	0.130	0.186
F_{12}	0	Best fitness	4.4972E-01	2.0504E-01	5.0413E-01	8.8038E-02
		Time/Iter (sec)	0.044	0.099	0.160	0.324
F_{13}	0	Best fitness	2.4324E + 00	1.5618E + 00	1.4256E + 00	5.8883E-01
		Time/Iter (sec)	0.042	0.096	0.170	0.290

Table 11 and Figure 11 illustrates the impact of population size on the capability and efficiency of E-CMBO in optimizing high-dimensional multimodal functions F_8 to F_{13}. Unlike the previous unimodal functions, this table clearly shows that a larger population size can lead to improved fitness function values, as evidenced by the consistently lower values achieved with a population size of 1,000 for all functionss. However, execution time remains a concern, as it increases significantly with larger population sizes, similar to the results observed previously.

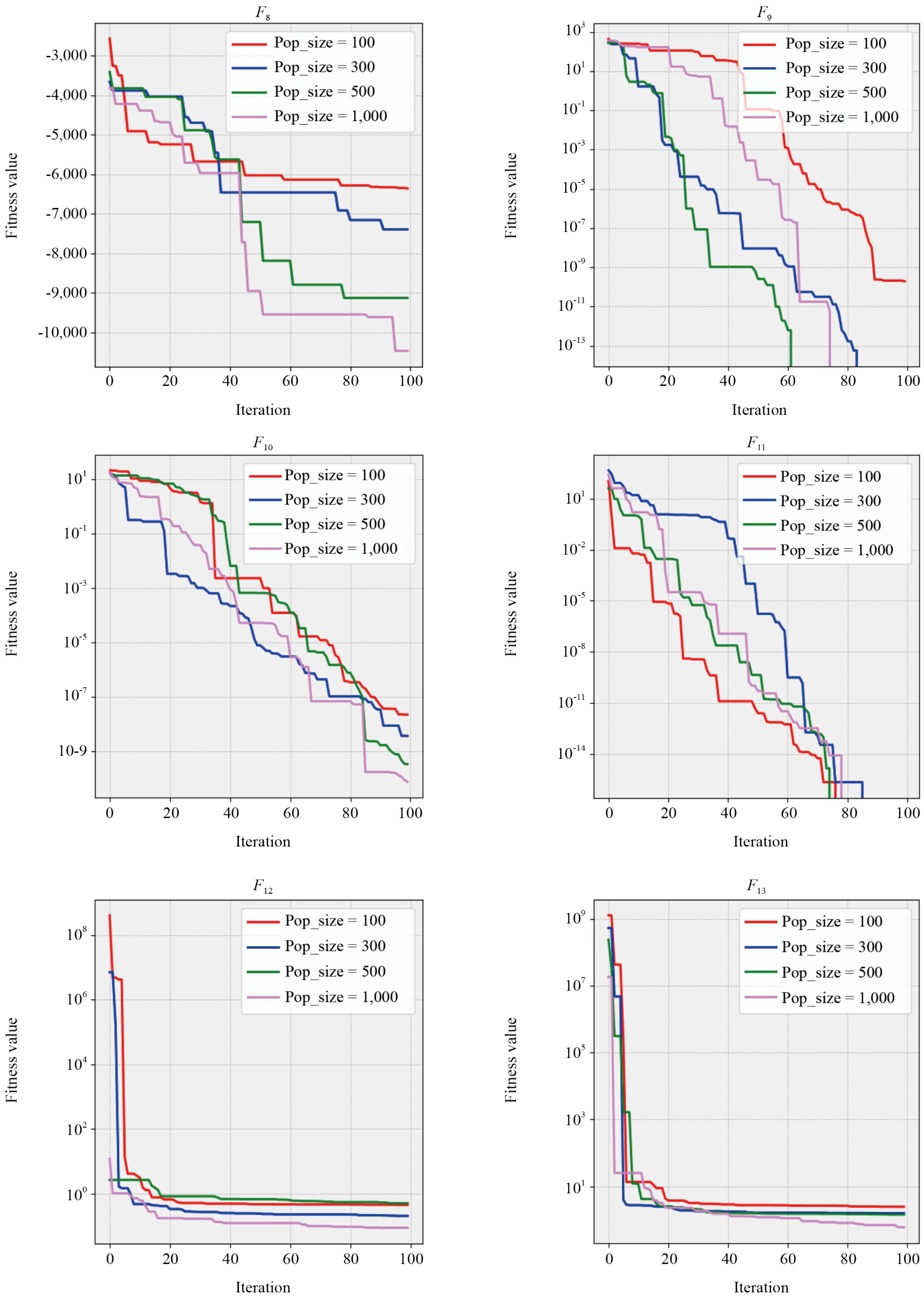
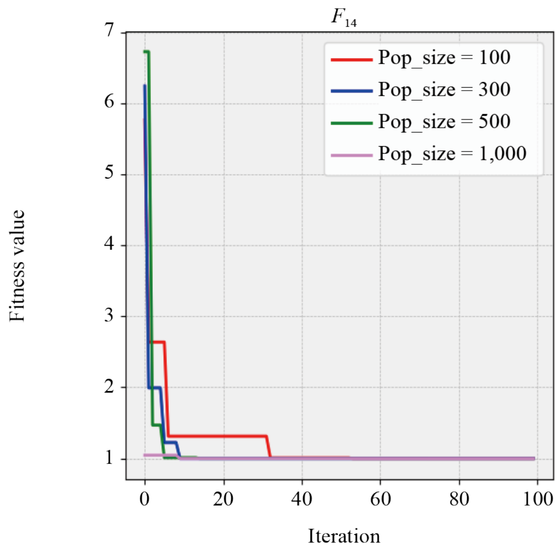
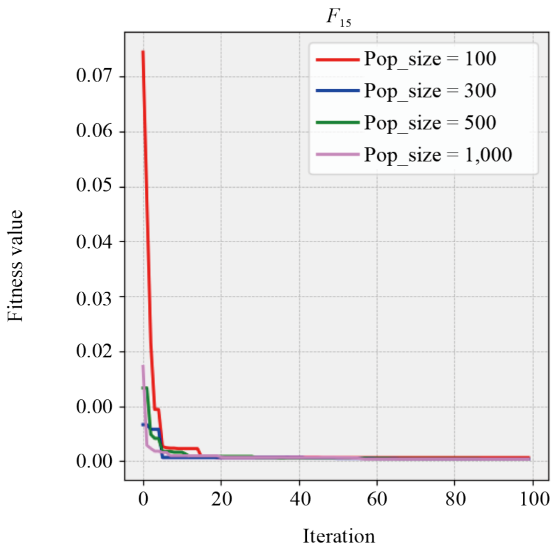
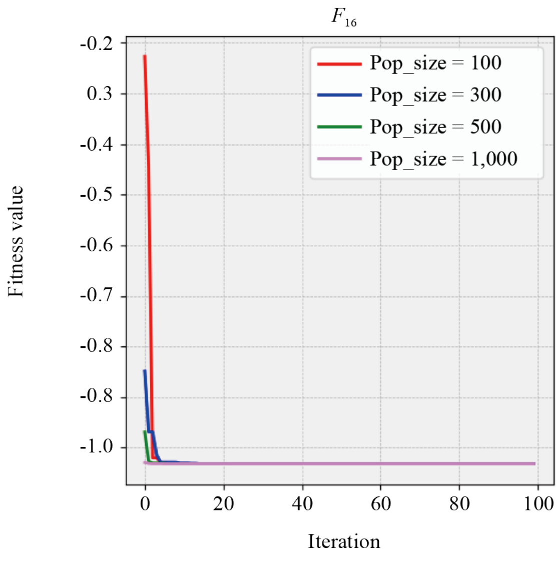
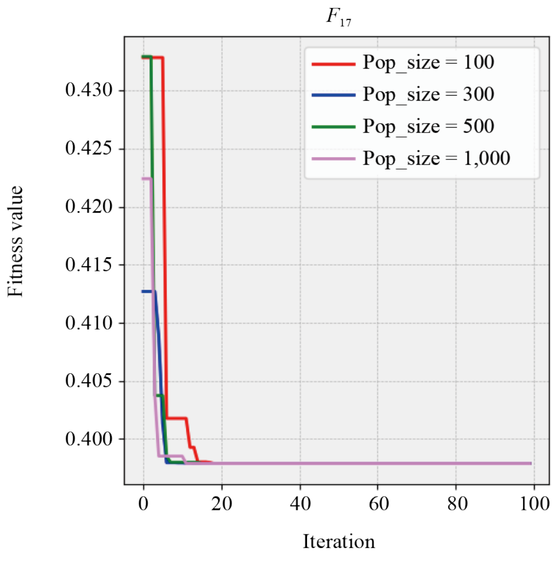


