

Article

Improved Hybrid Firefly Algorithm with Probability Attraction Model

Jin-Ling Bei ^{1,†}, Ming-Xin Zhang ^{2,†}, Ji-Quan Wang ^{1,*} , Hao-Hao Song ¹ and Hong-Yu Zhang ¹¹ College of Engineering, Northeast Agricultural University, Harbin 150030, China² Computer Department, Shijiazhuang Posts and Telecommunications Technical College, Shijiazhuang 050020, China

* Correspondence: wangjiquan@neau.edu.cn

† These authors contributed equally to this work.

Abstract: An improved hybrid firefly algorithm with probability attraction model (IHFAPA) is proposed to solve the problems of low computational efficiency and low computational accuracy in solving complex optimization problems. First, the method of square-root sequence was used to generate the initial population, so that the initial population had better population diversity. Second, an adaptive probabilistic attraction model is proposed to attract fireflies according to the brightness level of fireflies, which can minimize the brightness comparison times of the algorithm and moderate the attraction times of the algorithm. Thirdly, a new location update method is proposed, which not only overcomes the deficiency in that the relative attraction of two fireflies is close to 0 when the distance is long but also overcomes the deficiency that the relative attraction of two fireflies is close to infinity when the distance is small. In addition, a combinatorial variational operator based on selection probability is proposed to improve the exploration and exploitation ability of the firefly algorithm (FA). Later, a similarity removal operation is added to maintain the diversity of the population. Finally, experiments using CEC 2017 constrained optimization problems and four practical problems in engineering show that IHFAPA can effectively improve the quality of solutions.

Keywords: improved hybrid firefly algorithm; probability attraction model; constrained optimization problem; remove similarity operation; combined mutation

MSC: 6804



Citation: Bei, J.-L.; Zhang, M.-X.; Wang, J.-Q.; Song, H.-H.; Zhang, H.-Y. Improved Hybrid Firefly Algorithm with Probability Attraction Model. *Mathematics* **2023**, *11*, 389. <https://doi.org/10.3390/math11020389>

Academic Editor: Ioannis E. Livieris

Received: 9 December 2022

Revised: 5 January 2023

Accepted: 9 January 2023

Published: 11 January 2023



Copyright: © 2023 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

Optimization problems widely exist in various fields of life. Traditional solving methods, such as the Newton method, conjugate gradient method and simplex method, need to traverse the entire search space, resulting in a combination explosion of search, that is, the search cannot be completed in polynomial time. In view of the complexity, constraint, nonlinearity and modelling difficulties in practical engineering problems, the research on meta-heuristic algorithms is particularly important. A meta-heuristic algorithm has the following advantages in solving complex engineering problems: (1) simple principle, fewer parameters to be adjusted and easy to implement; (2) do not need the derivative information of objective function; (3) differentiability and convexity are not required; (4) widely used to solve problems difficult to define with mathematical models or complex optimization problems; (5) strong, robust, versatile and suitable for parallel processing. Compared with traditional optimization methods, the probability of successfully escaping from the local optimal region is higher when solving complex problems, such as multi-extremum optimization. Therefore, meta-heuristic algorithms have great advantages in solving complex optimization problems and have been widely used in many fields.

Meta-heuristic algorithm includes the evolutionary algorithm as a key component. Evolutionary algorithms are primarily used to identify the best answer to optimization

problems through the use of selection, crossover, mutation and other processes. The most well-known evolutionary algorithms are the genetic algorithm (GA) [1] and the differential evolution algorithm (DE) [2]. Population intelligence algorithms, as another important part of metaheuristic algorithms, update the position of individuals by simulating the behavior of animals, such as foraging, hunting and finding mates, to find the optimal solution of the optimization problem. In the past few decades, people have proposed many intelligence optimization algorithms, such as particle swarm optimization (PSO) [3], firefly algorithm (FA) [4], artificial bee colony algorithm (ABC) [5] and other swarm intelligence optimization algorithms. In 2008, Yang et al. [6] were inspired by fireflies' luminous characteristics to attract mates and avoid natural enemies at night and proposed an FA. FA has the advantages of simple principles, few parameters, easy implementation, high precision and fast convergence. Therefore, the firefly algorithm is implemented to solve various problems in different domains, such as neural networks [7–9], scheduling problems [10–12], image processing [9,13], wireless parameter optimization of sensor networks [14–16] and big data processing [17,18].

