

Article

A Novel Ensemble of Arithmetic Optimization Algorithm and Harris Hawks Optimization for Solving Industrial Engineering Optimization Problems

Jinyan Yao ¹, Yongbai Sha ¹ , Yanli Chen ^{1,*} and Xiaoying Zhao ²

¹ Key Laboratory of CNC Equipment Reliability, Ministry of Education, School of Mechanical and Aerospace Engineering, Jilin University, Changchun 130022, China; yaojy_3143@163.com (J.Y.); shayb@jlu.edu.cn (Y.S.)

² Changchun Vocational Institute of Technology, Changchun 130033, China; zxy_zhaoxy@163.com

* Correspondence: chenyanli@jlu.edu.cn

Abstract: Recently, numerous new meta-heuristic algorithms have been proposed for solving optimization problems. According to the Non-Free Lunch theorem, we learn that no single algorithm can solve all optimization problems. In order to solve industrial engineering design problems more efficiently, we, inspired by the algorithm framework of the Arithmetic Optimization Algorithm (AOA) and the Harris Hawks Optimization (HHO), propose a novel hybrid algorithm based on these two algorithms, named EAOAHHO in this paper. The pinhole imaging opposition-based learning is introduced into the proposed algorithm to increase the original population diversity and the capability to escape from local optima. Furthermore, the introduction of composite mutation strategy enhances the proposed EAOAHHO exploitation and exploration to obtain better convergence accuracy. The performance of EAOAHHO is verified on 23 benchmark functions and the IEEE CEC2017 test suite. Finally, we verify the superiority of the proposed EAOAHHO over the other advanced meta-heuristic algorithms for solving four industrial engineering design problems.



Citation: Yao, J.; Sha, Y.; Chen, Y.; Zhao, X. A Novel Ensemble of Arithmetic Optimization Algorithm and Harris Hawks Optimization for Solving Industrial Engineering Optimization Problems. *Machines* **2022**, *10*, 602. <https://doi.org/10.3390/machines10080602>

Academic Editor: Huosheng Hu

Received: 10 July 2022

Accepted: 21 July 2022

Published: 24 July 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: arithmetic optimization algorithm; Harris hawks optimization; pinhole imaging opposition-based learning; composite mutation strategy; industrial engineering design problems

1. Introduction

Optimization is the process of finding the best solution for all possible options for a particular problem [1,2]. Optimization problems have received increasing attention from different disciplines and engineering fields in recent years, and are divided into three main categories: constrained optimization, unconstrained optimization, and constrained engineering optimization problems [3]. For the majority of engineering applications, optimization is required to find the optimal results in terms of minimizing cost and energy consumption or maximizing profit, efficiency, and yield under certain constraints. With the increasing complexity of engineering optimization problems, it is extremely difficult for traditional optimization methods to satisfy the multi-constraint, high-dimensional, multi-objectives, and other characteristics of current optimization problems. Therefore, it is an essential research direction to construct an effective optimization strategy to solve the optimization problem. Compared with traditional methods, the meta-heuristic algorithms, which are characterized by simple structure, flexibility, gradient-free information, easy implementation, and strong capability of circumventing local constraints, can achieve global optimum. Currently, numerous meta-heuristic algorithms have been proposed to solve complex optimization problems, such as engineering optimization [4,5], path planning [6,7], feature selection [8], and image segmentation [9].

Meta-heuristic algorithms can be classified into three categories based on the design inspiration of the algorithms: evolutionary, physics-based, and swarm-based [10]. Evolutionary algorithms simulate Darwinian biological evolution. The solving process of such

algorithms is to initialize a random population and then evaluate the initial population using one or more operators such as crossover, mutation, and selection during the optimization process, which does not take into account the previous population execution [11]. The main common evolutionary algorithms include Genetic Algorithm (GA) [12], Differential Evolution (DE) [13], etc. The second class of algorithms is inspired by the principles of physics and physical phenomena in the universe, such as Simulated Annealing (SA) [14], Big Bang-Big Crunch (BB-BC) algorithm [15], Gravity Search Algorithm (GSA) [16], Artificial Chemical Reaction Optimization Algorithm (ACROA) [17], Galaxy-Based Search Algorithm (GBSA) [18], Multi-verse optimizer (MVO) [19], and Atom Search Optimization (ASO) [20]. The last category generally originates from the collective behavior of social creatures. The natural world is the key inspiration for most stochastic optimization strategies based on the population [21]. Particle Swarm Optimization (PSO) derived from the study on bird predation behavior is a well-known swarm intelligence algorithm [22,23]. The PSO algorithm has the advantages of simple structure and few parameter settings, but it is susceptible to being trapped in the local optimum [1]. In recent years, a large number of scholars have studied the group behavior of creatures, and better performing swarm intelligence algorithms have come out, such as Ant Lion Optimizer (ALO) [24], Salp Swarm Algorithm (SSA) [25], Grey Wolf Optimizer (GWO) [26], Harris hawks optimization (HHO) [27], Aquila Optimizer (AO) [28], Spotted Hyena Optimizer (SHO) [29], Lion Optimization Algorithm (LOA) [30], Whale Optimization Algorithm (WOA) [31], and so on.

However, the most general meta-heuristic algorithms have the disadvantages of easily falling into local optimum, slow convergence speed, and poor convergence accuracy when solving some complex optimization problems. As proved in No Free Lunch (NFL) theory, there is no one algorithm that can be used to address all optimization problems [32]. Therefore, it has been developed as a hot issue how to reasonably improve and apply algorithms to specific problems. For example, Fan et al. [33] proposed a refracted Salp Swarm Algorithm (RSSA). The basic Salp Swarm Algorithm is improved with the refracted opposition-based learning strategy, multi-leader mechanism, and adaptive conversion parameter strategy. The improved RSSA is significantly better than the basic SSA algorithm for applications in the field of structural parameter identification. Zhang et al. [1] propose an improved Harris hawks optimization based on adaptive cooperative foraging and dispersed foraging strategies, so as to remedy the shortcomings of the HHO in the exploration phase, where the population diversity is low and easily falls into local optimum. In addition, there are also some scholars who enhance the performance of the algorithm by hybridizing more than two algorithms. For example, in order to accelerate the global search phase of the existing Harris hawks Optimization and escape from local search space, Kamboj et al. [34] utilized the sine cosine algorithm to develop a hybrid variant of the Harris Hawks optimizer named the hybrid Harris Hawks-Sine Cosine Algorithm (hHHO-SCA).

In recent years, Harris Hawks Optimization (HHO) [27], a popular swarm intelligence algorithm, has been modified by a large number of scholars, and extensively applied to various optimization problems, such as engineering design [34], path planning [35], image segmentation [36], and fault diagnosis [37]. Compared with HHO, Arithmetic Optimization Algorithm (AOA) [38] proposed in 2021 is a novel meta-heuristic algorithm, which has more scope for performance enhancement. P. Arun Mozhi Devan et al. [39] propose an Arithmetic-Trigonometric Optimization Algorithm that combines the traditional sine cosine algorithm with arithmetic optimization algorithm to enhance the convergence speed and optimal search area in the exploration and exploitation phase. To achieve better convergence during exploration and exploitation phases, Namrata Panga et al. [40] proposed an improved arithmetic optimization algorithm, which has been achieved by incorporating other functions such as square, cube, sine, and cosine in the algorithms' stochastic scaling coefficient. In this paper, we proposed an ensemble of Arithmetic Optimization Algorithm and Harris Hawks Optimization, named EAOAHHO, which has greater performance than the basic AOA and HHO due to the integration mechanism of AOA and HHO, and

2.1.1. Initialization Phase

In the optimization process of AOA, a set of candidate solutions (X) is randomly generated as shown in the matrix of Equation (1). The best candidate solution in each iteration is considered to be the best solution or near-optimal solution obtained so far.

$$X = \begin{bmatrix} x_{1,1} & \cdots & \cdots & x_{1,j} & x_{1,n-1} & x_{1,n} \\ x_{2,1} & \cdots & \cdots & x_{2,j} & \cdots & x_{2,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{N-1,1} & \cdots & \cdots & x_{N-1,j} & \cdots & x_{N-1,n} \\ x_{N,1} & \cdots & \cdots & x_{N,j} & x_{N,n-1} & x_{N,n} \end{bmatrix} \quad (1)$$

Before the algorithm begins working, exploration or exploitation is selected by a factor calculated by the Mathematical Optimizer Acceleration (MOA). The Equation (2) is shown below.

$$MOA(t) = Min + t \times \left(\frac{Max - Min}{T} \right) \quad (2)$$

where $MOA(t)$ represents the i th iteration function value. t stands for the current number of iterations, and T is the maximum number of iterations. Min and Max are accelerated functions of Min. and Max. values, respectively.

2.1.2. Exploration Phase

This section describes the exploration phase of AOA. According to the arithmetic operators, mathematical calculations using the Division (D) operator or the Multiplication (M) operator have high distribution values dedicated to the exploration process. However, unlike the operators S and A, these operators are not easily accessible to the target due to their high dispersion. AOA exploration operators exploit the search field arbitrarily through many regions and seek a better alternative dependent on two key search techniques M and D search techniques as shown in Equation (3):

$$X_i(t+1) = \begin{cases} X_{best}(t) \div (MOP + eps) \times ((ub - lb) \times \mu + lb), & r_2 < 0.5 \\ X_{best}(t) \times MOP \times ((ub - lb) \times \mu + lb), & otherwise \end{cases} \quad (3)$$

where $X_i(t+1)$ denotes the i th solution of the next iteration. $X_{best}(t)$ is the position of optimum solution till now. ub and lb represent the upper bound values and the lower bound values. eps is a small integer value. μ is the control parameter to adjust the search process. In this paper, μ is set to 0.5.

$$MOP(t) = 1 - \frac{t^{\frac{1}{\alpha}}}{T^{\frac{1}{\alpha}}} \quad (4)$$

where the Math Optimizer probability (MOP) is a parameter. $Mop(t)$ is the i th iteration function value, and the dynamic parameter α is set to 5.

2.1.3. Exploitation Phase

AOA exploitation operators exploit the search field deeply through many regions and seek a better alternative dependent on two key search techniques A and S search techniques as shown in Equation (5):

$$X_i(t+1) = \begin{cases} X_{best}(t) - MOP \times ((ub - lb) \times \mu + lb), & r_3 < 0.5 \\ X_{best}(t) + MOP \times ((ub - lb) \times \mu + lb), & otherwise \end{cases} \quad (5)$$

The pseudo-code of the basic AOA is presented in Algorithm 1.

Algorithm 1 Pseudo-code of the basic AOA

```

1. Initialize the population size  $N$  and the maximum iterations  $T$ 
2. Initialize the positions of each search agent  $X_i (i = 1, 2, \dots, N)$ 
3. While  $t \leq T$ 
4.   Check if the position goes beyond the search space boundary and then adjust it
5.   Evaluate the fitness values of all search agents
6.   Set  $X_{best}$  as the position of current best solution
7.   Calculate the  $MOA$  value using Equation (2)
8.   Calculate the  $MOP$  value using Equation (4)
9.   For  $i = 1$  to  $N$ 
10.    If  $r_1 > MOA$  then //Exploration phase
11.      Update the search agent's position using Equation (3)
12.    Else //Exploitation phase
13.      Update the search agent's position using Equation (5)
14.    End If
15.  End For
16.   $t = t + 1$ 
17. End While
18. Return  $X_{best}$ 

```

2.2. Harris Hawks Optimization (HHO)

Harris hawks optimization (HHO) is a novel bionic, gradient-free, and population-based optimization algorithm proposed by Heidari et al. [27] in 2019. The inspiration for HHO derives from the foraging behavior of the Harris hawk, a famous bird of prey discovered in the southern half of Arizona, USA. Harris hawks commonly adopt a strategy known as “surprise pounce” to capture the prey. In this strategy, several hawks try to attack the potential prey (rabbit, etc.) outside the shelter simultaneously from different directions by means of good teamwork. Depending on the dynamic nature of the environment and the escaping energy of the prey, Harris hawks can exhibit a variety of chasing styles. These collaborative activities are beneficial for perplexing the escaping prey and help Harris hawks chase the detected prey to exhaustion, thus increasing its vulnerability. In this algorithm, Harris hawks are the candidate solutions and the best solution in each iteration is defined as the intended prey. Figure 2 illustrates the three main search phases of HHO including: the exploration phase, the transition from exploration to exploitation phase, and the exploitation phase, which are briefly described as follows.

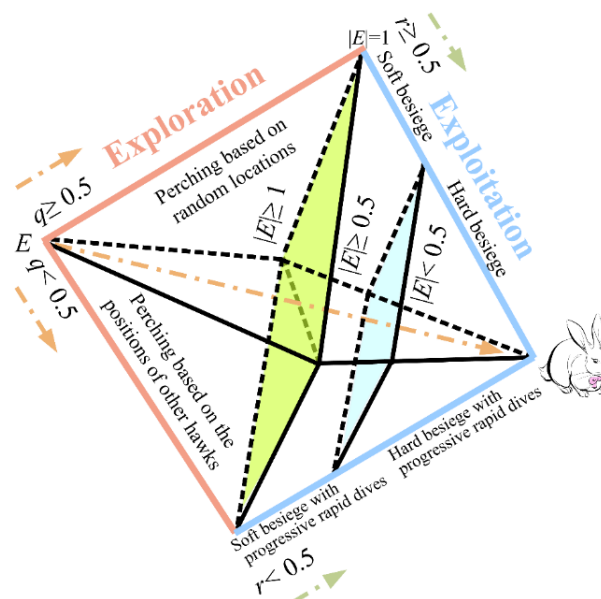


Figure 2. Different search phases of HHO.

2.2.1. Exploration Phase

Considering the habits of Harris hawks, they randomly perch on tall trees or perch according to the locations of other family members to search for the prey. To simulate these two perching mechanisms, the control parameter q is introduced in this phase to choose between them with an equal probability of 0.5. The mathematical model is as follows:

$$X_i(t+1) = \begin{cases} X_r(t) - r_4|X_r(t) - 2r_5X_i(t)|, & q \geq 0.5 \\ (X_{best}(t) - X_m(t)) - r_6(lb + r_7(ub - lb)), & q < 0.5 \end{cases} \quad (6)$$

where $X_i(t+1)$ denotes the position vector of the i -th hawk in the next iteration t , $X_i(t)$ denotes the current position of the i -th hawk, $X_r(t)$ represents a random hawk selected from the population, X_{best} implies the position of the prey (best solution obtained so far), q , r_4 , r_5 , r_6 , r_7 are random numbers between 0 and 1, ub and lb describe the upper and lower boundaries of the search range, respectively. In addition, the average position $X_m(t)$ of all hawks in the current population can be calculated using Equation (7):

$$X_m(t) = \frac{1}{N} \sum_{i=1}^N X_i(t) \quad (7)$$

where N refers to the total number of hawks (population size).

2.2.2. Transition from Exploration to Exploitation

At this stage, the algorithm transfers from the global exploration phase to the local exploitation phase according to the escaping energy of the prey, and then carries out different exploitative behaviors. As the intended prey tries to escape from the pursuit of Harris hawks, its retained energy will decrease significantly, which can be modeled by Equation (8).

$$E = 2E_0(1 - \frac{t}{T}) \quad (8)$$

where E denotes the escaping energy of the prey, T denotes the maximum iteration, and E_0 represents the initial energy of the prey, which is a random number ranged from -1 to 1 .

The variable E establishes a good basis for the smooth transition from exploration to exploitation. In the case of $|E| \geq 1$, the hawk searches the target area as much as possible to explore the location of the prey, which is also called the exploration phase. On the other hand, when $|E| < 1$, the hawk attacks the prey found in the earlier phase, implying that HHO switches to the exploitation.

2.2.3. Exploitation Phase

In the exploitation phase, four possible mechanisms are proposed to simulate the attacking process of Harris hawks. Suppose r is a random number in the range $[0, 1]$, which symbolizes the chance of a prey in successfully escaping from dangerous situations ($r < 0.5$) or not before attack ($r \geq 0.5$). Regardless of what the prey does, the hawks will perform a soft or hard besiege to capture the prey based on the escaping energy E . A soft besiege is conducted when $|E| \geq 0.5$, otherwise, the hard besiege occurs.

- Soft besiege

When $r \geq 0.5$ and $|E| \geq 0.5$, the prey still has enough energy to escape, so Harris hawks softly surround it to make the prey more exhausted before the surprise pounce is implemented. In this situation, the current position of each hawk is updated as follows:

$$X_i(t+1) = \Delta X_i(t) - E|JX_{best}(t) - X_i(t)| \quad (9)$$

$$\Delta X_i(t) = X_{best}(t) - X_i(t) \quad (10)$$

$$J = 2(1 - r_8) \quad (11)$$

where $\Delta X_i(t)$ is the difference between the location of the prey and the present position of i -th hawk in iteration t , r_8 denotes a random number between 0 and 1, J denotes the random jump strength of the prey.

- Hard besiege

If $r \geq 0.5$ and $|E| < 0.5$, the prey has a low energy to escape and the Harris hawks hardly encircle the intended prey and then attack it. This behavior is modeled as follows:

$$X_i(t+1) = X_{best}(t) - E|\Delta X_i(t)| \quad (12)$$

- Soft besiege with progressive rapid dives

When $r < 0.5$ and $|E| \geq 0.5$, the prey

$$Y = X_{best}(t) - E|JX_{best}(t) - X_i(t)| \quad (13)$$

$$Z = Y + S \times \text{Levy}(D) \quad (14)$$

where D is the dimension of the problem and S is a random vector by size $1 \times D$, while $\text{Levy}(\cdot)$ indicates the levy flight function, which is expressed as follows:

$$\text{Levy}(x) = 0.01 \times \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}}, \sigma = \left(\frac{\Gamma(1+\beta) \times \sin(\frac{\pi\beta}{2})}{\Gamma(1+\beta) \times \beta \times 2^{(\frac{\beta-1}{2})}} \right)^{1/\beta} \quad (15)$$

where u and v are random numbers within the interval $[0, 1]$, $\Gamma(\cdot)$ denotes the standard gamma function, and β is a constant value fixed to 1.5.

$$X_i(t+1) = \begin{cases} Y, & \text{if } F(Y) < F(X_i(t)) \\ Z, & \text{if } F(Z) < F(X_i(t)) \end{cases} \quad (16)$$

- Hard besiege with progressive rapid dives

In the case of $r < 0.5$ and $|E| < 0.5$, the prey is too exhausted to escape, so the hawks attempt to shrink the distance from their average position to the prey before constructing a hard besiege to attack and kill the prey. The mathematical model of this behavior can be described as follows:

$$Y = X_{best}(t) - E|JX_{best}(t) - X_m(t)| \quad (17)$$

$$Z = Y + S \times \text{Levy}(D) \quad (18)$$

$$X_i(t+1) = \begin{cases} Y, & \text{if } F(Y) < F(X_i(t)) \\ Z, & \text{if } F(Z) < F(X_i(t)) \end{cases} \quad (19)$$

where $X_m(t)$ is also calculated using Equation (7). It is worth noting that in each step, only the better position Y or Z will be considered as the next location.

The pseudo-code of the basic HHO is presented in Algorithm 2.