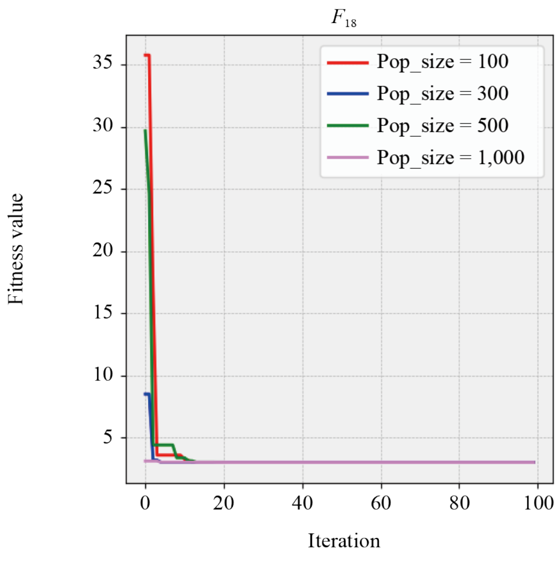
Figure 11. Population size analysis of E-CMBO on high-dimensional multimodal function

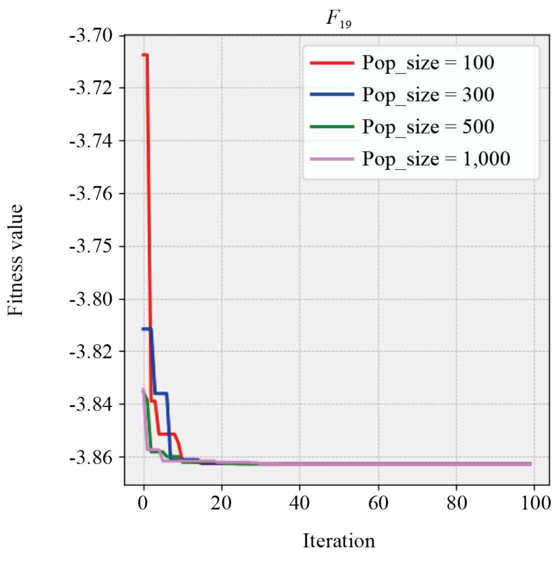

F_{14}
Pop_size = 100
Pop_size = 300
Pop_size = 500
Pop_size = 1,000
Fitness value
Iteration


F_{15}
Pop_size = 100
Pop_size = 300
Pop_size = 500
Pop_size = 1,000
Fitness value
Iteration


F_{16}
Pop_size = 100
Pop_size = 300
Pop_size = 500
Pop_size = 1,000
Fitness value
Iteration


F_{17}
Pop_size = 100
Pop_size = 300
Pop_size = 500
Pop_size = 1,000
Fitness value
Iteration


F_{18}
Pop_size = 100
Pop_size = 300
Pop_size = 500
Pop_size = 1,000
Fitness value
Iteration


F_{19}
Pop_size = 100
Pop_size = 300
Pop_size = 500
Pop_size = 1,000
Fitness value
Iteration

