processes

MDPI

Article

Improved Honey Badger Algorithm Based on Elite Tangent Search and Differential Mutation with Applications in Fault Diagnosis

He Ting [1], Chang Yong [2,*] and Chen Peng [2]

[1] Gansu Natural Energy Research Institute, Lanzhou 730046, China; heting@gneri.com.cn
[2] School of Electronics and Information Engineering, Lanzhou Petrochemical Vocational and Technical University, Lanzhou 730060, China; chenpeng@lzpuvt.edu.cn
* Correspondence: changyong@lzpuvt.edu.cn

Abstract: This paper presents a critique of the Honey Badger Algorithm (HBA) with regard to its limited exploitation capabilities, susceptibility to local optima, and inadequate pre-exploration mechanisms. In order to address these issues, we propose the Improved Honey Badger Algorithm (IHBA), which integrates the Elite Tangent Search Algorithm (ETSA) and differential mutation strategies. Our approach employs cubic chaotic mapping in the initialization phase and a random value perturbation strategy in the pre-iterative stage to enhance exploration and prevent premature convergence. In the event that the optimal population value remains unaltered across three iterations, the elite tangent search with differential variation is employed to accelerate convergence and enhance precision. Comparative experiments on partial CEC2017 test functions demonstrate that the IHBA achieves faster convergence, greater accuracy, and improved robustness. Moreover, the IHBA is applied to the fault diagnosis of rolling bearings in electric motors to construct the IHBA-VMD-CNN-BiLSTM fault diagnosis model, which quickly and accurately identifies fault types. Experimental verification confirms that this method enhances the speed and accuracy of rolling bearing fault identification compared to traditional approaches.

Keywords: honey badger algorithm (HBA); elite tangent search algorithm (ETSA); differential mutation strategy; cubic chaotic mapping; convergence speed and solution accuracy; bearing fault diagnosis

check for updates

Academic Editor: Olympia Roeva

Received: 3 December 2024
Revised: 13 January 2025
Accepted: 14 January 2025
Published: 17 January 2025

Citation: Ting, H.; Yong, C.; Peng, C. Improved Honey Badger Algorithm Based on Elite Tangent Search and Differential Mutation with Applications in Fault Diagnosis. *Processes* **2025**, *13*, 256. https://doi.org/10.3390/pr13010256

Copyright: © 2025 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/).

1. Introduction

Electric motors are essential components in various industries, converting electrical energy into mechanical energy. Rolling bearings are critical for the smooth operation of many motor components, and their failure is one of the leading causes of motor malfunction. Bearing failures account for over 40% of motor faults [1,2], and if not detected early, can lead to severe consequences, such as prolonged downtime, financial losses, and even risks to human health and safety. Therefore, it is crucial to develop effective fault diagnosis techniques to ensure the reliable operation of electric motors, especially in environments with fluctuating loads and harsh conditions.

Recent advancements in swarm intelligence algorithms have provided promising solutions for fault diagnosis. These algorithms, which mimic the collective behavior of natural organisms, have been successfully applied to various optimization problems, including motor fault diagnosis. Popular algorithms include the Sparrow Search Algorithm (SSA) [3], Whale Optimization Algorithm (WOA) [4], Pelican Optimization Algorithm

(POA) [5], Gray Wolf Optimization (GWO) [6], Harris Hawk Algorithm (HHO) [7], etc. These algorithms have demonstrated the capacity for effective global search and adaptability. However, despite their advantages, these algorithms face significant challenges, including premature convergence, limited exploitation capabilities, and susceptibility to local optima, which hinder their performance in complex fault diagnosis tasks.

The Honey Badger Algorithm (HBA), introduced by Fatma et al. in 2021 [8], is inspired by the foraging behavior of honey badgers. While it offers several advantages, such as a simple structure and few adjustable parameters, HBA struggles with issues like inadequate exploration and a slow convergence rate. In response to these shortcomings, numerous modifications have been proposed, such as incorporating reverse learning strategies [9], chaotic mapping [10], and hybrid approaches [11]. However, these improvements often involve increased complexity and do not fully address the underlying limitations in the algorithm's exploration and exploitation balance.

In this paper, we introduce the Improved Honey Badger Algorithm (IHBA), which integrates the Elite Tangent Search Algorithm (ETSA) and differential mutation strategies to overcome the limitations of traditional HBA. Our proposed method improves exploration by employing cubic chaotic mapping during the initialization phase and utilizes a random value perturbation strategy in the pre-iterative stage to prevent premature convergence. Moreover, when the optimal population value remains unchanged for three consecutive iterations, an elite tangent search with differential variation is triggered to enhance convergence speed and precision. We validate the performance of the IHBA through comparative experiments on several CEC2017 test functions, demonstrating that the IHBA significantly outperforms the traditional HBA in terms of convergence speed, accuracy, and robustness. To further assess the advantages of the IHBA, we conduct ablation experiments comparing it with the original HBA, Dung Beetle Optimizer (DBO) [12], Coati Optimization Algorithm (COA) [13], Osprey Optimization Algorithm (OOA) [14], Beluga Whale Optimization (BWO) [15], Osprey–Cauchy–Sparrow Search Algorithm (OCSSA) [16], and Harris Hawk Algorithm (HHO) [7]. The experimental results clearly show that the IHBA achieves a notable improvement in both the speed of convergence and the accuracy of the optimization process, with enhanced robustness. Furthermore, the IHBA is successfully applied to the fault diagnosis of rolling bearings in electric motors, where we develop an IHBA-VMD-CNN-BiLSTM fault diagnosis model. The experimental results validate that this method significantly improves the speed and accuracy of fault detection, providing a more reliable solution for motor condition monitoring than traditional approaches.

The contributions of this paper are twofold:

(1) The development of the IHBA addresses the key shortcomings of the original HBA by enhancing exploration and accelerating convergence.
(2) The successful application of the IHBA in motor bearing fault diagnosis, providing a novel approach to improving fault detection accuracy in industrial settings.

2. Honey Badger Algorithm (HBA)

The Honey Badger Algorithm (HBA) is a metaheuristic algorithm designed to tackle continuous optimization problems. It mimics the intelligent foraging behavior exhibited by honey badgers in nature and boasts advantages such as high efficiency, a straightforward structure, and ease of implementation [8]. The HBA is structured into two distinct phases as follows, the "Digging Phase" and the "Honey Phase", which are elaborated upon as follows:

Step 1: Initialization phase. We initialize the number of honey badgers (population size N_s) and their respective positions based on Equation (1) as follows:

$$x_i = lb_i + r_1 \times (ub_i + lb_i), \quad (1)$$

where x_i is the i^{th} honey badger position referring to a candidate solution in a population of N_s, while lb_i and ub_i are, respectively, the lower and upper bounds of the search domain, $i \in [1, 2, \ldots, N_s]$, and r_1 is a random number between 0 and 1.

Step 2: Defining intensity (I_i). Intensity is related to concentration strength of the prey and distance between it and the i^{th} honey badger. I_i is the smell intensity of the prey; if the smell intensity is high, the motion will be fast and vice versa, as defined by Equation (2).

$$I_i = r_2 \times \frac{S}{4\pi d_i^2}, \tag{2}$$

$$S = (x_i - x_{i+1})^2, \tag{3}$$

$$d_i = x_{\text{best}} - x_i, \tag{4}$$

where r_2 is a random number between 0 and 1, S is the source strength or concentration strength, d_i denotes the distance between prey and the i^{th} badger, and x_{best} is the current global optimal food source location, i.e., the optimal solution.

Step 3: Update density factor. The density factor (α) controls time-varying randomization to ensure the smooth transition from exploration to exploitation. We update the decreasing factor α that decreases with iterations to decrease randomization with time, using Equation (5) as follows:

$$\alpha = C \times \exp(\frac{-t}{t_{\max}}), \tag{5}$$

where t is the number of current iterations and $t_{\max}$ is the maximum number of iterations, $C \geq 1$ (default = 2).

Step 4: Digging phase. In the digging phase, a honey badger performs actions in a Cardioid shape. The Cardioid motion can be simulated by Equation (4) as follows:

$$x_{\text{new}} = x_{\text{best}} + F \times \beta \times I \times x_{\text{best}} + F \times r_3 \times \alpha \times d_i \times |\cos(2\pi r_4) \times [1 - \cos(2\pi r_5)]|, \tag{6}$$

where x_{new} is the updated position of the honey badger individual $\beta \geq 1$; r_3, r_4, and r_5 are three different random numbers between 0 and 1; and F works as the flag that alters search direction, which is determined using Equation (7) as follows:

$$F = \begin{cases} 1, & r_6 < 0.5 \\ -1, & \text{else} \end{cases}, \tag{7}$$

where r_6 is a random number between 0 and 1. In the digging phase, a honey badger heavily relies on the smell intensity, I_i, of the prey, x_{best}; the distance between the badger and the prey, d_i; and the time-varying search influence factor, α. Moreover, during the digging phase, a badger may receive disturbances, F, allowing it to find an even better prey location.

Step 5: Honey phase. The case of a honey badger following a honey guide bird to reach beehive can be simulated using Equation (8) as follows:

$$x_{\text{new}} = x_{\text{best}} + F \times r_7 \times \alpha \times d_i, \tag{8}$$

where r_7 is a random number between 0 and 1.

3. Improved Honey Badger Algorithm (IHBA)

Like other heuristic algorithms, HBA is premature and susceptible to local optimization in some complex problems. This phenomenon constrains the algorithm's precision and offers significant potential for enhancement. The improved algorithm introduces

several new mechanisms, including a cubic chaotic mapping mechanism, a random value perturbation strategy, an elite tangent search, and a differential mutation strategy.

3.1. Cubic Chaotic Mapping

As the initial population of the basic honey badger search algorithm is randomly generated, it is impossible to guarantee that the initial positions of the individuals are uniformly distributed in the search space. This has an impact on the search speed and optimization performance of the algorithm. The IHBA initialization process incorporates cubic mapping to enhance the traversal of the initial population.

$$\begin{cases} z_{i+1} = \rho z_i (1 - z_i^2) \\ x_{i+1} = lb_{i+1} + z_{i+1} \times (ub_{i+1} - lb_{i+1}) \end{cases}, \tag{9}$$

where z_i is cubic chaotic sequence, $\rho = 3$.

In the initial population generation process utilizing chaotic mapping, solutions with inferior or intermediate positions may be produced. These solutions may prove unfavorable for the subsequent search for the optimal solution or may impede the convergence of the algorithm. Accordingly, a population filtering mechanism is employed in the HBA of elite tangent search with differential variation, whereby solutions with superior initial positions are excluded. As illustrated in Figure 1, the number of solutions is initially generated at twice the expected value during the initialization phase. Subsequently, the optimal half of the solutions is selected as the initial population based on the calculated fitness values.

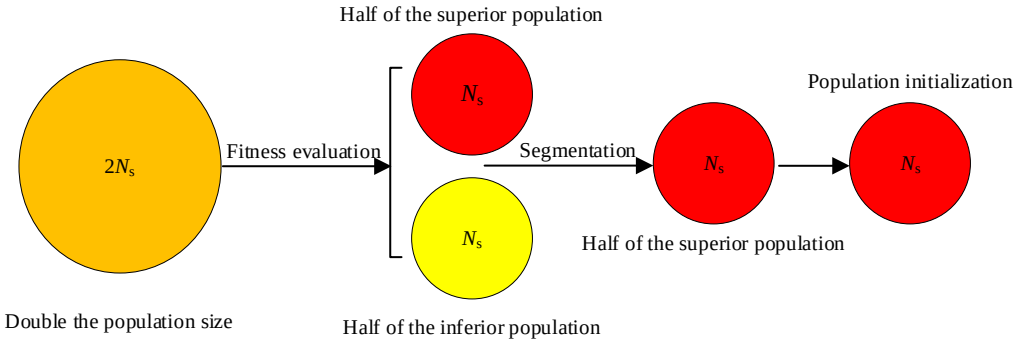


Figure 1. Schematic diagram of the population filtering process.

The combination of cubic chaotic mapping with elite tangent search and differential variation enables the HBA to generate populations that are more closely aligned with the optimal solution location during the initialization phase. The pseudocode of the population filtering mechanism using cubic chaotic mapping is illustrated in Algorithm 1.

Algorithm 1: Population filtering mechanism

Initialization of honey badger population size N_s;
1. Use Equation (9) to generate a population x_o with $2N_s$ solutions with random positions;
2. Calculate the fitness function value for each solutions;
3. Sort all individuals according to their fitness function values in descending order;
4. **For** $i = 1$ to N_s do
5. $x_i \leftarrow x_o$;
6. **end**
7. **Return** x_i.

3.2. Random Value Perturbation Strategy

In the IHBA, a key coefficient, A, is introduced to determine the strategy of honey badger position updating, aimed at balancing the algorithm's global search capability and local convergence speed. The design of this coefficient A incorporates the ideas of linear decreasing and stochastic perturbation in order to dynamically adjust the search strategy so as to avoid premature convergence and enhance the algorithm's global optimization seeking ability. The expression for A is as follows:

$$A = 2mr - m \tag{10}$$

where $m = 2 - 2t/t_{\max}$, whose value decreases linearly from 2 to 0, and r is a random number within (0, 1).