Numerous academics have worked tirelessly over the past ten years to boost the FA's effectiveness, with some promising study findings. In the FA, the attraction model greatly impacts the computational TC, convergence speed and solution quality in the algorithm. The more times of attraction in the attraction model, the faster the convergence speed of the algorithm and the worse the diversity of the population. The higher the TC, the easier it is to fall into a local optimum. In addition, the comparison times of fitness are different for different attraction models. The more fitness comparisons, the higher the time complexity (TC) in the algorithm. The attraction model used by the standard firefly algorithm (SFA) [6] is a complete attraction model (CAM). SFA can have an excess of attraction because each firefly can be attracted to another, with brighter fireflies in the population. Too much attraction can lead to oscillations or fluctuations in the algorithm search process, resulting in a poor quality algorithm solution. In 2016, Wang et al. proposed a firefly algorithm with random attraction (RaFA) [19]. RaFA uses a random attraction model (RAM) in which each firefly in the population is compared with another firefly chosen at random. If the brightness of the randomly selected firefly is higher than that of the current firefly, the current firefly is attracted by the randomly selected firefly; otherwise, the current firefly is not attracted. The RAM reduces the TC of the algorithm due to its smaller number of attractions, but also slows down the algorithm's convergence due to the smaller number of attractions. In 2017, Wang et al. proposed a neighborhood attraction firefly algorithm (NaFA) [4], which uses a neighborhood attraction model (NAM), where brighter fireflies in the k -neighborhood are attracted to the current firefly. Although the NAM has a relatively moderate number of attractions, which has a greater possibility of reducing the premature convergence phenomenon, too many comparisons of too much brightness will lead to too much TC in the algorithm. In 2021, Cheng et al. proposed the hybrid firefly algorithm for group attraction (GAHFA) [20], which uses the grouping attraction model (GAM). GAHFA population size was even n , and all fireflies in the population were sorted and grouped according to the OFV, consisting of $n/2$ groups. Fireflies with high brightness in each group attract fireflies with low brightness, and fireflies with the highest brightness in the population attract fireflies with high brightness in each group. Although GAHFA has moderate attraction times and low TC in the algorithm, the attraction relationship between fireflies in the GAM is fixed, and it is easy to become stuck in a local optimum. To solve this problem, a new firefly attraction model, probabilistic attraction model, is proposed.

Use the example of an optimization problem where the goal function is the minimum value to demonstrate the issue. First, all fireflies are sorted according to fitness value from small to large. After sorting, the first firefly is the brightest firefly and the best firefly. Second, the first firefly in the population moves randomly. Third, starting from the second firefly, calculate the selection probability of each firefly. According to the selection probability of fireflies, select a brighter firefly from the fireflies in front of the current firefly. Fireflies are attracted to the brighter fireflies selected and so on. The probability attraction model can

decrease the quantity of attraction times and brightness comparisons while preventing a situation in which there are insufficient numbers of both. In addition, the fireflies selected by the probability attraction model are random, which will reduce the possibility of the algorithm being trapped in local optimum. At the same time, the brightness of the firefly chosen by the probability attraction model is higher, which overcomes the drawback that the firefly chosen by the RAM cannot attract the present firefly.

The position-updating formula of the FA in the existing literature may have a phenomenon that the relative attractiveness tends to zero at the beginning of the iteration. The method conducts a random search because the relative attractiveness is 0, which causes a slow convergence of the algorithm and poor solution quality. In response to this problem, a formula for updating the position of the firefly with adaptive changes in relative attractiveness is presented. The relative attractiveness of adaptive changes will not approach zero in the iterative process. In addition, the location update formula not only considers the influence of high-brightness fireflies on the location-update fireflies but also considers the guiding effect of the best firefly in the population on the position-update fireflies.

In the later iteration of the algorithm, fireflies will gradually gather near the best fireflies, so the fitness difference among fireflies will be very small. In light of this circumstance, a formula is suggested to calculate the degree of similarity S among population members. The higher the similarity, the worse the diversity in the population and the lower the exploration ability of the algorithm; there is a high probability of getting trapped by a local optimum. The updating strategy of some individuals in the population is proposed to eliminate the similarity between individuals in the population. The update strategy is that, when S is less than the threshold, update some individuals in the population.

A combined mutation based on selection probability is proposed to dynamically select a single mutation operator with strong exploration and exploitation ability. This will further improve the solution quality and convergence speed of the algorithm and better balance the FA's exploration and exploitation ability. In the early stages of the algorithm's iteration, the combined mutation operator based on selection probability has a large chance of selecting a single mutation operator with strong global ability; in the later phases of the iteration, the probability of selecting a single mutation operator with strong local ability is high. First, according to the exploration and exploitation ability of each single mutation operator, multiple mutation operators are divided into two categories. The single mutation operator in the first category has a strong exploration ability, and the single mutation operator in the second category has a strong exploitation ability. Second, a formula is designed to calculate the selection probabilities of the two types of mutation operators based on improvements in the solutions to the optimization problems of the two types of mutation operators. A type of mutation operator has a bigger selection probability if it can enhance the quality of the problem's solution, whereas if it cannot, it has a reduced selection probability. Each mutation operator within each category is chosen at random.