2.3. Pinhole Imaging Opposition-Based Learning

Pinhole imaging opposition-based learning (PIOBL), integrating both the conventional opposition-based learning (OBL) [41] and pinhole imaging principle, is a powerful technique in machine intelligence and has been successfully used to improve different basic meta-heuristic algorithms [42–44]. The main ideology of PIOBL is to simultaneously estimate the fitness values of the current position as well as its inverse position, and then the fitter one is retained to participate in the subsequent iterative calculation. The optimization procedure often starts with a stochastic initial position. If this initial position is near the global optimum, the algorithm converges quickly. On the contrary, the initial position may be far from the optimum or just at the opposite direction, which will cause it to take a quite long time to converge or even fall into a stagnant state. Therefore, PIOBL can effectively

increase the search efficiency and the probability of finding a better candidate position. The mathematical definition of PIOBL is expressed as follows.

Algorithm 2 Pseudo-code of the basic HHO

1. Initialize the population size N and the maximum iterations T
 2. Generate the random position of each hawk $X_i (i = 1, 2, \dots, N)$
 3. **While** $t \leq T$
 4. Check if any hawk position goes beyond the search space boundary and then adjust it
 5. Evaluate the fitness values of all hawks
 6. Set X_{best} as the position of current best solution
 7. **For** $i = 1$ to N
 8. Calculate the prey energy E using Equation (8)
 9. **If** $|E| \geq 1$ **then** //Exploration phase
 10. Update the hawk's position using Equation (6)
 11. **End If**
 12. **If** $|E| < 1$ **then** //Exploitation phase
 13. **If** $r \geq 0.5$ and $|E| \geq 0.5$ **then**
 14. Update the hawk's position using Equation (9)
 15. **Else if** $r \geq 0.5$ and $|E| < 0.5$ **then**
 16. Update the hawk's position using Equation (12)
 17. **Else if** $r < 0.5$ and $|E| \geq 0.5$ **then**
 18. Update the hawk's position using Equation (16)
 19. **Else if** $r < 0.5$ and $|E| < 0.5$ **then**
 20. Update the hawk's position using Equation (19)
 21. **End If**
 22. **End For**
 23. $t = t + 1$
 24. **End While**
 25. **Return** X_{best}
-

Pinhole imaging is a common optical phenomenon that specifically refers to when the light source passes diagonally from one side of the pinhole plate into the other side, an inverted image will be formed on the receiving screen at this moment. As illustrated in Figure 3, where the cardinal point O denotes the midpoint of the search range $[lb, ub]$ and the y -axis is considered as a pinhole plate. Additionally, a light source p with height h is placed at the point X_i (X_i is the current location of the i -th search agent in the population). Through pinhole imaging, the corresponding image p^* with the height h^* can be obtained and its projection on the coordinate axis is $\tilde{X}_i$ ($\tilde{X}_i$ is the opposite position of X_i). The geometrical relationship in this figure is formulated as follows:

$$\frac{(lb + ub)/2 - X_i}{\tilde{X}_i - (lb + ub)/2} = \frac{h}{h^*} \quad (20)$$

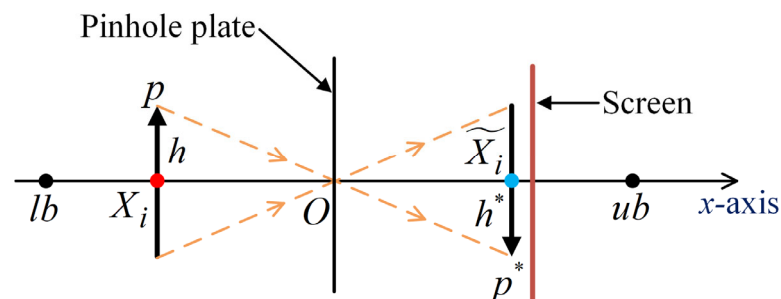


Figure 3. Pinhole imaging opposition-based learning.

Let the distance coefficient $k = h/h^*$, the opposite position $\widetilde{X}_i$ based on the theory of pinhole imaging is calculated by modifying the Equation (20):

$$\widetilde{X}_i = \frac{(lb + ub)}{2} + \frac{(lb + ub)}{2k} - \frac{X_i}{k} \quad (21)$$

Interestingly, if $k = 1$, the Equation (21) can be simplified as the standard formula of OBL strategy as follows:

$$\widetilde{X}_i = (lb + ub) - X_i \quad (22)$$

Thus, OBL is just a peculiar case of PIOBL. Compared with the former, we can adjust the coefficient k of PIOBL to attain dynamic opposite positions and a broader search range, thereby further enhancing the capability of the algorithm to avoid the local optima.

Generally, most optimization problems are multi-dimensional, so the above Equation (22) can be also extended into D-dimensional space as follows:

$$\widetilde{X}_{i,j} = \frac{(lb_j + ub_j)}{2} + \frac{(lb_j + ub_j)}{2k} - \frac{X_{i,j}}{k}, j = 1, 2, \dots, D \quad (23)$$

where $X_{i,j}$ and $\widetilde{X}_{i,j}$ are the j -dimensional components of X_i and $\widetilde{X}_i$, respectively, lb_j and ub_j are the lower and upper boundaries in the j -th dimension.

2.4. Ensemble/Composite Mutation Strategy (CMS)

The ensemble/composite mutation is a novel search strategy used to facilitate the exploration and exploitation capability of an algorithm and provide additional population diversity during the optimization [45], which was developed by Zhang et al. [46] based on the rand local mutation scheme of composite DE [47]. In this strategy, three different mutation operators: DE/rand/1/bin, DE/rand/2/bin, and DE/current-to-rand/2/bin, are executed in parallel to generate candidate positions for each of the search agents. The mathematical formula of these operators is shown as follows:

$$V_{i1} = \begin{cases} X_{R1} + F_1 \times (X_{R2} - X_{R3}), r_9 < C_1 \\ X_i, r_9 \geq C_1 \end{cases} \quad (24)$$

$$V_{i2} = \begin{cases} X_{R4} + F_2 \times (X_{R5} - X_{R6}) + F_2 \times (X_{R7} - X_{R8}), r_{10} < C_2 \\ X_i, r_{10} \geq C_2 \end{cases} \quad (25)$$

$$V_{i3} = \begin{cases} X_i + F_3 \times (X_{R9} - X_i) + F_3 \times (X_{R10} - X_{R11}), r_{11} < C_3 \\ X_i, r_{11} \geq C_3 \end{cases} \quad (26)$$

where V_{i1} , V_{i2} and V_{i3} are the new mutant positions of the i -th search agent in the population, $R1 \sim R11$ are different integer indices in the interval $[1, N]$, $r_9 \sim r_{11}$ are random numbers between 0 and 1. F_1 , F_2 and F_3 denote the scale factors, which are equal to 1.0, 0.8, and 1.0, respectively. Moreover, C_1 , C_2 and C_3 express the crossover rates, which are set as 0.1, 0.2, and 0.9, respectively [46].

After these mutant candidate positions V_{i1} , V_{i2} , and V_{i3} have been generated, the best one of them with the lowest fitness V_i is selected to update the original position of i -th search agent by using Equation (27) as follows:

$$X_i \leftarrow \begin{cases} V_i, \text{if } F(V_i) < F(X_i) \\ X_i, \text{otherwise} \end{cases} \quad (27)$$

where $F(\cdot)$ is the objective function for a given problem. If the fitness value of V_i is better than that of X_i , then X_i will be replaced by V_i , otherwise it keeps unchanged.

3. The Proposed EAOAHHO Algorithm

3.1. The Detailed Design of EAOAHHO

In this section, we give a detailed introduction to the process of the EAOAHHO. Our proposed algorithm is based on an ensemble of AOA and HHO, and introduces PIOBL strategy and CMS to enhance the performance. During each iteration, we first calculate *MOA* and *MOP* using Equations (2) and (4) in AOA. Furthermore, the PIOBL strategy is performed to update the best solution. Then, according to the value of *rand*, the next step to determine the AOA or HHO is used for updating X_i . Ultimately, we generate the mutant position V_i by CMS strategy. When the current number of iterations reaches the maximum number of iterations, the global optimal solution is outputted. The flow chart and pseudo-code of the EAOAHHO are shown in Figure 4 and Algorithm 3, respectively.

Algorithm 3 Pseudo-code of the proposed EAOAHHO

```

1. Initialize the population size  $N$  and the maximum iterations  $T$ 
2. Initialize the positions of each search agent  $X_i (i = 1, 2, \dots, N)$ 
3. While  $t \leq T$ 
4.   Check if the position goes beyond the search space boundary and then adjust it
5.   Evaluate the fitness values of all search agents
6.   Set  $X_{best}$  as the position of current best solution
7.   Calculate the MOA value using Equation (2)
8.   Calculate the MOP value using Equation (4)
9.   Generate the opposite positions of all search agents using Equation (21) and save the ones
       with better fitness // PIOBL
10.  For  $i = 1$  to  $N$ 
11.    If  $rand < 0.5$  then //AOA  $\Rightarrow rand$  is a random number between 0 and 1
12.      If  $r_1 > MOA$  then
13.        Update the position using Equation (3)
14.      Else
15.        Update the position using Equation (5)
16.      End If
17.    Else //HHO
18.      Calculate the prey energy  $E$  using Equation (8)
19.      If  $|E| \geq 1$  then
20.        Update the position using Equation (6)
21.      End If
22.      If  $|E| < 1$  then
23.        If  $r \geq 0.5$  and  $|E| \geq 0.5$  then
24.          Update the position using Equation (9)
25.        Else if  $r \geq 0.5$  and  $|E| < 0.5$  then
26.          Update the position using Equation (12)
27.        Else if  $r < 0.5$  and  $|E| \geq 0.5$  then
28.          Update the position using Equation (16)
29.        Else if  $r < 0.5$  and  $|E| < 0.5$  then
30.          Update the position using Equation (19)
31.        End If
32.      Generate the new mutant positions  $V_{i1}, V_{i2}, V_{i3}$  by using Equations (24)–(26) //CMS
33.      Set  $V_i$  as the best trial vector with the lowest fitness from  $V_{i1}, V_{i2}, V_{i3}$ 
34.      If  $F(V_i) < F(X_i)$  then
35.         $X_i = V_i$ 
36.      End If
37.    End If
38.  End For
39.   $t = t + 1$ 
40. End While
41. Return  $X_{best}$ 

```

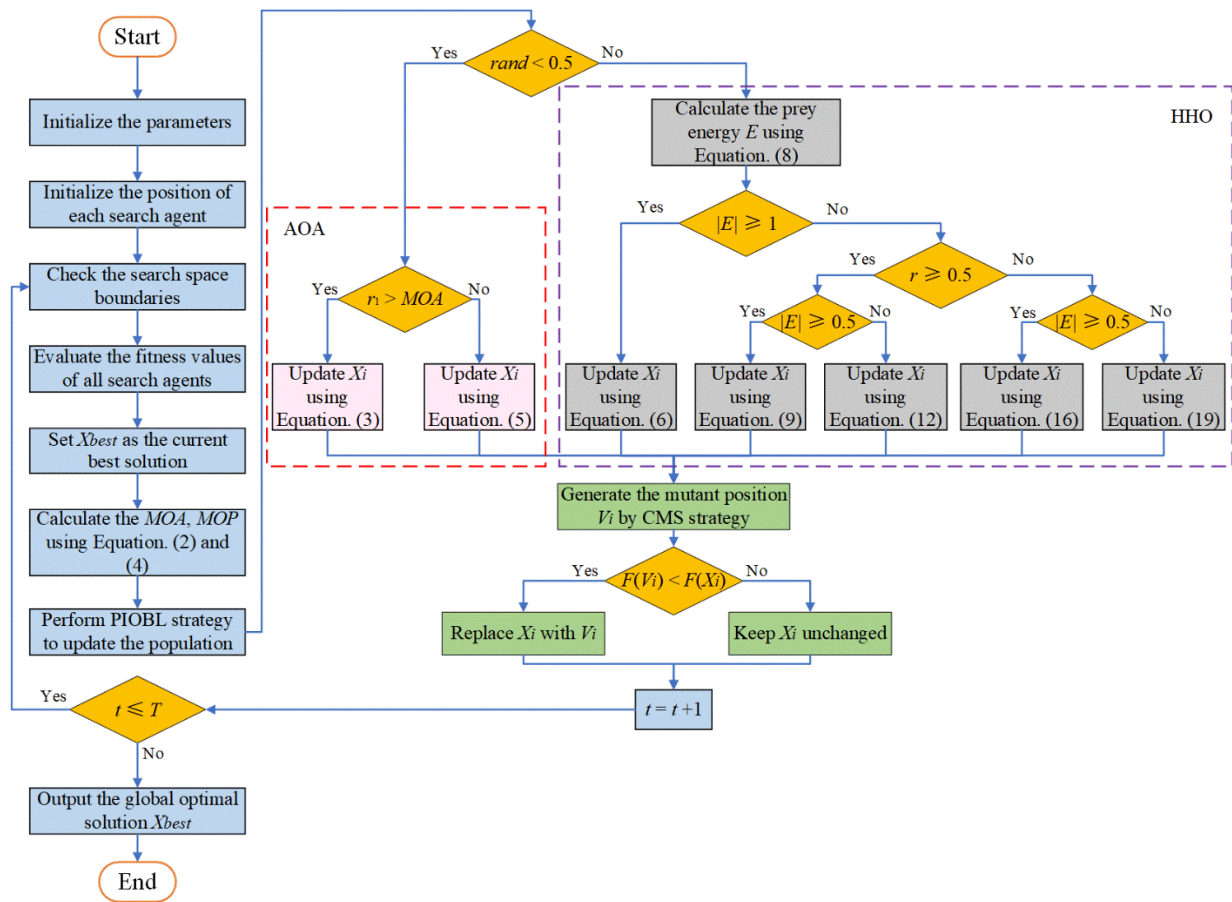


Figure 4. Flow chart of the proposed EAOAHHO algorithm.

3.2. Computational Complexity Analysis

Computational complexity is a vital metric to measure the time consumption of an algorithm during operation. The computational complexity of the proposed EAOAHHO is mainly associated with three components: initialization, fitness evaluation, and updating of positions. In the initialization phase, the positions of all search agents are generated randomly in the search domain, which needs computational complexity $O(N)$, where N is the population size. Then, in the iteration procedure, the algorithm evaluates the fitness value of each individual and updates the population positions sequentially, so the computational complexity is $O(T \times N + T \times N \times D)$, where T denotes the maximum number of iterations and D denotes the dimension of specific problems. In addition, PIOBL and CMS are introduced in the proposed algorithm to further update the position vector of each individual. The computational complexity of the two strategies is $O(T \times N \times D)$ and $O(3 \times T \times N \times D)$, respectively. Thus, the total computational complexity of EAOAHHO should be $O(N \times (1 + T + 5TD))$. As per the literature [27,38], the computational complexity of both AOA and HHO is $O(N \times (1 + T + TD))$. In addition, the complexity of all other algorithms (AO [28], WOA [31], MFO [48], TSA [49] and ChOA [50]) used in this paper is $O(N \times (1 + T + TD))$. Compared to the basic algorithms, the computational complexity of EAOAHHO increases to some extent. However, these extra time cost can significantly improve the search performance of the algorithm and acquire a better solution, which is acceptable based on NFL theory.

4. Experimental Results and Discussions

In this section, we verify the performance of the proposed EAOAHHO on 23 classical benchmark functions and 29 IEEE CEC2017 benchmark functions. These two sets of

functions verify the effectiveness of the proposed algorithm in handling simple numerical problems and complex numerical problems, respectively. To verify the superiority of the proposed algorithm, we compare EAOAHHO with the original AOA [38], HHO [27], and several other advanced meta-heuristics, namely Aquila optimizer (AO) [28], whale optimization algorithm (WOA) [31], moth-flame optimization algorithm (MFO) [48], salp swarm algorithm (SSA) [25], tunicate swarm algorithm (TSA) [49], and chimp optimization algorithm (ChOA) [50]. For all experiments, the maximum number of iterations was set to 500, the population size was 30, and each algorithm was run 30 times independently. The parameter settings for each algorithm are shown in Table 1. Furthermore, we chose the average fitness (Avg) and standard deviation (Std) as the evaluation metrics for the algorithm performance. All the experimental series are implemented in MATLAB R2017a software (version 9.2.0) with Microsoft Windows 10 system, and the hardware platform of the computer is configured as Intel® Core™ i5-10300H CPU @ 2.50 GHz and 16 GB RAM.

Table 1. Parameter settings of different algorithms.

Algorithm	Parameter Setting
AO [28]	$U = 0.00565; r_1 = 10; \omega = 0.005; \alpha = 0.1; \delta = 0.1; G_1 \in [-1, 1]; G_2 = [2, 0]$
WOA [31]	$b = 1; a_1 = [2, 0]; a_2 = [-2, -1]$
MFO [48]	$b = 1; t = [-1, 1]; a \in [-1, -2]$
SSA [25]	$c_1 = [1, 0]; c_2, c_3 \in [0, 1]$
TSA [49]	$P_{min} = 1; P_{max} = 4$
HHO [27]	$E_0 \in [-1, 1]$
AOA [38]	$\alpha = 5; \mu = 0.5; Min = 0.1; Max = 1$
ChOA [50]	$f = [2.5, 0]; m = \text{Gauss chaotic}$
EAOAHHO	$\alpha = 5; \mu = 0.5; Min = 0.1; Max = 1; E_0 \in [-1, 1], k = 12,000,$ $F_1 = 1.0; F_2 = 0.8; F_3 = 1.0; C_1 = 0.1; C_2 = 0.2; C_3 = 0.9$

4.1. Experiment 1: Classical Benchmark Functions

In this subsection, a total of 23 classical benchmark functions selected from the literature [27] are utilized to evaluate the performance of EAOAHHO. The 23 benchmark functions are classified into three different categories on the basis of their features: unimodal, multimodal, and fix-dimension multimodal. The unimodal benchmark functions ($F_1 \sim F_7$) have only one global optimal value and are usually applied to check the exploitation competence and convergence rate of the algorithm. On the other hand, the multimodal benchmark functions ($F_8 \sim F_{13}$) are characterized by multiple local minima. This type of functions is designed to examine the exploration capability and the local optima avoidance of the algorithm. The fix-dimension multimodal benchmark functions ($F_{14} \sim F_{23}$) can be regarded as a combination of the first two categories of functions but with a lower dimension, and they are used to study the stability of the algorithm between the exploration and exploitation. The expression, spatial dimension (*Dim*), search range, and a theoretical minimum of each function are outlined in Tables 2–4. Figure 5 visualizes the search space of some representative benchmark functions.

Table 2. Unimodal benchmark functions.

Function	Dim	Range	F_{min}
$F_1(x) = \sum_{i=1}^D x_i^2$	30	$[-100, 100]$	0
$F_2(x) = \sum_{i=1}^D x_i + \prod_{i=1}^D x_i $	30	$[-10, 10]$	0
$F_3(x) = \sum_{i=1}^D (\sum_{j=1}^D x_j)^2$	30	$[-100, 100]$	0
$F_4(x) = \max_i \{ x_i , 1 \leq i \leq D\}$	30	$[-100, 100]$	0
$F_5(x) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	$[-30, 30]$	0
$F_6(x) = \sum_{i=1}^D (x_i + 0.5)^2$	30	$[-100, 100]$	0
$F_7(x) = \sum_{i=1}^D ix_i^4 + \text{random}[0, 1)$	30	$[-1.28, 1.28]$	0

Table 3. Multimodal benchmark functions.

