processes

MDPI

Article

An Improved Hybrid Aquila Optimizer and Harris Hawks Algorithm for Solving Industrial Engineering Optimization Problems

Shuang Wang [1], Heming Jia [1,*], Laith Abualigah [2,3,4], Qingxin Liu [5] and Rong Zheng [1]

[1] School of Information Engineering, Sanming University, Sanming 365004, China; shuang.wang@fjsmu.edu.cn (S.W.); zhengr@fjsmu.edu.cn (R.Z.)
[2] Research and Innovation Department, Skyline University College, Sharjah 1797, United Arab Emirates; Aligah.2020@gmail.com
[3] Faculty of Computer Sciences and Informatics, Amman Arab University, Amman 11953, Jordan
[4] School of Computer Science, Universiti Sains Malaysia, Gelugor 11800, Pulau Pinang, Malaysia
[5] School of Computer Science and Technology, Hainan University, Haikou 570228, China; lqxass@fjsmu.edu.cn
* Correspondence: jiaheming@fjsmu.edu.cn

Abstract: Aquila Optimizer (AO) and Harris Hawks Optimizer (HHO) are recently proposed meta-heuristic optimization algorithms. AO possesses strong global exploration capability but insufficient local exploitation ability. However, the exploitation phase of HHO is pretty good, while the exploration capability is far from satisfactory. Considering the characteristics of these two algorithms, an improved hybrid AO and HHO combined with a nonlinear escaping energy parameter and random opposition-based learning strategy is proposed, namely IHAOHHO, to improve the searching performance in this paper. Firstly, combining the salient features of AO and HHO retains valuable exploration and exploitation capabilities. In the second place, random opposition-based learning (ROBL) is added in the exploitation phase to improve local optima avoidance. Finally, the nonlinear escaping energy parameter is utilized better to balance the exploration and exploitation phases of IHAOHHO. These two strategies effectively enhance the exploration and exploitation of the proposed algorithm. To verify the optimization performance, IHAOHHO is comprehensively analyzed on 23 standard benchmark functions. Moreover, the practicability of IHAOHHO is also highlighted by four industrial engineering design problems. Compared with the original AO and HHO and five state-of-the-art algorithms, the results show that IHAOHHO has strong superior performance and promising prospects.

Keywords: Aquila Optimizer; Harris Hawks Optimizer; hybrid algorithm; nonlinear escaping energy parameter; random opposition-based learning

Citation: Wang, S.; Jia, H.; Abualigah, L.; Liu, Q.; Zheng, R. An Improved Hybrid Aquila Optimizer and Harris Hawks Algorithm for Solving Industrial Engineering Optimization Problems. *Processes* **2021**, *9*, 1551. https://doi.org/10.3390/pr9091551

Academic Editor: Jean-Pierre Corriou

Received: 20 July 2021
Accepted: 27 August 2021
Published: 30 August 2021

1. Introduction

Meta-heuristic optimization algorithms inspired by nature are becoming more and more popular in real-world applications [1]. Meta-heuristics usually mimic biological or physical phenomena and only consider inputs and outputs, making them flexible and straightforward. Furthermore, meta-heuristics is a kind of stochastic optimization technique. This property assists them to effectively avoid local optima, which usually occurs in real problems. Because of the advantages of simplicity, flexibility, and ability to avoid local optima, meta-heuristic optimization algorithms outperform heuristic optimization algorithms to solve various complex and tricky optimization problems in the real world [2].

Three dominant categories are divided from meta-heuristic optimization algorithms: evolutionary, physics-based, and swarm intelligence techniques. Evolutionary algorithms are inspired by the laws of evolution in nature. The randomly generated population evolves over subsequent generations as the number of iterations increases. Each generation of individuals is always formed by the combination of best individuals so that the

population can be optimized over several generations of evolution. The most popular evolutionary technique is Genetic Algorithms (GA) [3], which simulates Darwin's theory of evolution. There are several other popular evolutionary algorithms, such as Differential Evolution Algorithm (DE) [4], Genetic Programming (GP) [5], Evolution Strategy (ES) [6], Biogeography-Based Optimizer (BBO) [7], Evolutionary Deduction Algorithm (ED) [8], and Probability-Based Incremental Learning (PBIL) [9]. Physics-based methods are inspired by the physical rules of the universe. The most popular algorithms in this category are Simulated Annealing (SA) [10], Big-Bang Big-Crunch (BBBC) [11], Gravity Search Algorithm (GSA) [12], Gravitational Local Search (GLSA) [13], Heat Transfer Relation-based Optimization Algorithm (HTOA) [14], Charged System Search (CSS) [15], Artificial Chemical Reaction Optimization Algorithm (ACROA) [16], Central Force Optimization (CFO) [17], Ray Optimization (RO) [18] algorithm, Black Hole (BH) [19] algorithm, Small-World Optimization Algorithm (SWOA) [20], Galaxy-based Search Algorithm (GbSA) [21], Curved Space Optimization (CSO) [22], Multi-Verse Optimizer (MVO) [23], Sine Cosine Algorithm (SCA) [24], and Arithmetic Optimization Algorithm (AOA) [25].

The third category is swarm intelligence algorithms, which simulate the behaviour of swarms of creatures in nature. The most well-known swarm intelligence technique is Particle Swarm Optimization (PSO), first proposed by Kennedy and Eberhart [26]. PSO mimics the behaviour of bird flocks in navigating and foraging, and the birds achieve the optimal position through collective cooperation. Particles update positions not only considering their own best positions but also according to the best position of the swarm obtained so far. Other representative algorithms include Ant Colony Optimization Algorithm (ACO) [27], Monkey Search [28], Firefly Algorithm [29], Bat Algorithm (BA) [30], Krill Herd (KH) [31], Grey Wolf Optimizer (GWO) [32], Cuckoo Search (CS) Algorithm [33], Fruit Fly Optimization (FFO) [34], Dolphin Partner Optimization (DPO) [35], Ant Lion Optimizer (ALO) [36], Remora Optimization Algorithm (ROA) [37], Whale Optimization Algorithm (WOA) [38], Salp Swarm Algorithm (SSA) [39], Bald Eagle Search (BES) algorithm [40], and Slime Mould Algorithm (SMA) [41].

As one of the swarm intelligence algorithms, the Harris Hawks Optimizer (HHO) [42] was proposed in 2019. HHO simulates several hunting strategies of Harris's hawk and attracted several researchers to apply it to solve practical problems [43–47]. The exploitation phase of HHO includes four strategies, but the exploration phase is insufficient, and the balance between the exploration and exploitation phases is not good enough. Therefore, many improved and hybrid researches have been proposed to enhance the performance of HHO. Yousri et al. [48] proposed an enhanced algorithm based on the fractional calculus (FOC) memory concept to improve the performance of exploration phase, which is known as FMHHO. The hawk moves with a fractional-order velocity, and the escaping energy of the prey is adaptively adjusted based on FOC parameters to avoid local optima stagnation. Gupta et al. [49] introduced a nonlinear energy parameter, different settings for rapid dives, opposition-based learning strategy, and a greedy selection mechanism into HHO to enhance the search efficiency and avoid premature convergence. Hussien and Amin [50] proposed an improved HHO called IHHO to enhance the performance of HHO. The proposed IHHO applied opposition-based learning (OBL) in the initialization phase to diverse the initial population as well as Chaotic Local Search (CLS) strategy and a self-adaptive technique to improve its performance and speed up the convergence of the algorithm. Sihwail et al. [51] proposed a new search mechanism and then applied it and elite opposite-based learning (EOBL) technique to HHO. The improved HHO raised the search capabilities by mutation, mutation neighborhood search (MNS), and rollback strategy. It can avoid local optimum entrapment and improve population diversity, convergence accuracy, and rate. Bao et al. [52] proposed HHO-DE by hybridizing HHO and Differential Evolution (DE) algorithms. HHO and DE were used to update the positions of two equal subpopulations respectively. The proposed HHO-DE has high accuracy, ability to avoid local optima, and remarkable stability. Houssein et al. [53] combined HHO with cuckoo search (CS) and chaotic maps to propose a hybrid algorithm called CHHO-CS. CS was

used to control the main position vectors of HHO to achieve a better balance between exploration and exploitation phases, and chaotic maps were adopted to update the control energy parameters to avoid premature convergence. Kaveh et al. [54] proposed an effective algorithm called ICHHO by hybridizing HHO with Imperialist Competitive Algorithm (ICA). Combination of the exploration strategy of ICA and exploitation technique of HHO helps to achieve a better search strategy. These improved and hybrid algorithms have proven that HHO is a valuable algorithm. Aquila Optimizer (AO) [55] is the latest swarm intelligence algorithm, proposed in 2021. This algorithm simulates different hunting methods of Aquila for different kinds of prey. The hunting methods for fast-moving prey reflect the global exploration ability of the algorithm, and the hunting methods for slow-moving prey reflect the local exploitation ability of the algorithm. AO algorithm possesses strong global exploration ability, high search efficiency, and fast convergence speed, but its local exploitation ability is insufficient, so it is easy to fall into local optima. Due to the short time that has elapsed since the algorithm has been proposed, there is no research on the improvement of AO yet.

Therefore, we tried a kind of hybridization to improve the performance of HHO and AO. As far as we know, this kind of hybridization of HHO with AO has not been used before. We propose a new, improved hybrid Aquila Optimizer and Harris Hawks Optimization (IHAOHHO) by combining the salient features of AO and HHO. In this paper, we integrate the exploitation strategy of HHO into the AO algorithm, which is added random opposition-based learning (ROBL) in the exploitation phase to avoid local optima stagnation. At the same time, the nonlinear escaping energy parameter balances the exploration and exploitation phases of the algorithm. The 23 standard benchmark functions and four engineering design problems were applied to test the exploration and exploitation capabilities of IHAOHHO. The proposed algorithm is compared with original AO, HHO, and several well-known algorithms, including SMA, SSA, WOA, GWO, and PSO. The experimental results show that the proposed IHAOHHO algorithm performs better than the state-of-the-art meta-heuristic algorithms.

The rest of this paper is organized as follows (Figure 1): Section 2 provides a brief overview of the related work: original Harris Hawks Optimization algorithm and Aquila Optimizer, as well as two improvement strategies. Section 3 describes in detail the proposed hybrid algorithm. Section 4 conducts simulation experiments and results analysis. Finally, Section 5 concludes the paper.

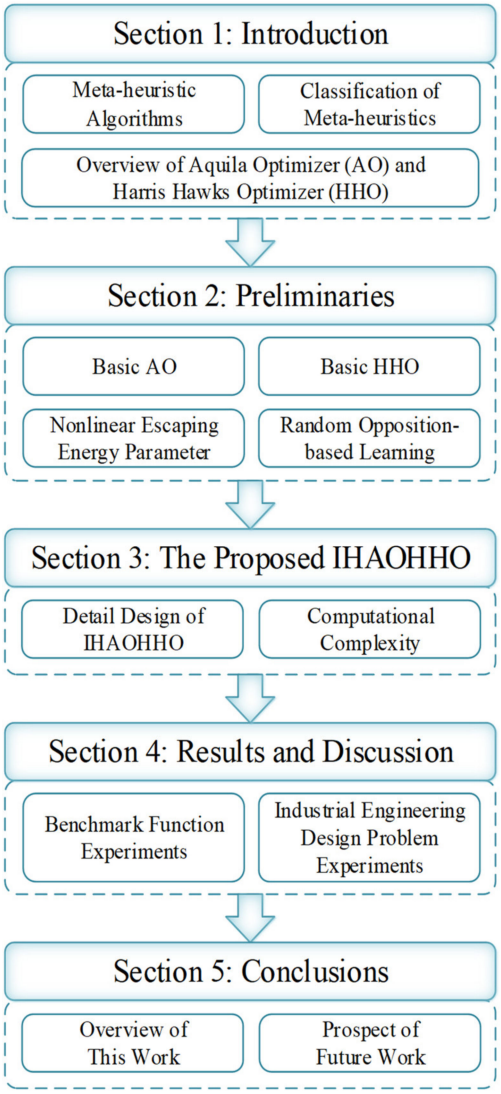

Figure 1. The overview sketch of this paper.

2. Preliminaries

2.1. Aquila Optimizer (AO)

AO is a latest novel swarm intelligence algorithm proposed by Abualigah et al. in 2021. There are four hunting behaviors of Aquila for different kinds of prey. Aquila can switch hunting strategies flexibly for different prey and then uses its fast speed united with sturdy feet and claws to attack prey. The brief description of mathematical model can be described as follows.

Step 1: Expanded exploration (X_1): high soar with a vertical stoop

In this method, the Aquila flies high over the ground and explores the search space widely, and then a vertical dive is taken once the Aquila determines the area of the prey. The mathematical representation of this behaviour is written as:

$$X_1(t+1) = X_{best}(t) \times (1 - \frac{t}{T}) + (X_M(t) - X_{best}(t) \times r_1) \quad (1)$$

$$X_M(t) = \frac{1}{N}\sum_{i=1}^{N} X_i(t) \quad (2)$$

where $X_{best}(t)$ represents the best position obtained so far, and $X_M(t)$ denotes the average position of all Aquilas in current iteration. t and T is the current iteration and the maximum

number of iterations, respectively. N is the population size, and r_1 is a random number between 0 and 1.