At the beginning of the iteration (t is small), the value of m is close to 2, which makes the absolute value of A greater than or equal to 1. Especially when r is close to 1, the algorithm tends to perform a search strategy with random individual perturbations, which increases the traversal of the search space and helps discover the globally optimal solution. As the number of iterations t increases, m gradually decreases, resulting in the absolute value of A tending to 0. In the later stages of the iteration, the algorithm relies more on the population optimum x_{best} for position updating, which facilitates the local convergence of the algorithm and accelerates the convergence to the optimal solution.

When $|A| \geq 1$, the algorithm performs a search strategy with randomized individual perturbations. This means that the honey badger's position update does not only depend on the current best individual, x_{best}, but is also influenced by a random individual, which helps the algorithm go beyond the local optimum and increase the diversity of the search.

***Digging phase*:**

$$x_{\text{new}} = x_{\text{rand}} + F \times \beta \times I_i \times x_{\text{rand}} + F \times r_3 \times \alpha \times (x_{\text{rand}} - x_i) \times |\cos(2\pi r_4) \times [1 - \cos(2\pi r_5)]|, \tag{11}$$

***Honey phase*:**

$$x_{\text{new}} = x_{\text{rand}} + F \times r_7 \times \alpha \times (x_{\text{rand}} - x_i), \tag{12}$$

When $|A| < 1$, the algorithm returns to the x_{best}-based position update strategy, which helps the algorithm to perform a fine-grained search in the vicinity of the discovered high-quality solutions, thus speeding up the convergence.

3.3. Elite Tangent Search and Differential Mutation Strategy

The following elite tangent search with a differential variation strategy is performed when the population is iterating and the best value found three times remains unchanged.

Elite subpopulation: In each iteration, the top half of individuals with lower fitness (i.e., better performance, since usually lower fitness values indicate better solutions) are classified into the elite subpopulation based on their fitness value. The elite subpopulation contains the best individuals in the current population, retaining the best solutions. This aims to protect these excellent solutions from being corrupted by subsequent genetic manipulations while allowing them to serve as the basis for future iterations.

Migration strategy for elite subpopulation: When an individual is close to the current optimal solution, it searches the area around it. This behavior, called local search, can improve the convergence speed and solution accuracy. Because the fitness of the elite subpopulation is close to the current optimal solution, the elite subpopulation is allowed to develop locally to improve convergence speed and solution accuracy.

The migration strategy of the elite subpopulation is updated by incorporating the Tangent Search Algorithm (TSA) of Reference [17] used for position updating.

$$\begin{cases} x_{\text{new}} = x_{\text{best}} + step\tan(\theta)(\text{rand} \cdot x_{\text{best}} - optS) & \text{if } x_{\text{best}} = optS \ \ (\text{a}) \\ x_{\text{new}} = x_{\text{best}} + step\tan(\theta)(x_{\text{best}} - optS) & \text{if } x_{\text{best}} \neq optS \ \ (\text{b}) \end{cases}, \tag{13}$$

$$step = 10\text{sign}(\text{rand} - 0.5)\text{norm}(optS)\log(1 + 10\text{dim}/t), \tag{14}$$

where $optS$ is the best current solution used to guide the search process towards the best solution, $\theta = \text{rand} * \pi/2.1$.

Exploration Subpopulation: The second half of the individuals are classified into the exploration subpopulation as opposed to the elite subpopulation. These individuals are usually more highly adapted, i.e., poorer performers. The main purpose of the exploring subpopulation is to explore new regions in the solution space through genetic manipulation to find potentially superior solutions. Since these individuals are not the current best, they are more likely to generate new, possibly superior solutions through operations such as mutation.

Exploration of the evolutionary strategy of the subpopulation: From Equation (13), it can be seen that in the HBA, the position of the honey badger individuals in the population is updated by generating new individuals in the vicinity of the current individual and the current optimal individual x_{best}, which means that the other individuals in the population move towards x_{best}. If x_{best} is a local optimal solution, as the iteration continues, the honey badger individuals in the population gather around x_{best}, which leads to poor population diversity and makes the algorithm converge prematurely; therefore, to solve these problems, this paper adopts a differential mutation strategy. Inspired by the variation strategy in the differential evolutionary algorithm, a stochastic difference is performed using the current honey badger individual, the current optimal individual, and a randomly selected honey badger individual in the population in order to generate a new individual, which is implemented using Equation (15).

$$x_{\text{new}} = x_{\text{rand1}} + F_0 \times (x_{\text{rand2}} - x_{\text{rand3}}), \tag{15}$$

where F_0 is the differential evolutionary contraction factor set to 0.4; x_{rand1}, x_{rand2}, and x_{rand3} are randomly selected honey badger individuals. The pseudocode of the IHBA is shown in Algorithm 2.

Algorithm 2: Pseudocode of the IHBA.

Input:
Set honey badger population size N_s;
Maximum number of iterations $t_{\max}$;
Dimension D;
Lower bound lb, upper bound ub;
Set parameters: constant C, attraction factor β;
Output:
f_{best}: Global best fitness;
x_{best}: Position of the global best individual;
1. **Initialize** the population x_i using Algorithm 1;
2. Evaluate the fitness of each honey badger position x_i using objective function and assign to f_i, $i \in [1, 2, \ldots, N_s]$;
3. Save best position x_{best} and assign fitness to f_{best};
4. **while** $t < t_{\max}$ **do**
5. Update the decreasing factor α using Equation (5);

Algorithm 2: *Cont.*

6. **For** $i = 1$ **to** N_s **do**
7. Calculate the intensity I_i using Equation (2);
8. Compute the adaptation parameter A using Equation (10);
9. **if** $|A| \geq 1$ **then**
10. **if** $r < 0.5$ **then**
11. Update the position x_{new} using Equation (11), Select a random individual x_{rand} instead of the globally optimal x_{best};
12. **else**
13. Update the position x_{new} using Equation (12);
14. **end if**
15. **else**
16. **if** $r < 0.5$ **then**
17. Update the position x_{new} using Equation (6);
18. else
19. Update the position x_{new} using Equation (8);
20. **end if**
21. **end if**
22. Evaluate the new individual x_{new} and assign to the fitness f_{new};
23. **if** $f_{new} \leq f_i$ **then**
24. Set $x_i = x_{new}$ and $f_i = f_{new}$;
25. **end if**
26. **if** $f_{new} \leq f_{best}$ **then**
27. Set $x_{best} = x_{new}$ and $f_{best} = f_{new}$;
28. **end if**
29. **if** x_{best} remains unchanged for three consecutive iterations, then execute the elite tangent and differential mutation strategies **then**
30. Sort the population fitness in ascending order;
31. Divide the population into two groups (top $N_s/2$ (optimal half) as elite subgroups and after $N_s/2$ as exploration subgroups);
32. **for** $i = 1$ to $N_s/2$ **do**
33. Generate a random angle θ in the range [0, π/2.1];
34. Calculate the step size using Equation (14);
35. **if** $optS$==x_{best} **then**
36. Update the position x_{new} using Equation (13(a));
37. **else**
38. Update the position x_{new} using Equation (13(b));
39. **end if**
40. **end for**
41. **for** $i = N_s/2 + 1$ to N_s **do**
42. Randomly select three distinct individuals *rand*1, *rand*2, *rand*3 from the population;
43. Update the position x_{new} using Equation (15);
44. **end for**
45. **end if**
46. **if** $f_{new} \leq f_i$ **then**
47. Set $x_i = x_{new}$ and $f_i = f_{new}$;
48. **end if**
49. **if** $f_{new} \leq f_{best}$ **then**
50. Set x_{best}=x_{new} and $f_{best} = f_{new}$;
51. **end if**

Algorithm 2: *Cont.*

52. **end for**
53. **end while** Stop criteria satisfied.
54. **Return** f_{best}, x_{best}.

4. Experiment and Results

To assess the optimization performance of the Improved Honey Badger Algorithm (IHBA), we selected 12 benchmark test functions from the CEC2017 suite, each possessing distinct characteristics. For the comparative experiments, we evaluated the IHBA against the following seven other algorithms: the original HBA, COA, DBO, OCSSA, BWO, OOA, and HHO.

To guarantee the fairness of our experimental evaluations, we established a consistent simulation environment using MATLAB R2023a on a Windows 11 operating system powered by an Intel (R) Core (TM) i7-11800H @ 2.3 GHz CPU (The manufacturer of the equipment is Intel Corporation, sourced from Santa Clara, CA, USA.) with 16.0 GB of RAM. Before presenting the results, we outline the experimental setup and crucial parameter settings, detailed in Table 1. The significance of these settings lies in ensuring a level playing field for all algorithms being compared. Specifically, we set the maximum number of iterations to 500 for all algorithms, maintained a population size of 30, and conducted each experiment independently 30 times to ensure statistical reliability.

Table 1. Main parameter settings.

Algorithm	Parameter Setting
COA	a decreases linearly from 2 to 0, $N_c = 5$
DBO	$\gamma = 0.1$, $k = 0.1$, $u = 0.3$, $s = 0.5$, $\alpha \in \{-1, 1\}$
OCSSA	$PD = 0.2$, $SD = 0.1$, $R_2 = 0.8$, $ST = 0.8$
BWO	W_f is probability of whale fall decreased at interval [0.1, 0.05]
OOA	$r_{i,j}$ are random numbers at interval [0, 1], $I_{i,j}$ are random numbers from the set {1, 2}
HHO	E_0 variable changes from -1 to 1, $J = [0, 2]$
HBA	$\beta = 6$, $C = 2$
IHBA	$\beta = 6$, $C = 2$, $F_0 = 0.4$

4.1. Test Function

The 13 test functions selected in this paper are shown in Table 2. f_1 and f_3 are unimodal functions that have no local minima but only global minima, and these functions test the convergence of the algorithm. f_4, f_7, f_8, and f_9 are simple multimodal functions with local extreme points that are used to test the ability of the algorithm to jump out of the local optimum. f_{13}, f_{14}, f_{15}, and f_{16} are hybrid functions, with each subfunction being assigned a certain weight. f_{21} and f_{23} are composition functions composed of at least three hybrid functions or CEC2017 benchmark functions rotated and shifted, and each subfunction has an additional bias value and a weight, which further increase the optimization difficulty of the algorithm, and multiple test functions can fully verify the algorithm's solution performance.

Table 2. Test function.

Type	No.	Function	D	Initial Range	$F_i^* = F_i(x^*)$
Unimodal function	f_1	Shifted and rotated Bent Cigar Function	30	$[-100, 100]^D$	100
	f_3	Shifted and rotated Zakharov Function	30	$[-100, 100]^D$	300
Simple multi-modal function	f_4	Shifted and rotated Rosenbrock Function	30	$[-100, 100]^D$	400
	f_7	Shifted and rotated Lunacek Bi-Rastrigin Function	30	$[-100, 100]^D$	700
	f_8	Shifted and rotated Non-Continuous Rastrigin Function	30	$[-100, 100]^D$	800
	f_9	Shifted and rotated Levy Function	30	$[-100, 100]^D$	900
Hybrid function	f_{13}	Hybrid Function 3 (N = 3)	30	$[-100, 100]^D$	1300
	f_{14}	Hybrid Function 4 (N = 4)	30	$[-100, 100]^D$	1400
	f_{15}	Hybrid Function 5 (N = 4)	30	$[-100, 100]^D$	1500
	f_{16}	Hybrid Function 6 (N = 4)	30	$[-100, 100]^D$	1600
Composition function	f_{21}	Composition Function 1 (N = 3)	30	$[-100, 100]^D$	2100
	f_{23}	Composition Function 3 (N = 4)	30	$[-100, 100]^D$	2200

4.2. Results and Discussions

4.2.1. CEC2017 Test Function Set Optimization Experiment

This section presents and discusses the experimental results obtained by evaluating eight algorithms across 13 test functions in 10-dimensional, 30-dimensional, and 50-dimensional spaces. Each experiment was independently replicated 30 times to ensure statistical reliability, and the comprehensive results are summarized in Table 3. We undertake a detailed analysis of the performance of each algorithm, focusing on their mean and variance in achieving optimal solutions. This analysis highlights key consistent observations and patterns across different dimensions and test functions.

Table 3. Optimization results and comparison for CEC-2017 test functions in three different dimensions.