Function	Dim	Range	$F_{\min}$
$F_8(x) = \sum_{i=1}^D -x_i \sin(\sqrt{ x_i })$	30	$[-500, 500]$	$-418.9829 \times D$
$F_9(x) = \sum_{i=1}^D [x_i^2 - 10 \cos(2\pi x_i) + 10]$	30	$[-5.12, 5.12]$	0
$F_{10}(x) = -20 \exp(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^D x_i^2}) - \exp(\frac{1}{n} \sum_{i=1}^D \cos(2\pi x_i)) + 20 + e$	30	$[-32, 32]$	0
$F_{11}(x) = \frac{1}{4000} \sum_{i=1}^D x_i^2 - \prod_{i=1}^D \cos(\frac{x_i}{\sqrt{i}}) + 1$	30	$[-600, 600]$	0
$F_{12}(x) = \frac{\pi}{D} \{10 \sin(\pi y_1) + \sum_{i=1}^{D-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] + (y_D - 1)^2\} + \sum_{i=1}^D u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}, u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m x_i & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m x_i & x_i < -a \end{cases}$	30	$[-50, 50]$	0
$F_{13}(x) = 0.1 \{\sin^2(3\pi x_1) + \sum_{i=1}^D (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] + (x_D - 1)^2 [1 + \sin^2(2\pi x_D)]\} + \sum_{i=1}^D u(x_i, 5, 100, 4)$	30	$[-50, 50]$	0

Table 4. Fix-dimension multimodal benchmark functions.

Function	Dim	Range	$F_{\min}$
$F_{14}(x) = (\frac{1}{500} + \sum_{j=1}^{25} (j + \sum_{i=1}^n (x_i - a_{ij})^6)^{-1})^{-1}$	2	$[-65, 65]$	0.998
$F_{15}(x) = \sum_{i=1}^{11} [a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}]^2$	4	$[-5, 5]$	0.00030
$F_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5, 5]$	-1.0316
$F_{17}(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi}) \cos x_1 + 10$	2	$[-5, 5]$	0.398
$F_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_2 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	$[-2, 2]$	3
$F_{19}(x) = -\sum_{i=1}^4 c_i \exp(-\sum_{j=1}^3 a_{ij}(x_j - p_{ij})^2)$	3	$[-1, 2]$	-3.8628
$F_{20}(x) = -\sum_{i=1}^4 c_i \exp(-\sum_{j=1}^6 a_{ij}(x_j - p_{ij})^2)$	6	$[0, 1]$	-3.32
$F_{21}(x) = -\sum_{i=1}^5 [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0, 10]$	-10.1532
$F_{22}(x) = -\sum_{i=1}^7 [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0, 10]$	-10.4028
$F_{23}(x) = -\sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0, 10]$	-10.5363

4.1.1. Impacts of Components

In the previous chapters, we introduced the basic flow of the proposed EAOAHHO, and two strategies for modifying the original algorithm, namely PIOBL and CMS. In order to verify the effectiveness of the introduced strategies, we have designed three other algorithms in this subsection. EAOAHHO-1 is a hybrid algorithm for the basic AOA and HHO. EAOAHHO-2 denotes the EAOAHHO-1 only with PIOBL. And EAOAHHO-3 denotes the EAOAHHO-1 only with CMS. To verify the impact of each component on the original algorithm, we tested the performance of EAOAHHO-1, EAOAHHO-2, EAOAHHO-3, and EAOAHHO-1 on 23 classical benchmark functions with the same experimental parameters. The results of the experiment are shown in Table 5. In this table, we can see that EAAHO achieves optimal values in all 23 benchmark functions, and is much better than the remaining three algorithms. In addition, EAOAHHO-2 and EAOAHHO-3 achieve optimal values or solutions converging to the optimal values of EAOAHHO in

most of the benchmark functions. Moreover, both EAOAHHO-2 and EAOAHHO-3 achieve optimal solutions more often than the unimproved EAOAHHO-1, which indicates that the two modified strategies we have introduced can improve the performance of the original algorithm to some extent.

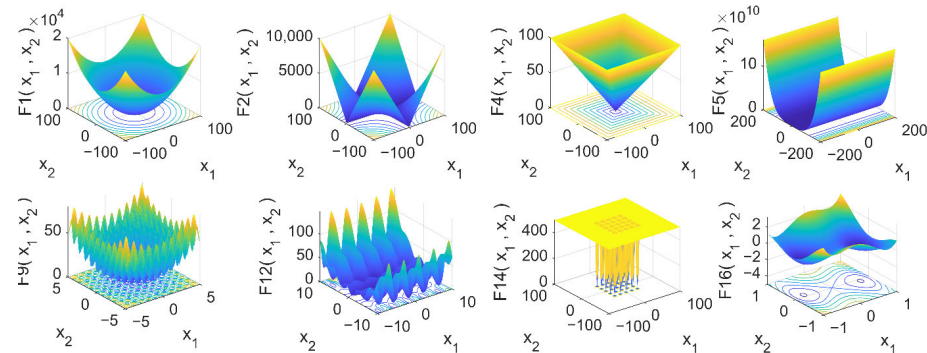


Figure 5. 3D view of the search space for some representative benchmark functions.

Table 5. Comparison results of EAOAHHO-1, EAOAHHO-2, EAOAHHO-3, and EAOAHHO.

F_n	Criteria	EAOAHHO-1	EAOAHHO-2	EAOAHHO-3	EAOAHHO
F_1	Avg	0.00×10^0	0.00×10^0	0.00×10^0	0.00×10^0
	Std	0.00×10^0	0.00×10^0	0.00×10^0	0.00×10^0
F_2	Avg	0.00×10^0	0.00×10^0	0.00×10^0	0.00×10^0
	Std	0.00×10^0	0.00×10^0	0.00×10^0	0.00×10^0
F_3	Avg	0.00×10^0	0.00×10^0	0.00×10^0	0.00×10^0
	Std	0.00×10^0	0.00×10^0	0.00×10^0	0.00×10^0
F_4	Avg	0.00×10^0	0.00×10^0	0.00×10^0	0.00×10^0
	Std	0.00×10^0	0.00×10^0	0.00×10^0	0.00×10^0
F_5	Avg	1.92×10^{-3}	1.04×10^{-3}	1.38×10^{-4}	7.13×10^{-5}
	Std	6.74×10^{-3}	1.84×10^{-3}	6.88×10^{-4}	1.00×10^{-4}
F_6	Avg	1.83×10^{-3}	3.39×10^{-4}	1.41×10^{-5}	2.09×10^{-6}
	Std	3.18×10^{-3}	5.99×10^{-4}	3.89×10^{-5}	2.62×10^{-6}
F_7	Avg	3.58×10^{-5}	2.78×10^{-5}	3.29×10^{-5}	1.98×10^{-5}
	Std	4.29×10^{-5}	2.22×10^{-5}	2.65×10^{-5}	2.20×10^{-5}
F_8	Avg	$-12,561.135$	$-12,564.854$	$-12,569.327$	$-12,569.451$
	Std	2.02×10^1	1.34×10^1	6.02×10^{-1}	1.16×10^{-1}
F_9	Avg	0.00×10^0	0.00×10^0	0.00×10^0	0.00×10^0
	Std	0.00×10^0	0.00×10^0	0.00×10^0	0.00×10^0
F_{10}	Avg	8.88×10^{-16}	8.88×10^{-16}	8.88×10^{-16}	8.88×10^{-16}
	Std	0.00×10^0	0.00×10^0	0.00×10^0	0.00×10^0
F_{11}	Avg	0.00×10^0	0.00×10^0	0.00×10^0	0.00×10^0
	Std	0.00×10^0	0.00×10^0	0.00×10^0	0.00×10^0
F_{12}	Avg	2.94×10^{-5}	3.01×10^{-6}	1.35×10^{-6}	2.21×10^{-8}
	Std	8.65×10^{-5}	4.30×10^{-6}	2.98×10^{-6}	3.77×10^{-8}
F_{13}	Avg	3.01×10^{-4}	1.57×10^{-5}	5.67×10^{-6}	1.86×10^{-7}
	Std	6.55×10^{-4}	2.95×10^{-5}	8.39×10^{-6}	4.48×10^{-7}
F_{14}	Avg	9.98×10^{-1}	9.98×10^{-1}	9.98×10^{-1}	9.98×10^{-1}
	Std	1.67×10^{-10}	1.12×10^{-10}	0.00×10^0	0.00×10^0
F_{15}	Avg	3.73×10^{-4}	3.50×10^{-4}	3.39×10^{-4}	3.07×10^{-4}
	Std	9.91×10^{-5}	4.34×10^{-5}	1.67×10^{-4}	1.34×10^{-19}
F_{16}	Avg	-1.0316	-1.0316	-1.0316	-1.0316
	Std	8.23×10^{-16}	8.63×10^{-16}	6.92×10^{-16}	6.78×10^{-16}
F_{17}	Avg	3.98×10^{-1}	3.98×10^{-1}	3.98×10^{-1}	3.98×10^{-1}
	Std	1.34×10^{-5}	7.07×10^{-6}	0.00×10^0	0.00×10^0
F_{18}	Avg	3.00×10^0	3.00×10^0	3.00×10^0	3.00×10^0
	Std	4.01×10^{-13}	1.17×10^{-14}	8.08×10^{-16}	5.34×10^{-16}
F_{19}	Avg	-3.8628	-3.8628	-3.8628	-3.8628
	Std	1.55×10^{-5}	1.04×10^{-5}	5.82×10^{-15}	2.71×10^{-15}
F_{20}	Avg	-3.2363	-3.2416	-3.2586	-3.2863
	Std	8.46×10^{-2}	8.04×10^{-2}	6.03×10^{-2}	5.54×10^{-2}
F_{21}	Avg	-9.7978	-10.1531	-10.1532	-10.1532
	Std	1.29×10^0	3.17×10^{-4}	6.90×10^{-15}	6.79×10^{-15}
F_{22}	Avg	-9.6291	-10.4028	-10.4029	-10.4029
	Std	2.33×10^0	2.32×10^{-4}	8.73×10^{-16}	7.38×10^{-16}
F_{23}	Avg	-9.1795	-10.5354	-10.3130	-10.5364
	Std	2.53×10^0	2.43×10^{-3}	1.22×10^0	2.29×10^{-15}

The best result obtained is highlighted in **bold**.

4.1.2. Comparison of EAOAHHO with Other Meta-Heuristic Algorithms

In this subsection, the basic AOA, HHO, and six well-known meta-heuristic algorithms, namely AO, WOA, MFO, SSA, TSA, and ChOA are employed to compare with EAOAHHO for 23 benchmark functions based on the numerical analysis, boxplot, convergence curve, and Wilcoxon test. The parameter settings for each algorithm have been shown in Table 1 above. After 30 independent runs on each benchmark function, comparison results attained by EAOAHHO and other algorithms on 23 benchmark functions are recorded in Table 6. According to the data in the table, we can observe that the EAOAHHO achieved the best Avg and Std on all 23 benchmark functions and even reaches the theoretical optimum on some functions, such as F_1 – F_4 , F_9 , F_{11} , and so on. There is only one global optimum for the unimodal benchmark function, therefore we can measure the exploitation capability of the algorithm with the functions F_1 – F_7 . The proposed EAOAHHO has superiority over the other eight meta-heuristic algorithms in solving such problems. Only on the function F_2 , AOA obtains the same optimal value as EAOAHHO. The above results demonstrate that the proposed algorithm has great exploitation capability. There are a large number of local optimal solutions in the multimodal benchmark functions. Therefore, the capability of the algorithm to escape local optima can be analysed by using functions F_8 – F_{13} . In most cases, EAOAHHO obtained optimal solutions that outperformed the other algorithms. EAOAHHO achieved the same optimal value as some of the algorithms, except in functions F_9 , F_{10} . In the rest of the cases, optimal solutions obtained by EAOAHHO are better than the other eight meta-heuristic algorithms. The fix-dimension multimodal benchmark functions can be utilised to examine the switching capability of the algorithm between exploration and exploitation. According to the experimental results in the table, we can notice that EAOAHHO achieves the optimal Avg and Std on functions F_{14} – F_{23} . For functions F_{16} – F_{19} , there are also other advanced meta-heuristic algorithms that achieve the same optimal values as EAOAHHO. Thus, in combination with the above results, it can be seen that the algorithm proposed in this paper is superior to the other eight meta-heuristics with regard to exploitation capability, avoidance of local extremes, and the ability to switch between exploration and exploitation.

Figure 6 shows the boxplot of EAOAHHO and other algorithms on some benchmark functions, and it can be noticed that the EAOAHHO has a relatively narrower box plot compared to other algorithms in most cases, which indicates great consistency in terms of median, maximum, and minimum values. In addition, Figure 7 illustrates the convergence curves of EAOAHHO and other algorithms on some benchmark functions. we can observe that the convergence accuracy and convergence speed of EAOAHHO outperforms other algorithms in most cases, especially for Functions F_1 – F_4 , and F_{11} . For Function F_7 and F_{13} , EAOAHHO demonstrates a great ability to escape from local optima, due to the introduction of PIOBL.

Table 6. Comparison results of EAOAHHO and other algorithms on 23 benchmark functions.

F_n	Criteria	AO	WOA	MFO	SSA	TSA	HHO	AOA	ChOA	EAOAHHO
F_1	Avg	2.49×10^{-112}	3.58×10^{-74}	2.01×10^3	1.83×10^{-7}	1.42×10^{-21}	4.69×10^{-92}	5.76×10^{-35}	9.89×10^{-6}	0.00×10^0
	Std	1.19×10^{-111}	1.25×10^{-73}	4.07×10^3	3.60×10^{-7}	2.78×10^{-21}	2.57×10^{-91}	3.16×10^{-34}	2.89×10^{-5}	0.00×10^0
F_2	Avg	1.89×10^{-56}	6.84×10^{-51}	3.38×10^1	1.75×10^0	1.00×10^{-13}	1.00×10^{-50}	0.00×10^0	4.69×10^{-5}	0.00×10^0
	Std	1.03×10^{-55}	3.35×10^{-50}	2.05×10^1	1.30×10^0	1.43×10^{-13}	4.20×10^{-50}	0.00×10^0	6.61×10^{-5}	0.00×10^0
F_3	Avg	2.35×10^{-101}	4.69×10^4	1.83×10^4	1.50×10^3	7.58×10^{-4}	4.66×10^{-75}	2.81×10^{-3}	1.99×10^2	0.00×10^0
	Std	1.29×10^{-100}	1.56×10^4	1.29×10^4	8.73×10^2	3.53×10^{-3}	2.43×10^{-74}	7.15×10^{-3}	4.14×10^2	0.00×10^0
F_4	Avg	1.92×10^{-52}	4.85×10^1	6.64×10^1	1.20×10^1	3.01×10^{-1}	8.85×10^{-48}	2.68×10^{-2}	3.32×10^{-1}	0.00×10^0
	Std	1.05×10^{-51}	3.19×10^1	7.66×10^0	4.32×10^0	4.17×10^{-1}	4.78×10^{-47}	2.04×10^{-2}	3.37×10^{-1}	0.00×10^0
F_5	Avg	3.88×10^{-3}	2.81×10^1	1.39×10^4	2.26×10^2	2.83×10^1	1.41×10^{-2}	2.84×10^1	2.89×10^1	1.64×10^{-4}
	Std	5.10×10^{-3}	4.39×10^{-1}	3.09×10^4	2.98×10^2	8.48×10^{-1}	1.46×10^{-2}	2.68×10^{-1}	1.68×10^{-1}	3.02×10^{-4}
F_6	Avg	9.27×10^{-5}	4.43×10^{-1}	3.01×10^3	5.83×10^{-7}	3.76×10^0	1.58×10^{-4}	3.14×10^0	3.66×10^0	2.26×10^{-7}
	Std	1.49×10^{-4}	2.24×10^{-1}	5.35×10^3	6.27×10^{-7}	6.28×10^{-1}	2.13×10^{-4}	3.10×10^{-1}	3.60×10^{-1}	2.34×10^{-7}
F_7	Avg	1.04×10^{-4}	4.01×10^{-3}	3.89×10^0	1.82×10^{-1}	1.02×10^{-2}	1.39×10^{-4}	5.25×10^{-5}	2.44×10^{-3}	2.61×10^{-5}
	Std	1.18×10^{-4}	5.17×10^{-3}	7.50×10^0	9.10×10^{-2}	4.09×10^{-3}	1.34×10^{-4}	4.60×10^{-5}	2.39×10^{-3}	2.58×10^{-5}
F_8	Avg	−7849.674	−10,295.569	−8411.505	−7585.108	−6021.436	−12,568.997	−5275.950	−5725.692	−12,569.445
	Std	3.91×10^3	1.99×10^3	8.90×10^2	7.78×10^2	5.06×10^2	7.71×10^{-1}	4.70×10^2	6.32×10^1	1.58×10^{-1}
F_9	Avg	0.00×10^0	0.00×10^0	1.60×10^2	5.37×10^1	1.83×10^2	0.00×10^0	0.00×10^0	7.50×10^0	0.00×10^0
	Std	0.00×10^0	0.00×10^0	3.18×10^1	2.06×10^1	3.63×10^1	0.00×10^0	0.00×10^0	7.60×10^0	0.00×10^0
F_{10}	Avg	8.88×10^{-16}	3.73×10^{-15}	1.40×10^1	2.42×10^0	1.73×10^0	8.88×10^{-16}	8.88×10^{-16}	2.00×10^1	8.88×10^{-16}
	Std	0.00×10^0	2.36×10^{-15}	7.87×10^0	1.01×10^0	1.56×10^0	0.00×10^0	0.00×10^0	9.97×10^{-4}	0.00×10^0
F_{11}	Avg	0.00×10^0	0.00×10^0	2.46×10^1	1.61×10^{-2}	1.25×10^{-2}	0.00×10^0	1.68×10^{-1}	1.94×10^{-2}	0.00×10^0
	Std	0.00×10^0	0.00×10^0	4.63×10^1	1.20×10^{-2}	1.75×10^{-2}	0.00×10^0	1.55×10^{-1}	3.16×10^{-2}	0.00×10^0
F_{12}	Avg	1.02×10^{-6}	2.62×10^{-2}	2.42×10^1	7.02×10^0	7.44×10^0	8.34×10^{-6}	5.01×10^{-1}	4.75×10^{-1}	4.37×10^{-8}
	Std	1.34×10^{-6}	2.92×10^{-2}	8.38×10^1	3.59×10^0	4.58×10^0	7.05×10^{-6}	4.96×10^{-2}	1.98×10^{-1}	9.03×10^{-8}
F_{13}	Avg	3.36×10^{-5}	5.70×10^{-1}	1.37×10^7	1.46×10^1	2.95×10^0	9.44×10^{-5}	2.83×10^0	2.76×10^0	1.73×10^{-7}
	Std	4.72×10^{-5}	2.99×10^{-1}	7.49×10^7	1.45×10^1	6.26×10^{-1}	2.29×10^{-4}	1.20×10^{-1}	1.09×10^{-1}	2.56×10^{-7}
F_{14}	Avg	2.89×10^0	3.84×10^0	1.89×10^0	1.10×10^0	8.56×10^0	1.33×10^0	1.02×10^1	1.13×10^0	9.98×10^{-1}
	Std	3.50×10^0	3.97×10^0	1.31×10^0	4.00×10^{-1}	5.08×10^0	9.47×10^{-1}	3.51×10^0	5.03×10^{-1}	0.00×10^0
F_{15}	Avg	5.21×10^{-4}	8.37×10^{-4}	1.01×10^{-3}	1.59×10^{-3}	3.94×10^{-3}	3.91×10^{-4}	1.73×10^{-2}	1.31×10^{-3}	3.07×10^{-4}
	Std	1.34×10^{-4}	5.31×10^{-4}	3.66×10^{-4}	3.55×10^{-3}	7.48×10^{-3}	2.22×10^{-4}	2.88×10^{-2}	4.50×10^{-5}	1.27×10^{-19}
F_{16}	Avg	−1.0312	−1.0316	−1.0316	−1.0316	−1.0284	−1.0316	−1.0316	−1.0316	−1.0316
	Std	3.77×10^{-4}	7.51×10^{-10}	6.78×10^{-16}	2.42×10^{-14}	9.65×10^{-3}	6.71×10^{-10}	1.05×10^{-7}	1.25×10^{-5}	6.78×10^{-16}
F_{17}	Avg	3.98×10^{-1}	3.98×10^{-1}	3.98×10^{-1}	3.98×10^{-1}	3.98×10^{-1}	3.98×10^{-1}	4.11×10^{-1}	3.99×10^{-1}	3.98×10^{-1}
	Std	3.22×10^{-4}	4.54×10^{-6}	0.00×10^0	3.65×10^{-14}	4.34×10^{-5}	1.60×10^{-5}	4.11×10^{-2}	9.51×10^{-4}	0.00×10^0
F_{18}	Avg	3.03×10^0	3.00×10^0	3.00×10^0	3.00×10^0	1.77×10^1	3.00×10^0	8.40×10^0	3.00×10^0	3.00×10^0
	Std	3.90×10^{-2}	1.85×10^{-4}	1.53×10^{-15}	2.05×10^{-13}	2.88×10^1	1.06×10^{-6}	1.10×10^1	2.37×10^{-4}	1.42×10^{-15}
F_{19}	Avg	−3.8571	−3.8541	−3.8628	−3.8628	−3.8626	−3.8578	−3.8529	−3.8528	−3.8628
	Std	4.19×10^{-3}	1.31×10^{-2}	1.65×10^{-10}	4.56×10^{-9}	3.41×10^{-4}	1.04×10^{-2}	3.52×10^{-3}	2.40×10^{-3}	2.71×10^{-15}
F_{20}	Avg	−3.1913	−3.1980	−3.2479	−3.2310	−3.2729	−3.0784	−3.0334	−2.5754	−3.2863
	Std	6.99×10^{-2}	1.68×10^{-1}	6.27×10^{-2}	6.11×10^{-2}	8.50×10^{-2}	1.12×10^{-1}	1.06×10^{-1}	4.67×10^{-1}	5.54×10^{-2}
F_{21}	Avg	−10.1395	−7.8528	−6.7116	−7.8105	−6.1621	−5.3826	−3.7548	−2.8730	−10.1532
	Std	3.27×10^{-2}	2.69×10^0	3.19×10^0	3.22×10^0	3.44×10^0	1.26×10^0	1.37×10^0	2.13×10^0	7.01×10^{-15}
F_{22}	Avg	−10.3873	−6.9247	−7.1985	−8.1209	−6.6915	−5.0843	−4.1673	−4.6058	−10.4029
	Std	2.14×10^{-2}	2.95×10^0	3.54×10^0	3.34×10^0	3.60×10^0	4.52×10^{-3}	1.49×10^0	1.66×10^0	1.23×10^{-15}
F_{23}	Avg	−10.5298	−7.1441	−7.3743	−8.5583	−6.0191	−5.2153	−3.5062	−4.3408	−10.5364
	Std	7.63×10^{-3}	3.40×10^0	3.72×10^0	3.14×10^0	3.80×10^0	1.07×10^0	1.77×10^0	1.54×10^0	2.06×10^{-15}