Step 2: Narrowed exploration (X_2): contour flight with short glide attack

This is the most commonly used hunting method for Aquila. It uses short gliding to attack the prey after descending within the selected area and flying around the prey. The position update formula is represented as:

$$X_2(t+1) = X_{best}(t) \times LF(D) + X_R(t) + (y - x) \times r_2 \tag{3}$$

where $X_R(t)$ represents a random position of the Aquila, D is the dimension size, and r_2 is a random number within (0, 1). $LF(D)$ represents Levy flight function, which is presented as follows:

$$LF(D) = s \times \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}} \tag{4}$$

$$\sigma = \left(\frac{\Gamma(1+\beta) \times \sin(\frac{\pi\beta}{2})}{\Gamma(\frac{1+\beta}{2}) \times \beta \times 2^{(\frac{\beta-1}{2})}} \right) \tag{5}$$

where s and β are constant values equal to 0.01 and 1.5, respectively, and u and v are random numbers between 0 and 1. y and x are used to present the spiral shape in the search, which are calculated as follows:

$$\begin{cases} x = r \times \sin(\theta) \\ y = r \times \cos(\theta) \\ r = r_3 + 0.00565 \times D_1 \\ \theta = -\omega \times D_1 + \frac{3 \times \pi}{2} \end{cases} \tag{6}$$

where r_3 means the number of search cycles between 1 and 20, D_1 is composed of integer numbers from 1 to the dimension size (D), and ω is equal to 0.005.

Step 3: Expanded exploitation (X_3): low flight with a slow descent attack

In the third method, when the area of prey is roughly determined, the Aquila descends vertically to do a preliminary attack. AO exploits the selected area to get close and attack the prey. This behaviour is presented as follows:

$$X_3(t+1) = (X_{best}(t) - X_M(t)) \times \alpha - r_4 + ((UB - LB) \times r_5 + LB) \times \delta \tag{7}$$

where $X_{best}(t)$ denotes to the best position obtained so far, and $X_M(t)$ means the average value of the current positions. α and δ are the exploitation adjustment parameters fixed to 0.1, UB and LB are the upper and lower bound of the problem, and r_4 and r_5 are random numbers within (0, 1).

Step 4: Narrowed exploitation (X_4): walking and grabbing prey

In this method, the Aquila chases the prey in the light of its escape trajectory and then attacks the prey on the ground. The mathematical representation of this behaviour is as follows:

$$\begin{cases} X_4(t+1) & = QF \times X_{best}(t) - (G_1 \times X(t) \times r_6) \\ & \quad -G_2 \times LF(D) + r_7 \times G_1 \\ QF(t) & = t^{\frac{2 \times rand() - 1}{(1-T)^2}} \\ G_1 = 2 \times & r_8 - 1 \\ G_2 = 2 \times & (1 - \frac{t}{T}) \end{cases} \tag{8}$$

where $X(t)$ is the current position, and $QF(t)$ represents the quality function value, which used to balance the search strategy. G_1 denotes the movement parameter of Aquila during

tracking prey, which is a random number between [–1,1]. G_2 denotes the flight slope when chasing prey, which decreases linearly from 2 to 0. r_6, r_7, and r_8 are random numbers between 0 and 1.

2.2. Harris's Hawks Optimizer (HHO)

HHO is a new meta-heuristic optimization algorithm proposed by Heidari et al. in 2019. It is inspired by the unique cooperative foraging activities of Harris's hawk. Harris's hawk can show a variety of chasing patterns according to the dynamic nature of the environment and the escaping patterns of the prey. These switching activities are conducive to confuse the running prey, and these cooperative strategies can help Harris's hawk chase the detected prey to exhaustion, which increases its vulnerability. The brief description of mathematical model is as follows.

2.2.1. Exploration Phase

The Harris's hawks usually perch on some random locations, wait, and monitor the desert to detect the prey. There are two perching strategies based on the positions of other family members and the prey or random tall trees, which is selected in accordance with the random q value.

$$X(t+1) = \begin{cases} X_r(t) - r_1|X_r(t) - 2r_2X(t)| & q \geq 0.5 \\ (X_{prey}(t) - X_m(t)) - r_3(LB + r_4(UB - LB)) & q < 0.5 \end{cases} \quad (9)$$

$$X_m(t) = \frac{1}{N}\sum_{i=1}^{N} X_i(t) \quad (10)$$

where $X_r(t)$ is the position of a random hawk, $X_{prey}(t)$ represents the position of the prey, that is the best position obtained so far, and $X_m(t)$ denotes the average position of the current population. N is total number of hawks, UB and LB are the upper and lower bound of the problem, and q, r_1, r_2, r_3, and r_4 are random numbers between 0 and 1.

2.2.2. Transition from Exploration to Exploitation Phase

The HHO algorithm has a transition mechanism from exploration to exploitation phase based on the escaping energy of the prey and then changes the different exploitative behaviors. The energy of the prey is modelled as follows, which decreases during the escaping behaviour.

$$E = 2E_0(1 - \frac{t}{T}) \quad (11)$$

where E represents the escaping energy of the prey, E_0 is the initial state of the energy, and t and T are the current and maximum number of iterations, respectively. When $|E| \geq 1$, the algorithm performs the exploration stage, and when $|E| < 1$, the algorithm performs the exploitation phase.

2.2.3. Exploitation Phase

In this phase, four different chasing and attacking strategies are proposed on the basis of the escaping energy of the prey and chasing styles of the Harris's hawks. Except for the escaping energy, parameter r is also utilized to choose the chasing strategy, which indicates the chance of the prey in successfully escaping ($r < 0.5$) or not ($r \geq 0.5$) before attack.

Soft besiege

When $r \geq 0.5$ and $|E| \geq 0.5$, the prey still has enough energy and tries to escape, so the Harris's hawks encircle it softly to make the prey more exhausted and then attack it. This behaviour is modeled as follows:

$$X(t+1) = \Delta X(t) - E\left|JX_{prey}(t) - X(t)\right| \quad (12)$$

$$\Delta X(t) = X_{prey}(t) - X(t) \tag{13}$$

$$J = 2(1 - r_5) \tag{14}$$

where $\Delta X(t)$ indicates the difference between the position of the prey and the current position, J represents the random jump strength of the prey, $X_{prey}(t)$ represents the position of the prey, $X(t)$ is the current position, and r_5 is a random number within (0, 1).

Hard besiege

When $r \geq 0.5$ and $|E| < 0.5$, the prey has a low escaping energy, and the Harris's hawks encircle the prey readily and finally attack it. In this situation, the positions are updated as follows:

$$X(t+1) = X_{prey}(t) - E\left|\Delta X(t)\right| \tag{15}$$

Soft besiege with progressive rapid dives

When $|E| \geq 0.5$ and $r < 0.5$, the prey has enough energy to successfully escape, so the Harris's hawks perform soft besiege with several rapid dives around the prey and try to progressively correct its position and direction. This behaviour is modeled as follows:

$$Y = X_{prey}(t) - E\left|JX_{prey}(t) - X(t)\right| \tag{16}$$

$$Z = Y + S \times LF(D) \tag{17}$$

$$LF(x) = 0.01 \times \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}} \tag{18}$$

$$\sigma = \left(\frac{\Gamma(1+\beta) \times sin(\frac{\pi\beta}{2})}{\Gamma(\frac{1+\beta}{2}) \times \beta \times 2^{(\frac{\beta-1}{2})}} \right)^{\frac{1}{\beta}} \tag{19}$$

$$X(t+1) = \begin{cases} Y & if\ F(Y) < F(X(t)) \\ Z & if\ F(Z) < F(X(t)) \end{cases} \tag{20}$$

where D is the dimension size of the problem, and S is a random vector. LF is Levy flight function, which is utilized to mimic the deceptive motions of the prey. u and v are random values between 0 and 1, and β is a constant number equal to 1.5. Note that only the better position between Y and Z is selected as the next position.

Hard besiege with progressive rapid dives

When $|E| < 0.5$ and $r < 0.5$, the prey does not have enough energy to escape, so the hawks perform a hard besiege to decrease the distance between their average position and the prey and finally attack and kill the prey. The mathematical representation of this behaviour is as follows:

$$Y = X_{prey}(t) - E\left|JX_{prey}(t) - X_m(t)\right| \tag{21}$$

$$Z = Y + S \times LF(D) \tag{22}$$

$$X(t+1) = \begin{cases} Y & if\ F(Y) < F(X(t)) \\ Z & if\ F(Z) < F(X(t)) \end{cases} \tag{23}$$

Note that only the better position between Y and Z will be the next position for the new iteration.

2.3. Nonlinear Escaping Energy Parameter

In the original HHO algorithm, the escaping energy E is used to control the transition from exploration to exploitation phase. The parameter E is linearly reduced from 2 to 0, that is, only local search is performed in the second half of the iterations, which is easy to

fall into local optima. In order to overcome this shortcoming of the algorithm, another way to update the escaping energy E is utilized [56]:

$$E = E_1(2 \times \text{rand} - 1) \tag{24}$$

$$E_1 = 2 \times (1 - \left(\frac{t}{T}\right)^{1/3}{}^{1/3}) \tag{25}$$

where t and T are the current and maximum number of iterations, respectively. It can be seen from Figure 2a that E_1 decreases rapidly in the early stage of the iterations, which controls the global search ability of the algorithm and changes slowly in the middle of the iterations. E_1 also balances the global and local search capabilities and decreases rapidly in the later stage of the iterations to speed up the local search. E can perform global search and local search in the whole iterative process. It mainly performs global search in the early stage and retains the possibility of global search while mainly performing local search in the later stage, as shown in Figure 2b.

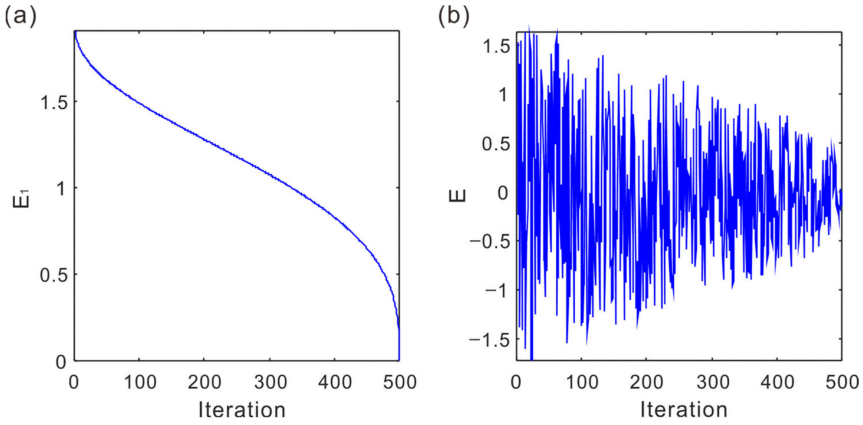

Figure 2. (**a**) E_1 curve and (**b**) E curve.

2.4. Random Opposition-Based Learning (ROBL)

Opposition-based learning (OBL) is a powerful optimization tool proposed by Tizhoosh [57]. The main idea of OBL is simultaneously considering the fitness of an estimate and its corresponding opposite estimate to obtain a better candidate solution. The OBL concept has successfully been used in varieties of meta-heuristics algorithms [58–62] to improve the convergence speed. Different from the original OBL, this paper utilizes an improved OBL strategy, called random opposition-based learning (ROBL) [63], which is defined by:

$$\hat{x}_j = l_j + u_j - rand \times x_j, \quad j = 1, 2, \ldots, n \tag{26}$$

where $\hat{x}_j$ represents the opposite solution, l_j and u_j are the lower and upper bound of the problem in jth dimension, and *rand* is a random number within (0, 1). The opposite solution described by Equation (26) is more random than the original OBL and can effectively help the population jump out of the local optima.

3. The Proposed IHAOHHO Algorithm

3.1. The Detail Design of IHAOHHO

The exploration phase of AO mimics the hunting behaviour for fast-moving prey with a wide flying area, making AO have a strong global search ability and fast convergence speed. However, the selected search space is not exhaustively searched during the exploitation phase. The effect of Levy flight is relatively weak, leading to premature convergence. In a word, the AO algorithm possesses strong randomness and fast convergence speed in the global exploration phase. However, it is easy to fall into local optima in the local exploitation stage. For the HHO algorithm, the transition from global to local search is

realized based on the energy attenuation of the prey. In the early iterations, which reflect the exploration phase, the diversity of the population is insufficient, and the convergence speed is slow. As the number of iterations increases, the energy of prey decreases, and the algorithm enters the stage of local exploitation. Four different hunting strategies are adopted in the light of the energy and escape probability of the prey. The Levy flight term is added in the exploitation phase. Whether to use Levy flight to update positions is decided by fitness values so that the algorithm can jump out of the local optima to a certain extent.

Therefore, we combine the global exploration phase of AO and the local exploitation phase of HHO to give full play to the advantages of these two algorithms. The global search capability, faster convergence speed, and the ability to jump out of the local optima of the algorithm are all retained. Meanwhile, a nonlinear escaping energy mechanism is utilized to control the transition from exploration to exploitation phase, which retains the possibility of global search in the later iterations. ROBL strategy is added to the exploitation phase to enhance further the ability to jump out of the local optima. All these strategies improve the convergence speed and accuracy of the hybrid algorithm and effectively enhance the overall optimization performance of the algorithm. This improved hybrid Aquila Optimizer and Harris Hawks Optimization algorithm is named IHAOHHO. Different phases of IHAOHHO are shown in Figure 3. The pseudo-code of IHAOHHO is given in Algorithm 1, and the summarized flowchart is illustrated in Figure 4.

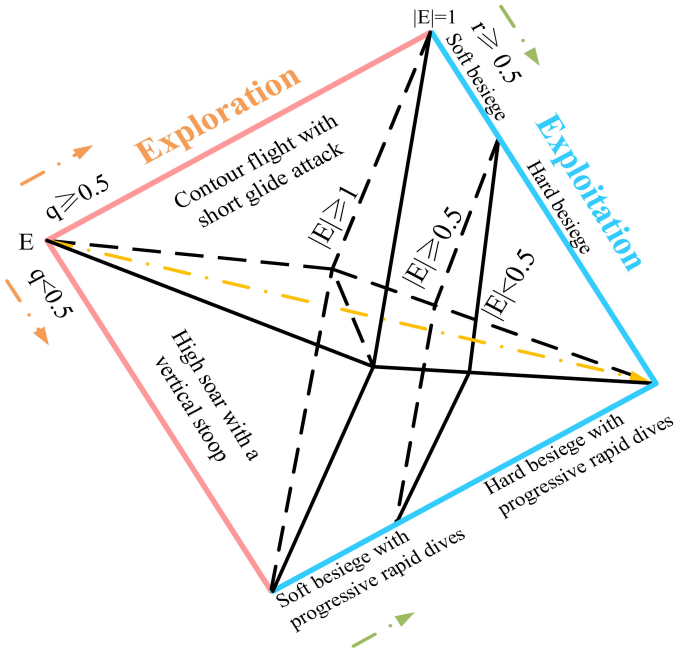

Figure 3. Different phases of IHAOHHO.