Fun.	D	Meas.	IHBA	OCSSA	HBA	HHO	BWO	OOA	DBO	COA
f_1	10	Best	$\mathbf{1.00 \times 10^2}$	1.04×10^2	1.33×10^2	1.90×10^5	4.39×10^9	1.55×10^9	1.84×10^8	4.24×10^9
		Std	2.47×10^3	$\mathbf{2.22 \times 10^3}$	3.50×10^3	3.01×10^5	3.83×10^9	2.74×10^9	5.45×10^8	2.86×10^9
		Ave	$\mathbf{2.06 \times 10^3}$	3.04×10^3	3.44×10^3	6.90×10^5	1.42×10^{10}	8.21×10^9	1.18×10^9	8.15×10^9
	30	Best	$\mathbf{6.43 \times 10^2}$	1.37×10^4	3.95×10^3	4.24×10^7	5.48×10^{10}	3.27×10^{10}	2.04×10^{10}	3.51×10^{10}
		Std	4.83×10^5	$\mathbf{3.60 \times 10^5}$	8.40×10^5	3.74×10^7	4.97×10^9	7.04×10^9	3.73×10^9	7.54×10^9
		Ave	$\mathbf{1.31 \times 10^5}$	2.35×10^5	2.06×10^5	8.84×10^7	6.77×10^{10}	5.12×10^{10}	2.74×10^{10}	5.29×10^{10}
	50	Best	$\mathbf{1.57 \times 10^6}$	6.12×10^7	6.91×10^6	7.22×10^8	1.10×10^{11}	8.76×10^{10}	5.97×10^{10}	8.83×10^{10}
		Std	$\mathbf{1.57 \times 10^7}$	9.29×10^7	3.67×10^8	7.45×10^8	5.55×10^9	8.24×10^9	5.36×10^9	8.32×10^9
		Ave	$\mathbf{2.06 \times 10^7}$	2.07×10^8	2.54×10^8	1.55×10^9	1.22×10^{11}	1.09×10^{11}	6.90×10^{10}	1.09×10^{11}
f_3	10	Best	$\mathbf{3.00 \times 10^2}$	3.00×10^2	3.00×10^2	3.02×10^2	7.61×10^3	4.45×10^3	5.01×10^2	3.65×10^3
		Std	$\mathbf{2.68 \times 10^{-7}}$	3.99×10^{-3}	3.05×10^{-5}	5.33×10^1	1.93×10^3	3.05×10^3	1.49×10^3	2.94×10^3
		Ave	$\mathbf{3.00 \times 10^2}$	3.00×10^2	3.00×10^2	3.40×10^2	1.19×10^4	1.22×10^4	3.28×10^3	9.47×10^3
	30	Best	$\mathbf{2.00 \times 10^3}$	2.76×10^4	1.33×10^4	3.18×10^4	6.91×10^4	6.90×10^4	6.50×10^4	6.52×10^4
		Std	$\mathbf{3.10 \times 10^3}$	8.38×10^3	6.84×10^3	7.26×10^3	1.08×10^4	7.71×10^3	9.02×10^3	7.46×10^3
		Ave	$\mathbf{5.86 \times 10^3}$	4.19×10^4	2.76×10^4	4.45×10^4	9.32×10^4	8.49×10^4	8.17×10^4	8.50×10^4
	50	Best	$\mathbf{4.52 \times 10^4}$	1.33×10^5	7.60×10^4	1.17×10^5	1.73×10^5	1.34×10^5	1.68×10^5	1.63×10^5
		Std	$\mathbf{1.29 \times 10^4}$	2.85×10^4	2.29×10^4	1.71×10^4	2.32×10^4	3.18×10^4	4.48×10^4	1.58×10^4
		Ave	$\mathbf{6.48 \times 10^4}$	1.85×10^5	1.24×10^5	1.51×10^5	2.13×10^5	2.12×10^5	2.26×10^5	1.92×10^5

Table 3. *Cont.*

Fun.	*D*	Meas.	IHBA	OCSSA	HBA	HHO	BWO	OOA	DBO	COA
f_4	10	Best	$\mathbf{4.00 \times 10^{2}}$	4.00×10^{2}	4.00×10^{2}	4.00×10^{2}	8.44×10^{2}	4.80×10^{2}	4.40×10^{2}	6.21×10^{2}
		Std	$\mathbf{5.54 \times 10^{-1}}$	2.23×10^{1}	1.43	3.93×10^{1}	4.47×10^{2}	2.71×10^{2}	2.68×10^{1}	3.24×10^{2}
		Ave	$\mathbf{4.01 \times 10^{2}}$	4.12×10^{2}	4.02×10^{2}	4.31×10^{2}	1.62×10^{3}	8.58×10^{2}	4.82×10^{2}	9.58×10^{2}
	30	Best	$\mathbf{4.10 \times 10^{2}}$	4.74×10^{2}	4.46×10^{2}	5.08×10^{2}	1.11×10^{4}	8.30×10^{3}	3.16×10^{3}	7.46×10^{3}
		Std	2.79×10^{1}	2.44×10^{1}	$\mathbf{2.33 \times 10^{1}}$	5.66×10^{1}	2.44×10^{3}	2.78×10^{3}	1.93×10^{3}	2.89×10^{3}
		Ave	$\mathbf{4.99 \times 10^{2}}$	5.10×10^{2}	5.01×10^{2}	6.06×10^{2}	1.70×10^{4}	1.33×10^{4}	6.37×10^{3}	1.37×10^{4}
	50	Best	5.27×10^{2}	5.13×10^{2}	$\mathbf{4.81 \times 10^{2}}$	7.24×10^{2}	4.08×10^{4}	1.79×10^{4}	1.34×10^{4}	2.48×10^{4}
		Std	$\mathbf{5.48 \times 10^{1}}$	6.64×10^{1}	8.01×10^{1}	2.19×10^{2}	3.10×10^{3}	7.04×10^{3}	2.08×10^{3}	5.95×10^{3}
		Ave	$\mathbf{6.09 \times 10^{2}}$	6.22×10^{2}	6.47×10^{2}	1.14×10^{3}	4.61×10^{4}	3.50×10^{4}	1.70×10^{4}	3.62×10^{4}
f_7	10	Best	$\mathbf{7.15 \times 10^{2}}$	7.24×10^{2}	7.19×10^{2}	7.54×10^{2}	8.09×10^{2}	7.56×10^{2}	7.61×10^{2}	7.62×10^{2}
		Std	**4.71**	2.59×10^{1}	9.86	1.38×10^{1}	8.72	1.82×10^{1}	1.06×10^{1}	1.84×10^{1}
		Ave	$\mathbf{7.23 \times 10^{2}}$	7.62×10^{2}	7.32×10^{2}	7.84×10^{2}	8.30×10^{2}	7.90×10^{2}	7.80×10^{2}	7.95×10^{2}
	30	Best	$\mathbf{7.58 \times 10^{2}}$	8.58×10^{2}	8.32×10^{2}	1.12×10^{3}	1.33×10^{3}	1.26×10^{3}	1.11×10^{3}	1.27×10^{3}
		Std	$\mathbf{2.89 \times 10^{1}}$	1.51×10^{2}	5.24×10^{1}	6.52×10^{1}	3.57×10^{1}	6.42×10^{1}	3.43×10^{1}	5.49×10^{1}
		Ave	$\mathbf{8.00 \times 10^{2}}$	1.09×10^{3}	9.10×10^{2}	1.29×10^{3}	1.47×10^{3}	1.39×10^{3}	1.20×10^{3}	1.42×10^{3}
	50	Best	$\mathbf{8.69 \times 10^{2}}$	1.08×10^{3}	1.02×10^{3}	1.66×10^{3}	1.99×10^{3}	1.81×10^{3}	1.58×10^{3}	1.88×10^{3}
		Std	4.44×10^{1}	1.92×10^{2}	1.12×10^{2}	8.83×10^{1}	$\mathbf{3.14 \times 10^{1}}$	9.47×10^{1}	6.95×10^{1}	7.10×10^{1}
		Ave	$\mathbf{9.40 \times 10^{2}}$	1.56×10^{3}	1.20×10^{3}	1.86×10^{3}	2.09×10^{3}	2.01×10^{3}	1.76×10^{3}	2.03×10^{3}
f_8	10	Best	$\mathbf{8.01 \times 10^{2}}$	8.19×10^{2}	8.09×10^{2}	8.11×10^{2}	8.41×10^{2}	8.21×10^{2}	8.26×10^{2}	8.32×10^{2}
		Std	5.92	9.05	6.26	8.85	5.95	9.22	**5.87**	9.20
		Ave	$\mathbf{8.12 \times 10^{2}}$	8.35×10^{2}	8.18×10^{2}	8.29×10^{2}	8.54×10^{2}	8.49×10^{2}	8.35×10^{2}	8.54×10^{2}
	30	Best	$\mathbf{8.28 \times 10^{2}}$	9.13×10^{2}	8.42×10^{2}	9.32×10^{2}	1.13×10^{3}	1.08×10^{3}	1.01×10^{3}	1.05×10^{3}
		Std	1.63×10^{1}	3.15×10^{1}	2.45×10^{1}	2.35×10^{1}	$\mathbf{1.58 \times 10^{1}}$	2.49×10^{1}	2.44×10^{1}	2.62×10^{1}
		Ave	$\mathbf{8.51 \times 10^{2}}$	9.70×10^{2}	8.96×10^{2}	9.78×10^{2}	1.16×10^{3}	1.12×10^{3}	1.06×10^{3}	1.13×10^{3}
	50	Best	$\mathbf{8.72 \times 10^{2}}$	1.08×10^{3}	9.88×10^{2}	1.14×10^{3}	1.48×10^{3}	1.42×10^{3}	1.31×10^{3}	1.40×10^{3}
		Std	3.79×10^{1}	4.42×10^{1}	3.56×10^{1}	3.69×10^{1}	$\mathbf{2.25 \times 10^{1}}$	3.28×10^{1}	4.06×10^{1}	3.48×10^{1}
		Ave	$\mathbf{9.40 \times 10^{2}}$	1.15×10^{3}	1.05×10^{3}	1.21×10^{3}	1.52×10^{3}	1.47×10^{3}	1.39×10^{3}	1.48×10^{3}
f_9	10	Best	$\mathbf{9.00 \times 10^{2}}$	$\mathbf{9.00 \times 10^{2}}$	$\mathbf{9.00 \times 10^{2}}$	1.12×10^{3}	1.45×10^{3}	9.77×10^{2}	9.53×10^{2}	1.04×10^{3}
		Std	$\mathbf{2.27 \times 10^{-2}}$	1.60×10^{2}	2.51	2.00×10^{2}	1.28×10^{2}	1.89×10^{2}	9.49×10^{1}	2.18×10^{2}
		Ave	$\mathbf{9.00 \times 10^{2}}$	1.01×10^{3}	9.01×10^{2}	1.50×10^{3}	1.80×10^{3}	1.34×10^{3}	1.06×10^{3}	1.41×10^{3}
	30	Best	$\mathbf{9.06 \times 10^{2}}$	2.81×10^{3}	1.06×10^{3}	5.72×10^{3}	9.35×10^{3}	5.84×10^{3}	4.57×10^{3}	4.95×10^{3}
		Std	$\mathbf{5.99} \times 10^{1}$	7.57×10^{2}	9.18×10^{2}	9.25×10^{2}	7.96×10^{2}	1.21×10^{3}	1.45×10^{3}	1.80×10^{3}
		Ave	$\mathbf{9.76 \times 10^{2}}$	5.07×10^{3}	2.60×10^{3}	7.93×10^{3}	1.12×10^{4}	9.17×10^{3}	7.86×10^{3}	9.92×10^{3}
	50	Best	$\mathbf{1.14 \times 10^{3}}$	8.28×10^{3}	4.60×10^{3}	1.89×10^{4}	3.37×10^{4}	2.94×10^{4}	2.14×10^{4}	2.76×10^{4}
		Std	$\mathbf{4.02 \times 10^{2}}$	1.99×10^{3}	4.17×10^{3}	3.87×10^{3}	2.16×10^{3}	3.04×10^{3}	3.95×10^{3}	3.69×10^{3}
		Ave	$\mathbf{1.80 \times 10^{3}}$	1.32×10^{4}	1.01×10^{4}	2.88×10^{4}	3.82×10^{4}	3.40×10^{4}	3.13×10^{4}	3.45×10^{4}
f_{13}	10	Best	$\mathbf{1.31 \times 10^{3}}$	2.23×10^{3}	1.48×10^{3}	2.11×10^{3}	3.57×10^{5}	2.37×10^{3}	2.81×10^{3}	2.44×10^{3}
		Std	$\mathbf{1.07 \times 10^{2}}$	3.09×10^{3}	1.27×10^{4}	1.34×10^{4}	2.02×10^{7}	1.23×10^{4}	1.63×10^{4}	6.92×10^{3}
		Ave	$\mathbf{1.38 \times 10^{3}}$	1.46×10^{4}	9.24×10^{3}	1.49×10^{4}	1.59×10^{7}	1.36×10^{4}	2.39×10^{4}	1.05×10^{4}
	30	Best	$\mathbf{3.01 \times 10^{3}}$	5.12×10^{3}	7.38×10^{3}	2.78×10^{5}	7.33×10^{9}	9.86×10^{8}	2.11×10^{8}	7.82×10^{8}
		Std	$\mathbf{2.58 \times 10^{4}}$	4.25×10^{4}	3.56×10^{4}	7.35×10^{5}	4.41×10^{9}	3.69×10^{9}	1.13×10^{9}	3.70×10^{9}
		Ave	$\mathbf{3.16 \times 10^{4}}$	3.43×10^{4}	4.35×10^{4}	9.50×10^{5}	1.50×10^{9}	7.00×10^{9}	1.83×10^{9}	6.95×10^{9}
	50	Best	$\mathbf{4.63 \times 10^{3}}$	1.35×10^{4}	1.25×10^{4}	3.23×10^{6}	4.73×10^{10}	2.48×10^{10}	3.42×10^{9}	1.80×10^{10}
		Std	$\mathbf{1.51 \times 10^{4}}$	4.75×10^{4}	6.52×10^{4}	9.01×10^{6}	1.21×10^{10}	1.19×10^{10}	5.57×10^{9}	1.07×10^{10}
		Ave	$\mathbf{2.10 \times 10^{4}}$	7.86×10^{4}	6.94×10^{4}	9.69×10^{6}	6.67×10^{10}	4.58×10^{10}	1.23×10^{10}	3.84×10^{10}
f_{14}	10	Best	$\mathbf{1.40 \times 10^{3}}$	1.43×10^{3}	1.44×10^{3}	1.48×10^{3}	1.57×10^{3}	1.45×10^{3}	1.52×10^{3}	1.46×10^{3}
		Std	**7.90**	6.85×10^{1}	5.64×10^{1}	3.82×10^{2}	2.01×10^{2}	2.11×10^{3}	7.10×10^{2}	3.35×10^{1}
		Ave	$\mathbf{1.42 \times 10^{3}}$	1.50×10^{3}	1.52×10^{3}	1.68×10^{3}	1.80×10^{3}	2.55×10^{3}	2.15×10^{3}	1.51×10^{3}
	30	Best	$\mathbf{1.56 \times 10^{3}}$	5.38×10^{3}	2.72×10^{3}	6.68×10^{3}	3.15×10^{6}	1.28×10^{5}	5.56×10^{4}	1.41×10^{5}
		Std	$\mathbf{2.15 \times 10^{3}}$	3.40×10^{4}	1.95×10^{4}	7.07×10^{5}	7.65×10^{6}	2.50×10^{6}	4.64×10^{5}	1.95×10^{6}
		Ave	$\mathbf{2.65 \times 10^{3}}$	4.87×10^{4}	2.67×10^{4}	7.96×10^{5}	1.33×10^{7}	2.56×10^{6}	5.94×10^{5}	2.08×10^{6}
	50	Best	$\mathbf{6.85 \times 10^{3}}$	1.20×10^{5}	3.17×10^{4}	1.81×10^{5}	5.03×10^{7}	1.29×10^{7}	4.76×10^{5}	5.02×10^{6}
		Std	$\mathbf{5.38 \times 10^{4}}$	3.67×10^{5}	5.14×10^{5}	1.97×10^{6}	1.28×10^{8}	6.41×10^{7}	7.42×10^{6}	2.87×10^{7}
		Ave	$\mathbf{6.27 \times 10^{4}}$	4.96×10^{5}	2.50×10^{5}	3.06×10^{6}	2.32×10^{8}	9.59×10^{7}	8.46×10^{6}	4.63×10^{7}