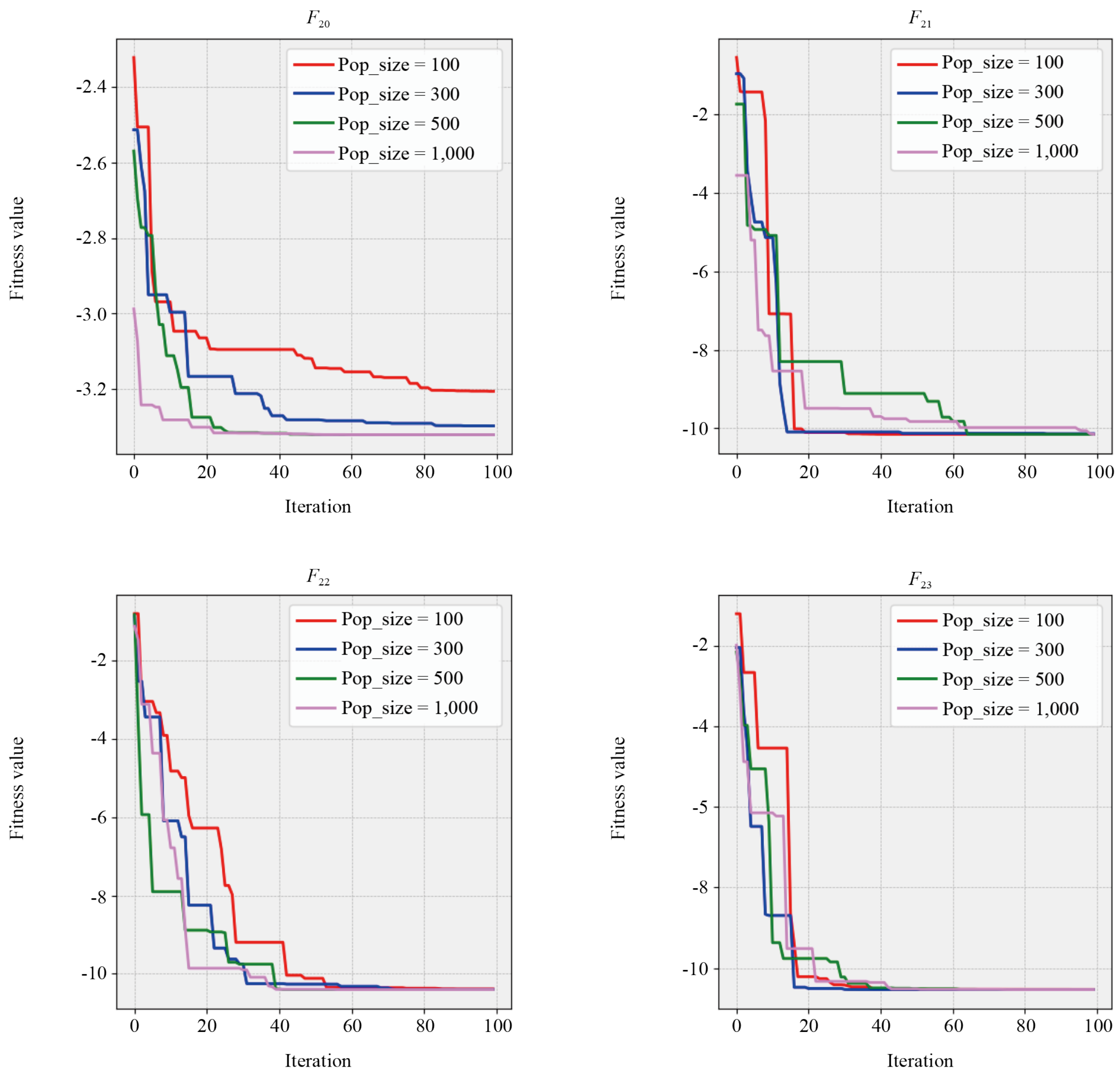


Figure 12. Population size analysis of E-CMBO on fixed-dimensional multimodal functions

Finally, the comparison of results for fixed-dimensional multimodal functions, evaluated based on population size, is presented in Table 12 and illustrated in Figure 12. This subsequent analysis provides deeper insights into the algorithm's behavior in lower-dimensional yet complex optimization landscapes, complementing the findings from the high-dimensional scenarios.

The experimental results for fixed-dimensional multimodal functions reveal an important pattern: as the population size of the algorithm increases, E-CMBO demonstrates a tendency to converge toward quasi-optimal solutions, particularly for functions F_{21}, F_{22}, and F_{23}. Conversely, for functions F_{14}, F_{16}, F_{17}, F_{18}, and F_{19}, all tested population sizes successfully achieved the optimal solutions. However, functions F_{15} and F_{20} presented a challenge, as none of the population sizes, even up to 1,000, reached the optimal solutions.

Table 12. Population size comparison of the algorithm concerning on the fixed-dimensional multimodal functions

Function	F_{min}	Values	Pop size			
			100	300	500	1,000
F_{14}	0.998	Best fitness	0.998	0.998	0.998	0.998
		Time/Iter (sec)	0.096	0.240	0.372	0.721
F_{15}	0.00030	Best fitness	0.00061	0.00033	0.000381	0.000344
		Time/Iter (sec)	0.028	0.063	0.099	0.252
F_{16}	-1.0316	Best fitness	-1.0316	-1.0316	-1.0316	-1.0316
		Time/Iter (sec)	0.021	0.038	0.134	0.325
F_{17}	0.398	Best fitness	0.398	0.398	0.398	0.398
		Time/Iter (sec)	0.022	0.056	0.075	0.131
F_{18}	3	Best fitness	3	3	3	3
		Time/Iter (sec)	0.024	0.061	0.089	0.195
F_{19}	-3.68	Best fitness	-3.863	-3.863	-3.863	-3.863
		Time/Iter (sec)	0.041	0.067	0.126	0.241
F_{20}	-3.22	Best fitness	-3.20	-3.29	-3.32	-3.32
		Time/Iter (sec)	0.030	0.075	0.105	0.217
F_{21}	-10.1532	Best fitness	-9.9674	-10.1532	-10.1530	-10.1532
		Time/Iter (sec)	0.023	0.052	0.080	0.147
F_{22}	-10.4029	Best fitness	-10.3870	-10.3969	-10.4028	-10.4029
		Time/Iter (sec)	0.027	0.057	0.088	0.185
F_{23}	-10.5364	Best fitness	-10.5306	-10.5364	-10.5355	-10.5364
		Time/Iter (sec)	0.031	0.062	0.104	0.254