3.2. Computational Complexity of IHAOHHO

Computation complexity is a key metric for an algorithm to evaluate its time consumption during operation. The computational complexity of the IHAOHHO algorithm depends on three processes: initialization, evaluation of fitness, and updating of hawks. In the initialization stage, the computational complexity of positions generated of N hawks is O(N × D), where D is dimension size of the problem. Then, the computational complexity of fitness evaluation for the best solution is O(N) during the iteration process. Considering the worst condition, the computational complexities of position updating of hawks and fitness comparison are O(3 × N × D) and O(3 × N), respectively. In a word, the total computational complexity of the proposed IHAOHHO algorithm is O(N × D + (3 × D + 4) × N × T).

Algorithm 1 Pseudo-code of IHAOHHO.

Set initial values of the population size N and the maximum number of iterations T
Initialize positions of the population X
While t < T
  For i = 1 to N
    Check if the position goes out of the search space boundary, and bring it back.
    Calculate the fitness of X_i
    Update X_{best}
  End for
  Update x, y, QF, G_1, G_2, E_1
  For i = 1 to N
    Update E using Equation (24) % Nonlinear escaping energy parameter
    If $|E| \geq 1$ % Exploration part of AO
      If rand < 0.5
        Update the position of $Xnew_i$ using Equation (1)
          If $f(Xnew_i) < f(X_i)$
            $X_i = Xnew_i$
          End if
        Else
        Update the position of $Xnew_i$ using Equation (3)
          If $f(Xnew_i) < f(X_i)$
            $X_i = Xnew_i$
          End if
      End if
    Else % Exploitation part of HHO
      If $r \geq 0.5$ and $|E| \geq 0.5$
        Update the position of X_i using Equation (12)
      End if
      If $r \geq 0.5$ and $|E| < 0.5$
        Update the position of X_i using Equation (15)
      End if
      If $r < 0.5$ and $|E| \geq 0.5$
        Update the position of $Xnew_i$ using Equation (16)
        If $f(Xnew_i) < f(X_i)$
          $X_i = Xnew_i$
        Else
          Update the position of $Xnew_i$ using Equation (17)
          If $f(Xnew_i) < f(X_i)$
            $X_i = Xnew_i$
          End if
        End if
      End if
      If $r < 0.5$ and $|E| < 0.5$
        Update the position of $Xnew_i$ using Equation (21)
        If $f(Xnew_i) < f(X_i)$
          $X_i = Xnew_i$
        Else
          Update the position of $Xnew_i$ using Equation (22)
          If $f(Xnew_i) < f(X_i)$
            $X_i = Xnew_i$
          End if
        End if
      End if
      Update the position of $Xnew_i$ using Equation (26) % ROBL
      If $f(Xnew_i) < f(X_i)$
        $X_i = Xnew_i$
      End if
    End if
    t = t + 1
  End for
End while
Return X_{best}

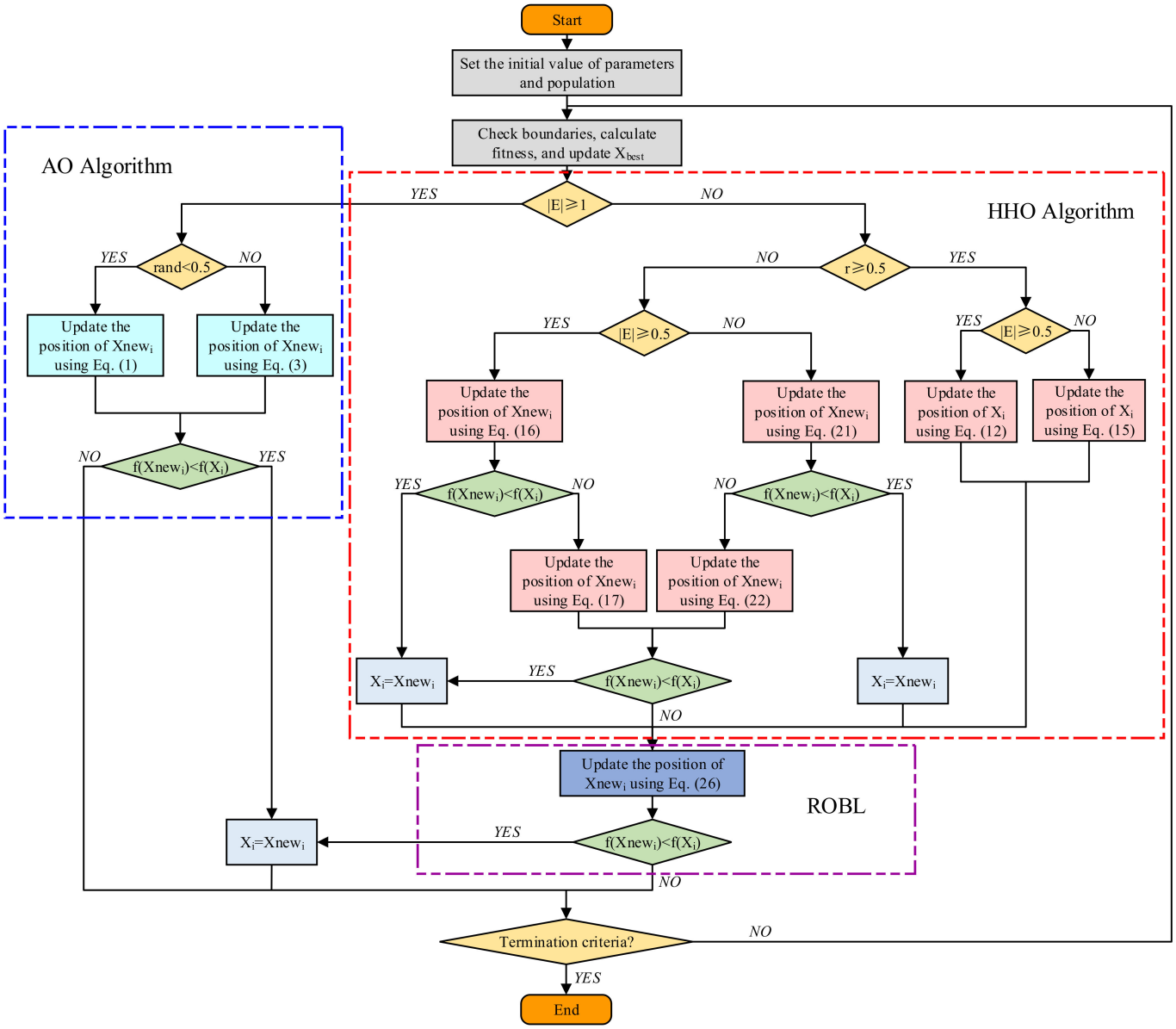

Figure 4. IHAOHHO algorithm flowchart.

4. Results and Discussion

In this section, two main experiments were carried out to evaluate the performance of the IHAOHHO algorithm. The first kind of experiments is benchmark function experiments, which aimed to evaluate the performance of IHAOHHO in solving 23 numerical optimization problems. The second experiment is industrial engineering design problems, which aimed to evaluate the performance of IHAOHHO in solving real-world problems. All experiments are implemented in MATLAB R2016a on a PC with Intel (R) core (TM) i5-9500 CPU @ 3.00 GHz and RAM 2 GB memory on OS windows 10.

4.1. Benchmark Function Experiments

To investigate the performance of the IHAOHHO algorithm, 23 standard benchmark functions of three different types were utilized for testing [64]. The main characteristic of the first type, namely unimodal benchmark functions, is that there is only one global optimum with no local optima. These test functions can be used to evaluate the exploitation capability and convergence rate of an algorithm. The second type, namely multimodal benchmark functions, has a global optimum and multiple local optima, which includes general and

fixed-dimension multimodal test functions. This type of functions was utilized to evaluate the exploitation and local optima avoidance capability of the algorithm. The benchmark function details, including dimensions, ranges, and optima, are listed in Tables 1–3.

Table 1. Unimodal benchmark functions.

Function	Dim	Range	Fmin
$F_1(x) = \sum_{i=1}^{n} x_i^2$	30	$(-100, 100)$	0
$F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$(-10, 10)$	0
$F_3(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j\right)^2$	30	$(-100, 100)$	0
$F_4(x) = \max_i\{\lvert x_i \rvert, 1 \le i \le n\}$	30	$(-100, 100)$	0
$F_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	$(-30,30)$	0
$F_6(x) = \sum_{i=1}^{n} (x_i + 5)^2$	30	$(-100, 100)$	0
$F_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	30	$(-1.28, 1.28)$	0

Table 2. Multimodal benchmark functions.

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

Table 3. Fixed-dimension multimodal benchmark functions.

Function	Dim	Range	Fmin
$F_{14}(x) = (\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6})^{-1}$	2	$(-65, 65)$	1
$F_{15}(x) = \sum_{i=1}^{11} [a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}]^2$	4	$(-5, 5)$	0.00030
$F_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + x_2^4$	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	$(-1, 2)$	-3.86
$F_{20}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)$	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

For verification of the results, the IHAOHHO algorithm was compared with the original AO and HHO; SMA as one of the recent algorithms; SSA, WOA, and GWO as several classical meta-heuristic algorithms; and PSO as the most well-known swarm intelligence algorithm. For all these algorithms, we set the population size N = 30, dimension size D = 30, maximum number of iterations T = 500, and ran them 30 times independently. The parameter settings of each algorithm are shown in Table 4. After all, the average and standard deviation results of these test functions are exhibited in Tables 4–6. Figure 5 shows the convergence curves of 23 test functions. The partial search history, trajectory and average fitness maps are represented in Figure 6. Wilcoxon signed-rank test results are also listed in Table 6. The detailed data analysis is given in the following subsections.

Table 4. Parameter settings for the comparative algorithms.

Algorithm	Parameters
AO	U = 0.00565; r1 = 10; ω = 0.005; α = 0.1; δ = 0.1; G1 $\in$ [−1, 1]; G2 = [2, 0]
HHO	q $\in$ [0, 1]; r $\in$ [0, 1]; E0 $\in$ [−1, 1]; E1 = [2, 0]; E $\in$ [−2, 2];
SMA	z = 0.03
SSA	c1 = [1, 0]; c2 $\in$ [0, 1]; c3 $\in$ [0, 1]
WOA	a1 = [2, 0]; a2 = [−1, −2]; b = 1
GWO	a = [2, 0]
PSO	c1 = 2; c2 = 2; vmax = 6

4.1.1. Evaluation of Exploitation Capability (Functions F1–F7)

Functions F1–F7 are used to investigate the exploitation capability of the algorithm since they have only one global optimum and no local optima. It can be seen from Table 5 that IHAOHHO can achieve much better results than other meta-heuristic algorithms excluding F6. For F1 and F3, IHAOHHO can find the theoretical optimum. For all unimodal functions excluding F6, IHAOHHO gets the smallest average values and standard deviations compared to other algorithms, which indicate the best accuracy and stability. Hence, the exploitation capability of the proposed IHAOHHO algorithm is excellent.

4.1.2. Evaluation of Exploration Capability (Functions F8–F23)

Multimodal functions F8–F23 contain plentiful local optima whose number increases exponentially with the dimension size of the problem. This kind of functions is very useful to evaluate the exploration ability of the algorithm. From the results shown in Table 5, IHAOHHO outperforms other algorithms in most of the multimodal and fixed-dimension multimodal functions. For multimodal functions F8–F13, IHAOHHO almost obtains all the best average values and standard deviations. Among ten fixed-dimensions multimodal functions F14–F23, IHAOHHO achieves the best accuracy of eight functions and best stability of four functions. These results indicate that IHAOHHO also provides robust exploitation capability.

Table 5. Results of algorithms on 23 benchmark functions.