Table 3. *Cont.*

Fun.	*D*	Meas.	IHBA	OCSSA	HBA	HHO	BWO	OOA	DBO	COA
f_{15}	10	Best	$\mathbf{1.50 \times 10^3}$	1.63×10^3	1.56×10^8	2.05×10^3	3.15×10^3	1.82×10^3	1.94×10^3	1.69×10^3
		Std	$\mathbf{4.82 \times 10^1}$	7.73×10^2	9.54×10^2	3.02×10^3	2.18×10^3	4.54×10^3	1.64×10^3	2.89×10^3
		Ave	$\mathbf{1.53 \times 10^3}$	2.22×10^3	1.84×10^3	6.58×10^3	8.69×10^3	1.12×10^4	4.08×10^3	4.61×10^3
	30	Best	$\mathbf{1.68 \times 10^3}$	2.04×10^3	1.96×10^3	3.37×10^4	8.99×10^7	5.54×10^6	3.61×10^5	4.78×10^6
		Std	$\mathbf{1.36 \times 10^4}$	1.42×10^4	1.64×10^4	4.89×10^4	4.60×10^8	4.38×10^8	4.89×10^6	4.02×10^8
		Ave	1.21×10^4	$\mathbf{9.61 \times 10^3}$	1.91×10^4	9.07×10^4	1.04×10^9	3.83×10^8	4.59×10^6	4.11×10^8
	50	Best	$\mathbf{2.12 \times 10^3}$	5.89×10^3	6.06×10^3	2.73×10^5	8.59×10^9	2.38×10^9	1.38×10^8	1.44×10^9
		Std	$\mathbf{9.93 \times 10^3}$	2.09×10^4	1.18×10^4	4.41×10^5	2.76×10^9	2.96×10^9	9.10×10^8	3.29×10^9
		Ave	$\mathbf{1.44 \times 10^4}$	3.71×10^4	2.51×10^4	9.50×10^5	1.36×10^{10}	7.64×10^9	1.62×10^9	7.90×10^9
f_{16}	10	Best	$\mathbf{1.60 \times 10^3}$	1.60×10^3	1.60×10^3	1.61×10^3	1.92×10^3	1.71×10^3	1.63×10^3	1.70×10^3
		Std	$\mathbf{5.54 \times 10^1}$	1.89×10^2	1.00×10^2	1.44×10^2	1.05×10^2	1.38×10^2	9.35×10^1	1.34×10^2
		Ave	$\mathbf{1.64 \times 10^3}$	1.98×10^3	1.73×10^3	1.87×10^3	2.19×10^3	1.97×10^3	1.78×10^3	1.97×10^3
	30	Best	$\mathbf{1.75 \times 10^3}$	1.99×10^3	1.75×10^3	2.67×10^3	6.00×10^3	3.90×10^3	3.53×10^3	3.76×10^3
		Std	3.30×10^2	3.83×10^2	5.68×10^2	4.91×10^2	1.78×10^3	6.78×10^2	$\mathbf{2.71 \times 10^2}$	8.40×10^2
		Ave	$\mathbf{2.35 \times 10^3}$	2.80×10^3	2.72×10^3	3.56×10^3	8.39×10^3	5.11×10^3	3.98×10^3	5.35×10^3
	50	Best	$\mathbf{2.15 \times 10^3}$	2.58×10^3	2.41×10^3	3.52×10^3	7.75×10^3	6.51×10^3	4.62×10^3	6.26×10^3
		Std	5.67×10^2	$\mathbf{4.11 \times 10^2}$	4.60×10^2	7.80×10^2	1.74×10^3	1.20×10^3	5.28×10^2	1.53×10^3
		Ave	$\mathbf{3.25 \times 10^3}$	3.64×10^3	3.46×10^3	4.67×10^3	1.19×10^4	8.41×10^3	5.64×10^3	8.81×10^3
f_{21}	10	Best	$\mathbf{2.20 \times 10^3}$	2.20×10^3	2.20×10^3	2.20×10^3	2.23×10^3	2.23×10^3	2.21×10^3	2.23×10^3
		Std	5.51×10^1	6.09×10^1	5.52×10^1	6.24×10^1	5.45×10^1	4.93×10^1	**8.33**	4.44×10^1
		Ave	2.27×10^3	2.31×10^3	2.28×10^3	2.33×10^3	2.31×10^3	2.33×10^3	$\mathbf{2.22 \times 10^3}$	2.35×10^3
	30	Best	2.33×10^3	$\mathbf{2.20 \times 10^3}$	2.36×10^3	2.48×10^3	2.58×10^3	2.58×10^3	2.33×10^3	2.64×10^3
		Std	$\mathbf{1.39 \times 10^1}$	1.51×10^2	3.11×10^1	4.27×10^1	6.86×10^1	4.73×10^1	9.83×10^1	4.01×10^1
		Ave	2.35×10^3	$\mathbf{2.33 \times 10^3}$	2.40×10^3	2.57×10^3	2.70×10^3	2.69×10^3	2.53×10^3	2.71×10^3
	50	Best	$\mathbf{2.37 \times 10^3}$	2.49×10^3	2.41×10^3	2.72×10^3	3.16×10^3	3.02×10^3	2.89×10^3	3.08×10^3
		Std	$\mathbf{2.41 \times 10^1}$	6.94×10^1	4.49×10^1	9.14×10^1	8.19×10^1	5.92×10^1	4.68×10^1	8.02×10^1
		Ave	$\mathbf{2.41 \times 10^3}$	2.63×10^3	2.52×10^3	2.91×10^3	3.30×10^3	3.13×10^3	2.97×10^3	3.20×10^3
f_{23}	10	Best	$\mathbf{2.60 \times 10^3}$	2.61×10^3	2.61×10^3	2.61×10^3	2.66×10^3	2.66×10^3	2.63×10^3	2.66×10^3
		Std	**9.08**	1.00×10^1	1.47×10^1	2.91×10^1	2.94×10^1	2.21×10^1	1.73×10^1	2.55×10^1
		Ave	$\mathbf{2.62 \times 10^3}$	2.62×10^3	2.62×10^3	2.66×10^3	2.73×10^3	2.69×10^3	2.66×10^3	2.69×10^3
	30	Best	$\mathbf{2.66 \times 10^3}$	2.75×10^3	2.73×10^3	3.04×10^3	3.54×10^3	3.27×10^3	3.03×10^3	3.25×10^3
		Std	$\mathbf{2.27 \times 10^1}$	7.59×10^1	3.53×10^1	1.22×10^2	1.97×10^2	1.53×10^2	9.57×10^1	1.16×10^2
		Ave	$\mathbf{2.71 \times 10^3}$	2.85×10^3	2.79×10^3	3.20×10^3	3.84×10^3	3.59×10^3	3.16×10^3	3.49×10^3
	50	Best	$\mathbf{2.80 \times 10^3}$	2.97×10^3	2.88×10^3	3.51×10^3	4.58×10^3	3.97×10^3	3.51×10^3	4.12×10^3
		Std	$\mathbf{3.64 \times 10^1}$	1.11×10^2	7.79×10^1	1.82×10^2	1.44×10^2	2.02×10^2	1.65×10^2	1.72×10^2
		Ave	$\mathbf{2.87 \times 10^3}$	3.14×10^3	3.02×10^3	3.85×10^3	4.83×10^3	4.50×10^3	3.84×10^3	4.40×10^3

Table 3 demonstrates that the optimized IHBA outperforms the seven comparison algorithms across most functions and dimensions, showcasing superior optimization capabilities and achieving outstanding results.

In the 10-dimensional case, the IHBA achieves the optimal average value across all tested functions, demonstrating superior optimization performance compared to the other seven comparison algorithms. Specifically, for functions f_3 and f_9, the IHBA attains the best optimal values, average values, and standard deviations, indicating its strong consistency and solving ability in both unimodal and simple multimodal problems. For function f_1, the standard deviation of the IHBA is slightly inferior to that of OCSSA; however, the IHBA still outperforms the other six comparison algorithms, showcasing high competitiveness. Regarding function f_4, the IHBA matches the optimal values of the OCSSA, HBA, and HHO and surpasses the other four comparison algorithms. For functions f_7, f_{13}, f_{14}, f_{15}, f_{16}, and f_{23}, the IHBA exhibits superior optimal values, average values, and standard deviations compared to the other seven comparison algorithms, indicating strong robustness on multimodal and hybrid functions. In the case of function f_{21}, the average value of the IHBA is slightly inferior to that of the DBO algorithm but exceeds the other six comparison algorithms, and its standard deviation is slightly lower than BWO, OOA, DBO, and COA

but better than the other three comparison algorithms. For function f_8, the standard deviation of the IHBA is slightly inferior to that of the DBO algorithm, but its optimal and average values are superior to the other six comparison algorithms.

In the 30-dimensional case, the IHBA demonstrates superior optimization capabilities, particularly in simple multimodal and hybrid functions, while maintaining strong stability and global exploration. Specifically, for unimodal function f_1, the IHBA achieves superior optimal and average values compared to the other seven comparison algorithms, with its standard deviation being slightly inferior to OCSSA but better than the other six comparison algorithms. For unimodal function f_3, the IHBA outperforms the other seven comparison algorithms in terms of optimal value, standard deviation, and average value. Regarding the simple multimodal function f_4, the IHBA achieves better optimal and average values than the other seven comparison algorithms, although its standard deviation is slightly inferior to the HBA. For the simple multimodal function f_8, the IHBA attains superior optimal and average values compared to the other seven comparison algorithms, with its standard deviation being slightly inferior to the BWO and COA. In the case of simple multimodal functions f_7 and f_9, the IHBA achieves superior optimal values, standard deviations, and average values compared to the other seven comparison algorithms. For hybrid functions f_{13} and f_{14}, the IHBA demonstrates better optimal values, average values, and standard deviations than the other seven comparison algorithms. Regarding the hybrid function f_{15}, the IHBA achieves a superior optimal value and standard deviation compared to the other seven comparison algorithms, although its average value is slightly inferior to the OCSSA and better than the other six comparison algorithms. For hybrid function f_{16}, the IHBA's optimal and average values surpass those of the other seven comparison algorithms, while its standard deviation is slightly inferior to that of the DBO. In the case of the composition function f_{21}, the IHBA exhibits a better standard deviation than the other seven comparison algorithms, though its optimal and average values are slightly inferior to the OCSSA. Lastly, for the composition function f_{23}, the IHBA achieves the best optimal values, average values, and standard deviations.