The best result obtained is highlighted in **bold**.

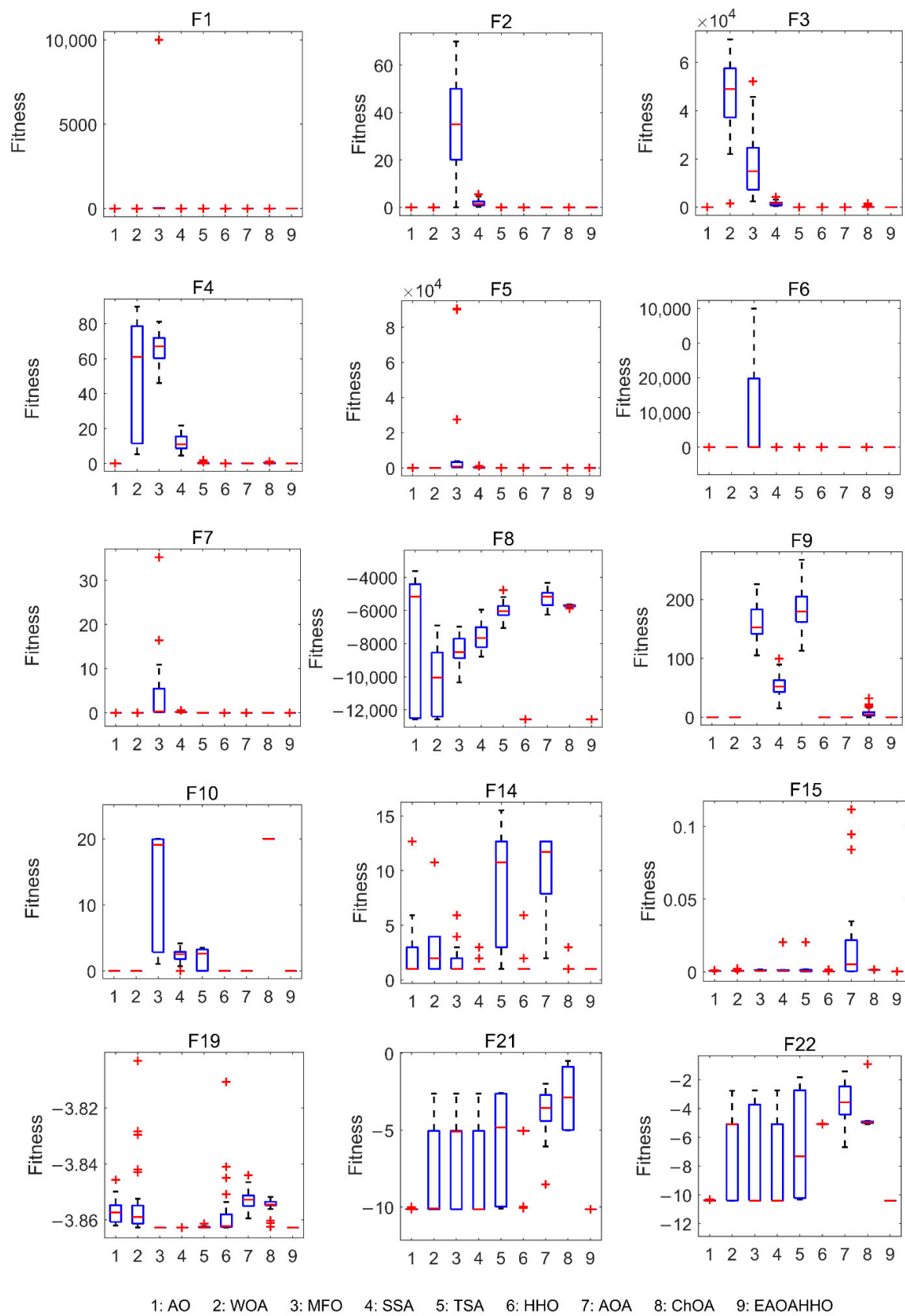


Figure 6. Boxplot of EAOAHHO and other algorithms on some benchmark functions.

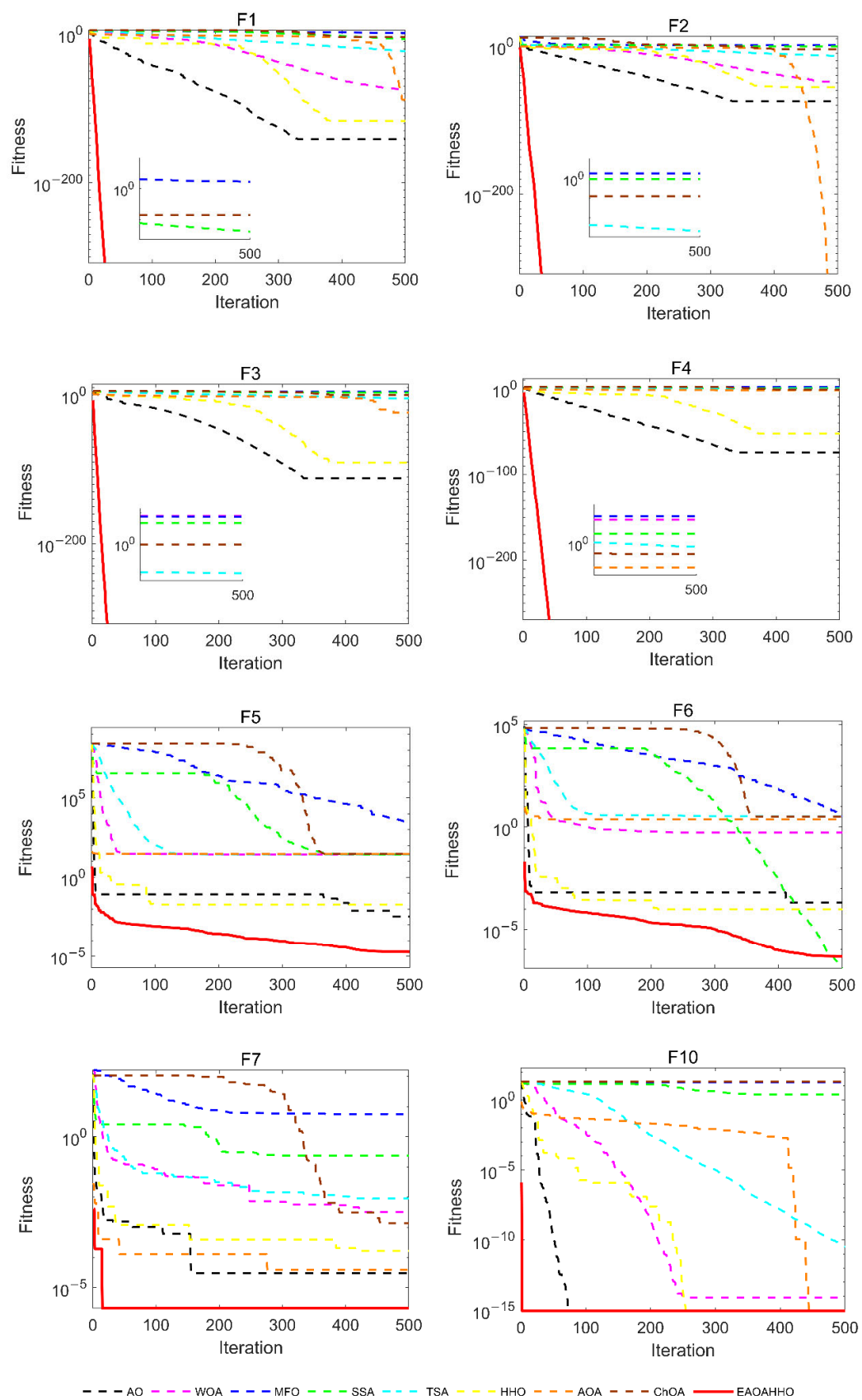


Figure 7. Cont.

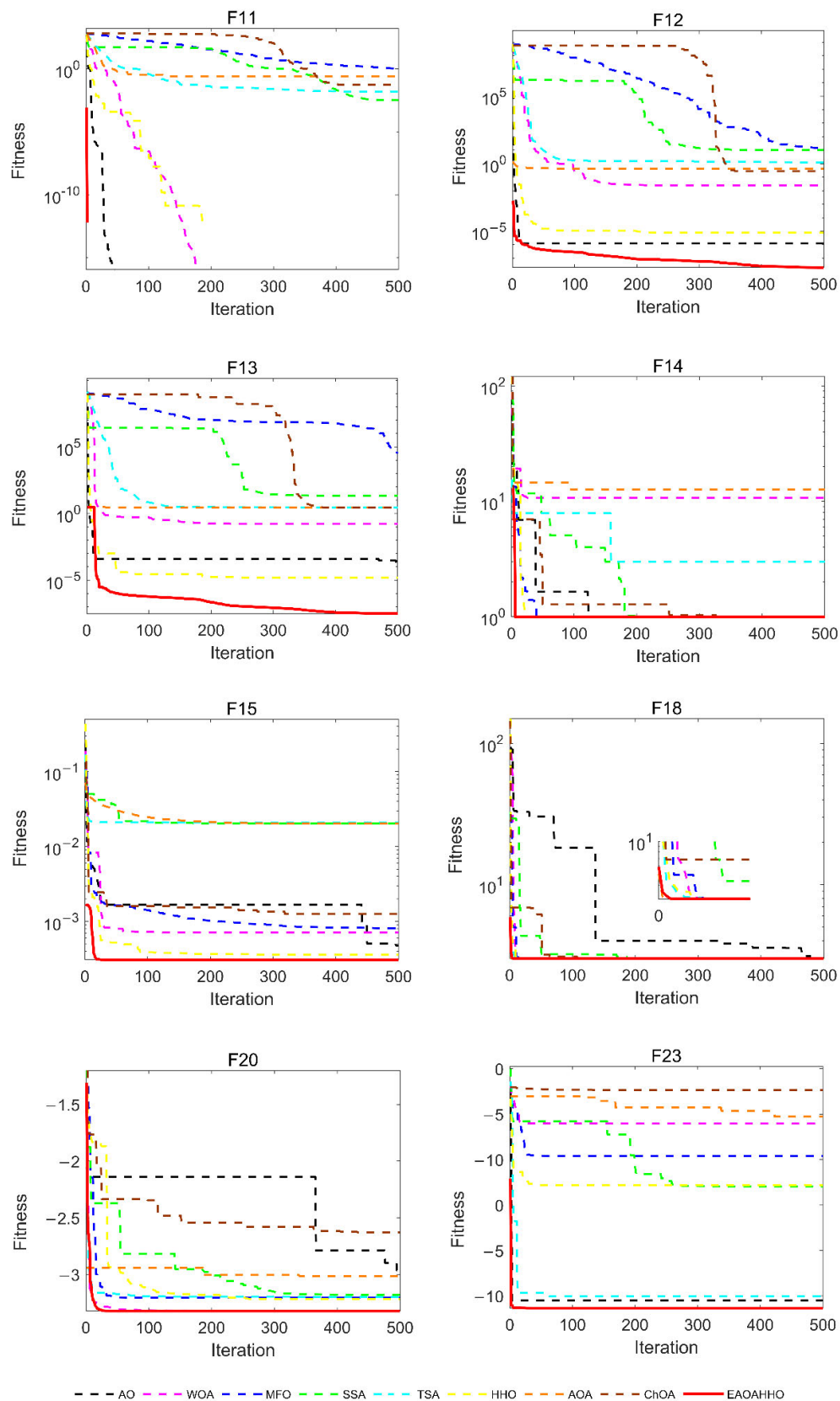


Figure 7. Convergence curves of EAOAHHO and other algorithms on some benchmark functions.

4.1.3. Scalability Test

Scalability is an essential metric to measure a newly proposed algorithm. By adjusting the dimension of the benchmark function, we can effectively judge the impact of the dimension expansion on the execution efficiency of the algorithm. In this section, we extend the dimensions of functions F_1 – F_{13} to 100, 300, and 500, and we perform scalability tests on EAOAHHO and the 8 advanced algorithms used in the previous section. The results on the 100, 300, and 500 dimensions are shown in Tables 7–9, respectively. According to the data comparison in Tables 7–9, it can be seen that as the function dimension increases, the performance of EOAHHO and the other 8 meta-heuristic algorithms gradually decreases. The main reason for the above results is that the dimensionality of the function increases, which leads to a complex search space and an increase in the number of factors that need to be optimized. In addition, the optimization results of EAOAHHO are better than the other 8 algorithms in most cases, and the gap becomes more obvious as the dimension increases. Therefore, EAOAHHO has more favorable scalability compared to the original AOA, HHO, and other 6 meta-heuristic algorithms.

4.1.4. Computational Time Analysis

In this section, we focus on the computation time for EAOAHHO and several other meta-heuristic algorithms, and the statistical results are listed in Table 10. We can observe that WOA outperforms the other eight algorithms in computation time in most cases. In addition, MFO, TSA and AOA also perform relatively well in computation time. EAOAHHO take the longest time to solve these benchmark functions. One of the main reasons for this phenomenon is that the HHO itself takes longer to compute compared with WOA, MFO and TSA. Secondly, we have introduced PIOBL and CMS strategies into the EAOAHHO, which enhance the performance of the algorithm but increase the computational cost.

4.1.5. Statistical Test

In the previous subsection, we evaluated the algorithms using Std and Avg criteria. However, the meta-heuristic algorithm obtains results randomly, and we introduce mean absolute error (MAE) and Wilcoxon rank-sum test to further assess the performance of the different algorithms.

MAE is a metric that indicates the distance between the estimated value and the true value. It is calculated by the formula Equation (28):

$$MAE = \frac{1}{NF} \sum_{i=1}^{NF} |f_i - f^*| \quad (28)$$

where, NF represents the number of test functions, f_i is the optimization result obtained by i th function, and f^* is its global optimum.

Table 11 shows the mean of EAOAHHO and other algorithms on 23 benchmark functions. The results show that the EAOAHHO ranks first and has a much smaller MAE value than the other algorithms.

Table 7. Comparison results of EAOAHHO and other algorithms on 23 benchmark functions ($D = 100$).