In summary, this analysis underscores the role of larger population sizes in enhancing the effectiveness of the E-CMBO algorithm, particularly for multimodal functions. While unimodal functions are less affected by this parameter and tend to find solutions close to zero regardless of population size, the benefits for multimodal functions are significant. However, increasing population size comes at the cost of substantially higher execution times. For simpler functions, a smaller population size is often sufficient to achieve quasi-optimal results, as evidenced in the earlier tables and figures. Therefore, careful adjustment of population size is essential to balance accuracy and efficiency, ensuring that the algorithm performs optimally across diverse and complex objective functions. These findings provide valuable insights for practitioners and researchers seeking to optimize E-CMBO, enabling the algorithm to deliver precise and efficient solutions tailored to specific applications.

Assertion 1 (Impact of Population Size on E-CMBO Algorithm Performance) Let F be a set of objective functions categorized as either unimodal or multimodal, and let P denote the population size of the E-CMBO algorithm. Then, the following holds:

1. For multimodal functions $F_m \subseteq F$, an increase in P significantly enhances the algorithm's convergence to quasi-optimal solutions. Specifically, there exists a threshold P_{min} such that for $P \geq P_{min}$, the algorithm achieves near-optimal performance.

2. For unimodal functions $F_u \subseteq F$, the algorithm's performance is relatively insensitive to increases in P, and solutions are consistently close to the optimal value, even for small P.

3. Increasing P results in a proportional increase in execution time, denoted by $T(P)$, such that $T(P)$ is monotonically increasing with respect to P.

Consequently, the optimal choice of P depends on the complexity of the objective function F. For simpler functions, a smaller P suffices to achieve quasi-optimal results, whereas for more complex multimodal functions, a larger P is recommended to balance accuracy and efficiency.

Explanation: Let F denote the set of objective functions, where $F = F_u \cup F_m$, with F_u representing unimodal functions and F_m representing multimodal functions. Let P be the population size of the E-CMBO algorithm, and let $T(P)$ denote the execution time as a function of P.

1. Multimodal Functions (F_m): Multimodal functions contain multiple local optima. The E-CMBO algorithm needs a sufficiently large population size P to explore the solution space effectively and avoid convergence to local optima. From experimental data:

$$\lim_{P\to\infty} \mathbb{E}[F_m(P)] = F^*$$

where F^* is the global optimum.

Additionally, there exists a threshold P_{min} such that:

$$P \geq P_{min} \implies \mathbb{E}[F_m(P)] \to F^*.$$

Here, $\mathbb{E}[F_m(P)]$ denotes the expected solution quality for multimodal functions. This shows that larger P significantly improves convergence for F_m.

2. Unimodal Functions (F_u): Unimodal functions have a single global optimum, making them less sensitive to population size. From experimental results:

$$\forall P,\, \mathbb{E}[F_u(P)] = F^*,$$

where F^* is the global optimum. This indicates that any P is sufficient for convergence:

$$\lim_{P\to 0} \mathbb{E}[F_u(P)] = F^* \quad \text{and} \quad \lim_{P\to\infty} \mathbb{E}[F_u(P)] = F^*.$$

Thus, for F_u, the choice of P does not affect convergence.

3. Execution Time ($T(P)$): The execution time $T(P)$ increases monotonically with population size P, as each individual in the population requires evaluation at each iteration. From experimental observations:

$$T(P) = c \cdot P + b,$$

where $c > 0$ is a constant representing the time per individual, and $b \geq 0$ accounts for baseline overhead. Thus:

$$\frac{\partial T(P)}{\partial P} > 0 \quad \text{and} \quad T(P) \to \infty \quad \text{as} \quad P \to \infty.$$

By synthesizing the observations and mathematical relationships discussed above, we derive the following key insights that establish the validity of the theorem:

1. For F_m, the expected solution improves with P:

$$P \geq P_{min} \implies \mathbb{E}[F_m(P)] \to F^*.$$

2. For F_u, the expected solution is independent of P:

$$\forall P, \mathbb{E}[F_u(P)] = F^*.$$

3. The execution time increases linearly with P:

$$T(P) = c \cdot P + b.$$

Thus, careful tuning of P is necessary to balance solution quality and computational cost.

5. Conclusion

In conclusion, the experimental results underscore the efficacy of the E-CMBO algorithm in addressing various types of optimization challenges, particularly those posed by unimodal and high-dimensional multimodal objective functions. Across the tested scenarios, E-CMBO consistently demonstrates its capability to generate fitness function values closer to the minimum and achieve superior convergence speeds compared to CMBO. This indicates its potential to offer faster and more efficient solutions in complex search spaces.