F		IHAOHHO	AO	HHO	SMA	SSA	WOA	GWO	PSO
F1	Avg	0.0000×10^{0}	2.5120×10^{-128}	1.7359×10^{-98}	6.7559×10^{-287}	2.0918×10^{-7}	7.0172×10^{-75}	2.7553×10^{-27}	1.7920×10^{-4}
	Std	0.0000×10^{0}	1.3759×10^{-127}	3.8748×10^{-98}	0.0000×10^{0}	2.5521×10^{-7}	2.0985×10^{-74}	7.4745×10^{-27}	2.1473×10^{-4}
F2	Avg	3.1773×10^{-283}	3.0714×10^{-51}	3.6162×10^{-49}	1.7722×10^{-136}	2.1400×10^{0}	2.1103×10^{-49}	7.2224×10^{-17}	2.2676×10^{-1}
	Std	0.0000×10^{0}	1.6823×10^{-50}	1.9747×10^{-48}	9.7069×10^{-136}	1.5737×10^{0}	1.1221×10^{-48}	4.3158×10^{-17}	2.0215×10^{-2}
F3	Avg	0.0000×10^{0}	2.3884×10^{-101}	7.9368×10^{-70}	2.7958×10^{-305}	1.5707×10^{3}	4.8346×10^{4}	1.9688×10^{-5}	8.7992×10^{1}
	Std	0.0000×10^{0}	9.262×10^{-101}	4.3417×10^{-69}	0.0000×10^{0}	1.0057×10^{3}	1.5295×10^{4}	8.5080×10^{-5}	3.7192×10^{1}
F4	Avg	1.1105×10^{-281}	1.0656×10^{-53}	1.2768×10^{-49}	1.0217×10^{-160}	1.1623×10^{1}	5.4222×10^{1}	9.2533×10^{-7}	1.0783×10^{0}
	Std	0.0000×10^{0}	5.8309×10^{-53}	4.4293×10^{-49}	5.5961×10^{-160}	3.3373×10^{0}	2.9852×10^{1}	9.1688×10^{-7}	2.1854×10^{-1}
F5	Avg	2.8203×10^{-3}	6.4303×10^{-3}	1.1390×10^{-2}	9.4019×10^{0}	3.1709×10^{2}	2.7969×10^{1}	2.7412×10^{1}	1.0424×10^{2}
	Std	4.4716×10^{-3}	9.1289×10^{-3}	1.2058×10^{-2}	1.2466×10^{1}	8.0601×10^{2}	4.5551×10^{-1}	8.8086×10^{-1}	9.9130×10^{1}
F6	Avg	4.2411×10^{-6}	1.1861×10^{-4}	1.1430×10^{-4}	5.2584×10^{-3}	3.5188×10^{-7}	3.6078×10^{-1}	8.0826×10^{-1}	1.1828×10^{-4}
	Std	6.2092×10^{-6}	2.1625×10^{-4}	1.4084×10^{-4}	3.1160×10^{-3}	7.3563×10^{-7}	1.8848×10^{-1}	3.3042×10^{-1}	1.3013×10^{-4}
F7	Avg	7.1381×10^{-5}	9.2969×10^{-5}	1.4408×10^{-4}	2.2317×10^{-4}	1.7310×10^{-1}	2.6756×10^{-3}	2.2547×10^{-3}	1.8040×10^{-1}
	Std	7.6852×10^{-5}	1.1466×10^{-4}	1.5482×10^{-4}	1.6750×10^{-4}	7.8997×10^{-2}	2.3949×10^{-3}	1.1317×10^{-3}	7.5627×10^{-2}
F8	Avg	−12,447.8654	−7073.9882	−12,568.7811	−12,568.9426	−7591.3246	−10,430.3986	−6049.3246	−5317.3115
	Std	4.5359×10^{2}	3.5511×10^{3}	1.3999×10^{0}	4.0261×10^{-1}	6.9106×10^{2}	1.9097×10^{3}	8.0214×10^{2}	1.5005×10^{3}
F9	Avg	0.0000×10^{0}	0.0000×10^{0}	0.0000×10^{0}	0.0000×10^{0}	5.5253×10^{1}	1.8948×10^{-15}	4.8419×10^{0}	5.6659×10^{1}
	Std	0.0000×10^{0}	0.0000×10^{0}	0.0000×10^{0}	0.0000×10^{0}	1.9037×10^{1}	1.0378×10^{-14}	6.2042×10^{0}	1.5111×10^{1}
F10	Avg	8.8818×10^{-16}	8.8818×10^{-16}	8.8818×10^{-16}	8.8818×10^{-16}	2.7561×10^{0}	3.9672×10^{-15}	1.0356×10^{-13}	2.0903×10^{-1}
	Std	0.0000×10^{0}	0.0000×10^{0}	0.0000×10^{0}	0.0000×10^{0}	1.9773×10^{0}	2.4210×10^{-15}	2.1323×10^{-14}	4.4871×10^{-1}
F11	Avg	0.0000×10^{0}	0.0000×10^{0}	0.0000×10^{0}	0.0000×10^{0}	1.7030×10^{-2}	5.9385×10^{-3}	2.5384×10^{-3}	4.9459×10^{-3}
	Std	0.0000×10^{0}	0.0000×10^{0}	0.0000×10^{0}	0.0000×10^{0}	1.9430×10^{-2}	3.2527×10^{-2}	8.7348×10^{-3}	9.4682×10^{-3}
F12	Avg	5.3164×10^{-7}	4.6513×10^{-6}	9.2636×10^{-6}	5.0331×10^{-3}	7.0564×10^{0}	2.5778×10^{-2}	3.7730×10^{-2}	6.9126×10^{-3}
	Std	9.6698×10^{-7}	8.9371×10^{-6}	1.2911×10^{-5}	6.3463×10^{-3}	3.0595×10^{0}	2.0942×10^{-2}	1.8369×10^{-2}	2.6301×10^{-2}
F13	Avg	1.1694×10^{-5}	3.3938×10^{-5}	1.2604×10^{-4}	7.3800×10^{-3}	1.7887×10^{1}	5.8549×10^{-1}	6.1135×10^{-1}	4.4120×10^{-3}
	Std	1.7961×10^{-5}	3.2363×10^{-5}	1.5375×10^{-4}	8.9329×10^{-3}	1.5307×10^{1}	2.9719×10^{-1}	1.7136×10^{-1}	6.6275×10^{-3}
F14	Avg	1.7919×10^{0}	1.5940×10^{0}	1.1635×10^{0}	9.9800×10^{-1}	1.1637×10^{0}	5.0748×10^{0}	5.2681×10^{0}	3.5906×10^{0}
	Std	9.1746×10^{-1}	2.1763×10^{0}	4.5784×10^{-1}	1.1156×10^{-12}	3.7678×10^{-1}	4.4603×10^{0}	4.6022×10^{0}	2.904×10^{0}
F15	Avg	3.5291×10^{-4}	5.5590×10^{-4}	4.0350×10^{-4}	5.1576×10^{-4}	2.8218×10^{-3}	6.6118×10^{-4}	6.3719×10^{-3}	9.3864×10^{-4}
	Std	4.8766×10^{-5}	1.1640×10^{-4}	2.3353×10^{-4}	3.0066×10^{-4}	5.9580×10^{-3}	7.1226×10^{-4}	1.2424×10^{-2}	2.6081×10^{-4}
F16	Avg	-1.0316×10^{0}	-1.0311×10^{0}	-1.0316×10^{0}	-1.0316×10^{0}	-1.0316×10^{0}	-1.0316×10^{0}	-1.0316×10^{0}	-1.0316×10^{0}
	Std	1.0379×10^{-10}	3.7614×10^{-4}	2.5745×10^{-9}	4.3934×10^{-10}	2.0489×10^{-14}	6.1164×10^{-10}	1.4772×10^{-8}	6.4539×10^{-16}
F17	Avg	3.9789×10^{-1}	3.9812×10^{-1}	3.9790×10^{-1}	3.9789×10^{-1}	3.9789×10^{-1}	3.9789×10^{-1}	3.9789×10^{-1}	3.9789×10^{-1}
	Std	5.4022×10^{-7}	2.2378×10^{-4}	2.4237×10^{-5}	2.4814×10^{-8}	1.4663×10^{-14}	8.5493×10^{-6}	8.9987×10^{-7}	0.0000×10^{0}
F18	Avg	3.0000×10^{0}	3.0439×10^{0}	3.0000×10^{0}	3.0000×10^{0}	3.0000×10^{0}	3.0000×10^{0}	3.0000×10^{0}	3.0000×10^{0}
	Std	2.714×10^{-7}	6.4693×10^{-2}	1.6198×10^{-7}	4.7705×10^{-10}	9.5042×10^{-14}	2.6269×10^{-4}	4.7607×10^{-5}	1.639×10^{-15}

Table 5. *Cont.*

F		IHAOHHO	AO	HHO	SMA	SSA	WOA	GWO	PSO
F19	Avg	-3.8628×10^{0}	-3.8539×10^{0}	-3.8616×10^{0}	-3.8628×10^{0}	-3.8628×10^{0}	-3.8597×10^{0}	-3.8593×10^{0}	-3.8628×10^{0}
	Std	1.8351×10^{-4}	6.0669×10^{-3}	1.7013×10^{-3}	3.0254×10^{-7}	8.1972×10^{-13}	3.1652×10^{-3}	4.2427×10^{-3}	2.6823×10^{-15}
F20	Avg	-3.1298×10^{0}	-3.1572×10^{0}	-3.0533×10^{0}	-3.2425×10^{0}	-3.2215×10^{0}	-3.2391×10^{0}	-3.2442×10^{0}	-3.2665×10^{0}
	Std	1.1264×10^{-1}	1.0448×10^{-1}	1.1671×10^{-1}	5.7177×10^{-2}	5.1720×10^{-2}	1.3596×10^{-1}	9.0427×10^{-2}	6.0328×10^{-2}
F21	Avg	-1.0152×10^{1}	-1.0142×10^{1}	-5.5370×10^{0}	-1.0152×10^{1}	-7.3774×10^{0}	-9.0891×10^{0}	-9.1419×10^{0}	-6.7868×10^{0}
	Std	5.6352×10^{-4}	1.8288×10^{-2}	1.484×10^{0}	2.2592×10^{-3}	2.9079×10^{0}	2.0545×10^{0}	2.3491×10^{0}	3.2622×10^{0}
F22	Avg	-1.0402×10^{1}	-1.0388×10^{1}	-5.2528×10^{0}	-1.0402×10^{1}	-8.1232×10^{0}	-7.5395×10^{0}	-1.0401×10^{1}	-8.1542×10^{0}
	Std	6.3272×10^{-4}	2.4782×10^{-2}	9.3628×10^{-1}	7.5981×10^{-4}	3.3371×10^{0}	3.1570×10^{0}	8.9128×10^{-4}	3.2898×10^{0}
F23	Avg	-1.0535×10^{1}	-1.0525×10^{1}	-5.2858×10^{0}	-1.0535×10^{1}	-7.6861×10^{0}	-6.6213×10^{0}	-1.0535×10^{1}	-1.0087×10^{1}
	Std	9.8617×10^{-4}	6.9516×10^{-3}	8.8012×10^{-1}	1.3006×10^{-3}	3.6004×10^{0}	3.0127×10^{0}	9.0143×10^{-4}	1.7472×10^{0}

Table 6. *p*-Values from the Wilcoxon signed-rank test for the results in Table 5.

F	IHAOHHO vs. AO	IHAOHHO vs. HHO	IHAOHHO vs. SMA	IHAOHHO vs. SSA	IHAOHHO vs. WOA	IHAOHHO vs. GWO	IHAOHHO vs. PSO
F1	6.1035×10^{-5}	6.1035×10^{-5}	N/A	6.1035×10^{-5}	6.1035×10^{-5}	6.1035×10^{-5}	6.1035×10^{-5}
F2	6.1035×10^{-5}	6.1035×10^{-5}	6.1035×10^{-5}	6.1035×10^{-5}	6.1035×10^{-5}	6.1035×10^{-5}	6.1035×10^{-5}
F3	6.1035×10^{-5}	6.1035×10^{-5}	N/A	6.1035×10^{-5}	6.1035×10^{-5}	6.1035×10^{-5}	6.1035×10^{-5}
F4	6.1035×10^{-5}	6.1035×10^{-5}	1.2207×10^{-4}	6.1035×10^{-5}	6.1035×10^{-5}	6.1035×10^{-5}	6.1035×10^{-5}
F5	6.7877×10^{-1}	6.3867×10^{-1}	6.1035×10^{-5}	6.1035×10^{-5}	6.1035×10^{-5}	6.1035×10^{-5}	6.1035×10^{-5}
F6	1.5076×10^{-2}	8.5449×10^{-4}	6.1035×10^{-5}	8.5449×10^{-4}	6.1035×10^{-5}	6.1035×10^{-5}	6.1035×10^{-5}
F7	8.0396×10^{-1}	4.2725×10^{-3}	3.0518×10^{-4}	6.1035×10^{-5}	1.2207×10^{-4}	6.1035×10^{-5}	6.1035×10^{-5}
F8	1.0699×10^{-3}	5.5359×10^{-3}	6.7139×10^{-3}	8.5449×10^{-4}	7.2998×10^{-2}	6.1035×10^{-5}	6.1035×10^{-5}
F9	N/A	N/A	N/A	6.1035×10^{-5}	N/A	6.1035×10^{-5}	6.1035×10^{-5}
F10	N/A	N/A	N/A	6.1035×10^{-5}	4.8828×10^{-4}	6.1035×10^{-5}	6.1035×10^{-5}
F11	N/A	N/A	N/A	6.1035×10^{-5}	N/A	6.2500×10^{-2}	6.1035×10^{-5}
F12	9.7797×10^{-1}	2.7686×10^{-1}	6.1035×10^{-5}	6.1035×10^{-5}	6.1035×10^{-5}	6.1035×10^{-5}	5.2448×10^{-1}
F13	8.9038×10^{-1}	3.5339×10^{-2}	6.1035×10^{-5}	6.1035×10^{-5}	6.1035×10^{-5}	6.1035×10^{-5}	2.1545×10^{-2}
F14	3.5339×10^{-2}	3.8940×10^{-2}	1.2207×10^{-4}	4.7913×10^{-2}	6.1035×10^{-4}	1.0699×10^{-1}	2.1545×10^{-2}
F15	3.3569×10^{-3}	7.1973×10^{-1}	4.7913×10^{-2}	6.1035×10^{-5}	1.1597×10^{-3}	2.1545×10^{-2}	6.1035×10^{-5}
F16	6.1035×10^{-5}	3.0151×10^{-2}	8.5449×10^{-4}	4.0283×10^{-3}	4.2725×10^{-3}	6.1035×10^{-5}	1.2207×10^{-4}
F17	6.1035×10^{-5}	3.0280×10^{-1}	1.0254×10^{-2}	6.1035×10^{-5}	6.7139×10^{-3}	2.5574×10^{-2}	6.1035×10^{-5}
F18	6.1035×10^{-5}	8.3618×10^{-3}	8.3618×10^{-3}	3.0518×10^{-4}	1.2207×10^{-4}	6.1035×10^{-5}	6.1035×10^{-5}
F19	N/A	N/A	N/A	N/A	N/A	N/A	6.1035×10^{-5}
F20	7.2998×10^{-2}	1.8762×10^{-1}	7.2998×10^{-2}	1.0699×10^{-2}	2.7686×10^{-1}	1.0254×10^{-2}	3.3569×10^{-3}
F21	1.8762×10^{-1}	6.1035×10^{-5}	4.8871×10^{-1}	4.2120×10^{-1}	8.5449×10^{-4}	5.9949×10^{-3}	2.5574×10^{-2}
F22	4.7913×10^{-2}	6.1035×10^{-5}	1.8066×10^{-2}	8.0396×10^{-1}	1.2207×10^{-4}	2.0776×10^{-1}	8.3618×10^{-3}
F23	6.1035×10^{-5}	6.1035×10^{-5}	5.5359×10^{-2}	8.3252×10^{-2}	6.1035×10^{-5}	8.3252×10^{-2}	8.3252×10^{-2}