Fn	Criteria	AO	WOA	MFO	SSA	TSA	HHO	AOA	ChOA	EAOAHHO
F_1	Avg	1.91×10^{-106}	4.91×10^{-70}	6.26×10^4	1.42×10^3	5.30×10^{-10}	1.90×10^{-92}	2.42×10^{-2}	7.46×10^{-1}	0.00×10^0
	Std	1.05×10^{-105}	2.68×10^{-69}	1.61×10^4	4.17×10^2	9.45×10^{-10}	7.70×10^{-92}	9.72×10^{-3}	8.66×10^{-1}	0.00×10^0
F_2	Avg	3.74×10^{-69}	5.79×10^{-51}	2.36×10^2	4.79×10^1	3.29×10^{-7}	2.82×10^{-50}	3.62×10^{-42}	9.12×10^{-2}	0.00×10^0
	Std	2.03×10^{-68}	1.43×10^{-50}	3.77×10^1	5.63×10^0	3.62×10^{-7}	8.96×10^{-50}	1.98×10^{-41}	5.87×10^{-2}	0.00×10^0
F_3	Avg	5.70×10^{-106}	1.04×10^6	2.19×10^5	5.32×10^4	1.18×10^4	2.22×10^{-58}	1.02×10^0	6.81×10^4	0.00×10^0
	Std	3.12×10^{-105}	2.40×10^5	4.89×10^4	2.81×10^4	6.78×10^3	1.22×10^{-57}	9.30×10^{-1}	3.11×10^4	0.00×10^0
F_4	Avg	4.32×10^{-53}	7.98×10^1	9.31×10^1	2.79×10^1	5.54×10^1	6.78×10^{-50}	9.59×10^{-2}	7.40×10^1	0.00×10^0
	Std	1.65×10^{-52}	1.96×10^1	2.02×10^0	3.01×10^0	1.55×10^1	2.27×10^{-49}	9.35×10^{-3}	1.65×10^1	0.00×10^0
F_5	Avg	2.37×10^{-2}	9.81×10^1	1.58×10^8	1.58×10^5	9.85×10^1	5.04×10^{-2}	9.88×10^1	3.33×10^2	2.07×10^{-3}
	Std	4.26×10^{-2}	2.36×10^{-1}	6.63×10^7	8.01×10^4	2.41×10^{-1}	6.49×10^{-2}	1.75×10^{-1}	5.47×10^2	4.26×10^{-3}
F_6	Avg	6.93×10^{-4}	4.25×10^0	5.96×10^4	1.35×10^3	1.44×10^1	5.07×10^{-4}	1.82×10^1	2.27×10^1	4.87×10^{-4}
	Std	1.51×10^{-3}	1.18×10^0	1.58×10^4	5.09×10^2	1.22×10^0	6.79×10^{-4}	7.37×10^{-1}	2.76×10^0	4.83×10^{-4}
F_7	Avg	9.14×10^{-5}	4.04×10^{-3}	3.00×10^2	2.57×10^0	5.51×10^{-2}	1.39×10^{-4}	6.79×10^{-5}	2.32×10^{-2}	2.59×10^{-5}
	Std	8.92×10^{-5}	4.57×10^{-3}	1.25×10^2	5.17×10^{-1}	2.01×10^{-2}	1.78×10^{-4}	5.69×10^{-5}	1.59×10^{-2}	1.82×10^{-5}
F_8	Avg	-1.07×10^4	-3.48×10^4	-2.27×10^4	-2.15×10^4	-1.34×10^4	-4.19×10^4	-9.88×10^3	-1.81×10^4	-4.19×10^4
	Std	2.12×10^3	5.78×10^3	1.97×10^3	2.07×10^3	1.19×10^3	1.83×10^0	7.35×10^2	1.11×10^2	6.87×10^0
F_9	Avg	0.00×10^0	7.58×10^{-15}	8.58×10^2	2.47×10^2	9.64×10^2	0.00×10^0	0.00×10^0	3.66×10^1	0.00×10^0
	Std	0.00×10^0	4.15×10^{-14}	7.98×10^1	4.76×10^1	1.48×10^2	0.00×10^0	0.00×10^0	1.91×10^1	0.00×10^0
F_{10}	Avg	4.44×10^{-16}	3.76×10^{-15}	1.98×10^1	1.03×10^1	1.83×10^{-1}	4.44×10^{-16}	3.58×10^{-4}	2.00×10^1	4.44×10^{-16}
	Std	0.00×10^0	2.46×10^{-15}	1.90×10^{-1}	1.56×10^0	6.97×10^{-1}	0.00×10^0	9.32×10^{-4}	4.29×10^{-3}	0.00×10^0
F_{11}	Avg	0.00×10^0	2.11×10^{-2}	5.49×10^2	1.50×10^1	1.24×10^{-2}	0.00×10^0	5.62×10^2	3.89×10^{-1}	0.00×10^0
	Std	0.00×10^0	8.12×10^{-2}	1.50×10^2	3.47×10^0	1.58×10^{-2}	0.00×10^0	1.62×10^2	2.45×10^{-1}	0.00×10^0
F_{12}	Avg	1.12×10^{-6}	4.49×10^{-2}	2.74×10^8	3.34×10^1	1.22×10^1	1.60×10^{-6}	9.03×10^{-1}	1.20×10^0	6.28×10^{-7}
	Std	2.38×10^{-6}	2.01×10^{-2}	1.54×10^8	1.08×10^1	5.04×10^0	2.10×10^{-6}	2.28×10^{-2}	3.14×10^{-1}	9.68×10^{-7}
F_{13}	Avg	5.30×10^{-5}	2.87×10^0	6.03×10^8	4.37×10^3	1.81×10^1	1.26×10^{-4}	9.96×10^0	1.05×10^1	6.87×10^{-6}
	Std	1.17×10^{-4}	9.34×10^{-1}	3.54×10^8	6.77×10^3	2.71×10^1	1.68×10^{-4}	7.28×10^{-2}	1.61×10^0	1.03×10^{-5}

The best result obtained is highlighted in **bold**.

Table 8. Comparison results of EAOAHHO and other algorithms on 23 benchmark functions ($D = 300$).

F_n	Criteria	AO	WOA	MFO	SSA	TSA	HHO	AOA	ChOA	EAOAHHO
F_1	Avg	3.97×10^{-98}	1.32×10^{-69}	5.59×10^5	4.17×10^4	2.61×10^{-4}	5.17×10^{-94}	2.81×10^{-1}	1.31×10^2	0.00×10^0
	Std	1.80×10^{-97}	6.55×10^{-69}	2.29×10^4	4.07×10^3	1.72×10^{-4}	2.81×10^{-93}	2.50×10^{-2}	6.40×10^1	0.00×10^0
F_2	Avg	1.95×10^{-57}	5.17×10^{-48}	3.86×10^{24}	2.81×10^2	6.11×10^{-4}	1.89×10^{-48}	2.27×10^{-9}	3.85×10^0	0.00×10^0
	Std	1.07×10^{-56}	2.34×10^{-47}	2.12×10^{25}	1.34×10^1	3.54×10^{-4}	7.02×10^{-48}	1.23×10^{-8}	1.44×10^0	0.00×10^0
F_3	Avg	6.27×10^{-101}	1.09×10^7	1.93×10^6	5.06×10^5	4.54×10^5	1.14×10^{-49}	1.31×10^1	1.02×10^6	0.00×10^0
	Std	3.43×10^{-100}	3.49×10^6	2.99×10^5	2.05×10^5	8.49×10^4	4.93×10^{-49}	8.36×10^0	2.50×10^5	0.00×10^0
F_4	Avg	1.47×10^{-51}	7.74×10^1	9.82×10^1	3.70×10^1	9.82×10^1	2.57×10^{-48}	1.50×10^{-1}	9.61×10^1	0.00×10^0
	Std	7.97×10^{-51}	2.25×10^1	5.68×10^{-1}	3.46×10^0	1.31×10^0	1.11×10^{-47}	1.59×10^{-2}	1.72×10^0	0.00×10^0
F_5	Avg	7.66×10^{-2}	2.97×10^2	2.24×10^9	1.22×10^7	3.47×10^2	1.11×10^{-1}	2.99×10^2	4.10×10^4	2.24×10^{-2}
	Std	1.58×10^{-1}	2.37×10^{-1}	1.94×10^8	2.72×10^6	5.92×10^1	1.37×10^{-1}	5.36×10^{-2}	4.67×10^4	4.04×10^{-2}
F_6	Avg	1.22×10^{-3}	1.72×10^1	5.65×10^5	4.03×10^4	5.54×10^1	1.11×10^{-3}	6.62×10^1	2.27×10^2	2.98×10^{-3}
	Std	1.98×10^{-3}	4.99×10^0	2.97×10^4	3.85×10^3	1.88×10^0	1.37×10^{-3}	9.33×10^{-1}	7.57×10^1	3.80×10^{-3}
F_7	Avg	6.49×10^{-5}	3.13×10^{-3}	1.04×10^4	6.16×10^1	4.30×10^{-1}	1.75×10^{-4}	8.75×10^{-5}	5.39×10^{-1}	3.77×10^{-5}
	Std	5.08×10^{-5}	4.07×10^{-3}	1.09×10^3	9.95×10^0	1.81×10^{-1}	3.44×10^{-4}	9.67×10^{-5}	4.00×10^{-1}	4.04×10^{-5}
F_8	Avg	-2.38×10^4	-1.06×10^5	-4.77×10^4	-4.52×10^4	-2.40×10^4	-1.26×10^5	-1.77×10^4	-5.19×10^4	-1.25×10^5
	Std	6.18×10^3	1.88×10^4	3.90×10^3	3.45×10^3	1.75×10^3	8.02×10^0	1.10×10^3	4.17×10^2	3.36×10^3
F_9	Avg	0.00×10^0	0.00×10^0	3.77×10^3	1.57×10^3	3.47×10^3	0.00×10^0	1.34×10^{-7}	1.39×10^2	0.00×10^0
	Std	0.00×10^0	0.00×10^0	1.09×10^2	9.60×10^1	3.47×10^2	0.00×10^0	7.34×10^{-7}	4.18×10^1	0.00×10^0
F_{10}	Avg	4.44×10^{-16}	4.12×10^{-15}	2.01×10^1	1.37×10^1	1.96×10^{-3}	4.44×10^{-16}	6.56×10^{-3}	2.00×10^1	4.44×10^{-16}
	Std	0.00×10^0	2.55×10^{-15}	1.11×10^{-1}	3.95×10^{-1}	1.23×10^{-3}	0.00×10^0	4.81×10^{-4}	1.58×10^{-2}	0.00×10^0
F_{11}	Avg	0.00×10^0	0.00×10^0	5.05×10^3	3.64×10^2	2.68×10^{-2}	0.00×10^0	4.19×10^3	2.18×10^0	0.00×10^0
	Std	0.00×10^0	0.00×10^0	2.27×10^2	3.33×10^1	4.53×10^{-2}	0.00×10^0	6.26×10^2	6.03×10^{-1}	0.00×10^0
F_{12}	Avg	1.45×10^{-6}	8.63×10^{-2}	5.11×10^9	1.05×10^5	2.87×10^4	2.86×10^{-6}	1.05×10^0	2.73×10^3	1.32×10^{-6}
	Std	4.39×10^{-6}	3.07×10^{-2}	5.14×10^8	1.08×10^5	7.43×10^4	3.17×10^{-6}	1.73×10^{-2}	1.39×10^4	2.21×10^{-6}
F_{13}	Avg	1.26×10^{-4}	1.01×10^1	1.03×10^{10}	9.04×10^6	5.57×10^3	4.87×10^{-4}	3.01×10^1	2.96×10^4	2.99×10^{-5}
	Std	1.78×10^{-4}	2.78×10^0	7.97×10^8	3.41×10^6	6.81×10^3	7.16×10^{-4}	1.92×10^{-2}	1.42×10^5	4.18×10^{-5}

The best result obtained is highlighted in **bold**.

Table 9. Comparison results of EAOAHHO and other algorithms on 23 benchmark functions ($D = 500$).

F_n	Criteria	AO	WOA	MFO	SSA	TSA	HHO	AOA	ChOA	EAOAHHO
F_1	Avg	1.70×10^{-99}	8.31×10^{-71}	1.14×10^6	9.27×10^4	2.42×10^{-2}	7.09×10^{-95}	6.49×10^{-1}	7.28×10^2	0.00×10^0
	Std	9.28×10^{-99}	2.87×10^{-70}	4.05×10^4	4.96×10^3	1.62×10^{-2}	2.72×10^{-94}	3.72×10^{-2}	3.21×10^2	0.00×10^0
F_2	Avg	1.29×10^{-58}	4.90×10^{-47}	5.80×10^{114}	5.36×10^2	7.69×10^{-3}	8.35×10^{-49}	8.38×10^{-4}	1.23×10^1	0.00×10^0
	Std	6.61×10^{-58}	2.21×10^{-46}	3.08×10^{115}	1.42×10^1	3.63×10^{-3}	3.42×10^{-48}	1.12×10^{-3}	2.71×10^0	0.00×10^0
F_3	Avg	3.05×10^{-94}	3.14×10^7	4.74×10^6	1.31×10^6	1.32×10^6	1.16×10^{-25}	3.57×10^1	4.00×10^6	0.00×10^0
	Std	1.67×10^{-93}	9.95×10^6	8.05×10^5	5.78×10^5	2.39×10^5	6.35×10^{-25}	1.56×10^1	1.43×10^6	0.00×10^0
F_4	Avg	3.90×10^{-52}	7.55×10^1	9.89×10^1	3.99×10^1	9.92×10^1	1.09×10^{-47}	1.83×10^{-1}	9.70×10^1	0.00×10^0
	Std	2.11×10^{-51}	2.89×10^1	4.34×10^{-1}	3.52×10^0	2.49×10^{-1}	5.94×10^{-47}	1.96×10^{-2}	1.28×10^0	0.00×10^0
F_5	Avg	8.41×10^{-2}	4.96×10^2	5.01×10^9	3.63×10^7	1.08×10^5	2.00×10^{-1}	4.99×10^2	4.61×10^5	4.66×10^{-2}
	Std	1.19×10^{-1}	4.67×10^{-1}	2.44×10^8	4.50×10^6	9.22×10^4	2.87×10^{-1}	6.49×10^{-2}	4.69×10^5	1.20×10^{-1}
F_6	Avg	8.53×10^{-4}	3.07×10^1	1.16×10^6	9.36×10^4	1.03×10^2	2.48×10^{-3}	1.16×10^2	8.75×10^2	6.58×10^{-3}
	Std	1.28×10^{-3}	7.50×10^0	2.77×10^4	6.33×10^3	1.70×10^0	3.68×10^{-3}	1.29×10^0	3.74×10^2	5.05×10^{-3}
F_7	Avg	1.01×10^{-4}	4.32×10^{-3}	3.89×10^4	2.81×10^2	2.83×10^0	2.01×10^{-4}	8.69×10^{-5}	4.52×10^0	2.52×10^{-5}
	Std	1.04×10^{-4}	4.28×10^{-3}	1.96×10^3	3.47×10^1	1.24×10^0	2.34×10^{-4}	7.01×10^{-5}	3.93×10^0	3.02×10^{-5}
F_8	Avg	-4.25×10^4	-1.78×10^5	-6.13×10^4	-6.09×10^4	-3.16×10^4	-2.09×10^5	-2.27×10^4	-8.51×10^4	-2.09×10^5
	Std	1.23×10^4	3.38×10^4	4.89×10^3	5.37×10^3	2.48×10^3	3.10×10^1	1.56×10^3	5.54×10^2	2.28×10^2
F_9	Avg	3.03×10^{-14}	0.00×10^0	6.97×10^3	3.11×10^3	5.68×10^3	0.00×10^0	7.28×10^{-6}	3.23×10^2	0.00×10^0
	Std	1.66×10^{-13}	0.00×10^0	1.47×10^2	1.48×10^2	7.77×10^2	0.00×10^0	7.47×10^{-6}	5.27×10^1	0.00×10^0
F_{10}	Avg	4.44×10^{-16}	4.47×10^{-15}	2.03×10^1	1.43×10^1	1.07×10^{-2}	4.44×10^{-16}	7.98×10^{-3}	2.01×10^1	4.44×10^{-16}
	Std	0.00×10^0	2.76×10^{-15}	1.52×10^{-1}	2.19×10^{-1}	5.07×10^{-3}	0.00×10^0	3.85×10^{-4}	7.95×10^{-3}	0.00×10^0
F_{11}	Avg	0.00×10^0	0.00×10^0	1.04×10^4	8.42×10^2	4.55×10^{-2}	0.00×10^0	9.50×10^3	7.85×10^0	0.00×10^0
	Std	0.00×10^0	0.00×10^0	3.03×10^2	5.57×10^1	8.45×10^{-2}	0.00×10^0	2.90×10^3	2.87×10^0	0.00×10^0
F_{12}	Avg	1.46×10^{-6}	9.02×10^{-2}	1.21×10^{10}	1.50×10^6	3.71×10^6	2.34×10^{-6}	1.08×10^0	1.29×10^5	1.23×10^{-6}
	Std	2.89×10^{-6}	3.98×10^{-2}	5.87×10^8	7.13×10^5	3.58×10^6	2.99×10^{-6}	1.07×10^{-2}	3.52×10^5	1.75×10^{-6}
F_{13}	Avg	5.14×10^{-4}	1.66×10^1	2.25×10^{10}	3.33×10^7	1.87×10^6	5.90×10^{-4}	5.02×10^1	2.18×10^5	6.32×10^{-5}
	Std	9.17×10^{-4}	5.31×10^0	1.20×10^9	7.38×10^6	1.90×10^6	6.37×10^{-4}	2.92×10^{-2}	4.57×10^5	1.08×10^{-4}

The best result obtained is highlighted in **bold**.

Table 10. Average computational time of EAOAHHO and other algorithms on 23 benchmark functions (unit: s).

F_n	AO	WOA	MFO	SSA	TSA	HHO	AOA	ChOA	EAOAHHO
F_1	1.91×10^{-1}	6.27×10^{-2}	1.22×10^{-1}	1.20×10^{-1}	1.58×10^{-1}	1.13×10^{-1}	1.03×10^{-1}	1.53×10^0	4.22×10^{-1}
F_2	1.65×10^{-1}	6.90×10^{-2}	1.08×10^{-1}	1.06×10^{-1}	1.39×10^{-1}	8.98×10^{-2}	9.28×10^{-2}	1.48×10^0	4.18×10^{-1}
F_3	3.83×10^{-1}	1.70×10^{-1}	2.09×10^{-1}	2.17×10^{-1}	2.63×10^{-1}	3.56×10^{-1}	2.04×10^{-1}	1.53×10^0	1.44×10^0
F_4	1.72×10^{-1}	6.07×10^{-2}	1.07×10^{-1}	1.11×10^{-1}	1.51×10^{-1}	1.08×10^{-1}	9.71×10^{-2}	1.54×10^0	3.90×10^{-1}
F_5	2.03×10^{-1}	7.75×10^{-2}	1.17×10^{-1}	1.19×10^{-1}	1.64×10^{-1}	1.80×10^{-1}	1.17×10^{-1}	1.49×10^0	5.48×10^{-1}
F_6	1.58×10^{-1}	6.00×10^{-2}	9.90×10^{-2}	1.15×10^{-1}	1.48×10^{-1}	1.33×10^{-1}	9.13×10^{-2}	1.52×10^0	4.31×10^{-1}
F_7	2.87×10^{-1}	1.32×10^{-1}	1.58×10^{-1}	1.78×10^{-1}	2.02×10^{-1}	2.47×10^{-1}	1.52×10^{-1}	1.48×10^0	1.11×10^0
F_8	2.07×10^{-1}	7.37×10^{-2}	1.06×10^{-1}	1.41×10^{-1}	1.63×10^{-1}	2.06×10^{-1}	1.24×10^{-1}	1.49×10^0	5.70×10^{-1}
F_9	1.77×10^{-1}	7.91×10^{-2}	1.23×10^{-1}	1.08×10^{-1}	1.73×10^{-1}	1.64×10^{-1}	9.99×10^{-2}	1.53×10^0	3.90×10^{-1}
F_{10}	1.86×10^{-1}	7.26×10^{-2}	1.15×10^{-1}	1.23×10^{-1}	1.71×10^{-1}	1.57×10^{-1}	1.05×10^{-1}	1.52×10^0	4.24×10^{-1}
F_{11}	1.97×10^{-1}	7.68×10^{-2}	1.54×10^{-1}	1.35×10^{-1}	1.75×10^{-1}	1.75×10^{-1}	1.13×10^{-1}	1.53×10^0	7.94×10^{-1}
F_{12}	5.00×10^{-1}	2.10×10^{-1}	2.52×10^{-1}	2.71×10^{-1}	2.99×10^{-1}	5.27×10^{-1}	2.46×10^{-1}	1.59×10^0	1.39×10^0
F_{13}	5.10×10^{-1}	2.15×10^{-1}	2.65×10^{-1}	2.94×10^{-1}	3.03×10^{-1}	5.16×10^{-1}	2.64×10^{-1}	1.58×10^0	1.45×10^0
F_{14}	7.77×10^{-1}	3.74×10^{-1}	3.45×10^{-1}	3.98×10^{-1}	3.46×10^{-1}	8.94×10^{-1}	3.77×10^{-1}	4.50×10^{-1}	2.63×10^0
F_{15}	1.23×10^{-1}	4.17×10^{-2}	5.01×10^{-2}	7.04×10^{-2}	4.55×10^{-2}	1.19×10^{-1}	4.64×10^{-2}	2.84×10^{-1}	4.18×10^{-1}
F_{16}	1.19×10^{-1}	4.16×10^{-2}	4.11×10^{-2}	6.78×10^{-2}	4.03×10^{-2}	1.11×10^{-1}	3.76×10^{-2}	1.55×10^{-1}	3.93×10^{-1}
F_{17}	1.18×10^{-1}	3.29×10^{-2}	3.94×10^{-2}	3.30×10^{-2}	3.15×10^{-2}	1.11×10^{-1}	4.16×10^{-2}	1.56×10^{-1}	3.80×10^{-1}
F_{18}	1.06×10^{-1}	3.22×10^{-2}	4.30×10^{-2}	6.91×10^{-2}	3.16×10^{-2}	9.89×10^{-2}	3.70×10^{-2}	1.43×10^{-1}	3.51×10^{-1}
F_{19}	1.40×10^{-1}	4.08×10^{-2}	5.88×10^{-2}	7.07×10^{-2}	4.52×10^{-2}	1.38×10^{-1}	4.82×10^{-2}	2.06×10^{-1}	4.33×10^{-1}
F_{20}	1.52×10^{-1}	5.13×10^{-2}	5.83×10^{-2}	8.53×10^{-2}	6.71×10^{-2}	1.48×10^{-1}	5.89×10^{-2}	3.86×10^{-1}	4.49×10^{-1}
F_{21}	1.67×10^{-1}	4.73×10^{-2}	5.97×10^{-2}	9.26×10^{-2}	6.01×10^{-2}	1.61×10^{-1}	6.11×10^{-2}	2.91×10^{-1}	5.04×10^{-1}
F_{22}	1.90×10^{-1}	6.34×10^{-2}	6.14×10^{-2}	9.65×10^{-2}	6.88×10^{-2}	1.91×10^{-1}	5.81×10^{-2}	2.84×10^{-1}	5.63×10^{-1}
F_{23}	2.21×10^{-1}	7.60×10^{-2}	8.47×10^{-2}	1.05×10^{-1}	8.09×10^{-2}	1.93×10^{-1}	8.66×10^{-2}	3.08×10^{-1}	6.70×10^{-1}

The best result obtained is highlighted in **bold**.