While E-CMBO exhibits competitive performance in most cases, its effectiveness slightly diminishes when applied to fixed-dimensional multimodal objective functions, where it falls short of surpassing CMBO in one particular test function. Despite this, E-CMBO still showcases comparable performance overall, highlighting its robustness across diverse problem landscapes.

The results are further supported by the theoretical analysis, which establishes that for multimodal objective functions, a larger population size (P) enhances the algorithm's convergence to quasi-optimal solutions, provided the population size exceeds a critical threshold ($P \geq P_{min}$). For unimodal functions, the performance is less sensitive to changes in P, with smaller population sizes sufficing to achieve optimal results. However, increasing P comes at the cost of higher computational time, as execution time $T(P)$ grows linearly with population size.

Overall, the study suggests that E-CMBO is a promising optimization tool, offering significant advantages in terms of convergence speed and solution quality, particularly for unimodal and high-dimensional multimodal optimization problems. The theoretical insights provide a deeper understanding of the relationship between population size and algorithm performance, emphasizing the importance of careful tuning of parameters to balance accuracy and efficiency. Further research could delve deeper into the specific characteristics of different objective functions to enhance our understanding of optimization algorithm behaviors and improve their applicability in real-world scenarios.

Acknowledgement

This research is funded through Universiti Malaysia Terengganu and Universitas Ahmad Dahlan under the International Partnership Research Grant (IPRG), No. Vot: 55382.

Disclosure

All experiments and analyses were performed independently in accordance with standard scientific practices. The findings and interpretations reflect the authors' analysis and theoretical insights. Contributions from both institutions are acknowledged and appreciated.

Conflict of interest

The authors confirm that no conflicts of interest arose at any stage in relation to the research findings presented.

References

[1] Molnar I. Simulation and optimisation. *Society and Economy*. 2005; 27(2): 213-226. Available from: https://doi.org/10.1556/socec.27.2005.2.3.

[2] Mashwani WK, Shah H, Kaur M, Bakar MA, Miftahuddin M. Large-scale bound constrained optimization based on hybrid teaching learning optimization algorithm. *Alexandria Engineering Journal*. 2021; 60(6): 6013-6033. Available from: https://doi.org/10.1016/j.aej.2021.04.002.

[3] Houssein EH, Oliva D, Samee NA, Mahmoud NF, Emam MM. Liver Cancer Algorithm: A novel bio-inspired optimizer. *Computers in Biology and Medicine*. 2023; 165: 107389. Available from: https://doi.org/10.1016/j.compbiomed.2023.107389.

[4] Sadeghi A, Doumari SA, Dehghani M, Montazeri Z, Trojovskỳ P, Ashtiani HJ. A new "Good and Bad Groups-Based Optimizer" for solving various optimization problems. *Applied Sciences*. 2021; 11(10): 4382. Available from: https://doi.org/10.3390/app11104382.

[5] Yang XS. *Nature-Inspired Optimization Algorithms*. Cambridge, Massachusetts: Academic Press; 2020.

[6] Rakotonirainy RG, Van Vuuren JH. Improved metaheuristics for the two-dimensional strip packing problem. *Applied Soft Computing*. 2020; 92: 106268. Available from: https://doi.org/10.1016/j.asoc.2020.106268.

[7] Abd Elaziz M, Dahou A, Abualigah L, Yu L, Alshinwan M, Khasawneh AM, et al. Advanced metaheuristic optimization techniques in applications of deep neural networks: a review. *Neural Computing and Applications*. 2021; 33: 14079-14099. Available from: https://doi.org/10.1007/s00521-021-05960-5.

[8] Trojovskỳ P, Dehghani M. Pelican optimization algorithm: A novel nature-inspired algorithm for engineering applications. *Sensors*. 2022; 22(3): 855. Available from: https://doi.org/10.3390/s22030855.

[9] Vasuki A. *Nature-Inspired Optimization Algorithms*. USA: CRC Press; 2020.

[10] Eiben AE, Smith JE. *Introduction to Evolutionary Computing*. Berlin, Heidelberg: Springer; 2015. Available from: https://doi.org/10.1007/978-3-662-44874-8.

[11] Dokeroglu T, Sevinc E, Kucukyilmaz T, Cosar A. A survey on new generation metaheuristic algorithms. *Computers & Industrial Engineering*. 2019; 137: 106040. Available from: https://doi.org/10.1016/j.cie.2019.106040.

[12] Stegherr H, Heider M, Hähner J. Classifying metaheuristics: Towards a unified multi-level classification system. *Natural Computing*. 2022; 21(2): 155-171. Available from: https://doi.org/10.1007/s11047-020-09824-0.