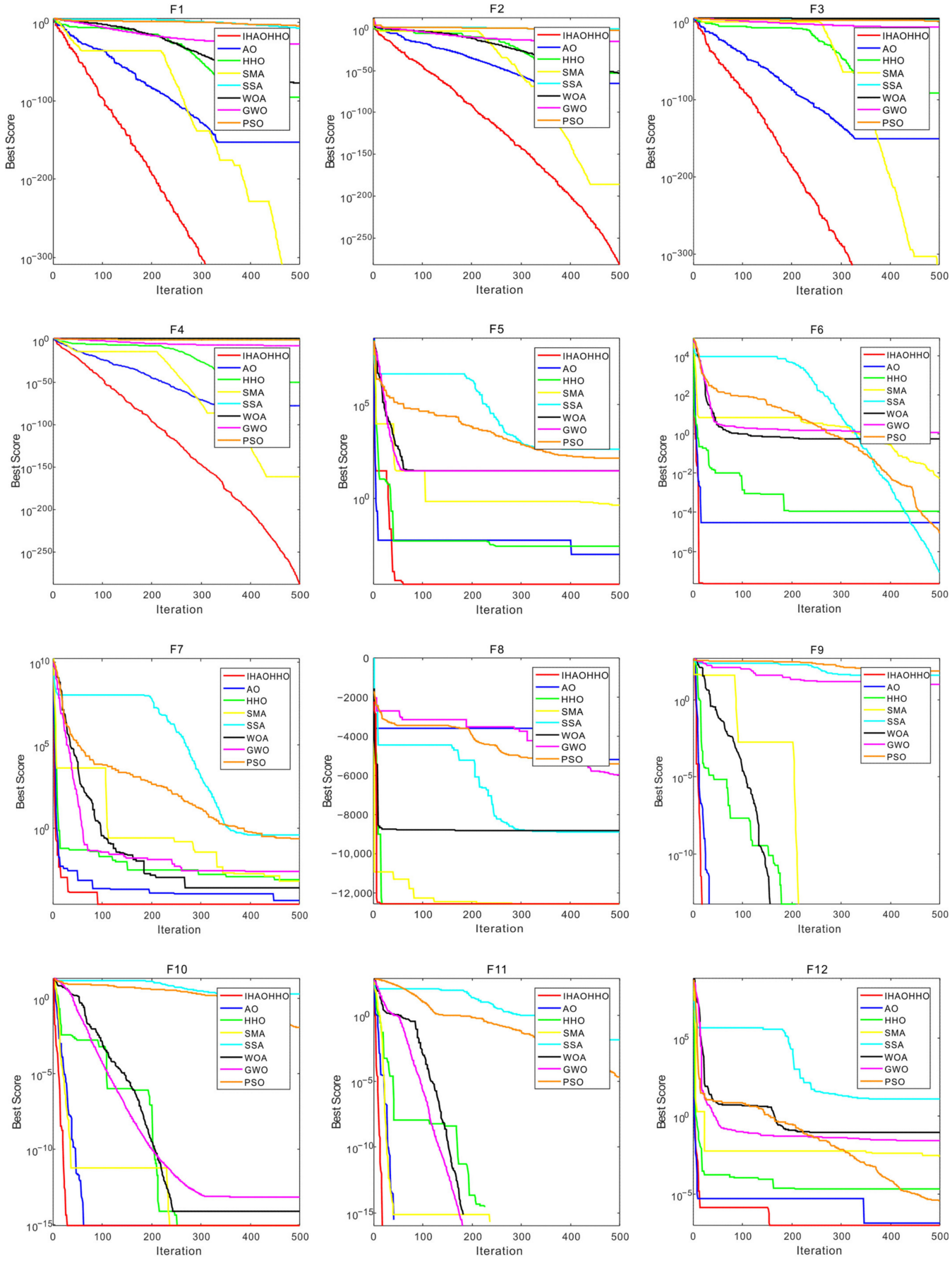

Figure 5. *Cont.*

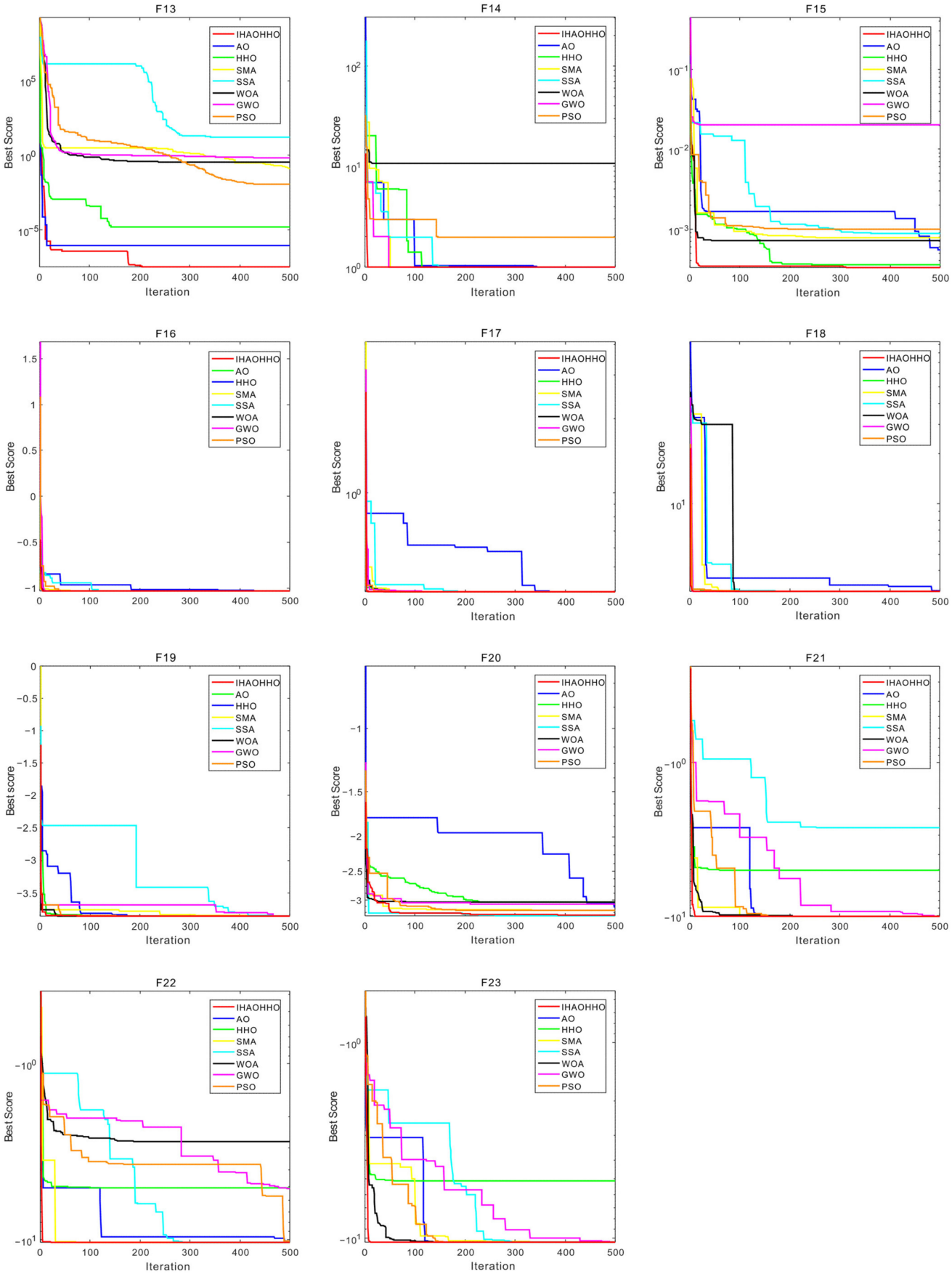

Figure 5. Convergence curves of 23 benchmark functions.

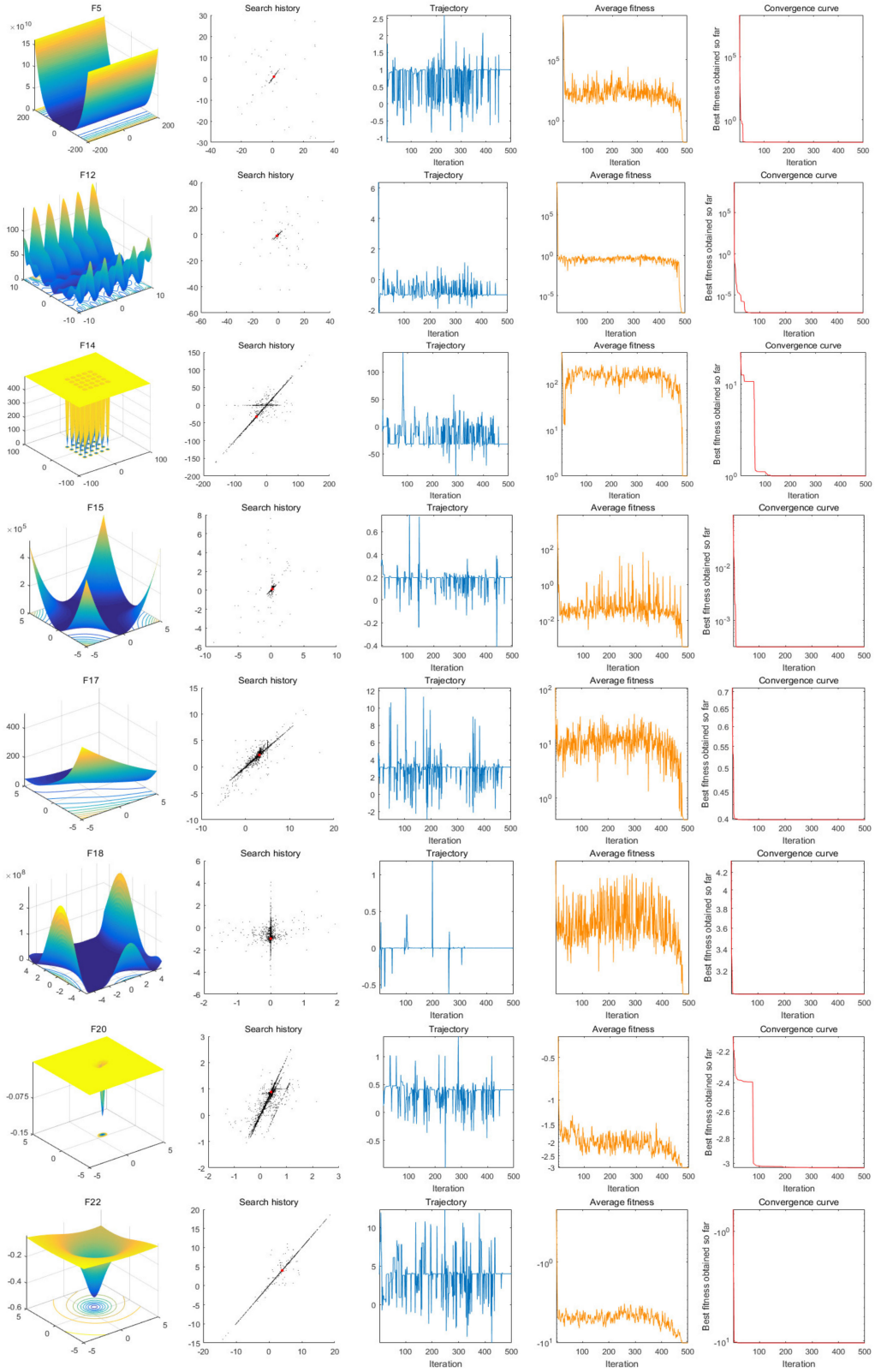

Figure 6. Parameter space, search history, trajectory, average fitness, and convergence curve of IHAOHHO.

4.1.3. Analysis of Convergence Behavior

In the light of the mathematical formula of the IHAOHHO algorithm, search agents tend to investigate promising regions of the search space widely and then exploit it in detail. Search agents change drastically in early iterations and then converge gradually as

the number of iterations increases. Convergence curves of the proposed IHAOHHO and AO, HHO, SMA, SSA, WOA, GWO, and PSO for 23 benchmark functions are provided in Figure 5, which shows the convergence rate of algorithms. It can be seen that IHAOHHO shows great superiority compared to other state-of-the-art algorithms. The IHAOHHO algorithm presents three different convergence behaviors during optimization processes. Firstly, for F1-F4, IHAOHHO gradually converges to the optimal values at a faster speed than other algorithms, and the optimal value is better than the others in three of the functions. The second behaviour is extremely fast convergence speed, as observed in F6, F8–F11, F14–F19, and F21–F23. For these functions, IHAOHHO can find the optimum at an extremely fast speed within 20 iterations, and the accurate approximation of the global optimum is almost the best. The last behaviour is observed in F5, F7, F12, F13, and F20 and shows the local optimum avoidance capability of IHAOHHO. The proposed algorithm jumps out of the local optimum after several times of stagnation. This is probably due to the effect of nonlinear escaping energy parameter. Overall, IHAOHHO can efficiently achieve great solutions for all these 23 standard benchmark functions.