In the 50-dimensional case, the IHBA continues to demonstrate strong optimization capabilities, particularly in unimodal, hybrid, and composition functions. However, the stability performance of certain simple multimodal functions shows variability. Specifically, for unimodal functions f_1 and f_3, the IHBA achieves superior optimal values, standard deviations, and average values compared to the other seven comparison algorithms. Regarding the simple multimodal function f_4, the optimal value of the IHBA is slightly inferior to that of the HBA, yet it surpasses the other six comparison algorithms in terms of optimal value, standard deviation, and average value. For simple multimodal functions f_7, f_8, and f_9, the IHBA attains the best optimal and average values, although its standard deviation is less favorable. In the case of hybrid functions f_{13}, f_{14}, and f_{15}, the IHBA records the highest optimal values, average values, and standard deviations. For the hybrid function f_{16}, the optimal and average values of the IHBA exceed those of the other seven comparison algorithms, while its standard deviation is marginally inferior to that of the OCSSA. Lastly, for composition functions f_{21} and f_{23}, the IHBA achieves the best optimal values, average values, and standard deviations.

The analysis of the optimization results indicates that the IHBA outperforms the other algorithms on the CEC 2017 benchmark, demonstrating superiority across multiple dimensions. It is also exceptionally effective in solving high-dimensional and complex problems, particularly compared to the HBA. Experimental findings show that the enhanced strategy proposed in this paper substantially enhances both the accuracy and stability of the HBA.

Evaluating the complexity of an algorithm by directly comparing its running time is an intuitive method. Table 4 provides a detailed list of the execution times of the IHBA and

HBA on 12 test functions. It is evident from the data in the table that the IHBA generally outperforms the HBA in terms of runtime across all test functions. In addition, with the increase in problem dimensions, both the IHBA and HBA show an upward trend in computation time.

Table 4. Average time for CEC-2017 test functions in three different dimensions.

Fun.	f_1			f_3			f_4		
Dim.	10	30	50	10	30	50	10	30	50
IHBA	7.86×10^{-2}	1.02×10^{-1}	1.40×10^{-1}	7.37×10^{-2}	1.16×10^{-1}	1.61×10^{-1}	6.99×10^{-2}	9.70×10^{-2}	1.37×10^{-1}
HBA	6.05×10^{-2}	8.30×10^{-2}	1.18×10^{-1}	5.81×10^{-2}	8.28×10^{-2}	1.19×10^{-1}	5.87×10^{-2}	8.29×10^{-2}	1.18×10^{-1}
Fun.	f_7			f_8			f_9		
Dim.	10	30	50	10	30	50	10	30	50
IHBA	9.24×10^{-2}	1.33×10^{-1}	1.85×10^{-1}	9.25×10^{-2}	1.36×10^{-1}	1.81×10^{-1}	8.80×10^{-2}	1.40×10^{-1}	2.06×10^{-1}
HBA	6.79×10^{-2}	1.02×10^{-1}	1.46×10^{-1}	6.47×10^{-2}	1.02×10^{-1}	1.45×10^{-1}	6.78×10^{-2}	1.05×10^{-1}	1.46×10^{-1}
Fun.	f_{13}			f_{14}			f_{15}		
Dim.	10	30	50	10	30	50	10	30	50
IHBA	8.65×10^{-2}	1.11×10^{-1}	1.51×10^{-1}	9.46×10^{-2}	1.45×10^{-1}	1.98×10^{-1}	8.57×10^{-2}	1.08×10^{-1}	1.45×10^{-1}
HBA	6.37×10^{-2}	9.54×10^{-2}	1.37×10^{-1}	6.71×10^{-2}	1.11×10^{-1}	1.62×10^{-1}	6.07×10^{-2}	9.08×10^{-2}	1.27×10^{-1}
Fun.	f_{16}			f_{21}			f_{23}		
Dim.	10	30	50	10	30	50	10	30	50
IHBA	9.42×10^{-2}	1.31×10^{-1}	1.81×10^{-1}	1.04×10^{-1}	1.89×10^{-1}	3.01×10^{-1}	1.26×10^{-1}	2.16×10^{-1}	3.66×10^{-1}
HBA	6.36×10^{-2}	9.92×10^{-2}	1.39×10^{-1}	7.96×10^{-2}	1.47×10^{-1}	2.38×10^{-1}	9.06×10^{-2}	1.72×10^{-1}	2.91×10^{-1}

Figure 2 presents a radar chart that analyzes the performance of the IHBA compared to seven other benchmark algorithms using the CEC2017 test functions. The IHBA demonstrates superior performance across multiple dimensions, highlighting its effectiveness in tackling complex optimization problems. Specifically, the IHBA excels in convergence speed, solution accuracy, and algorithm stability, outperforming its counterparts, especially in handling multimodal functions and high-dimensional optimization scenarios. Additionally, the IHBA achieves a balanced trade-off between exploration and exploitation, effectively preventing premature convergence while maintaining robust global search capabilities. In contrast, although some competing algorithms may exhibit strengths in specific test functions, their performance lacks the consistency and comprehensiveness demonstrated by the IHBA. The analysis derived from the radar chart underscores the IHBA's enhanced adaptability and robustness, positioning it as a highly competitive method within the evolutionary algorithm landscape. These findings suggest that the IHBA is well-suited for diverse optimization challenges, offering reliable and efficient solutions across various complex environments.

To further compare and analyze the optimization-seeking ability of the improved algorithms, the average fitness results of each algorithm are ranked, and the standard deviations are compared to see if they are equal. The rankings are tied if both the mean and standard deviation are equal. A lower average performance ranking result for the algorithm indicates a superior optimization ability and enhanced optimization search performance. Figure 3 demonstrates the average merit-seeking ranking results of various algorithms. The IHBA is ranked first, with an average ranking result of 1.33, while for the OCSSA, HBA, HHO, BWO, OOA, DBO, and COA, the average ranking results are 3.00, 2.17, 4.50, 7.83, 6.42, 5.25, and 5.50, respectively. From the average ranking results, it can be seen that the ability of the IHBA to find the best results is better than the other seven compared algorithms.

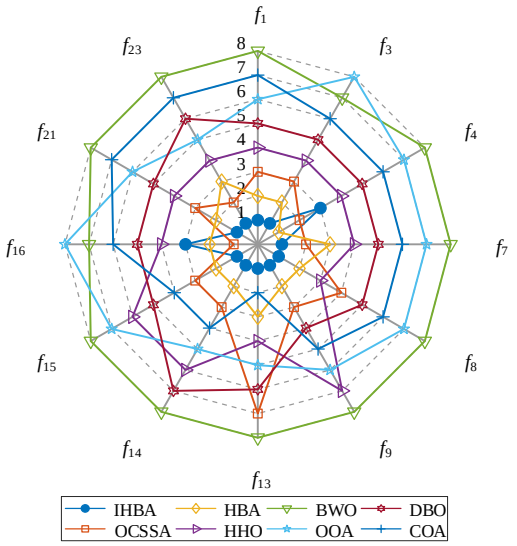


Figure 2. Radar chart of sort of algorithms.

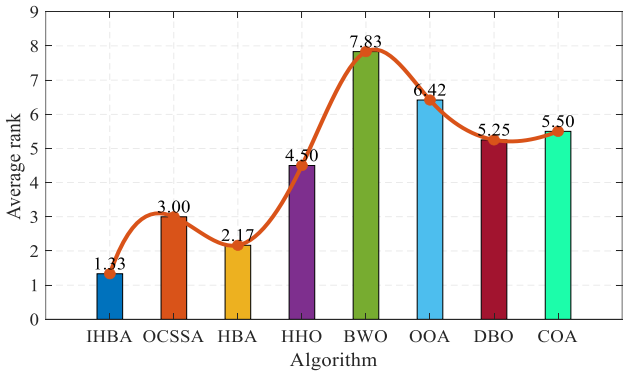


Figure 3. Results of the sort of average performance.

To analyze the distribution characteristics of the IHBA optimization results, box plots were drawn based on the results of each algorithm running independently for 30 times, as shown in Figure 4. The upper and lower boundaries of each box represent the upper and lower quartiles, respectively, and the center line of each box is the median of the 30 optimization results. The symbol "o" indicates the outliers within the box. It can be seen from the box plots that, except for Figure 4e,j, the 30 optimization results of the IHBA are relatively concentrated. Therefore, the improved algorithm in this paper is strong and robust.

Boxplots can reflect the stability of a set of data. The boxplots of the algorithm on 12 representative test functions are plotted in Figure 4, and it is found that the IHBA solves the problem in a more stable manner than the other algorithms, and the distribution of the convergence values is more centralized, which is obviously more superior to the other algorithms and indicates that the IHBA has a better robustness.

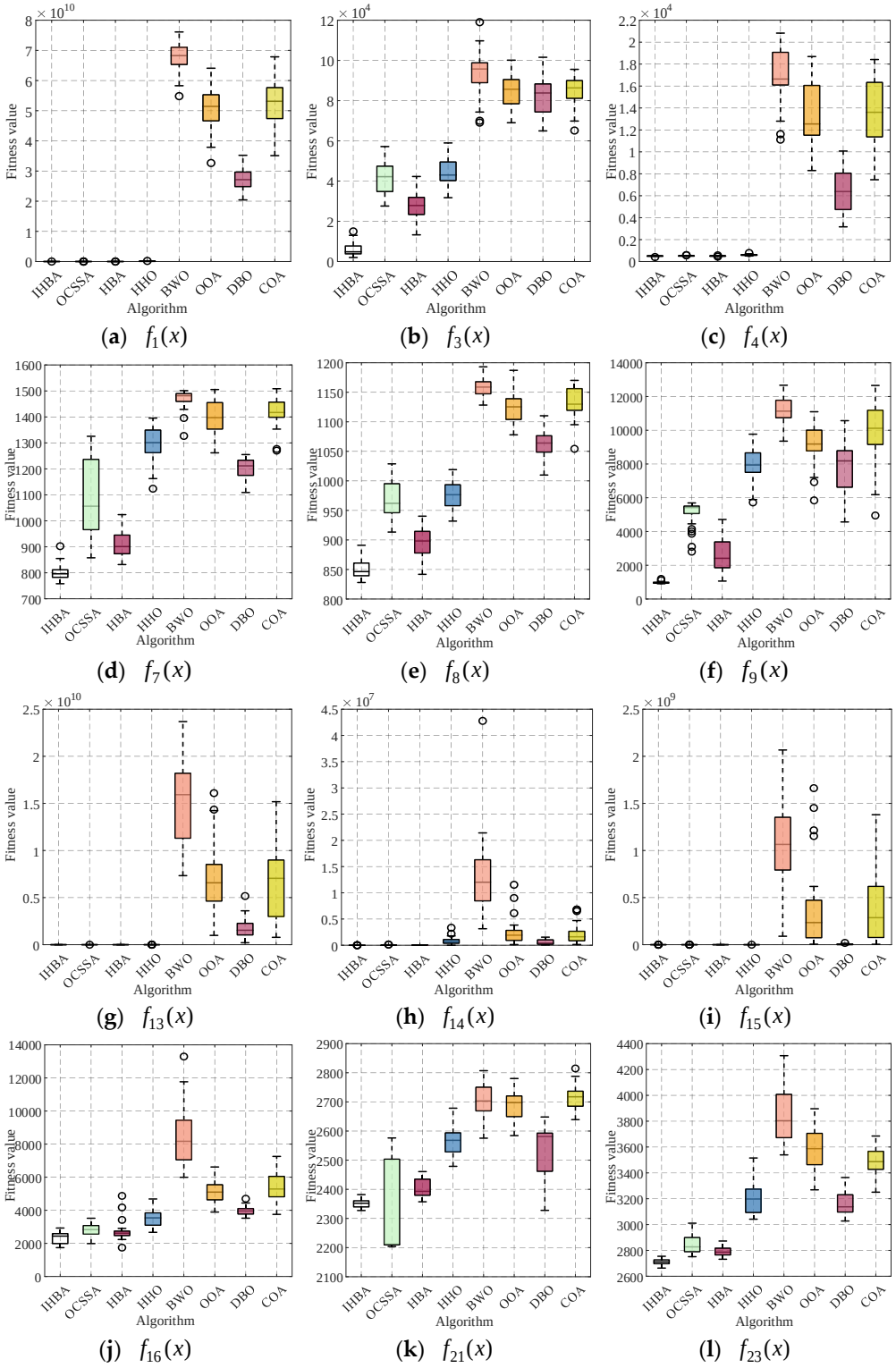


Figure 4. Convergence boxplot of the test function in 30-dimensional spaces.

4.2.2. Wilcoxon Rank Sum Test

The Wilcoxon rank sum test is a commonly used hypothesis test for analyzing significant algorithm differences. It is widely employed to verify the effectiveness of optimization algorithm improvement strategies and ascertain whether there are significant differences between various algorithms.