Table 11. Mean absolute error of EAOAHHO and other algorithms on 23 benchmark functions.

Algorithms	MAE	Rank
AO	205.300427	3
WOA	2141.9002	8
MFO	597,465.9	9
SSA	296.0374779	4
TSA	296.09092	5
HHO	0.7170996	2
AOA	320.163464	7
CHOA	309.871379	6
EAOAHHO	0.003309	1

In addition, we introduce the Wilcoxon rank-sum test in this subsection to calculate and count the differences between the different algorithms. In this study, we set the significance level at 0.05. The obtained p -values and statistical results of EAOAHHO using the Wilcoxon rank-sum test are listed in Table 12. In this table, the “+” sign denotes that the EAOAHHO has a better performance than the comparison algorithms, the “−” sign denotes that the EAOAHHO has worse performance than the comparison algorithms, and “=” represents the EAOAHHO is similar to the algorithms in the comparison. The last three rows of this table indicate the times of EAOAHHO obtained “+”, “=”, and “−” compared with each algorithm in the Wilcoxon rank-sum test. Among the 23 benchmark functions, EAOAHHO outperformed SSA, TSA, and ChOA 23 times; outperformed WOA 21 times; and outperformed AO, MFO, HHO, and AOA 20 times. Given the above statistics, it is evident that the EAOAHHO proposed in this paper is significantly enhanced compared to the original AOA and HHO and is the best optimizer among the eight advanced algorithms.

4.2. Experiment 2: IEEE CEC2017 Test Suite

In Experiment 1, we verify the excellent performance of the proposed EAOAHHO in solving simple problems with 23 classical benchmark functions. In order to further verify whether EAOAHHO is still superior to the original AOA, HHO, and other advanced meta-heuristics when dealing with complex problems, we choose IEEE CEC 2017 test suite [51], a set of more challenging test functions listed in Table 13, for Experiments 2. The experimental method is the same as the previous section. Comparison results of EAOAHHO and other algorithms on IEEE CEC2017 test suite are shown in Table 14. In addition, we counted the experimental results and plotted the ranking diagram of the 9 algorithms shown in Figure 8. Combining the results in Table 14 and Figure 8, we can find that EAOAHHO ranks first in the vast majority of cases. For functions CEC-5, CEC-7, CEC-8, CEC-9, CEC-16, CEC-20, and CEC-23, the proposed algorithm ranks fourth. Third place for functions CEC-10 and CEC-26. For functions CEC-17 and CEC-24, the proposed algorithm ranks second.

Table 12. *p*-values of the Wilcoxon rank-sum test between EAOAHHO and other algorithms.

F_n	EAOAHHO vs. AO	EAOAHHO vs. WOA	EAOAHHO vs. MFO	EAOAHHO vs. SSA	EAOAHHO vs. TSA	EAOAHHO vs. HHO	EAOAHHO vs. AOA	EAOAHHO vs. ChOA
F_1	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
F_2	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	NaN	1.21×10^{-12}
F_3	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
F_4	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
F_5	5.46×10^{-6}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	1.43×10^{-8}	3.02×10^{-11}	3.02×10^{-11}
F_6	1.69×10^{-9}	3.02×10^{-11}	3.02×10^{-11}	3.82×10^{-10}	3.02×10^{-11}	4.94×10^{-5}	3.02×10^{-11}	3.02×10^{-11}
F_7	5.56×10^{-4}	4.50×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	2.68×10^{-6}	1.44×10^{-2}	4.50×10^{-11}
F_8	3.00×10^{-11}	3.00×10^{-11}	3.00×10^{-11}	3.00×10^{-11}	3.00×10^{-11}	3.07×10^{-6}	3.00×10^{-11}	3.00×10^{-11}
F_9	NaN	NaN	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	NaN	NaN	1.21×10^{-12}
F_{10}	NaN	8.73×10^{-8}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	NaN	NaN	1.21×10^{-12}
F_{11}	NaN	NaN	1.21×10^{-12}	1.21×10^{-12}	8.87×10^{-7}	NaN	1.21×10^{-12}	1.21×10^{-12}
F_{12}	1.10×10^{-8}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	4.98×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
F_{13}	6.70×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	4.18×10^{-9}	3.02×10^{-11}	3.02×10^{-11}
F_{14}	1.21×10^{-12}	1.21×10^{-12}	6.58×10^{-5}	7.56×10^{-13}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
F_{15}	2.82×10^{-11}	2.82×10^{-11}	2.81×10^{-11}	2.82×10^{-11}	2.82×10^{-11}	2.82×10^{-11}	2.82×10^{-11}	2.82×10^{-11}
F_{16}	1.21×10^{-12}	1.21×10^{-12}	NaN	1.21×10^{-12}	1.21×10^{-12}	4.57×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
F_{17}	1.21×10^{-12}	1.21×10^{-12}	NaN	1.93×10^{-10}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
F_{18}	1.26×10^{-11}	1.26×10^{-11}	4.90×10^{-5}	1.26×10^{-11}	1.26×10^{-11}	1.26×10^{-11}	1.26×10^{-11}	1.26×10^{-11}
F_{19}	1.21×10^{-12}	1.21×10^{-12}	NaN	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
F_{20}	1.93×10^{-6}	9.73×10^{-7}	1.06×10^{-2}	2.43×10^{-8}	1.41×10^{-4}	1.60×10^{-10}	1.01×10^{-11}	2.59×10^{-11}
F_{21}	4.08×10^{-12}	4.08×10^{-12}	6.56×10^{-7}	4.08×10^{-12}	4.08×10^{-12}	4.08×10^{-12}	4.08×10^{-12}	4.08×10^{-12}
F_{22}	1.45×10^{-11}	1.45×10^{-11}	1.55×10^{-6}	1.45×10^{-11}	1.45×10^{-11}	1.45×10^{-11}	1.45×10^{-11}	1.45×10^{-11}
F_{23}	3.15×10^{-12}	3.15×10^{-12}	2.12×10^{-10}	3.15×10^{-12}	3.15×10^{-12}	3.15×10^{-12}	3.15×10^{-12}	3.15×10^{-12}
+	20	21	20	23	23	20	20	23
=	3	2	3	0	0	3	3	0
−	0	0	0	0	0	0	0	0

Table 13. Descriptions of IEEE CEC2017 test suite.

Function	Name	Dim	Range	$F_{\min}$
Unimodal functions				
CEC-01	Shifted and Rotated Bent Cigar Function	10	$[-100, 100]$	100
CEC-03	Shifted and Rotated Zakharov Function	10	$[-100, 100]$	300
Simple Multimodal Functions				
CEC-04	Shifted and Rotated Rosenbrock's Function	10	$[-100, 100]$	400
CEC-05	Shifted and Rotated Rastrigin's Function	10	$[-100, 100]$	500
CEC-06	Shifted and Rotated Expanded Scaffer's F6 Function	10	$[-100, 100]$	600
CEC-07	Shifted and Rotated Lunacek Bi_Rastrigin Function	10	$[-100, 100]$	700
CEC-08	Shifted and Rotated Non-Continuous Rastrigin's Function	10	$[-100, 100]$	800
CEC-09	Shifted and Rotated Levy Function	10	$[-100, 100]$	900
CEC-10	Shifted and Rotated Schwefel's Function	10	$[-100, 100]$	1000
Hybrid Functions				
CEC-11	Hybrid Function 1 ($N = 3$)	10	$[-100, 100]$	1100
CEC-12	Hybrid Function 2 ($N = 3$)	10	$[-100, 100]$	1200
CEC-13	Hybrid Function 3 ($N = 3$)	10	$[-100, 100]$	1300
CEC-14	Hybrid Function 4 ($N = 4$)	10	$[-100, 100]$	1400
CEC-15	Hybrid Function 5 ($N = 4$)	10	$[-100, 100]$	1500
CEC-16	Hybrid Function 6 ($N = 4$)	10	$[-100, 100]$	1600
CEC-17	Hybrid Function 6 ($N = 5$)	10	$[-100, 100]$	1700
CEC-18	Hybrid Function 6 ($N = 5$)	10	$[-100, 100]$	1800
CEC-19	Hybrid Function 6 ($N = 5$)	10	$[-100, 100]$	1900
CEC-20	Hybrid Function 6 ($N = 6$)	10	$[-100, 100]$	2000
Composition Functions				
CEC-21	Composition Function 1 ($N = 3$)	10	$[-100, 100]$	2100
CEC-22	Composition Function 2 ($N = 3$)	10	$[-100, 100]$	2200
CEC-23	Composition Function 3 ($N = 4$)	10	$[-100, 100]$	2300
CEC-24	Composition Function 4 ($N = 4$)	10	$[-100, 100]$	2400
CEC-25	Composition Function 5 ($N = 5$)	10	$[-100, 100]$	2500
CEC-26	Composition Function 6 ($N = 5$)	10	$[-100, 100]$	2600
CEC-27	Composition Function 7 ($N = 6$)	10	$[-100, 100]$	2700
CEC-28	Composition Function 8 ($N = 6$)	10	$[-100, 100]$	2800
CEC-29	Composition Function 9 ($N = 3$)	10	$[-100, 100]$	2900
CEC-30	Composition Function 10 ($N = 3$)	10	$[-100, 100]$	3000

Table 14. Comparison results of EAOAHHO and other algorithms on IEEE CEC2017 test suite.

F_n	Metric	AO	WOA	MFO	SSA	TSA	HHO	AOA	ChOA	EAOAHHO
CEC-1	Avg	3.78×10^7	5.79×10^7	1.41×10^8	4.01×10^3	3.19×10^9	9.59×10^6	1.07×10^{10}	2.32×10^9	3.30×10^3
	Std	9.04×10^7	6.42×10^7	4.37×10^8	3.49×10^3	3.75×10^9	4.71×10^7	4.38×10^9	1.70×10^9	3.18×10^3
CEC-3	Avg	2.31×10^3	7.56×10^3	8.82×10^3	3.25×10^2	9.41×10^3	6.46×10^2	1.29×10^4	5.01×10^3	3.00×10^2
	Std	7.94×10^2	5.10×10^3	7.52×10^3	1.77×10^0	5.44×10^3	2.71×10^2	3.42×10^3	2.33×10^3	4.29×10^{-2}
CEC-4	Avg	4.23×10^2	4.88×10^2	4.28×10^2	4.10×10^2	6.31×10^2	4.34×10^2	1.20×10^3	6.52×10^2	4.06×10^2
	Std	2.30×10^1	6.83×10^1	3.91×10^1	1.77×10^1	3.02×10^2	3.48×10^1	5.45×10^2	1.88×10^2	1.20×10^1
CEC-5	Avg	5.34×10^2	5.64×10^2	5.30×10^2	5.24×10^2	5.55×10^2	5.57×10^2	5.60×10^2	5.65×10^2	5.36×10^2
	Std	1.58×10^1	2.12×10^1	1.13×10^1	9.97×10^0	1.81×10^1	1.94×10^1	1.65×10^1	1.61×10^1	1.63×10^1
CEC-6	Avg	6.20×10^2	6.36×10^2	6.37×10^2	6.13×10^2	6.38×10^2	6.40×10^2	6.38×10^2	6.34×10^2	6.04×10^2
	Std	5.79×10^0	1.31×10^1	1.29×10^1	8.53×10^0	1.32×10^1	1.38×10^1	5.93×10^0	1.26×10^1	4.57×10^0
CEC-7	Avg	7.58×10^2	7.97×10^2	7.39×10^2	7.47×10^2	8.00×10^2	7.97×10^2	8.04×10^2	8.06×10^2	7.78×10^2
	Std	1.48×10^1	2.52×10^1	1.10×10^1	1.47×10^1	2.85×10^1	2.05×10^1	1.82×10^1	1.48×10^1	1.40×10^1
CEC-8	Avg	8.28×10^2	8.47×10^2	8.34×10^2	8.30×10^2	8.47×10^2	8.32×10^2	8.36×10^2	8.45×10^2	8.34×10^2
	Std	8.67×10^0	2.14×10^1	1.28×10^1	1.38×10^1	1.34×10^1	1.04×10^1	9.46×10^0	8.12×10^0	9.29×10^0
CEC-9	Avg	1.04×10^3	1.61×10^3	1.07×10^3	1.10×10^3	1.56×10^3	1.53×10^3	1.41×10^3	1.63×10^3	1.38×10^3
	Std	6.65×10^1	3.90×10^2	2.25×10^2	3.35×10^2	5.70×10^2	2.71×10^2	1.97×10^2	2.59×10^2	1.92×10^2
CEC-10	Avg	1.97×10^3	2.25×10^3	2.03×10^3	1.93×10^3	2.29×10^3	2.07×10^3	2.28×10^3	2.94×10^3	2.01×10^3
	Std	2.24×10^2	3.91×10^2	4.25×10^2	1.41×10^2	2.44×10^2	2.90×10^2	2.94×10^2	3.33×10^2	2.76×10^2
CEC-11	Avg	1.28×10^3	1.24×10^3	1.26×10^3	1.20×10^3	4.41×10^3	1.19×10^3	2.91×10^3	1.40×10^3	1.15×10^3
	Std	1.60×10^2	8.00×10^1	3.57×10^2	7.48×10^1	3.44×10^3	7.67×10^1	1.91×10^3	1.19×10^2	6.10×10^1
CEC-12	Avg	4.49×10^6	6.43×10^6	1.81×10^6	3.85×10^6	2.11×10^7	4.03×10^6	3.17×10^8	4.12×10^7	2.10×10^3
	Std	5.36×10^6	6.73×10^6	4.49×10^6	3.97×10^6	1.05×10^8	4.25×10^6	3.96×10^8	1.03×10^8	1.38×10^3
CEC-13	Avg	1.74×10^4	1.80×10^4	1.15×10^4	1.53×10^4	5.23×10^6	1.89×10^4	1.20×10^4	4.61×10^4	1.33×10^3
	Std	1.25×10^4	1.17×10^4	1.10×10^4	1.45×10^4	2.26×10^7	1.19×10^4	8.05×10^3	2.28×10^4	9.55×10^1
CEC-14	Avg	2.63×10^3	4.08×10^3	5.11×10^3	2.87×10^3	3.61×10^3	2.40×10^3	8.14×10^3	6.62×10^3	1.45×10^3
	Std	1.05×10^3	2.15×10^3	5.97×10^3	3.01×10^3	2.15×10^3	1.27×10^3	7.87×10^3	1.42×10^3	2.90×10^1
CEC-15	Avg	6.50×10^3	1.05×10^4	1.22×10^4	8.09×10^3	7.19×10^3	6.75×10^3	1.90×10^4	1.95×10^4	1.53×10^3
	Std	3.22×10^3	7.49×10^3	1.94×10^4	7.25×10^3	5.46×10^3	3.70×10^3	7.59×10^3	8.50×10^3	3.56×10^1
CEC-16	Avg	1.88×10^3	1.92×10^3	1.81×10^3	1.79×10^3	1.93×10^3	1.91×10^3	2.08×10^3	1.94×10^3	1.89×10^3
	Std	1.28×10^2	1.55×10^2	1.60×10^2	1.56×10^2	1.62×10^2	1.65×10^2	1.25×10^2	1.03×10^2	1.44×10^2
CEC-17	Avg	1.78×10^3	1.82×10^3	1.81×10^3	1.85×10^3	1.88×10^3	1.84×10^3	1.91×10^3	1.80×10^3	1.79×10^3
	Std	2.29×10^1	6.76×10^1	6.67×10^1	4.95×10^1	1.08×10^2	1.01×10^2	1.17×10^2	2.68×10^1	3.43×10^1
CEC-18	Avg	2.54×10^4	1.95×10^4	2.06×10^4	1.99×10^4	2.91×10^4	1.67×10^4	2.70×10^6	1.02×10^5	1.83×10^3
	Std	1.46×10^4	1.32×10^4	1.35×10^4	9.38×10^3	1.61×10^4	1.25×10^4	1.22×10^7	6.72×10^4	2.25×10^1

Table 14. Cont.