In addition, the search history, trajectory, and average fitness figures of several functions are given in Figure 6. Search history figures show us how the algorithm explores and exploits the search space while solving optimization problems. Trajectory figures reveal the order in which an algorithm explores and exploits the search space. Meanwhile, average fitness presents if exploration and exploitation are conducive to improve the first random population, and an accurate approximation of the global optimum can be found in the end. Inspecting Figure 6, the IHAOHHO algorithm samples the most promising areas observed from search histories. Because of the fast convergence, the vast majority of search agents are concentrated near the global optimum. From trajectory and average fitness maps, it can be noticed that exploration almost spread throughout the iterative process until the last 50 iterations focused on exploitation, and average fitness decreased abruptly and leveled off accordingly. The average fitness figures also show the great improvement of the first random population and the acquisition of the final global optimal accurate approximation.

4.1.4. The Wilcoxon Test

Furthermore, the Wilcoxon signed-rank test results are listed in Table 6, which is used to evaluate the statistical performance differences between the proposed IHAOHHO algorithm with other algorithms. It is worth noting that a p-value less than 0.05 means that there is a significant difference between the two compared algorithms. In the light of this criterion, IHAOHHO outperforms all other algorithms in varying degrees. This superiority is statistically significant on unimodal functions F1–F7, which indicates that IHAOHHO benefits from high exploitation. IHAOHHO also shows better results on multimodal function F8–F23, from which we may conclude that IHAOHHO has a high capability of exploration to investigate the most promising regions in the search space. To sum up, the IHAOHHO algorithm can provide better results on almost all benchmark functions than other comparative algorithms.

4.1.5. Computation Time

The computation time is useful to assess the efficiency for an algorithm in solving optimization problems. From the computation time results of all algorithms shown in Table 7, it is obvious that IHAOHHO spent more time in solving these benchmark functions than other comparative algorithms, especially the earlier classic methods of SSA, WOA, GWO, and PSO. The computation time of IHAOHHO is also slightly longer than the basic AO and HHO, which may be ascribed to the ROBL strategy. ROBL produces one more candidate solution in each iteration, increasing the computation time. However, the IHAOHHO took less time than SMA on most test functions. In view of the best search performance of IHAOHHO and the rapid development of the computational machines, it is acceptable for the proposed algorithm to improve the optimization performance.

Table 7. Computation time results of algorithms on 23 benchmark functions.

F	IHAOHHO	AO	HHO	SMA	SSA	WOA	GWO	PSO
F1	2.8539×10^{-1}	2.3253×10^{-1}	1.3713×10^{-1}	8.8997×10^{-1}	8.5420×10^{-2}	7.5875×10^{-2}	1.1491×10^{-1}	6.5132×10^{-2}
F2	2.8946×10^{-1}	2.5214×10^{-1}	1.4672×10^{-1}	9.1203×10^{-1}	1.0346×10^{-1}	1.1982×10^{-1}	1.2761×10^{-1}	7.3814×10^{-2}
F3	1.6030×10^{0}	9.2890×10^{-1}	9.3324×10^{-1}	1.2673×10^{0}	4.6382×10^{-1}	3.9400×10^{-1}	4.2700×10^{-1}	3.9204×10^{-1}
F4	2.8070×10^{-1}	1.9787×10^{-1}	1.5712×10^{-1}	9.5399×10^{-1}	8.2341×10^{-2}	7.3767×10^{-2}	1.1442×10^{-1}	6.4915×10^{-2}
F5	3.3725×10^{-1}	2.2214×10^{-1}	2.2123×10^{-1}	1.0204×10^{0}	9.8470×10^{-2}	8.7778×10^{-2}	1.2667×10^{-1}	7.8503×10^{-2}
F6	2.7707×10^{-1}	2.0399×10^{-1}	1.7800×10^{-1}	9.0977×10^{-1}	8.2725×10^{-2}	7.4251×10^{-2}	1.1248×10^{-1}	6.5708×10^{-2}
F7	5.0109×10^{-1}	3.0078×10^{-1}	2.8662×10^{-1}	9.5443×10^{-1}	1.3880×10^{-1}	1.2862×10^{-1}	1.6701×10^{-1}	1.1976×10^{-1}
F8	3.9395×10^{-1}	2.3581×10^{-1}	2.3276×10^{-1}	9.7695×10^{-1}	1.0531×10^{-1}	9.7443×10^{-2}	1.3674×10^{-1}	9.1720×10^{-2}
F9	3.2379×10^{-1}	1.9907×10^{-1}	1.9594×10^{-1}	9.5132×10^{-1}	9.5204×10^{-2}	7.9254×10^{-2}	1.1801×10^{-1}	7.4441×10^{-2}
F10	3.5602×10^{-1}	2.3037×10^{-1}	2.3125×10^{-1}	9.4870×10^{-1}	1.0399×10^{-1}	9.0064×10^{-2}	1.2725×10^{-1}	8.3986×10^{-2}
F11	4.0659×10^{-1}	2.4303×10^{-1}	2.4198×10^{-1}	9.3026×10^{-1}	1.1382×10^{-1}	1.0089×10^{-1}	1.3566×10^{-1}	9.2499×10^{-2}
F12	1.0131×10^{0}	6.0006×10^{-1}	6.9400×10^{-1}	1.1939×10^{0}	2.6401×10^{-1}	2.5229×10^{-1}	3.4237×10^{-1}	2.4517×10^{-1}
F13	1.0300×10^{0}	5.6112×10^{-1}	6.1205×10^{-1}	1.1549×10^{0}	2.7393×10^{-1}	2.7208×10^{-1}	3.3915×10^{-1}	2.4746×10^{-1}
F14	2.3159×10^{0}	1.2173×10^{0}	1.5168×10^{0}	8.9676×10^{-1}	5.9818×10^{-1}	6.0722×10^{-1}	5.9328×10^{-1}	5.5450×10^{-1}
F15	2.6135×10^{-1}	1.7086×10^{-1}	1.7031×10^{-1}	3.4136×10^{-1}	9.9034×10^{-2}	7.5482×10^{-2}	6.4104×10^{-2}	4.2546×10^{-2}
F16	2.0719×10^{-1}	1.4146×10^{-1}	1.3859×10^{-1}	2.7170×10^{-1}	5.9081×10^{-2}	4.9666×10^{-2}	6.0033×10^{-2}	4.1193×10^{-2}
F17	1.8138×10^{-1}	1.3529×10^{-1}	1.5833×10^{-1}	2.7311×10^{-1}	5.2979×10^{-2}	4.1321×10^{-2}	4.1556×10^{-2}	2.3066×10^{-2}
F18	1.8108×10^{-1}	1.3183×10^{-1}	1.2693×10^{-1}	2.7041×10^{-1}	5.4471×10^{-2}	4.0487×10^{-2}	4.1752×10^{-2}	2.2830×10^{-2}
F19	3.5119×10^{-1}	2.4635×10^{-1}	2.4016×10^{-1}	3.4125×10^{-1}	9.8544×10^{-2}	8.5903×10^{-2}	9.2049×10^{-2}	7.0253×10^{-2}
F20	3.7106×10^{-1}	2.2549×10^{-1}	2.4656×10^{-1}	4.0519×10^{-1}	1.0270×10^{-1}	9.0294×10^{-2}	9.8020×10^{-2}	7.2095×10^{-2}
F21	5.8451×10^{-1}	3.2169×10^{-1}	3.6385×10^{-1}	4.1713×10^{-1}	1.4920×10^{-1}	1.3664×10^{-1}	1.3885×10^{-1}	1.2170×10^{-1}
F22	7.2861×10^{-1}	3.9414×10^{-1}	4.3943×10^{-1}	4.9071×10^{-1}	1.8789×10^{-1}	1.7154×10^{-1}	1.7638×10^{-1}	1.5215×10^{-1}
F23	9.4549×10^{-1}	4.9412×10^{-1}	5.7464×10^{-1}	4.9527×10^{-1}	2.3551×10^{-1}	2.7717×10^{-1}	2.2549×10^{-1}	2.0513×10^{-1}

4.2. Experiments on Industrial Engineering Design Problems

Most optimization problems have constraints in the real world, so considering equality and inequality constraints during optimization is a necessary process. In this subsection, four well-known constrained industrial engineering design problems, which include pressure vessel design problem, speed reducer design problem, tension/compression spring design problem, and three-bar truss design problem, were solved to further verify the performance of the proposed IHAOHHO algorithm. The results of IHAOHHO were compared to various classical optimizers proposed in previous studies. The parameter settings were as same as the previous experiments.

4.2.1. Pressure Vessel Design Problem

The objective of this problem was to minimize the fabrication cost of the cylindrical pressure vessel to meet the pressure requirements. As shown in Figure 7, four structural parameters in this problem needed to be minimized, including the thickness of the shell (T_s), the thickness of the head (T_h), inner radius (R), and the length of the cylindrical section without the head (L). The formulation of four optimization constraints can be described as follows:

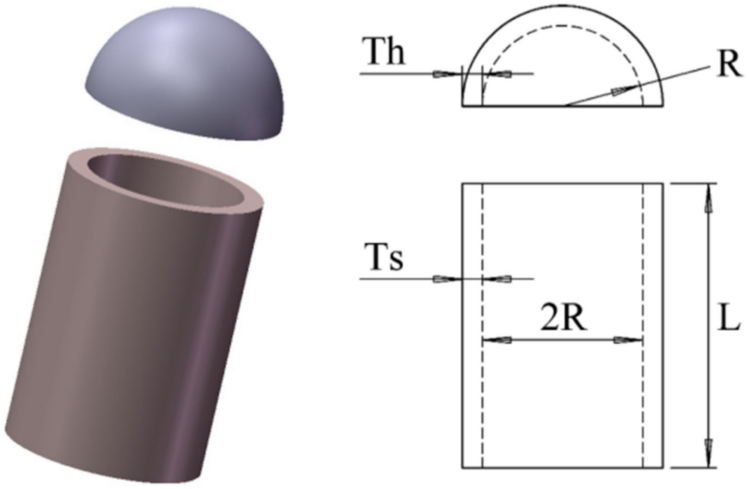

Figure 7. Pressure vessel design problem.

Consider

$$\vec{x} = [x_1\ x_2\ x_3\ x_4] = [T_s\ T_h\ \mathrm{R}\ L], \tag{27}$$

Minimize

$$f(\vec{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3, \tag{28}$$

Subject to

$$\begin{aligned} g_1(\vec{x}) &= -x_1 + 0.0193x_3 \le 0, \\ g_2(\vec{x}) &= -x_3 + 0.00954x_3 \le 0, \\ g_3(\vec{x}) &= -\pi x_3^2x_4 - \tfrac{4}{3}\pi x_3^3 + 1296000 \le 0, \\ g_4(\vec{x}) &= x_4 - 240 \le 0, \end{aligned} \tag{29}$$

Variable range

$$\begin{aligned} 0 \le x_1 \le 99, \\ 0 \le x_2 \le 99, \\ 10 \le x_3 \le 200, \\ 10 \le x_4 \le 200, \end{aligned} \tag{30}$$

From the results in Table 8, it is obvious that IHAOHHO can obtain superior optimal values compared to AO, HHO, SMA, WOA, GWO, MVO, GA, ES, and CPSO [65].

Table 8. Comparison of IHAOHHO results with other competitors for the pressure vessel design problem.

Algorithm	Optimum Variables				Optimum Cost
	T_s	T_h	R	L	
IHAOHHO	0.8363559	0.4127868	45.08462	142.9202	5932.3392
AO [55]	1.0540	0.182806	59.6219	38.8050	5949.2258
HHO [42]	0.81758383	0.4072927	42.09174576	176.7196352	6000.46259
SMA [41]	0.7931	0.3932	40.6711	196.2178	5994.1857
WOA [38]	0.8125	0.4375	42.0982699	176.638998	6059.7410
GWO [32]	0.8125	0.4345	42.0892	176.7587	6051.5639
MVO [23]	0.8125	0.4375	42.090738	176.73869	6060.8066
GA [3]	0.8125	0.4375	42.097398	176.65405	6059.94634
ES [6]	0.8125	0.4375	42.098087	176.640518	6059.74560
CPSO [65]	0.8125	0.4375	42.091266	176.7465	6061.0777

4.2.2. Speed Reducer Design Problem

This problem aims to optimize seven variables to minimize the speed reducer's total weights, which include the face width (x_1), module of teeth (x_2), a discrete design variable on behalf of the teeth in the pinion (x_3), length of the first shaft between bearings (x_4), length of the second shaft between bearings (x_5), diameters of the first shaft (x_6), and diameters of the second shaft (x_7). Four constraints—covering stress, bending stress of the gear teeth, stresses in the shafts, and transverse deflections of the shafts, as shown in Figure 8—should be satisfied. The mathematical formulation is represented as follows:

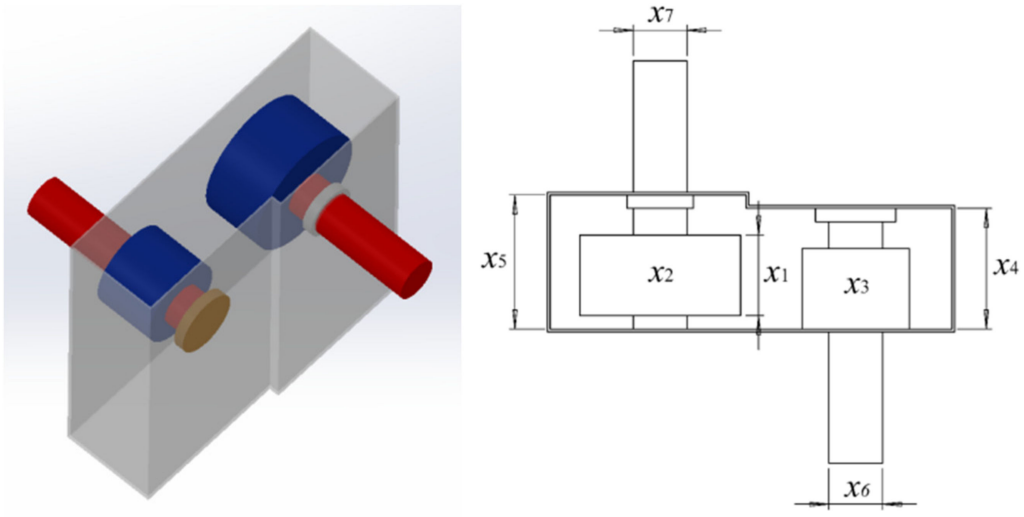

Figure 8. Speed reducer design problem.