[13] Fister Jr I, Yang XS, Fister I, Brest J, Fister D. A brief review of nature-inspired algorithms for optimization. *arXiv:13074186*. 2013. Available from: https://doi.org/10.48550/arXiv.1307.4186.

[14] Bäck T, Schwefel HP. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*. 1993; 1(1): 1-23. Available from: https://doi.org/10.1162/evco.1993.1.1.1.

[15] Price K, Storn RM, Lampinen JA. *Differential Evolution: A Practical Approach to Global Optimization*. Berlin, Heidelberg: Springer Science & Business Media; 2006. Available from: https://doi.org/10.1007/3-540-31306-0.

[16] Storn R, Price K. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*. 1997; 11: 341-359. Available from: https://doi.org/10.1023/A:1008202821328.

[17] Back T. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford, UK: Oxford University Press; 1996.

[18] Koza JR. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*. 1994; 4: 87-112. Available from: https://doi.org/10.1007/BF00175355.

[19] Poli R, Langdon WB, McPhee NF. *A Field Guide to Genetic Programming*. UK: Lulu Enterprises UK Ltd.; 2008.

[20] Kennedy J, Eberhart R. Particle swarm optimization. In: *Proceedings of ICNN'95-International Conference on Neural Networks*. Perth, WA, Australia: IEEE; 1995. p.1942-1948. Available from: https://doi.org/10.1109/ICNN.1995.488968.

[21] Karaboga D, Akay B. A comparative study of Artificial Bee Colony algorithm. *Applied mathematics and computation*. 2009; 214(1): 108-132. Available from: https://doi.org/10.1016/j.amc.2009.03.090.

[22] Meng X, Liu Y, Gao X, Zhang H. A new bio-inspired algorithm: chicken swarm optimization. In: *Advances in Swarm Intelligence: 5th International Conference, ICSI 2014*. Hefei, China: Springer; 2014. p.86-94. Available from: https://doi.org/10.1007/978-3-319-11857-4_10.

[23] Mirjalili S, Lewis A. The whale optimization algorithm. *Advances in Engineering Software*. 2016; 95: 51-67. Available from: https://doi.org/10.1016/j.advengsoft.2016.01.008.

[24] Rao RV, Savsani VJ, Vakharia DP. Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems. *Computer-Aided Design*. 2011; 43(3): 303-315. Available from: https://doi.org/10.1016/j.cad.2010.12.015.

[25] Matoušová I, Trojovskỳ P, Dehghani M, Trojovská E, Kostra J. Mother optimization algorithm: a new human-based metaheuristic approach for solving engineering optimization. *Scientific Reports*. 2023; 13(1): 10312. Available from: https://doi.org/10.1038/s41598-023-37537-8.

[26] Abedinpourshotorban H, Shamsuddin SM, Beheshti Z, Jawawi DNA. Electromagnetic field optimization: a physics-inspired metaheuristic optimization algorithm. *Swarm and Evolutionary Computation*. 2016; 26: 8-22. Available from: https://doi.org/10.1016/j.swevo.2015.07.002.

[27] Wolpert DH, Macready WG. *No Free Lunch Theorems for Search*. SFI working paper. Santa Fe Institute; 1995.

[28] Blum C, Roli A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*. 2003; 35(3): 268-308. Available from: https://doi.org/10.1145/937503.937505.

[29] Yang XS. *Nature-Inspired Metaheuristic Algorithms*. Bristol: Luniver Press; 2010.

[30] Yang XS. *Engineering Optimization: An Introduction with Metaheuristic Applications*. Hoboken, New Jersey: John Wiley & Sons; 2010.

[31] Fausto F, Reyna-Orta A, Cuevas E, Andrade ÁG, Perez-Cisneros M. From ants to whales: metaheuristics for all tastes. *Artificial Intelligence Review*. 2020; 53: 753-810. Available from: https://doi.org/10.1007/s10462-018-09676-2.

[32] Yang XS, Deb S. Eagle strategy using Lévy walk and firefly algorithms for stochastic optimization. In: *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*. Springer; 2010. p.101-111. Available from: https://doi.org/10.1007/978-3-642-12538-6_9.

[33] Yang XS, Deb S. Two-stage eagle strategy with differential evolution. *International Journal of Bio-Inspired Computation*. 2012; 4(1): 1-5. Available from: https://doi.org/10.1504/IJBIC.2012.044932.

[34] Xu J, Zhang J. Exploration-exploitation tradeoffs in metaheuristics: Survey and analysis. In: *Proceedings of the 33rd Chinese Control Conference*. Nanjing, China: IEEE; 2014. p.8633-8638. Available from: https://doi.org/10.1109/ChiCC.2014.6896450.