F_n	Metric	AO	WOA	MFO	SSA	TSA	HHO	AOA	ChOA	EAOAHHO
CEC-19	Avg	2.80×10^4	1.09×10^5	1.68×10^4	6.91×10^3	1.15×10^5	2.17×10^4	8.40×10^4	2.48×10^4	1.90×10^3
	Std	4.44×10^4	1.50×10^5	2.34×10^4	4.92×10^3	3.71×10^5	2.72×10^4	8.60×10^4	9.90×10^3	8.92×10^0
CEC-20	Avg	2.13×10^3	2.18×10^3	2.10×10^3	2.14×10^3	2.19×10^3	2.19×10^3	2.15×10^3	2.26×10^3	2.15×10^3
	Std	4.91×10^1	7.99×10^1	6.35×10^1	5.70×10^1	8.55×10^1	7.53×10^1	7.32×10^1	7.61×10^1	6.13×10^1
CEC-21	Avg	2.31×10^3	2.34×10^3	2.32×10^3	2.30×10^3	2.34×10^3	2.34×10^3	2.34×10^3	2.31×10^3	2.29×10^3
	Std	4.76×10^1	3.99×10^1	4.62×10^1	5.18×10^1	5.20×10^1	4.59×10^1	7.82×10^1	6.20×10^1	3.48×10^1
CEC-22	Avg	2.31×10^3	2.42×10^3	2.31×10^3	2.30×10^3	2.60×10^3	2.42×10^3	3.03×10^3	3.63×10^3	2.31×10^3
	Std	5.11×10^0	4.15×10^2	4.35×10^1	2.09×10^1	3.28×10^2	4.53×10^2	2.58×10^2	7.99×10^2	2.21×10^1
CEC-23	Avg	2.64×10^3	2.65×10^3	2.62×10^3	2.63×10^3	2.71×10^3	2.69×10^3	2.75×10^3	2.66×10^3	2.65×10^3
	Std	1.51×10^1	2.35×10^1	1.00×10^1	1.08×10^1	4.42×10^1	3.89×10^1	4.02×10^1	9.06×10^0	1.38×10^1
CEC-24	Avg	2.76×10^3	2.77×10^3	2.75×10^3	2.74×10^3	2.81×10^3	2.82×10^3	2.84×10^3	2.80×10^3	2.75×10^3
	Std	4.90×10^1	6.68×10^1	4.92×10^1	5.34×10^1	7.58×10^1	4.15×10^1	8.22×10^1	1.63×10^1	4.69×10^1
CEC-25	Avg	2.95×10^3	2.96×10^3	2.94×10^3	2.93×10^3	3.03×10^3	2.93×10^3	3.36×10^3	3.05×10^3	2.93×10^3
	Std	3.20×10^1	5.60×10^1	2.49×10^1	2.89×10^1	9.88×10^1	2.90×10^1	2.31×10^2	8.86×10^1	2.48×10^1
CEC-26	Avg	3.07×10^3	3.45×10^3	3.09×10^3	2.92×10^3	3.92×10^3	3.79×10^3	3.97×10^3	4.01×10^3	3.07×10^3
	Std	1.79×10^2	5.34×10^2	2.88×10^2	2.06×10^2	4.47×10^2	6.09×10^2	3.09×10^2	3.33×10^2	3.02×10^2
CEC-27	Avg	3.11×10^3	3.15×10^3	3.09×10^3	3.10×10^3	3.17×10^3	3.17×10^3	3.26×10^3	3.14×10^3	3.09×10^3
	Std	8.58×10^0	4.84×10^1	1.99×10^1	1.72×10^1	3.96×10^1	5.41×10^1	5.93×10^1	3.34×10^1	3.74×10^0
CEC-28	Avg	3.46×10^3	3.47×10^3	3.38×10^3	3.32×10^3	3.52×10^3	3.42×10^3	3.79×10^3	3.25×10^3	3.30×10^3
	Std	7.54×10^1	1.33×10^2	9.19×10^1	1.15×10^2	1.84×10^2	1.68×10^2	1.73×10^2	2.95×10^1	1.18×10^2
CEC-29	Avg	3.25×10^3	3.40×10^3	3.25×10^3	3.31×10^3	3.34×10^3	3.38×10^3	3.44×10^3	3.39×10^3	3.21×10^3
	Std	4.39×10^1	1.35×10^2	9.96×10^1	5.83×10^1	1.06×10^2	1.02×10^2	1.40×10^2	1.04×10^2	6.58×10^1
CEC-30	Avg	1.00×10^6	1.73×10^6	6.35×10^5	1.02×10^6	2.31×10^6	1.25×10^6	3.09×10^7	5.74×10^6	4.70×10^5
	Std	1.22×10^6	1.61×10^6	5.46×10^5	7.18×10^5	6.59×10^6	1.87×10^6	2.82×10^7	4.21×10^6	1.15×10^6

The best result obtained is highlighted in **bold**.

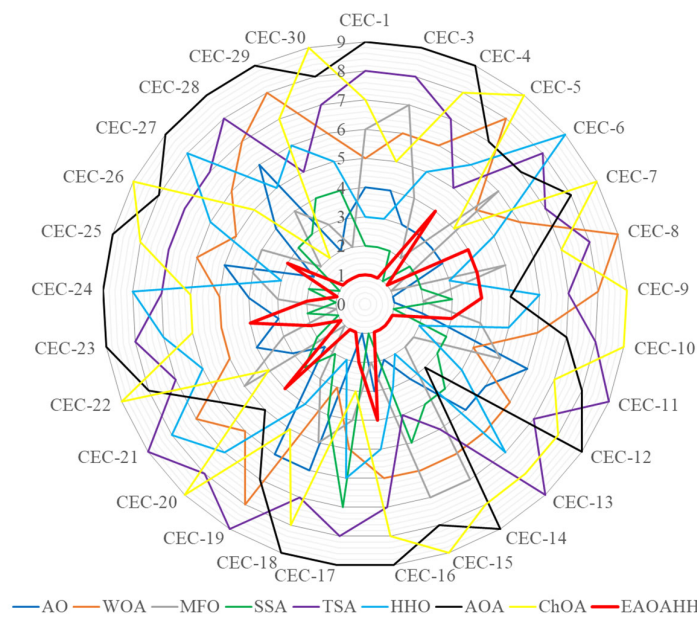


Figure 8. Ranking of different algorithms on IEEE CEC2017 test suite.

5. EAOAHHO for Solving Industrial Engineering Design Problems

Industrial engineering design problems are a class of nonlinear optimization problems accompanied by complex geometry and many constraints [52]. To further highlight the performance of the proposed EAOAHHO in real constrained optimization, four common industrial engineering design problems from the structural field are solved in this section, namely the three-bar truss design problem, tension/compression spring design problem, welded beam design problem, and rolling element bearing design problem. For the sake of convenience, the death penalty function [53] is introduced here to handle infeasible candidate solutions subject to those different constraints of equality and inequality. The proposed EAOAHHO runs independently 30 times on each project with the maximum iterations and population size are set to 500 and 30, respectively. The results obtained are compared with various state-of-the-art optimizers released in the previous studies.

5.1. Three-Bar Truss Design Problem

As it is named, the main objective of this optimization problem is to minimize the total weight of a three-bar truss. From the shape of the truss and its associated forces in Figure 9, two structural parameters need to be considered in this design, including the cross-sectional area of component 1 (A_1) and the cross-sectional area of component 2 (A_2). Stress, deflection, and buckling are the three dominant constraints. The mathematical formulation of this problem is described as follows:

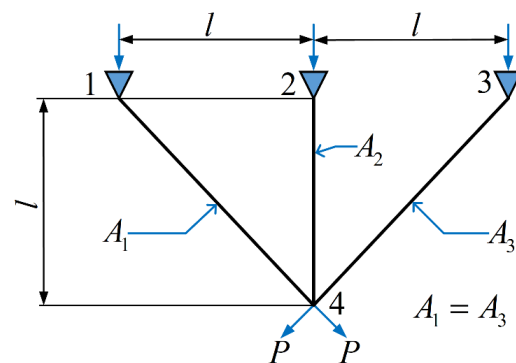


Figure 9. Schematic view of three-bar truss design problem.

Consider

$$\vec{z} = [z_1, z_2] = [A_1, A_2]$$

Minimize

$$f(\vec{z}) = (2\sqrt{2}z_1 + z_2) \times l$$

Subject to

$$\begin{aligned} g_1(\vec{z}) &= \frac{\sqrt{2}z_1 + z_2}{\sqrt{2}z_1^2 + 2z_1z_2} P - \sigma \leq 0 & g_3(\vec{z}) &= \frac{1}{\sqrt{2}z_2 + z_1} P - \sigma \leq 0 \\ g_2(\vec{z}) &= \frac{z_2}{\sqrt{2}z_1^2 + 2z_1z_2} P - \sigma \leq 0 \end{aligned}$$

Variable range

$$0 \leq z_1, z_2 \leq 1$$

where

$$l = 100\text{cm}, P = 2 \text{ KN/cm}^2, \sigma = 2 \text{ KN/cm}^2$$

Table 15 records the detailed results obtained by the proposed EAOAHHO and other meta-heuristics such as MFO, SSA, HHO, AOA, MVO, GEO, GOA, and AHA on this application. Based on the data in the table, it can be noticed that EAOAHHO outperforms all comparison algorithms and reveals the minimum weight $f_{min}(\vec{z}) = 263.87285$ corresponding to the optimal solution $\vec{z} = [0.78859304 \ 0.40825052]$. Therefore, it is reasonable to believe that EAOAHHO has promising potential to solve such a problem with very confined search space.

Table 15. Comparison results of different algorithms for solving three-bar truss design problem.

Algorithm	Optimal Values for Variables		Minimum Weight
	$A_1(z_1)$	$A_2(z_2)$	
MFO [48]	0.78824477	0.40946691	263.89598
SSA [25]	0.78866541	0.40827578	263.89584
HHO [27]	0.78866280	0.40828313	263.89584
AOA [38]	0.79369000	0.39426000	263.91540
MVO [19]	0.78860276	0.40845307	263.89585
GEO [54]	0.78867110	0.40825970	263.89584
GOA [55]	0.78889756	0.40761957	263.89588
AHA [56]	0.78868300	0.40822460	263.89584
EAOAHHO	0.78859304	0.40825052	263.87285

The best result obtained is highlighted in **bold**.

5.2. Tension/Compression Spring Design Problem

The final purpose of this engineering case is to decrease the weight of a tension/compression spring as much as possible. As illustrated in Figure 10, this problem has three decision variables that need to be optimized, namely the wire diameter (d), the mean coil diameter (D), and the number of active coils (N). Furthermore, constraints on shear stress, floating frequency, and limited floating deflection should not be violated during the minimization process. The problem is expressed mathematically as follows:



Figure 10. Schematic view of tension/compression spring design problem.

Consider

$$\vec{z} = [z_1, z_2, z_3] = [d, D, N]$$

Minimize

$$f(\vec{z}) = (z_3 + 2)z_2z_1^2$$

Subject to

$$\begin{aligned} g_1(\vec{z}) &= 1 - \frac{z_2^3 z_3}{71785 z_1^4} \leq 0 \\ g_2(\vec{z}) &= \frac{4z_2^2 - z_1 z_2}{12566(z_2 z_1^3 - z_1^4)} + \frac{1}{5108 z_1^2} \leq 0 \\ g_3(\vec{z}) &= 1 - \frac{140.45 z_1}{z_2^2 z_3} \leq 0 \\ g_4(\vec{z}) &= \frac{z_1 + z_2}{1.5} - 1 \leq 0 \end{aligned}$$

Variable range

$$0.05 \leq z_1 \leq 2, 0.25 \leq z_2 \leq 1.30, 2.00 \leq z_3 \leq 15.00$$

The experimental results of EAOAHHO for this problem are compared with those of WOA, MFO, SSA, HHO, AOA, AHA, GWO, and INFO, as presented in Table 16. It can be observed from this table that the proposed EAOAHHO can effectively achieve the optimal solution at $\vec{z} = [0.052291 \ 0.360263 \ 10.179344]$ with the minimum weight $f_{min}(\vec{z}) = 0.01199749$, which proves the merits of EAOAHHO in addressing the tension/compression spring design problem.

Table 16. Comparison results of different algorithms for solving tension/compression spring design problem.

Algorithm	Optimal Values for Variables			Minimum Weight
	$d(z_1)$	$D(z_2)$	$N(z_3)$	
WOA [31]	0.051207	0.345215	12.004032	0.0126763
MFO [48]	0.051994	0.364109	10.868422	0.0126669
SSA [25]	0.051207	0.345215	12.004032	0.0126763
HHO [27]	0.051796	0.359305	11.138859	0.0126654
AOA [38]	0.050000	0.349809	11.863700	0.0121240
AHA [56]	0.051897	0.361748	10.689283	0.0126660
GWO [26]	0.051690	0.356737	11.288850	0.0126660
INFO [57]	0.051555	0.353499	11.480340	0.0126660
EAOAHHO	0.052291	0.360263	10.179344	0.01199749

The best result obtained is highlighted in **bold**.

5.3. Welded Beam Design Problem

The welded beam design problem is one of the most well-known case studies used to evaluate the performance of algorithms. It was first proposed by Coello [53] and aims to minimize the overall manufacturing cost of a welded beam with four decision variables, i.e., the weld thickness (h), the length of the joint beam (l), the height of the beam (t), and the thickness of the beam (b), as illustrated in Figure 11. The mathematical formulation and seven constraint functions of this problem are given as follows:

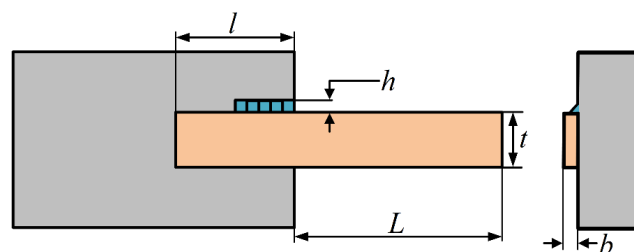


Figure 11. Schematic view of welded beam design problem.

Consider

$$\vec{z} = [z_1, z_2, z_3, z_4] = [h, l, t, b]$$

Minimize

$$f(\vec{z}) = 1.10471z_1^2z_2 + 0.04811z_3z_4(14 + z_2)$$

Subject to

$$\begin{aligned} g_1(\vec{z}) &= \tau(\vec{z}) - \tau_{\max} \leq 0 \\ g_2(\vec{z}) &= \sigma - \sigma_{\max} \leq 0 \\ g_3(\vec{z}) &= \delta - \delta_{\max} \leq 0 \\ g_4(\vec{z}) &= z_1 - z_4 \leq 0 \\ g_5(\vec{z}) &= P - P_C(\vec{z}) \leq 0 \\ g_6(\vec{z}) &= 0.125 - z_1 \leq 0 \\ g_7(\vec{z}) &= 1.10471z_1^2 + 0.04811z_3z_4(14 + z_2) - 5 \leq 0 \end{aligned}$$

Variable range

$$0.1 \leq z_1, z_4 \leq 2, 0.1 \leq z_2, z_3 \leq 10$$

where

$$\begin{aligned} \tau(\vec{z}) &= \sqrt{(\tau')^2 + 2\tau'\tau''\frac{z_2}{2R} + (\tau'')^2}, \tau' = \frac{P}{\sqrt{2}z_1z_2}, \tau'' = \frac{MR}{J}, M = P(L + \frac{z_2}{2}) \\ R &= \sqrt{\frac{z_2^2}{4} + (\frac{z_1+z_3}{2})^2}, J = 2\left\{\sqrt{2}z_1z_2\left[\frac{z_2^2}{4} + (\frac{z_1+z_3}{2})^2\right]\right\}, \sigma(\vec{z}) = \frac{6PL}{Ez_3^2z_4} \\ \delta(\vec{z}) &= \frac{6PL^3}{Ez_3^2z_4}, P_C(\vec{z}) = \frac{4.013E\sqrt{\frac{z_2^2z_4^2}{36}}}{L^2}\left(1 - \frac{z_3}{2L}\sqrt{\frac{E}{4G}}\right) \\ P &= 6000\text{lb}, L = 14\text{in}, E = 30 \times 10^6\text{psi}, G = 12 \times 10^6\text{psi}, \\ \delta_{\max} &= 0.25\text{in}, \tau_{\max} = 13600\text{psi}, \sigma_{\max} = 30000\text{psi}. \end{aligned}$$

This problem has been figured out by the proposed EAOAHHO and the remaining eight methods, such as WOA, AOA, MVO, GWO, ROA, HGS, AVOA, and IMFO. The optimal solutions are summarized in Table 17. It can be seen that when the four variables h, l, t , and b are set as 0.195539, 3.354588, 9.036630, and 0.205729, respectively, the minimum manufacturing cost of EAOAHHO is 1.693914. In this comparison, the results of EAOAHHO are evidently better than all the other methods. This shows that EAOAHHO has good competitiveness in dealing with the welded beam design problem.

Table 17. Comparison results of different algorithms for solving welded beam design problem.

Algorithm	Optimal Values for Variables				Minimum Cost
	$h(z_1)$	$l(z_1)$	$t(z_3)$	$b(z_4)$	
WOA [31]	0.205396	3.484293	9.037426	0.206276	1.730499
AOA [38]	0.194475	2.570920	10.00000	0.201827	1.716400
MVO [19]	0.205463	3.473193	9.044502	0.205695	1.726450
GWO [26]	0.205676	3.478377	9.036810	0.205778	1.726240
ROA [58]	0.200077	3.365754	9.011182	0.206893	1.706447
HGS [59]	0.207739	3.230642	8.988778	0.207926	1.703355
AVOA [60]	0.205730	3.470474	9.036621	0.205730	1.724852
IMFO [61]	0.205730	3.470200	9.037500	0.205730	1.724900
EAOAHHO	0.195539	3.354588	9.036630	0.205729	1.693914

The best result obtained is highlighted in **bold**.

5.4. Rolling Element Bearing Design Problem

Unlike the three test cases mentioned above, the primary goal of this problem is to maximize the dynamic load-carrying capacity of a rolling element bearing. In this optimum design, a total of ten geometric variables need to be taken into account, which are pitch diameter (D_m), ball diameter (D_b), the number of balls (Z), the inner and outer raceway

curvature radius coefficient (f_i and f_o), K_{dmin} , K_{dmax} , δ , e , and ζ (see Figure 12). The problem has nine constraints and its mathematical model is described as follows.

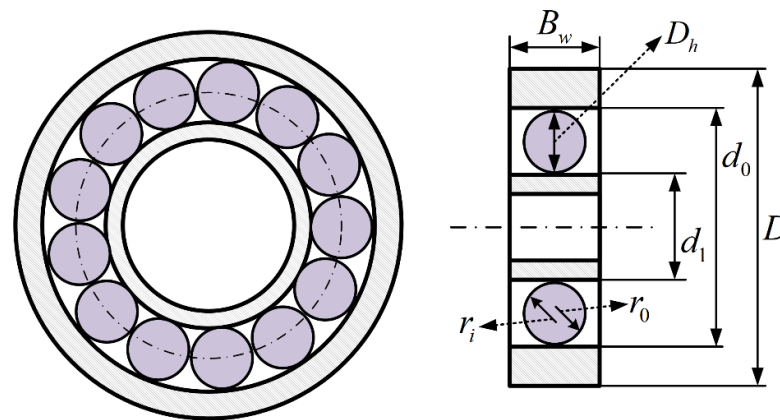


Figure 12. Schematic view of rolling element bearing design problem.

Maximize

$$C_d = \begin{cases} f_c Z^{2/3} D_b^{1.8}, & \text{if } D_b \leq 25.4 \text{ mm} \\ 3.647 f_c Z^{2/3} D_b^{1.4}, & \text{else} \end{cases}$$

Subject to

$$\begin{aligned} g_1(\vec{z}) &= \frac{\phi_0}{2 \sin^{-1}(D_b/D_m)} - Z + 1 \leq 0 \\ g_2(\vec{z}) &= 2D_b - K_{dmin}(D - d) > 0 \\ g_3(\vec{z}) &= K_{dmax}(D - d) - 2D_b \geq 0 \\ g_4(\vec{z}) &= \zeta B_w - D_b \leq 0 \\ g_5(\vec{z}) &= D_m - 0.5(D + d) \geq 0 \\ g_6(\vec{z}) &= (0.5 + e)(D + d) - D_m \geq 0 \\ g_7(\vec{z}) &= 0.5(D - D_m - D_b) - \delta D_b \geq 0 \\ g_8(\vec{z}) &= f_i \geq 0.515 \\ g_9(\vec{z}) &= f_o \geq 0.515 \end{aligned}$$

where

$$\begin{aligned} f_c &= 37.91 \left[1 + \left\{ 1.04 \left(\frac{1-\gamma}{1+\gamma} \right)^{1.72} \left(\frac{f_i(2f_o-1)}{f_o(2f_i-1)} \right)^{0.41} \right\}^{10/3} \right]^{-0.3} \times \left[\frac{\gamma^{0.3}(1-\gamma)^{1.39}}{(1+\gamma)^{1/3}} \right] \left[\frac{2f_i}{2f_i-1} \right]^{0.41} \\ x &= \left[\{(D-d)/2 - 3(T/4)\}^2 + \{D/2 - T/4 - D_b\}^2 - \{d/2 + T/4\}^2 \right] \\ y &= 2\{(D-d)/2 - 3(T/4)\}\{D/2 - T/4 - D_b\} \\ \phi_0 &= 2\pi - \cos^{-1}\left(\frac{x}{y}\right), \gamma = \frac{D_b}{D_m}, f_i = \frac{r_i}{D_b}, f_o = \frac{r_o}{D_b}, T = D - d - 2D_b \\ D &= 160, d = 90, B_w = 30, r_i = r_o = 11.033, 0.5(D + d) \leq D_m \leq 0.6(D + d) \\ 0.15(D - d) &\leq D_b \leq 0.45(D - d), 4 \leq Z \leq 50, 0.515 \leq f_i \text{ and } f_o \leq 0.6 \\ 0.4 \leq K_{dmin} &\leq 0.5, 0.6 \leq K_{dmax} \leq 0.7, 0.3 \leq \delta \leq 0.4, 0.02 \leq e \leq 0.1, 0.6 \leq \zeta \leq 0.85. \end{aligned}$$

The comparison results between different optimization techniques when solving the rolling element bearing design problem are summarized in Table 18. Compared with HHO, TLBO, RUN, RSA, and COOT, the proposed EAOAHHO algorithm in this paper is able to discover the best cost value with significant progress, which is 85,539.193. This example once again proves the effectiveness of EAOAHHO at the practical application level.