The test results are presented in Table 5, where the p-values determine whether a significant difference exists. The significance level is set at the commonly used value of 0.05.

If $p < 0.5$, the null hypothesis can be rejected, indicating a significant difference between the two compared algorithms. Based on the statistical test results, it is evident that the p-values for the IHBA, when compared to the other seven algorithms, are predominantly less than 0.05. This suggests a significant difference in search capability between the improved and comparative algorithms, indicating its statistical superiority.

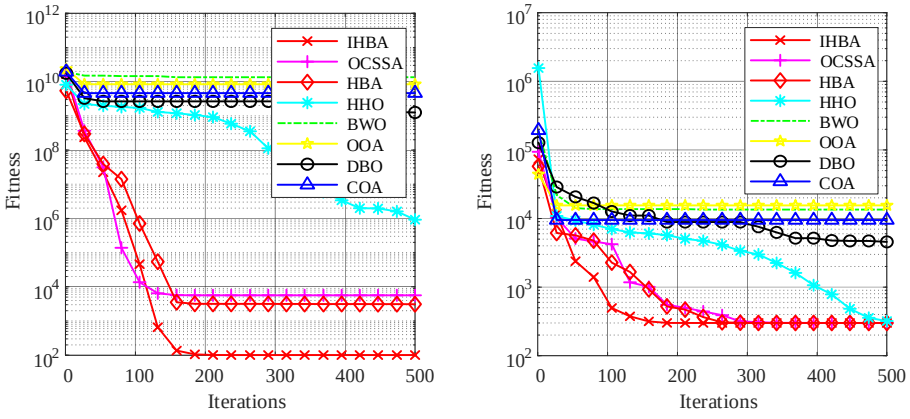
Table 5. The p-values of the Wilcoxon rank sum test (dimension 30) in CEC2017.

Function	OCSSA	HBA	HHO	BWO	OOA	DBO	COA
f_1	7.48×10^{-2}	8.50×10^{-2}	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$
f_3	$\mathbf{3.69 \times 10^{-11}}$	$\mathbf{2.03 \times 10^{-7}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$
f_4	$\mathbf{1.17 \times 10^{-4}}$	$\mathbf{5.09 \times 10^{-6}}$	$\mathbf{6.12 \times 10^{-10}}$	$\mathbf{3.02 \times 1^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$
f_7	$\mathbf{1.46 \times 10^{-10}}$	$\mathbf{5.97 \times 10^{-5}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$
f_8	$\mathbf{6.67 \times 10^{-11}}$	$\mathbf{5.40 \times 10^{-4}}$	$\mathbf{1.69 \times 10^{-9}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.33 \times 10^{-11}}$	$\mathbf{3.33 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$
f_9	$\mathbf{1.05 \times 10^{-9}}$	$\mathbf{7.16 \times 10^{-9}}$	$\mathbf{2.74 \times 10^{-11}}$	$\mathbf{2.74 \times 10^{-11}}$	$\mathbf{2.74 \times 10^{-11}}$	$\mathbf{2.74 \times 10^{-11}}$	$\mathbf{2.74 \times 10^{-11}}$
f_{13}	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{4.08 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$
f_{14}	$\mathbf{3.34 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$
f_{15}	$\mathbf{6.70 \times 10^{-11}}$	$\mathbf{2.23 \times 10^{-9}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.34 \times 10^{-11}}$
f_{16}	$\mathbf{4.69 \times 10^{-8}}$	$\mathbf{1.87 \times 10^{-5}}$	$\mathbf{9.26 \times 10^{-9}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{8.99 \times 10^{-11}}$	$\mathbf{4.69 \times 10^{-8}}$	$\mathbf{6.07 \times 10^{-11}}$
f_{21}	$\mathbf{1.34 \times 10^{-5}}$	6.79×10^{-2}	$\mathbf{3.09 \times 10^{-6}}$	$\mathbf{1.63 \times 10^{-2}}$	$\mathbf{2.00 \times 10^{-5}}$	1.86×10^{-1}	$\mathbf{2.20 \times 10^{-7}}$
f_{23}	$\mathbf{4.98 \times 10^{-4}}$	$\mathbf{1.68 \times 10^{-3}}$	$\mathbf{2.19 \times 10^{-8}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{3.02 \times 10^{-11}}$	$\mathbf{1.09 \times 10^{-10}}$	$\mathbf{3.02 \times 10^{-11}}$

4.2.3. CEC2017 Convergence Curve of the Test Function Set

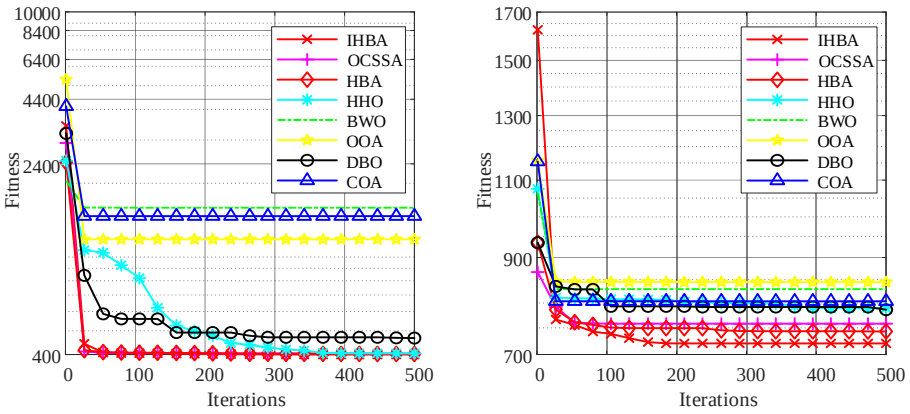
To verify the convergence performance of the improved algorithm and to visually compare the convergence speeds of different algorithms, we compared the convergence curves of swarm intelligence algorithms when solving function extremum problems.

Observing the aforementioned convergence curve, it can be concluded that the IHBA exhibits the highest optimization accuracy across all graphs. In Figure 5a,e,h, the IHBA did not converge as quickly as other algorithms during the initial stages of iteration. However, as the number of iterations increased after 50 iterations had been completed, the IHBA escaped from the local optimum and quickly converged to a better optimization result, demonstrating that the improved algorithm can escape from local optima. In Figure 5b–d,f,i–k, the IHBA quickly converged to the optimal value among all algorithms, and consistently maintained its lead throughout the iterative process, ultimately converging to the best optimization result. At the same time, it can be seen that the convergence curve of the MSDBO algorithm is relatively smooth, and it is less likely to fall into the situation of update stagnation. In Figure 5g,l, many algorithms quickly converge to local optima, but the HBA and IHBA converge near the optimum value, with the IHBA outperforming HBA.

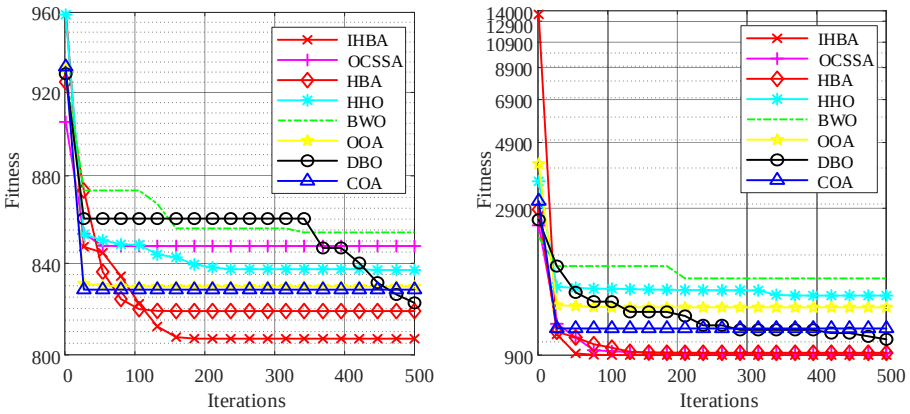


(**a**) $f_1(x)$ (**b**) $f_3(x)$

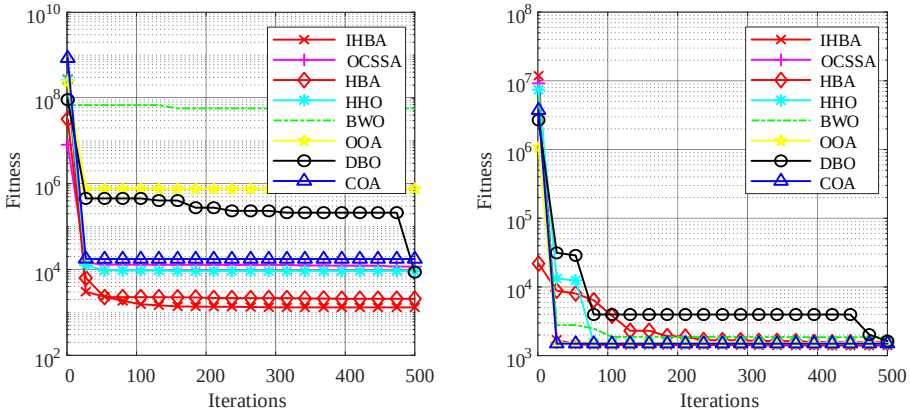
Figure 5. *Cont.*



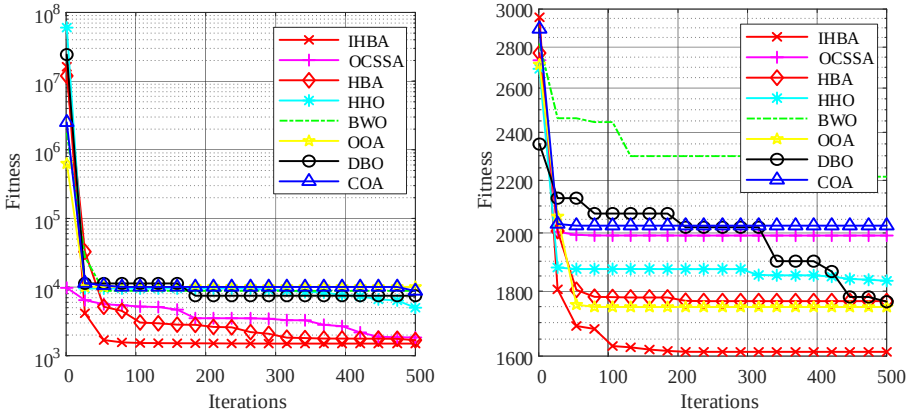
(**c**) $f_4(x)$ (**d**) $f_7(x)$



(**e**) $f_8(x)$ (**f**) $f_9(x)$

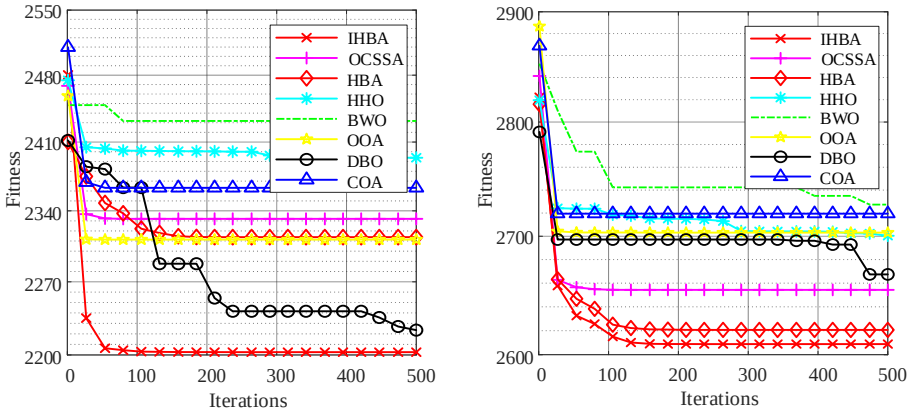


(**g**) $f_{13}(x)$ (**h**) $f_{14}(x)$



(**i**) $f_{15}(x)$ (**j**) $f_{16}(x)$

Figure 5. *Cont.*



(**k**) $f_{21}(x)$ (**l**) $f_{23}(x)$

Figure 5. Twelve classic test functions and convergence curves.

5. Engineering Optimization

To further demonstrate the effectiveness and engineering applicability of the IHBA, it is applied to the bearing fault diagnosis problem. The Variational Mode Decomposition (VMD) algorithm, an adaptive signal decomposition method, uses vital parameters like the penalty factor α and the number of modes K to enhance performance. The IHBA, a novel metaheuristic algorithm, was effectively applied to optimize the VMD parameters $[K, \alpha]$ for bearing fault diagnosis, quickly achieving notable results compared to other optimization techniques. Ultimately, VMD was integrated with convolutional neural networks and bidirectional long short-term memory (CNN-BILSTM) to facilitate the diagnosis of bearing faults.

5.1. Optimizing the Parameters of VMD

Different optimization algorithms are used to optimize the VMD parameters, and the adaptability curves are shown in Figure 6. By using the IHBA to optimize the key parameters of VMD, the signal analysis performance can be significantly enhanced. Especially when dealing with complex fault vibration signals, the algorithm in this paper can improve the decomposition quality, reduce the computational complexity, and obtain better accuracy and adaptability.