Minimize

$$f(\vec{x}) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3), \quad (31)$$

Subject to

$$
\begin{aligned}
g_1(\vec{x}) &= \frac{27}{x_1 x_2^2 x_3} - 1 \leq 0, \\
g_2(\vec{x}) &= \frac{397.5}{x_1 x_2^2 x_3^2} - 1 \leq 0, \\
g_3(\vec{x}) &= \frac{1.93 x_4^3}{x_2 x_3 x_6^4} - 1 \leq 0, \\
g_4(\vec{x}) &= \frac{1.93 x_5^3}{x_2 x_3 x_7^4} - 1 \leq 0, \\
g_5(\vec{x}) &= \frac{\sqrt{\left(\frac{745 x_4}{x_2 x_3}\right)^2 + 16.9 \times 10^6}}{110.0 x_6^3} - 1 \leq 0, \\
g_6(\vec{x}) &= \frac{\sqrt{\left(\frac{745 x_4}{x_2 x_3}\right)^2 + 157.5 \times 10^6}}{85.0 x_6^3} - 1 \leq 0, \\
g_7(\vec{x}) &= \frac{x_2 x_3}{40} - 1 \leq 0, \\
g_8(\vec{x}) &= \frac{5 x_2}{x_1} - 1 \leq 0, \\
g_9(\vec{x}) &= \frac{x_1}{12 x_2} - 1 \leq 0, \\
g_{10}(\vec{x}) &= \frac{1.5 x_6 + 1.9}{x_4} - 1 \leq 0, \\
g_{11}(\vec{x}) &= \frac{1.1 x_7 + 1.9}{x_5} - 1 \leq 0,
\end{aligned}
\tag{32}
$$

Variable range

$$
\begin{aligned}
2.6 \leq x_1 \leq 3.6, \\
0.7 \leq x_2 \leq 0.8, \\
17 \leq x_3 \leq 28, \\
7.3 \leq x_4 \leq 8.3, \\
7.8 \leq x_5 \leq 8.3, \\
2.9 \leq x_6 \leq 3.9, \\
5.0 \leq x_7 \leq 5.5,
\end{aligned}
\tag{33}
$$

Compared to AO, PSO, AOA, MFO [66], GA, SCA, HS [67], FA [68], and MDA [69], IHAOHHO can obviously achieve better results in the speed reducer design problem, as shown in Table 9.

Table 9. Comparison of IHAOHHO results with other competitors for the speed reducer design problem.

Algorithm	Optimum Variables							Optimum Weight
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	
IHAOHHO	3.49924	0.7	17	7.3	7.8191	3.35006	5.28531	2996.0935
AO [55]	3.5021	0.7	17	7.3099	7.7476	3.3641	5.2994	3007.7328
PSO [26]	3.5001	0.7	17.0002	7.5177	7.7832	3.3508	5.2867	3145.922
AOA [25]	3.50384	0.7	17	7.3	7.72933	3.35649	5.2867	2997.9157
MFO [66]	3.49745	0.7	17	7.82775	7.71245	3.35178	5.28635	2998.9408
GA [3]	3.51025	0.7	17	8.35	7.8	3.36220	5.28772	3067.561
SCA [24]	3.50875	0.7	17	7.3	7.8	3.46102	5.28921	3030.563
HS [67]	3.52012	0.7	17	8.37	7.8	3.36697	5.28871	3029.002
FA [68]	3.50749	0.7001	17	7.71967	8.08085	3.35151	5.28705	3010.13749
MDA [69]	3.5	0.7	17	7.3	7.67039	3.54242	5.24581	3019.58336

4.2.3. Tension/Compression Spring Design Problem

In this case, the intention is to minimize the weight of the tension/compression spring shown in Figure 9. Constraints on surge frequency, shear stress, and deflection must be satisfied during optimum design. There are three parameters that needed to be minimized,

including the wire diameter (d), mean coil diameter (D), and the number of active coils (N). The mathematical form of this problem can be written as follows:

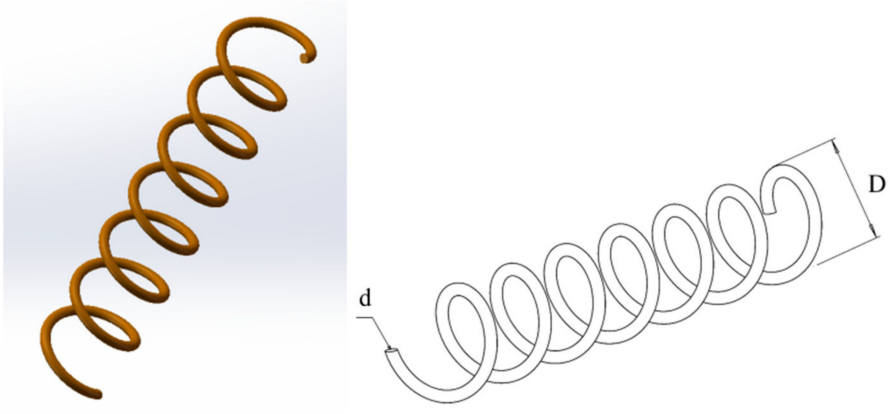

Figure 9. Tension/compression spring design problem.

Consider

$$\vec{x} = [x_1\ x_2\ x_3\ x_4] = [\mathrm{d\ D\ N}], \tag{34}$$

Minimize

$$f(\vec{x}) = (x_3 + 2)x_2x_1^2, \tag{35}$$

Subject to

$$\begin{aligned} g_1(\vec{x}) &= 1 - \frac{x_2^3 x_3}{71{,}785x_1^4} \leq 0, \\ g_2(\vec{x}) &= \frac{4x_2^2 - x_1x_2}{12{,}566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} \leq 0, \\ g_3(\vec{x}) &= 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0, \\ g_4(\vec{x}) &= \frac{x_1 + x_2}{1.5} - 1 \leq 0, \end{aligned} \tag{36}$$

Variable range

$$\begin{aligned} 0.05 \leq x_1 \leq 2.00, \\ 0.25 \leq x_2 \leq 1.30, \\ 2.00 \leq x_3 \leq 15.00, \end{aligned} \tag{37}$$

The proposed IHAOHHO is compared with AO, HHO, SSA, WOA, GWO, PSO, MVO, GA, and HS algorithms. Results are listed in Table 10 and show that the IHAOHHO can attain the best weight values compared to all other algorithms. Additionally, it is clear that the proposed method found a more accurate design with new parameter values.

Table 10. Comparison of IHAOHHO results with other competitors for the tension/compression spring design problem.

Algorithm	Optimum Variables			Optimum Weight
	d	D	N	
IHAOHHO	0.055883	0.52784	4.7603	0.011144
AO [55]	0.0502439	0.35262	10.5425	0.011165
HHO [42]	0.051796393	0.359305355	11.138859	0.012665443
SSA [39]	0.051207	0.345215	12.004032	0.0126763
WOA [38]	0.051207	0.345215	12.004032	0.0126763
GWO [32]	0.05169	0.356737	11.28885	0.012666
PSO [26]	0.051728	0.357644	11.244543	0.0126747
MVO [23]	0.05251	0.37602	10.33513	0.012790
GA [3]	0.051480	0.351661	11.632201	0.01270478
HS [67]	0.051154	0.349871	12.076432	0.0126706

4.2.4. Three-Bar Truss Design Problem

The three-bar truss design problem is a classical optimization application in civil engineering field. The main intention of this case is to minimize the weight of a truss with three bars by considering two structural parameters as illustrated in Figure 10. Deflection, stress, and buckling are the three main constrains. The mathematical formulation of this problem is given:

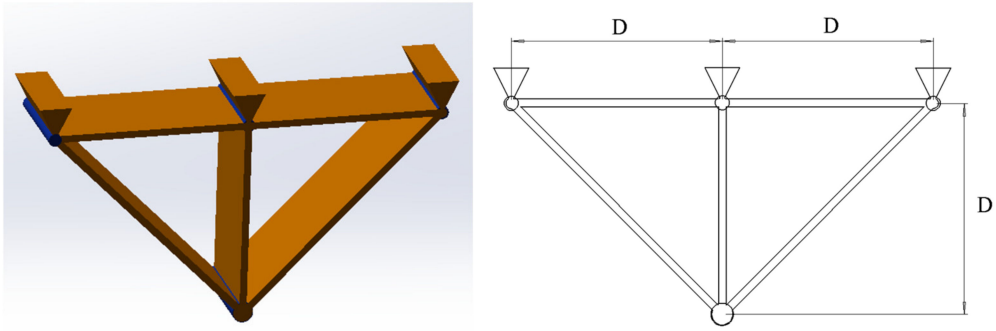

Figure 10. Three-bar truss design problem.

Consider

$$\vec{x} = [x_1\ x_2] = [A_1\ A_2], \tag{38}$$

Minimize

$$f(\vec{x}) = (2\sqrt{2}x_1 + x_2) * l, \tag{39}$$

Subject to

$$\begin{aligned} g_1(\vec{x}) &= \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \leq 0, \\ g_2(\vec{x}) &= \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \leq 0, \\ g_3(\vec{x}) &= \frac{1}{\sqrt{2}x_2 + x_1} P - \sigma \leq 0, \end{aligned} \tag{40}$$

Variable range

$$0 \leq x_1, x_2 \leq 1, \tag{41}$$

where $l = 100\text{cm}$, $P = 2\text{KN/cm}^2$, $\sigma = 2\text{KN/cm}^2$. Results of IHAOHHO for solving three-bar truss design problem are listed in Table 11, which are compared with AO, HHO, SSA, AOA, MVO, MFO, and GOA [70]. It can be observed that IHAOHHO outperforms other optimization algorithms published in the literature.

Table 11. Comparison of IHAOHHO results with other competitors for the three-bar truss design problem.

Algorithm	Optimum Variables		Optimum Weight
	x_1	x_2	
IHAOHHO	0.79002	0.40324	263.8622
AO [55]	0.7926	0.3966	263.8684
HHO [42]	0.788662816	0.408283133832900	263.8958434
SSA [39]	0.78866541	0.408275784	263.89584
AOA [25]	0.79369	0.39426	263.9154
MVO [23]	0.78860276	0.408453070000000	263.8958499
MFO [66]	0.788244771	0.409466905784741	263.8959797
GOA [70]	0.788897555578973	0.407619570115153	263.895881496069

As a summary, this section demonstrates the superiority of the proposed IHAOHHO algorithms in different characteristics and real case studies. IHAOHHO is able to outperform the original AO and HHO and other well-known algorithms with very competitive

results, which were derived from the robust exploration and exploitation capabilities of IHAOHHO. Excellent performance in solving industrial engineering design problems indicates that IHAOHHO can be widely used in real-world optimization problems.

5. Conclusions

This study proposed an improved hybrid Aquila Optimizer and Harris Hawks Optimization by combining the exploration part of AO with the exploitation part of HHO and a nonlinear escaping energy parameter and random opposition-based learning (ROBL) strategy. The proposed method integrated the mentioned search methods to tackle the weakness of the traditional search methods. The proposed IHAOHHO algorithm was tested using 23 mathematical benchmark functions to analyze its exploration, exploitation, local optima avoidance capabilities, and convergence behaviors. Results show competitive results compared to other state-of-the-art meta-heuristic algorithms. To further verify the superiority of IHAOHHO, four industrial engineering design problems were solved. The results are also competitive with other meta-heuristic algorithms.

As future perspectives, binary and multi-objective versions of IHAOHHO will be considered. More applications of this algorithm in different fields are valuable works, including text clustering, scheduling problems, appliances management, parameters estimation, multi-objective engineering problems, feature selection, test classification, image segmentation problems, network applications, sentiment analysis, etc.

Author Contributions: Conceptualization, H.J. and L.A.; methodology, S.W.; software, R.Z.; validation, S.W., Q.L. and R.Z.; formal analysis, S.W.; writing—original draft preparation, S.W.; writing—review and editing, S.W.; visualization, Q.L.; supervision, L.A.; project administration, H.J.; funding acquisition, S.W. and H.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Sanming University Introduces High-level Talents to Start Scientific Research Funding Support Project, grant number 20YG01, 20YG14; the Guiding Science and Technology Projects in Sanming City, grant number 2020-S-39, 2020-G-61, 2021-S-8; the Educational Research Projects of Young and Middle-aged Teachers in Fujian Province, grant number JAT200638, JAT200618; the Scientific Research and Development Fund of Sanming University, grant number B202029, B202009; Collaborative education project of industry university cooperation of the Ministry of Education, grant number 202002064014; School level education and teaching reform project of Sanming University, grant number J2010306, J2010305; and Higher education research project of Sanming University, grant number SHE2102, SHE2013.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Abualigah, L.; Diabat, A. Advances in sine cosine algorithm: A comprehensive survey. *Artif. Intell. Rev.* **2021**, *54*, 2567–2608. [CrossRef]
2. Abualigah, L.; Diabat, A. A comprehensive survey of the Grasshopper optimization algorithm: Results, variants, and applications. *Neural Comput. Appl.* **2020**, *32*, 15533–15556. [CrossRef]
3. Holland, J.H. Genetic algorithms. *Sci. Am.* **1992**, *267*, 66–72. [CrossRef]
4. Storn, R.; Price, K. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
5. Koza, J.R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*; MIT Press: Cambridge, MA, USA, 1992.
6. Rechenberg, I. Evolutionsstrategien. In *Simulationsmethoden in der Medizin und Biologie*; Springer: Berlin/Heidelberg, Germany, 1978; Volume 8, pp. 83–114.
7. Simon, D. Biogeography-based optimization. *IEEE Trans. Evol. Comput.* **2008**, *12*, 702–713. [CrossRef]
8. Yao, X.; Liu, Y.; Lin, G. Evolutionary programming made faster. *IEEE Trans. Evol. Comput.* **1999**, *3*, 82–102. [CrossRef]
9. Dasgupta, D.; Michalewicz, Z. *Evolutionary Algorithms in Engineering Applications*; DBLP: Trier, Germany, 1997.

10. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by simmulated annealing. *Science* **1983**, *220*, 671–680. [CrossRef] [PubMed]
11. Erol, O.K.; Eksin, I. A new optimization method: Big bang-big crunch. *Adv. Eng. Softw.* **2006**, *37*, 106–111. [CrossRef]
12. Rashedi, E.; Nezamabadi-Pour, H.; Saryazdi, S. GSA: A Gravitational Search Algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248. [CrossRef]
13. Webster, B.; Bernhard, P.J. A local search optimization algorithm based on natural principles of gravitation. In *Information & Knowledge Engineering, Proceedings of the 2003 International Conference on Information and Knowledge Engineering (IKE'03), Las Vegas, NV, USA, 23–26 June 2003*; DBLP: Trier, Germany, 2003.
14. Asef, F.; Majidnezhad, V.; Feizi-Derakhshi, M.R.; Parsa, S. Heat transfer relation-based optimization algorithm (HTOA). *Soft Comput.* **2021**, 1–30. [CrossRef]
15. Kaveh, A.; Talatahari, S. A novel heuristic optimization method: Charged system search. *Acta Mech.* **2010**, *213*, 267–289. [CrossRef]
16. Alatas, B. ACROA: Artificial Chemical Reaction Optimization Algorithm for global optimization. *Expert Syst. Appl.* **2011**, *38*, 13170–13180. [CrossRef]
17. Formato, R.A. Central force optimization: A new metaheuristic with applications in applied electromagnetics. *Prog. Electromag. Res.* **2007**, *77*, 425–491. [CrossRef]
18. Kaveh, A.; Khayatazad, M. A new meta-heuristic method: Ray optimization. *Comput. Struct.* **2012**, *112*, 283–294. [CrossRef]
19. Hatamlou, A. Black hole: A new heuristic optimization approach for data clustering. *Inf. Sci.* **2013**, *222*, 175–184. [CrossRef]
20. Du, H.; Wu, X.; Zhuang, J. Small-world optimization algorithm for function optimization. In *Advances in Natural Computation, Advances in Natural Computation, Second International Conference*; ICNC: Xi'an, China, 2006.
21. Shah-Hosseini, H. Principal components analysis by the galaxy-based search algorithm: A novel metaheuristic for continuous optimisation. *Int. J. Comput. Sci. Eng.* **2011**, *6*, 132–140. [CrossRef]
22. Moghaddam, F.F.; Moghaddam, R.F.; Cheriet, M. Curved space optimization: A random search based on general relativity theory. *arXiv* **2012**, arXiv:1208.2214.
23. Mirjalili, S.; Mirjalili, S.M.; Hatamlou, A. Multi-Verse Optimizer: A nature-inspired algorithm for global optimization. *Neural Comput. Appl.* **2015**, *27*, 495–513. [CrossRef]
24. Mirjalili, S. SCA: A Sine Cosine Algorithm for Solving Optimization Problems. *Knowl.-Based Syst.* **2016**, *96*. [CrossRef]
25. Abualigah, L.; Diabat, A.; Mirjalili, S.; Elaziz, M.A.; Gandomi, A.H. The Arithmetic Optimization Algorithm. *Comput. Methods Appl. Mech. Eng.* **2021**, *376*, 113609. [CrossRef]
26. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the 1995 IEEE International Conference on Neural Networks (ICNN '93), Perth, WA, Australia, 27 November–1 December 1995; IEEE: Piscataway, NJ, USA, 1995.
27. Dorigo, M.; Birattari, M.; Stutzle, T. Ant colony optimization. *IEEE Comput. Intell.* **2006**, *1*, 28–39. [CrossRef]
28. Mucherino, A.; Seref, O.; Seref, O.; Kundakcioglu, O.E.; Pardalos, P. Monkey search: A novel metaheuristic search for global optimization. *Am. Inst. Phys.* **2007**, *953*, 162–173. [CrossRef]
29. Yang, X.S. Firefly algorithm, stochastic test functions and design optimization. *Int. J. Bio-Inspired Comput.* **2010**, *2*, 78–84. [CrossRef]
30. Yang, X.S. A new metaheuristic bat-inspired algorithm. In *Nature Inspired Cooperative Strategies for Optimization (NICSO)*; Springer: Berlin/Heidelberg, Germany, 2010.
31. Gandomi, A.H.; Alavi, A.H. Krill Herd: A new bio-inspired optimization algorithm. *Commun. Nonlinear Sci. Numer. Simul.* **2012**, *17*, 4831–4845. [CrossRef]
32. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
33. Gandomi, A.H.; Yang, X.S.; Alavi, A.H. Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. *Eng. Comput.* **2013**, *29*, 17–35. [CrossRef]
34. Pan, W.T. A new fruit fly optimization algorithm: Taking the financial distress model as an example. *Knowl.-Based Syst.* **2012**, *26*, 69–74. [CrossRef]
35. Yang, S.; Jiang, J.; Yan, G. A dolphin partner optimization. In Proceedings of the 2009 WRI Global Congress on Intelligent Systems (GCIS 2009), Xiamen, China, 19–21 May 2009; IEEE: Piscataway, NJ, USA, 2009.
36. Mirjalili, S. The Ant Lion optimizer. *Adv. Eng. Softw.* **2015**, *83*, 80–98. [CrossRef]
37. Jia, H.; Peng, X.; Lang, C. Remora optimization algorithm. *Expert Syst. Appl.* **2021**, *185*, 115665. [CrossRef]
38. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
39. 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]
40. Alsattar, H.A.; Zaidan, A.A.; Zaidan, B.B. Novel meta-heuristic bald eagle search optimisation algorithm. *Artif. Intell. Rev.* **2020**, *53*, 2237–2264. [CrossRef]
41. Li, S.M.; Chen, H.L.; Wang, M.J.; Heidari, A.A.; Mirjalili, S. Slime mould algorithm: A new method for stochastic optimization. *Future Gener. Comput. Syst.* **2020**, *111*, 300–323. [CrossRef]
42. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H.L. Harris Hawks optimization: Algorithm and applications. *Future Gener. Comput. Syst.* **2019**, *97*, 849–872. [CrossRef]
43. Yousri, D.; Fathy, A.; Thanikanti, S.B. Recent methodology based Harris Hawks optimizer for designing load frequency control incorporated in multi-interconnected renewable energy plants. *Sustain. Energy Grids Netw.* **2020**, *22*, 100352. [CrossRef]
44. Bui, D.T.; Moayedi, H.; Kalantar, B.; Osouli, A.; Rashid, A. A Novel Swarm Intelligence Technique Harris Hawks Optimization for Spatial Assessment of Landslide Susceptibility. *Sensors* **2019**, *19*, 3590. [CrossRef]

45. Golilarz, N.A.; Gao, H.; Demirel, H. Satellite image de-noising with Harris Hawks meta heuristic optimization algorithm and improved adaptive generalized gaussian distribution threshold function. *IEEE Access* **2019**, *7*, 57459–57468. [CrossRef]
46. Jia, H.; Peng, X.; Kang, L.; Li, Y.; Sun, K. Pulse coupled neural network based on Harris Hawks optimization algorithm for image segmentation. *Multimed Tools Appl.* **2020**, *79*, 28369–28392. [CrossRef]
47. Jia, H.; Lang, C.; Oliva, D.; Song, W.; Peng, X. Dynamic Harris Hawks Optimization with Mutation Mechanism for Satellite Image Segmentation. *Remote Sens.* **2019**, *11*, 1421. [CrossRef]
48. Yousri, D.; Mirjalili, S.; Machado, J.A.T.; Thanikantie, S.B.; Elbaksawi, O.; Fathy, A. Efficient fractional-order modified Harris Hawks optimizer for proton exchange membrane fuel cell modeling. *Eng. Appl. Artif. Intell.* **2021**, *100*, 104193. [CrossRef]
49. Gupta, S.; Deep, K.; Heidari, A.A.; Moayedi, H.; Wang, M. Opposition-based Learning Harris Hawks Optimization with Advanced Transition Rules: Principles and Analysis. *Expert Syst. Appl.* **2020**, *158*, 113510. [CrossRef]
50. Hussien, A.G.; Amin, M. A self-adaptive Harris Hawks optimization algorithm with opposition-based learning and chaotic local search strategy for global optimization and feature selection. *Int. J. Mach. Learn. Cyber.* **2021**, 1–28. [CrossRef]
51. Sihwail, R.; Omar, K.; Ariffin, K.; Tubishat, M. Improved Harris Hawks Optimization Using Elite Opposition-Based Learning and Novel Search Mechanism for Feature Selection. *IEEE Access* **2020**, *8*, 121127–121145. [CrossRef]
52. Bao, X.; Jia, H.; Lang, C. A Novel Hybrid Harris Hawks Optimization for Color Image Multilevel Thresholding Segmentation. *IEEE Access* **2019**, *7*, 76529–76546. [CrossRef]
53. Houssein, E.H.; Hosney, M.E.; Elhoseny, M.; Oliva, D.; Hassaballah, M. Hybrid Harris Hawks Optimization with Cuckoo Search for Drug Design and Discovery in Chemoinformatics. *Sci. Rep.* **2020**, *10*, 14439. [CrossRef] [PubMed]
54. Kaveh, A.; Rahmani, P.; Eslamlou, A.D. An efficient hybrid approach based on Harris Hawks optimization and imperialist competitive algorithm for structural optimization. *Eng. Comput.* **2021**, 4598. [CrossRef]
55. Abualigah, L.; Yousri, D.; Elaziz, M.A.; 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]
56. Tang, A.D.; Han, T.; Xu, D.W.; Xie, L. Chaotic Elite Harris Hawk Optimization Algorithm. *J. Comput. Appl.* **2021**, 1–10. Available online: https://kns.cnki.net/kcms/detail/detail.aspx?dbcode=CAPJ&dbname=CAPJLAST&filename=JSJY2021011300H&v=5lc3RO%25mmd2BEUUC%25mmd2FhVq8jnE%25mmd2BxfkAnjCOOEL7xcSF5jPQfItuqOALm2aHD2u1aGLhSpw1 (accessed on 15 January 2021).
57. Tizhoosh, H. Opposition-based learning: A new scheme for machine intelligence. In *Control and Automation, Proceedings of the International Conference on Computational Intelligence for Modeling, Vienna, Austria, 28–30 November 2005*; IEEE: Piscataway, NJ, USA, 2005.
58. Rahnamayan, S.; Tizhoosh, H.R.; Salama, M.M.A. Opposition-based differential evolution. *IEEE Trans. Evol. Comput.* **2014**, *12*, 64–79. [CrossRef]
59. Jia, Z.; Li, L.; Hui, S. Artificial Bee Colony Using Opposition-Based Learning. *Adv. Intell. Syst. Comput.* **2015**, *329*, 3–10.
60. Elaziz, M.A.; Oliva, D.; Xiong, S. An improved Opposition-Based Sine Cosine Algorithm for global optimization. *Expert Syst. Appl.* **2017**, *90*, 484–500. [CrossRef]
61. Ewees, A.A.; Elaziz, M.A.; Houssein, E.H. Improved Grasshopper Optimization Algorithm using Opposition-based Learning. *Expert Syst. Appl.* **2018**, *112*, 156–172. [CrossRef]
62. Fan, C.; Zheng, N.; Zheng, J.; Xiao, L.; Liu, Y. Kinetic-molecular theory optimization algorithm using opposition-based learning and varying accelerated motion. *Soft Comput.* **2020**, *24*, 12709–12730. [CrossRef]
63. Long, W.; Jiao, J.; Liang, X.; Cai, S.; Xu, M. A Random Opposition-Based Learning Grey Wolf Optimizer. *IEEE Access* **2019**, *7*, 113810–113825. [CrossRef]
64. Molga, M.; Smutnicki, C. Test Functions for Optimization Needs. 2005. Available online: http://www.robertmarks.org/Classes/ENGR5358/Papers/functions.pdf (accessed on 1 January 2005).
65. He, Q.; Wang, L. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Eng. Appl. Artif. Intell.* **2007**, *20*, 89–99. [CrossRef]
66. Mirjalili, S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowl.-Based Syst.* **2015**, *89*, 228–249. [CrossRef]
67. Geem, Z.W.; Kim, J.H.; Loganathan, G.V. A new heuristic optimization algorithm: Harmony search. *Simulation* **2001**, *76*, 60–68. [CrossRef]
68. Baykasoğlu, A.; Ozsoydan, F.B. Adaptive firefly algorithm with chaos for mechanical design optimization problems. *Appl. Soft Comput.* **2015**, *36*, 152–164. [CrossRef]
69. Lu, S.; Kim, H.M. A regularized inexact penalty decomposition algorithm for multidisciplinary design optimization problems with complementarity constraints. *J. Mech. Des.* **2010**, *132*, 041005. [CrossRef]
70. Saremi, S.; Mirjalili, S.; Lewis, A. Grasshopper optimisation algorithm: Theory and application. *Adv. Eng. Softw.* **2017**, *105*, 30–47. [CrossRef]