[35] Du J, Wen Y, Wang L, Zhang P, Fei M, Pardalos PM. An adaptive human learning optimization with enhanced exploration-exploitation balance. *Annals of Mathematics and Artificial Intelligence*. 2023; 91(2): 177-216. Available from: https://doi.org/10.1007/s10472-022-09799-x.

[36] Mann PS, Singh S. Improved metaheuristic based energy-efficient clustering protocol for wireless sensor networks. *Engineering Applications of Artificial Intelligence*. 2017; 57: 142-152. Available from: https://doi.org/10.1016/j.engappai.2016.10.014.

[37] Sayed GI, Tharwat A, Hassanien AE. Chaotic dragonfly algorithm: an improved metaheuristic algorithm for feature selection. *Applied Intelligence*. 2019; 49: 188-205. Available from: https://doi.org/10.1007/s10489-018-1261-8.

[38] Dandy GC, Simpson AR, Murphy LJ. An improved genetic algorithm for pipe network optimization. *Water Resources Research*. 1996; 32(2): 449-458. Available from: https://doi.org/10.1029/95WR02917.

[39] Jiang Y, Hu T, Huang C, Wu X. An improved particle swarm optimization algorithm. *Applied Mathematics and Computation*. 2007; 193(1): 231-239. Available from: https://doi.org/10.1016/j.amc.2007.03.047.

[40] Ojugo A, Emudianughe J, Yoro R, Okonta E, Eboka A. A hybrid artificial neural network gravitational search algorithm for rainfall runoffs modeling and simulation in hydrology. *Progress in Intelligent Computing and Applications*. 2013; 2(1): 22-33.

[41] Nama S, Saha AK, Sharma S. A hybrid TLBO algorithm by quadratic approximation for function optimization and its application. In: *Recent Trends and Advances in Artificial Intelligence and Internet of Things*. Cham: Springer; 2020. p.291-341. Available from: https://doi.org/10.1007/978-3-030-32644-9_30.

[42] Tuo S, Yong L, Deng F, Li Y, Lin Y, Lu Q. HSTLBO: A hybrid algorithm based on Harmony Search and Teaching-Learning-Based Optimization for complex high-dimensional optimization problems. *PloS One*. 2017; 12(4): e0175114. Available from: https://doi.org/10.1371/journal.pone.0175114.

[43] Chen X, Yu K. Hybridizing cuckoo search algorithm with biogeography-based optimization for estimating photovoltaic model parameters. *Solar Energy*. 2019; 180: 192-206. Available from: https://doi.org/10.1016/j.solener.2019.01.025.

[44] Nassef AM, Abdelkareem MA, Maghrabie HM, Baroutaji A. Review of metaheuristic optimization algorithms for power systems problems. *Sustainability*. 2023; 15(12): 9434. Available from: https://doi.org/10.3390/su15129434.

[45] Dehghani M, Hubálovskỳ Š, Trojovskỳ P. Cat and mouse based optimizer: A new nature-inspired optimization algorithm. *Sensors*. 2021; 21(15): 5214. Available from: https://doi.org/10.3390/s21155214.

[46] Chu SC, Tsai PW, Pan JS. Cat swarm optimization. In: *PRICAI 2006: Trends in Artificial Intelligence: 9th Pacific Rim International Conference on Artificial Intelligence*. Guilin, China: Springer; 2006. p.854-858. Available from: https://doi.org/10.1007/978-3-540-36668-3_94.

[47] Yarat S, Senan S, Orman Z. A comparative study on PSO with other metaheuristic methods. *Applying Particle Swarm Optimization: New Solutions and Cases for Optimized Portfolios*. 2021; 306: 49-72. Available from: https://doi.org/10.1007/978-3-030-70281-6_4.

[48] Musa Z, Ibrahim Z, Shapiai MI. Teaching learning-based optimization for solving CEC2014 test suite: A comparative study. In: *Intelligent Manufacturing and Mechatronics. iM3F 2023*. Singapore; 2023. p.315-325. Available from: https://doi.org/10.1007/978-981-99-8819-8_25.

[49] Yiğit H, Ürgün S, Mirjalili S. Comparison of recent metaheuristic optimization algorithms to solve the SHE optimization problem in MLI. *Neural Computing and Applications*. 2023; 35(10): 7369-7388. Available from: https://doi.org/10.1007/s00521-022-07980-1.

[50] Van Thieu N, Mirjalili S. MEALPY: An open-source library for latest meta-heuristic algorithms in Python. *Journal of Systems Architecture*. 2023; 139: 102871. Available from: https://doi.org/10.1016/j.sysarc.2023.102871.