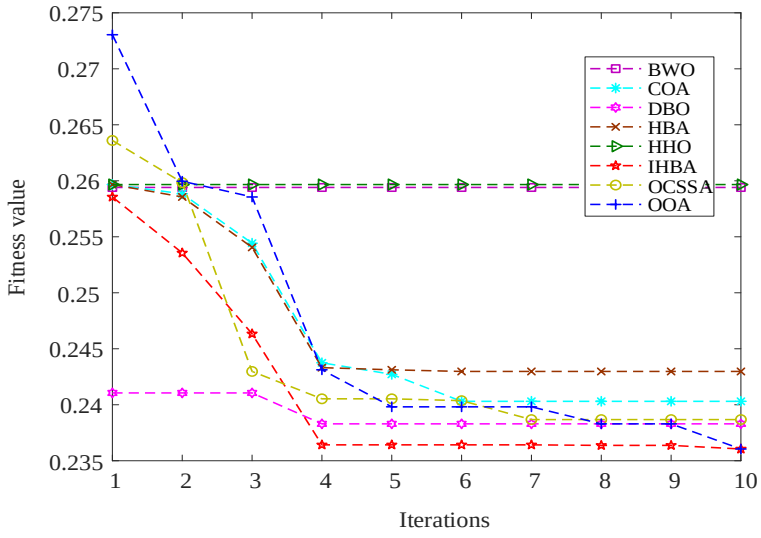
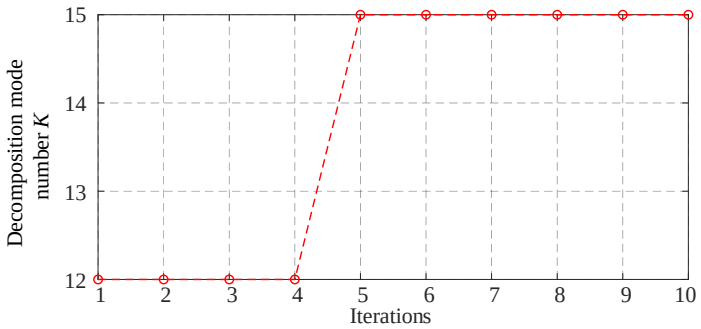
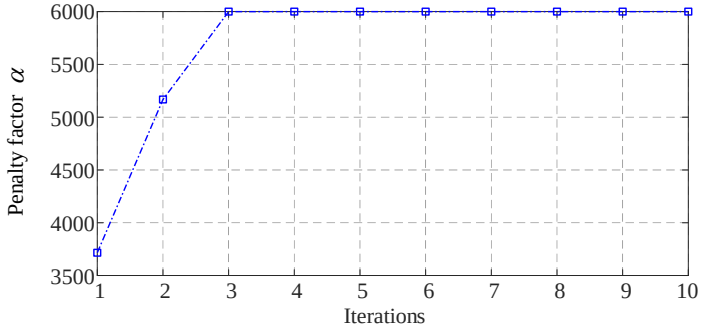


Figure 6. Fitness curves for VMD parameter optimization.

Figure 7 illustrates the convergence curve and parameter variation curve obtained during the optimization of the VMD parameters by the IHBA. As depicted in Figure 8, the convergence process shows that the algorithm tends to stabilize after four iterations, indicating convergence. At this point, the fitness function (minimum permutation entropy) value is 0.2364, with the optimal parameter combinations for the number of decomposition modes (K) and the penalty factor (α) being (15, 6000).

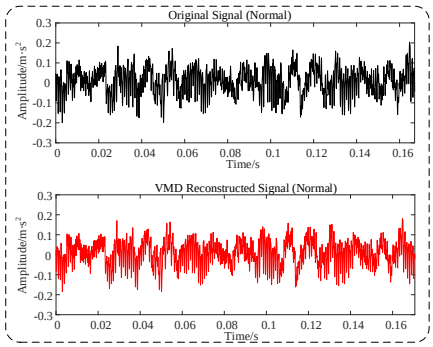


(**a**) Optimization process curve of K

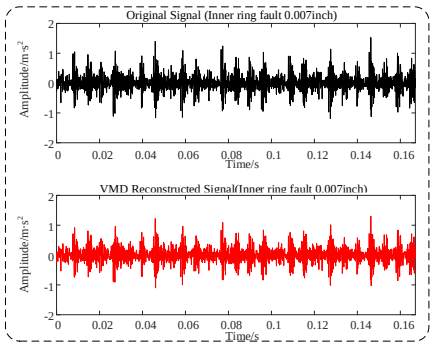


(**b**) Optimization curve of penalty factor α

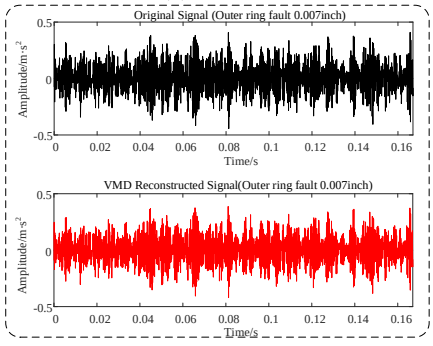
Figure 7. Optimization process curve of outer ring faulty bearing 0.007.



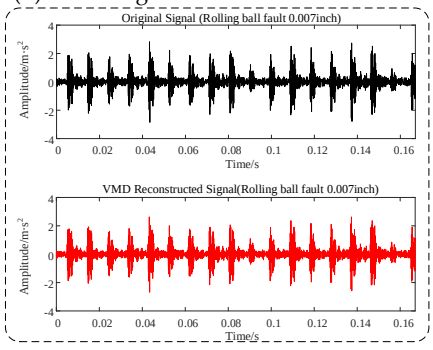
(**a**) Normal



(**b**) Inner ring fault diameter 0.007 inches



(**c**) Outer ring fault diameter of 0.007 inches



(**d**) Rolling ball fault diameter of 0.007 inches

Figure 8. Time-domain comparison between the original and the reconstructed signals of different faults.

Figure 8a–d shows the time-domain comparison diagrams before and after signal decomposition and reconstruction for a normal bearing and inner ring, outer ring, and rolling body bearing faults, respectively. These figures demonstrate that after optimized VMD reconstruction, the time-domain signals exhibit significant noise reduction, and the impact characteristics of the fault signals become more pronounced.

5.2. Rolling Bearing Fault Diagnosis

The experimental data used in this study were sourced from the Case Western Reserve University (CWRU) database on motor bearing failures [18]. As shown in Figure 9, a test bench was employed to simulate bearing failures, with a vibration signal sampling frequency of 1024 Hz. The experiment utilized a deep groove ball bearing model SKF6205, with failures simulated via electrical discharge machining (EDM). For diagnostic purposes, 10 fault categories were established, encompassing four types of states as follows: normal bearing, rolling element fault, outer ring fault, and inner ring fault. Each fault state was further categorized based on fault diameters of 0.007 inches, 0.014 inches, and 0.021 inches, resulting in nine distinct states representing varying degrees of failure. The dataset for each fault type consisted of 120 samples, with each sample containing 2048 data points. In the experimental setup, 90 samples were allocated for model training, while 30 samples were reserved for model testing. The division of the fault dataset is detailed in Table 6.

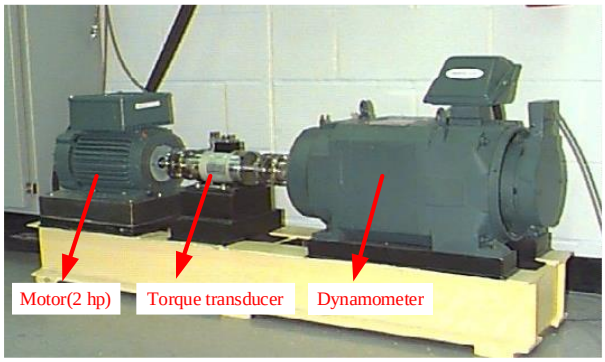


Figure 9. Bearing fault simulation test bench.

Table 6. Division results of the fault dataset.

Fault Type	Normal	Inner Ring Fault				Rolling Ball Fault			Outer Ring Fault	
Fault label	1	2	3	4	5	6	7	8	9	10
Fault diameter (d/inch)	-	0.007	0.014	0.021	0.007	0.014	0.021	0.007	0.014	0.021
Training sample	90	90	90	90	90	90	90	90	90	90
Test sample	30	30	30	30	30	30	30	30	30	30
Sample data size	2048	2048	2048	2048	2048	2048	2048	2048	2048	2048

Determining the relevant parameters is crucial in the VMD of the original vibration signals of rolling bearings. Specifically, the number of modal components K and the penalty parameter α significantly influence the decomposition results. The value of K dictates the number of decomposed modal components. If K is set too low, under-decomposition occurs, leading to a loss of key information and an inability to extract essential features. Conversely, if K is set too high, it results in an excessive number of modal components, causing overlapping center frequencies and making signal feature differentiation difficult. The penalty parameter α primarily affects the bandwidth of each modal component, with an appropriate value enhancing the accuracy of the reconstructed signal. In this paper, we propose an approach that optimizes VMD using an IHBA, combining it with a CNN-BiLSTM architecture for the fault diagnosis of motor bearings. This method effectively

extracts fault features from the original vibration signals. After the feature extraction process, each of the 10 states is represented by 120 samples, resulting in a 1200 × 9 matrix. Each row of this matrix is labeled with numbers 1–10 to indicate different fault types. The fault diagnosis process, as implemented in this study, is depicted in Figure 10.

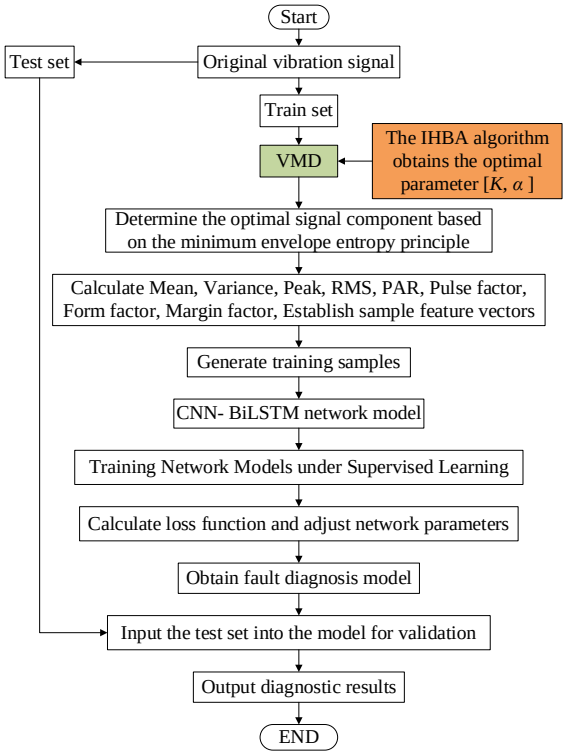


Figure 10. Based on the IHBA-VMD-CNN-BiLSTM rolling bearing failure prediction flowchart.

Figure 11 illustrates the accuracy curve of a fault diagnosis model based on IHBA-VMD-CNN-BiLSTM. The model performs excellently during both the training and testing phases, stabilizing after approximately 200 iterations and achieving a testing accuracy of up to 99.67% at 400 iterations.

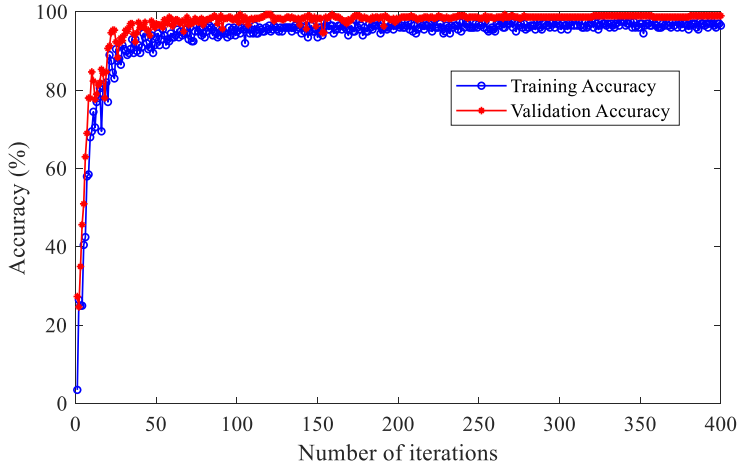


Figure 11. Accuracy curve of the fault diagnosis model based on IHBA-VMD-CNN-BiLSTM.

Figure 12 shows the loss function curve of a fault diagnosis model based on IHBA-VMD-CNN-BiLSTM. The model demonstrates excellent performance, with both training and testing losses rapidly decreasing and stabilizing at low values after around 50 iterations.