Taken together, the results of this section strongly demonstrate the superiority of the proposed EAOAHHO in different characteristics and real-world optimization tasks. Benefiting from the hybrid operation, pinhole imaging opposition-based learning, and composite mutation strategy, EAOAHHO possesses more robust exploration and exploitation capabilities, which can be regarded as a reliable alternative to the basic AOA, HHO, and

some existing algorithms. In light of its excellent performance, EAOAHHO may be used to tackle a wider range of real-world problems.

Table 18. Comparison results of different algorithms for solving rolling element bearing design problem.

Algorithm	EAOAHHO	HHO [27]	TLBO [62]	RUN [63]	RSA [64]	COOT [65]
D_m	125.7227	125	125.7191	125.2142	125.1722	125
D_b	21.42330	21.00000	21.42559	21.59796	21.29734	21.87500
Z	11.00116	11.09207	11.00000	11.40240	10.88521	10.77700
f_i	0.51500	0.51500	0.51500	0.51500	0.515253	0.51500
f_o	0.51500	0.51500	0.51500	0.51500	0.517764	0.51500
K_{dmin}	0.50000	0.40000	0.424266	0.40059	0.41245	0.43190
K_{dmax}	0.70000	0.60000	0.633948	0.61467	0.632338	0.65290
δ	0.30000	0.30000	0.30000	0.30530	0.301911	0.30000
e	0.02000	0.05047	0.068858	0.02000	0.024395	0.02000
ζ	0.600240	0.60000	0.799498	0.63665	0.6024	0.60000
Maximum cost	85,539.193	83,011.883	81,859.74	83,680.47	83,486.64	83,918.492

The best result obtained is highlighted in **bold**.

6. Conclusions and Future Work

In this study, we propose an ensemble of Arithmetic Optimization Algorithm and Harris Hawks Optimization, called EAOAHHO, which combines the basic AOA and HHO, and introduces PIOBL and CMS strategies. The PIOBL is introduced into the proposed algorithm to increase the original population diversity and the capability to escape from local optima. Furthermore, the introduction of CMS enhances the proposed EAOAHHO exploitation and exploration to obtain better convergence accuracy.

In order to evaluate the performance of the proposed algorithm, EAOAHHO is compared with the basic AOA, HHO, and six other advanced meta-heuristic algorithms based on 23 classical benchmark functions and the IEEE CEC2017 test suite. The experimental results indicate that the EAOAHHO proposed in this study has better convergence accuracy and capability of local optimal avoidance, compared with other algorithms, and can maintain a good balance between exploration and exploitation. In the end of the paper, we further verify the superiority of EAOAHHO by solving four industrial engineering design problems, and the results are also competitive with other meta-heuristic algorithms.

However, the performance of this algorithm on the IEEE CEC2017 test suite remains to be enhanced. We could further enhance the performance of the proposed algorithm by introducing strategies such as chaos-based initialization, and so on.

In the future, we expect to further enhance the computational efficiency of the proposed EAOAHHO and apply it to more practical engineering applications, such as path planning for autonomous intelligent unmanned systems, fault diagnosis for aero engines, and optimizing Support Vector Machine (SVM).

Author Contributions: Conceptualization, J.Y. and Y.S.; methodology, J.Y.; software, J.Y.; validation, J.Y., Y.S., X.Z. and Y.C.; formal analysis, Y.S.; investigation, X.Z.; data curation, J.Y.; writing—original draft preparation, J.Y.; writing—review and editing, J.Y.; funding acquisition, Y.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by China Geological Survey, grant number 12120113017600; and Program for JLU Science and Technology Innovative Research Team, grant number 2017TD-13.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing not applicable.

Acknowledgments: The authors are grateful to the editor and anonymous reviewers for their constructive comments and suggestions, which have improved this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhang, X.C.; Zhao, K.; Niu, Y. Improved harris hawks optimization based on adaptive cooperative foraging and dispersed foraging strategies. *IEEE Access* **2020**, *8*, 160297–160314. [\[CrossRef\]](#)
2. Xiao, Y.N.; Sun, X.; Guo, Y.L.; Li, S.P.; Zhang, Y.P.; Wang, Y.W. An Improved Gorilla Troops Optimizer Based on Lens Opposition-Based Learning and Adaptive β -Hill Climbing for Global Optimization. *Cmes-Comput. Model. Eng. Sci.* **2022**, *131*, 815–850. [\[CrossRef\]](#)
3. Ewees, A.A.; Abd Elaziz, M.; Houssein, E.H. Improved grasshopper optimization algorithm using opposition-based learning. *Expert Syst. Appl.* **2018**, *112*, 156–172. [\[CrossRef\]](#)
4. Hayyolalam, V.; Kazem, A.A.P. Black widow optimization algorithm: A novel meta-heuristic approach for solving engineering optimization problems. *Eng. Appl. Artif. Intell.* **2020**, *87*, 103249. [\[CrossRef\]](#)
5. Xiao, Y.; Sun, X.; Guo, Y.; Cui, H.; Wang, Y.; Li, J.; Li, S. An enhanced honey badger algorithm based on Lévy flight and refraction opposition-based learning for engineering design problems. *J. Intell. Fuzzy Syst.* **2022**, 1–24. [\[CrossRef\]](#)
6. Dewangan, R.K.; Shukla, A.; Godfrey, W.W. Three dimensional path planning using grey wolf optimizer for UAVs. *Appl. Intell.* **2019**, *49*, 2201–2217. [\[CrossRef\]](#)
7. Yao, J.; Sha, Y.; Chen, Y.; Zhang, G.; Hu, X.; Bai, G.; Liu, J. IHSSAO: An Improved Hybrid Salp Swarm Algorithm and Aquila Optimizer for UAV Path Planning in Complex Terrain. *Appl. Sci.* **2022**, *12*, 5634. [\[CrossRef\]](#)
8. Jia, H.; Zhang, W.; Zheng, R.; Wang, S.; Leng, X.; Cao, N. Ensemble mutation slime mould algorithm with restart mechanism for feature selection. *Int. J. Intell. Syst.* **2022**, *37*, 2335–2370. [\[CrossRef\]](#)
9. Tongbram, S.; Shimray, B.A.; Singh, L.S.; Dhanachandra, N. A novel image segmentation approach using fcm and whale optimization algorithm. *J. Ambient. Intell. Humaniz. Comput.* **2021**, 1–15. [\[CrossRef\]](#)
10. Yin, S.H.; Luo, Q.F.; Du, Y.L.; Zhou, Y.Q. p DTSMa: Dominant swarm with adaptive T-distribution mutation-based slime mould algorithm. *Math. Biosci. Eng.* **2022**, *19*, 2240–2285. [\[CrossRef\]](#)
11. Tang, C.M.; Zhou, Y.Q.; Tang, Z.H.; Luo, Q.F. Teaching-learning-based pathfinder algorithm for function and engineering optimization problems. *Appl. Intell.* **2021**, *51*, 5040–5066. [\[CrossRef\]](#)
12. Kuncheva, L.I.; Jain, L.C. Designing classifier fusion systems by genetic algorithms. *IEEE Trans. Evol. Comput.* **2000**, *4*, 327–336.
13. Qian, W.W.; Chai, J.R.; Xu, Z.G.; Zhang, Z.Y. Differential evolution algorithm with multiple mutation strategies based on roulette wheel selection. *Appl. Intell.* **2018**, *48*, 3612–3629. [\[CrossRef\]](#)
14. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by simulated annealing. *Science* **1983**, *220*, 671–680. [\[CrossRef\]](#) [\[PubMed\]](#)
15. Yilmaz, S.; Gokasan, M. Optimal trajectory planning by big bang-big crunch algorithm. In Proceedings of the 2nd International Conference on Control, Decision and Information Technologies (CoDIT), Ecole Natl Ingenieurs Metz, Metz, France, 3–5 November 2014; pp. 557–561.
16. Rashedi, E.; Nezamabadi-Pour, H.; Saryazdi, S. GSA: A gravitational search algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248. [\[CrossRef\]](#)
17. Alatas, B. ACROA: Artificial chemical reaction optimization algorithm for global optimization. *Expert Syst. Appl.* **2011**, *38*, 13170–13180. [\[CrossRef\]](#)
18. Recioui, A. Application of a galaxy-based search algorithm to mimo system capacity optimization. *Arab. J. Sci. Eng.* **2016**, *41*, 3407–3414. [\[CrossRef\]](#)
19. Mirjalili, S.; Mirjalili, S.M.; Hatamlou, A. Multi-verse optimizer: A nature-inspired algorithm for global optimization. *Neural Comput. Appl.* **2016**, *27*, 495–513. [\[CrossRef\]](#)
20. Hekimoglu, B. Optimal tuning of fractional order PID controller for DC motor speed control via chaotic atom search optimization algorithm. *IEEE Access* **2019**, *7*, 38100–38114. [\[CrossRef\]](#)
21. Krishna, A.B.; Saxena, S.; Kamboj, V.K. hSMA-PS: A novel memetic approach for numerical and engineering design challenges. *Eng. Comput.* **2021**, 1–35. [\[CrossRef\]](#)
22. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the 1995 IEEE International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; pp. 1942–1948.
23. Jiang, J.J.; Wei, W.X.; Shao, W.L.; Liang, Y.F.; Qu, Y.Y. Research on Large-scale bi-level particle swarm optimization algorithm. *IEEE Access* **2021**, *9*, 56364–56375. [\[CrossRef\]](#)
24. Mirjalili, S. The ant lion optimizer. *Adv. Eng. Softw.* **2015**, *83*, 80–98. [\[CrossRef\]](#)
25. Mirjalili, S.; Gandomi, A.H.; Mirjalili, S.Z.; Saremi, S.; Faris, H.; Mirjalili, S.M. Salp swarm algorithm: A bio-inspired optimizer for engineering design problems. *Adv. Eng. Softw.* **2017**, *114*, 163–191. [\[CrossRef\]](#)
26. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [\[CrossRef\]](#)
27. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris hawks optimization: Algorithm and applications. *Future Gener. Comp. Syst.* **2019**, *97*, 849–872. [\[CrossRef\]](#)
28. 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.* **2021**, *157*, 107250. [\[CrossRef\]](#)
29. Dhiman, G.; Kumar, V. Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications. *Adv. Eng. Softw.* **2017**, *114*, 48–70. [\[CrossRef\]](#)
30. Selvi, M.; Ramakrishnan, B. Lion optimization algorithm (LOA)-based reliable emergency message broadcasting system in VANET. *Soft Comput.* **2020**, *24*, 10415–10432. [\[CrossRef\]](#)
31. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [\[CrossRef\]](#)

32. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [\[CrossRef\]](#)
33. Fan, Q.; Chen, Z.J.; Li, Z.; Xia, Z.H.; Lin, Y.Q. An efficient refracted salp swarm algorithm and its application in structural parameter identification. *Eng. Comput.* **2022**, *38*, 175–189. [\[CrossRef\]](#)
34. Kamboj, V.K.; Nandi, A.; Bhadoria, A.; Sehgal, S. An intensify harris hawks optimizer for numerical and engineering optimization problems. *Appl. Soft Comput.* **2020**, *89*, 106018. [\[CrossRef\]](#)
35. Belge, E.; Altan, A.; Hacioglu, R. Metaheuristic Optimization-Based Path Planning and Tracking of Quadcopter for Payload Hold-Release Mission. *Electronics* **2022**, *11*, 1208. [\[CrossRef\]](#)
36. Abd Elaziz, M.; Heidari, A.A.; Fujita, H.; Moayedi, H. A competitive chain-based Harris Hawks Optimizer for global optimization and multi-level image thresholding problems. *Appl. Soft Comput.* **2020**, *95*, 106347. [\[CrossRef\]](#)
37. Shao, K.X.; Fu, W.L.; Tan, J.W.; Wang, K. Coordinated approach fusing time-shift multiscale dispersion entropy and vibrational Harris hawks optimization-based SVM for fault diagnosis of rolling bearing. *Measurement* **2021**, *173*, 108580. [\[CrossRef\]](#)
38. Abualigah, L.; Diabat, A.; Mirjalili, S.; Abd Elaziz, M.; Gandomi, A.H. The arithmetic optimization algorithm. *Comput. Meth. Appl. Mech. Eng.* **2021**, *376*, 113609. [\[CrossRef\]](#)
39. Devan, P.A.M.; Hussin, F.A.; Ibrahim, R.B.; Bingi, K.; Nagarajapandian, M.; Assaad, M. An Arithmetic-Trigonometric Optimization Algorithm with Application for Control of Real-Time Pressure Process Plant. *Sensors* **2022**, *22*, 617. [\[CrossRef\]](#)
40. Panga, N.; Sivaramakrishnan, U.; Abishek, R.; Bingi, K.; Chaudhary, J. An Improved Arithmetic Optimization Algorithm. In Proceedings of the 2021 IEEE Madras Section Conference (MASCION), Chennai, India, 27–28 August 2021; pp. 1–6.
41. Tizhoosh, H.R. Opposition-based learning: A new scheme for machine intelligence. In Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, Vienna, Austria, 28–30 November 2005; pp. 695–701.
42. Xiao, Y.; Sun, X.; Zhang, Y.; Guo, Y.; Wang, Y.; Li, J. An improved slime mould algorithm based on tent chaotic mapping and nonlinear inertia weight. *Int. J. Innov. Comp. Inf. Control* **2021**, *17*, 2151–2176. [\[CrossRef\]](#)
43. Long, W.; Jiao, J.; Liang, X.; Wu, T.; Xu, M.; Cai, S. Pinhole-imaging-based learning butterfly optimization algorithm for global optimization and feature selection. *Appl. Soft Comput.* **2021**, *103*, 107146. [\[CrossRef\]](#)
44. Li, M.; Xu, G.; Fu, B.; Zhao, X. Whale optimization algorithm based on dynamic pinhole imaging and adaptive strategy. *J. Supercomput.* **2021**, *78*, 6090–6120. [\[CrossRef\]](#)
45. Abualigah, L.; Diabat, A. Improved multi-core arithmetic optimization algorithm-based ensemble mutation for multidisciplinary applications. *J. Intell. Manuf.* **2022**, 1–42. [\[CrossRef\]](#)
46. Zhang, H.; Wang, Z.; Chen, W.; Heidari, A.A.; Wang, M.; Zhao, X.; Liang, G.; Chen, H.; Zhang, X. Ensemble mutation-driven salp swarm algorithm with restart mechanism: Framework and fundamental analysis. *Expert Syst. Appl.* **2021**, *165*, 113897. [\[CrossRef\]](#)
47. Wang, Y.; Cai, Z.; Zhang, Q. Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Trans. Evol. Comput.* **2011**, *15*, 55–66. [\[CrossRef\]](#)
48. Mirjalili, S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowl. Based Syst.* **2015**, *89*, 228–249. [\[CrossRef\]](#)
49. 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.* **2020**, *90*, 103541. [\[CrossRef\]](#)
50. Khishe, M.; Mosavi, M.R. Chimpanzee optimization algorithm. *Expert Syst. Appl.* **2020**, *149*, 113338. [\[CrossRef\]](#)
51. Awad, N.H.; Ali, M.Z.; Suganthan, P.N. Ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood for solving CEC2017 benchmark problems. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), Donostia, Spain, 5–8 June 2017; pp. 372–379.
52. Tang, A.; Zhou, H.; Han, T.; Xie, L. A modified manta ray foraging optimization for global optimization problems. *IEEE Access* **2021**, *9*, 128702–128721. [\[CrossRef\]](#)
53. Coello Coello, C.A. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. *Comput. Meth. Appl. Mech. Eng.* **2002**, *191*, 1245–1287. [\[CrossRef\]](#)
54. Mohammadi-Balani, A.; Dehghan Nayeri, M.; Azar, A.; Taghizadeh-Yazdi, M. Golden eagle optimizer: A nature-inspired metaheuristic algorithm. *Comput. Ind. Eng.* **2021**, *152*, 107050. [\[CrossRef\]](#)
55. Saremi, S.; Mirjalili, S.; Lewis, A. Grasshopper optimisation algorithm: Theory and application. *Adv. Eng. Softw.* **2017**, *105*, 30–47. [\[CrossRef\]](#)
56. Zhao, W.; Wang, L.; Mirjalili, S. Artificial hummingbird algorithm: A new bio-inspired optimizer with its engineering applications. *Comput. Meth. Appl. Mech. Eng.* **2022**, *388*, 114194. [\[CrossRef\]](#)
57. Ahmadianfar, I.; Heidari, A.A.; Noshadian, S.; Chen, H.; Gandomi, A.H. INFO: An efficient optimization algorithm based on weighted mean of vectors. *Expert Syst. Appl.* **2022**, *195*, 116516. [\[CrossRef\]](#)
58. Jia, H.; Peng, X.; Lang, C. Remora optimization algorithm. *Expert Syst. Appl.* **2021**, *185*, 115665. [\[CrossRef\]](#)
59. Yang, Y.; Chen, H.; Heidari, A.A.; Gandomi, A.H. Hunger games search: Visions, conception, implementation, deep analysis, perspectives, and towards performance shifts. *Expert Syst. Appl.* **2021**, *177*, 114864. [\[CrossRef\]](#)
60. Abdollahzadeh, B.; Gharehchopogh, F.S.; Mirjalili, S. African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems. *Comput. Ind. Eng.* **2021**, *158*, 107408. [\[CrossRef\]](#)
61. Li, Y.; Zhu, X.; Liu, J. An improved moth-flame optimization algorithm for engineering problems. *Symmetry* **2020**, *12*, 1234. [\[CrossRef\]](#)

62. Rao, R.V.; Savsani, V.J.; Vakharia, D.P. Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. *Comput. Aided Des.* **2011**, *43*, 303–315. [[CrossRef](#)]
63. Ahmadianfar, I.; Heidari, A.A.; Gandomi, A.H.; Chu, X.; Chen, H. RUN beyond the metaphor: An efficient optimization algorithm based on Runge Kutta method. *Expert Syst. Appl.* **2021**, *181*, 115079. [[CrossRef](#)]
64. Abualigah, L.; Elaziz, M.A.; Sumari, P.; Geem, Z.W.; Gandomi, A.H. Reptile search algorithm (RSA): A nature-inspired meta-heuristic optimizer. *Expert Syst. Appl.* **2022**, *191*, 116158. [[CrossRef](#)]
65. Naruei, I.; Keynia, F. A new optimization method based on COOT bird natural life model. *Expert Syst. Appl.* **2021**, *183*, 115352. [[CrossRef](#)]