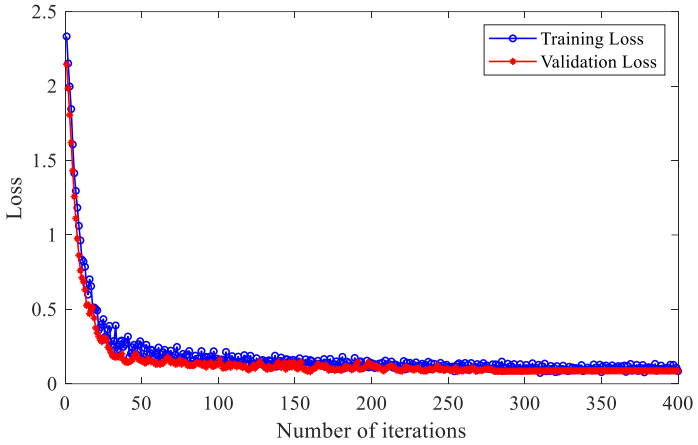


Figure 12. Loss function curve of the fault diagnosis model based on IHBA-VMD-CNN-BiLSTM.

To evaluate the performance of IHBA-VMD-CNN-BiLSTM, the proposed model is compared with other state-of-the-art (SOTA) models [19]. The confusion matrix and classification results for fault diagnosis are presented in Figure 13. After repeated validation, the IHBA-VMD-CNN-BiLSTM model demonstrates superior performance, achieving a recognition accuracy of 96.7% for normal faults, and 100% for all other fault types. The overall recognition accuracy for the 10 types of faults reaches an impressive 99.34%. Furthermore, the recognition speed of the IHBA-VMD-CNN-BiLSTM model is significantly improved, demonstrating its effectiveness in fault diagnosis.

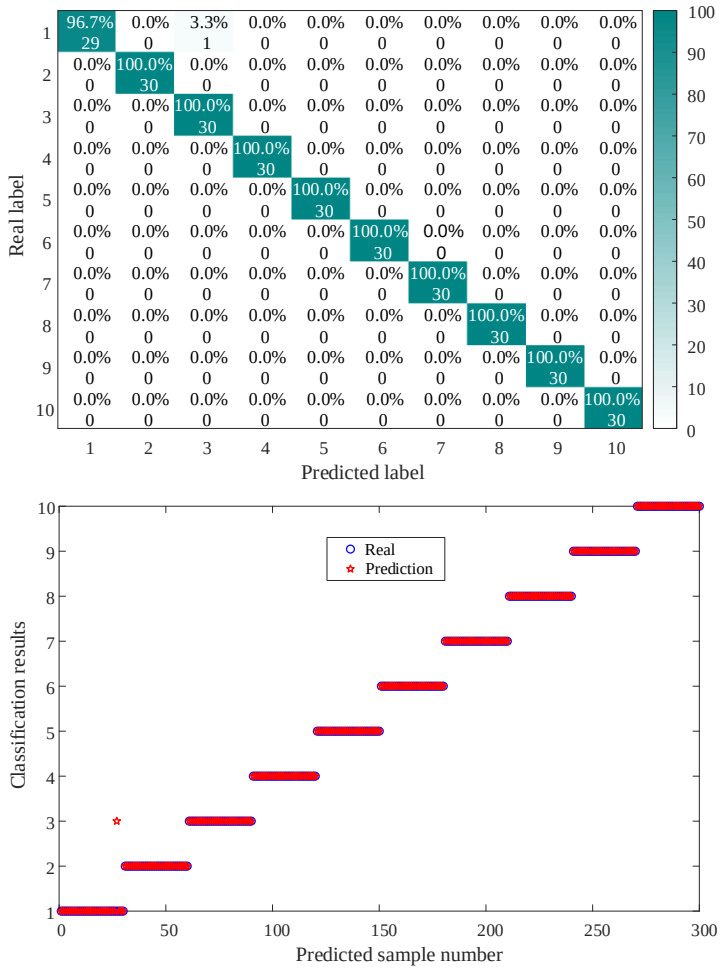
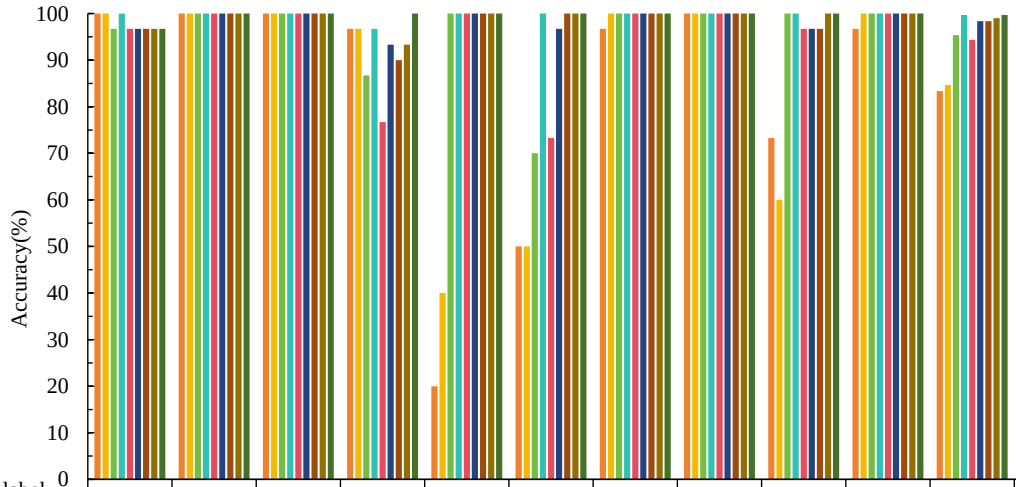


Figure 13. The accuracy of the test results for the eight methods.

Figure 14 shows the diagnostic accuracy of different experimental methods using the CWRU dataset. The figure presents the accuracy and average processing time of eight different fault diagnosis models applied to 10 types of faults. The IHBA-VMD-CNN-BiLSTM model stands out as the most effective method for fault diagnosis, achieving 100% accuracy for all faults except one, with an overall accuracy of 99.67%. It maintains a very short processing time of 3.34 s, second only to CNN-SVM, which has significantly lower accuracy. Its consistently high accuracy across different fault types demonstrates its robustness. The combination of optimization (IHBA) and advanced feature extraction (VMD) significantly enhances the performance of the CNN-BiLSTM model, making it the best choice for practical applications requiring high accuracy and efficiency in fault diagnosis.



	1	2	3	4	5	6	7	8	9	10	Avg.	Time(s)
CNN-LSTM	100	100	100	96.7	20	50	96.7	100	73.3	96.7	83.34	98.52
CNN-BiLSTM	100	100	100	96.7	40	50	100	100	60	100	84.67	148.37
VMD-CNN-BiLSTM	96.7	100	100	86.7	100	70	100	100	100	100	95.34	6.47
OCSSA-VMD-CNN-BiLSTM	100	100	100	96.7	100	100	100	100	100	100	99.67	6.67
BiGRU	96.7	100	100	76.7	100	73.3	100	100	96.7	100	94.34	21.08
CNN-BiLSTM-Attention	96.7	100	100	93.3	100	96.7	100	100	96.7	100	98.34	8.59
CNN-BiGRU-Attention	96.7	100	100	90	100	100	100	100	96.7	100	98.34	33.86
CNN-SVM	96.7	100	100	93.3	100	100	100	100	100	100	99	5.7
IHBA-VMD-CNN-BiLSTM	96.7	100	100	100	100	100	100	100	100	100	99.67	3.47

Figure 14. Diagnostic accuracy of different experimental methods.

6. Conclusions

To address the limitations of the HBA, including poor exploitation, susceptibility to local optima, and insufficient pre-exploration, this study introduces the following key enhancements: cubic chaotic mapping for population diversity, random value perturbation, and repeated random searches to prevent premature convergence and strengthen global optimization. Additionally, an elite tangent search with differential variation accelerates convergence and improves accuracy by leveraging optimal solution information. Simulation and bearing fault diagnosis experiments validate the proposed IHBA's robustness, stability, and practical applicability, demonstrating its effectiveness in enhancing decomposition quality, reducing computational complexity, and achieving superior precision and adaptability in complex signal analysis tasks. Additionally, experiments on bearing fault diagnosis confirmed the practical applicability of the IHBA in engineering. In summary, the IHBA has performed excellently in enhancing initial population traversal and improving global optimization capabilities. However, its limitations, such as sensitivity to algorithm parameter settings, high computational complexity, need for improvement in convergence

speed, and lack of universal validation, also require attention. Future research directions can explore and improve these limitations in depth to further enhance the performance and application scope of the IHBA.

Author Contributions: Conceptualization, H.T.; methodology, C.Y.; software, C.Y.; validation, C.P.; formal analysis, C.Y.; investigation, H.T.; resources, H.T.; data curation, C.P.; writing—original draft preparation, C.Y.; writing—review and editing, H.T.; supervision, project administration, funding acquisition, H.T. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (51967012), the Gansu Natural Science Foundation (23JRRA1664), and the Innovative Ability Enhancement Project of Gansu Provincial Higher Education (2023A-199), Gansu Province Longyuan Youth Innovation Talent Team Project (310100296012).

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author(s).

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

References

1. Alonso-González, M.; Díaz, V.G.; Pérez, B.L.; G-Bustelo, B.C.P.; Anzola, J.P. Bearing Fault Diagnosis With Envelope Analysis and Machine Learning Approaches Using CWRU Dataset. *IEEE Access* **2023**, *11*, 57796–57805. [CrossRef]
2. Ke, Z.; Di, C.; Bao, X. Adaptive Suppression of Mode Mixing in CEEMD Based on Genetic Algorithm for Motor Bearing Fault Diagnosis. *IEEE Trans. Magn.* **2022**, *58*, 1–6. [CrossRef]
3. Xue, J.K.; Shen, B. A novel swarm intelligence optimization approach: Sparrow search algorithm. *Syst. Sci. Control Eng.* **2020**, *8*, 22–34. [CrossRef]
4. Huang, K.W.; Wu, Z.X.; Jiang, C.L.; Huang, Z.H.; Lee, S.H. WPO: A Whale Particle Optimization Algorithm. *Int. J. Comput. Intell. Syst.* **2023**, *16*, 1–16. [CrossRef]
5. Trojovsky, P.; Dehghani, M. Pelican Optimization Algorithm: A Novel Nature-Inspired Algorithm for Engineering Applications. *Sensor* **2022**, *22*, 855. [CrossRef]
6. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
7. Heidari, A.; Mirjalili, S.; Faris, H.; Mafarja, M. Harris hawks optimization: Algorithm and applications. *Future Gener. Comput. Syst.* **2019**, *97*, 849–872. [CrossRef]
8. Hashim, F.A.; Houssein, E.H.; Talpur, K.; Mabrouk, M.; Al-Atabany, W. Honey badger algorithm: New metaheuristic algorithm for solving optimization problems. *Math. Comput. Simul.* **2022**, *192*, 84–110. [CrossRef]
9. Abasi, A.K.; Aloqaily, M. Optimization of CNN using modified honey badger algorithm for sleep apnea detection. *Expert Syst. Appl.* **2023**, *229*, 120484. [CrossRef]
10. Zhou, C.; Gao, B.; Yang, H.; Zhang, X.; Liu, J.; Li, L. Junction temperature prediction of insulated-gate bipolar transistors in wind power systems based on an improved honey badger algorithm. *Energies* **2022**, *15*, 7366. [CrossRef]
11. Düzenli, T.; Onay, F.K.; Aydemir, S.B. Improved honey badger algorithms for parameter extraction in photovoltaic models. *Optik* **2022**, *268*, 1169731. [CrossRef]
12. Xue, J.; Shen, B. Dung beetle optimizer: A new meta-heuristic algorithm for global optimization. *J. Supercomput.* **2023**, *79*, 7305–7336. [CrossRef]
13. Mohammad, D.; Zeinab, M.; Eva, T.; Pavel, T. Coati Optimization Algorithm: A new bio-inspired metaheuristic algorithm for solving optimization problems. *Knowl.-Based Syst.* **2023**, *259*, 110011.
14. Dehghani, M.; TrojovskýSec, P. Osprey optimization algorithm: A new bio-inspired metaheuristic algorithm for solving engineering optimization problems. *Engine Automot. Eng.* **2022**, *8*, 1126450. [CrossRef]
15. Zhong, C.; Li, G.; Meng, Z. Beluga whale optimization: A novel nature-inspired metaheuristic algorithm. *Knowl.-Based Syst.* **2022**, *251*, 109215. [CrossRef]
16. Yong, C.; Ting, H.; Peng, C. Enhancing sparrow search algorithm with OCSSA: Integrating osprey optimization and Cauchy mutation for improved convergence and precision. *Electron. Lett.* **2024**, *60*, e13127. [CrossRef]
17. Layeb, A. Tangent search algorithm for solving optimization problems. *Neural Comput. Appl.* **2022**, *34*, 8853–8884. [CrossRef]

18. Available online: https://engineering.case.edu/bearingdatacenter (accessed on 1 October 2024).
19. Yong, C.; Guangqing, B. Enhancing Rolling Bearing Fault Diagnosis in Motors using the OCSSA-VMD-CNN-BiLSTM Model: A Novel Approach for Fast and Accurate Identification. *IEEE Access* **2024**, *12*, 78463–78479.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.