The major contributions of this paper are:

1. According to the uniformity and diversity of the initial population generated by different initialization methods, the best method of population uniformity and diversity is selected as the population initialization method.
2. A probabilistic attraction model is proposed for the problems of various attraction models.
3. A firefly position update formula with an adaptive change in relative attraction is proposed to improve the convergence speed and solution quality of FA.
4. A combined mutation operator based on selection probability is proposed, which can adaptively select a single mutation operator with strong exploration ability and exploitation ability.
5. A remove similarity operation is added to the algorithm to enhance the exploration ability of the algorithm and maintain the diversity of the population.

6. The proposed IHFAPA is compared with other improved algorithms in the literature in parameter optimization, such as reducer and cantilever beam. IHFAPA is superior to them in solution quality.

The organization of the rest of this paper is: Section 2 reviews the existing firefly algorithms; in Section 3, an IHFAPA is proposed; in Section 4, the proposed IHFAPA algorithm is experimentally analyzed using the CEC 2017 test function set and the performance is compared with other algorithms; Section 5 utilizes the IHFAPA algorithm to solve four classical engineering optimization problems, which is then compared with other algorithms; Section 6 is the Conclusion.

2. Related Works

2.1. Firefly Algorithm

In 2008, Yang proposed an FA based on the luminous characteristics of fireflies and the principle of mutual attraction between individual fireflies [6]. In order to construct the FA, some characteristics of the firefly flash need to be idealized. The specific idealization criteria are as follows:

- (1) Fireflies are male and female, that is, fireflies are attracted to each other.
- (2) Attraction is proportional to the brightness of fireflies. For any two fireflies, the high-brightness fireflies attract the low-brightness fireflies and move toward them, and the brightest fireflies move randomly.
- (3) The brightness of fireflies is determined by the OFV of the problem to be optimized. Let n be the population size, the i -th firefly in the population is $X_i = (x_{i1}, x_{i2}, \dots, x_{iD})^T$ ($i = 1, 2, \dots, n$) and D is the dimension of the variable. The distance r_{ij} between any two fireflies i and firefly j in the population is:

$$r_{ij} = \|X_i - X_j\| = \sqrt{\sum_{k=1}^D (x_{ik} - x_{jk})^2} \quad (1)$$

where X_i and X_j are the i -th fireflies and j -th fireflies in the population, x_{ik} is the k component of the firefly i in the population and x_{jk} is the k component in the j -th firefly in the population.

The attraction of firefly i to firefly j $\beta_{ij}(r_{ij})$ is:

$$\beta_{ij}(r_{ij}) = \beta_0 e^{-\gamma r_{ij}^2} \quad (2)$$

where β_0 is the maximum attraction, that is, when the attraction $r_{ij} = 0$, usually $\beta_0 = 1$.

If the brightness of firefly j is higher than that of firefly i , firefly i is attracted by firefly j and moves toward firefly j . The location-update formula of firefly i is:

$$x_{ik}(t+1) = x_{ik}(t) + \beta_{ij} (x_{jk}(t) - x_{ik}(t)) + step * (r - 0.5) \quad (3)$$

where t is the number of iterations; $step$ is the step factor $step \in [0,1]$; r is a random number between $[0,1]$.

The pseudo code of the standard firefly algorithm is shown in Algorithm 1.

2.2. Brief Review of FA

FA has received wide attention from many scholars because of its simple concept, fewer parameter settings and easy implementation. In the past decade, the research results on FA improvement have mainly focused on the adaptive adjustment of parameters, the improvement in position update formula, the improvement in attraction model and the hybrid firefly algorithm.

Algorithm 1: Pseudo code of FA

Input: The population size of the population is n , and the dimension of the variable is D , T is maximum number of iterations;
Output: The final population;
 Randomly generate n initial fireflies;
 Calculate the fitness value of all initial fireflies;
 Parameter initialization, population size, maximum attraction and light attraction coefficient; let $t = 0$;
 While $t \leq T$ do
 $t \leftarrow t + 1$;
 for $i = 1$ to n do
 for $j = 1$ to n do
 If firefly j is brighter than firefly i then
 Generate a new firefly according to Equation (3);
 Evaluate the new solution;
 End if
 End for
 End for
 Rank the fireflies and find the current best;
 End While
 End

2.2.1. Adaptive Adjustment of Parameters

The performance of the FA is significantly influenced by the parameters in the position-update formula, and the adaptive adjustment parameters can significantly enhance the algorithm's performance. As a result, numerous researchers have examined parameter adaptive adjustments in the algorithm and proposed numerous adjustment methods. In 2012, Leandro et al. [21] proposed the light-absorption factor γ and step factor α adaptive adjustment firefly algorithm (CFA). CFA introduces Tinkerbell map in the light-absorption factor γ , which reduces the probability of the FA falling into a local optimum. The step size factor α in the CFA algorithm decreases linearly with the number of iterations, which better balances the exploration and exploitation ability of the algorithm. In 2013, Rizk-Allah et al. [22] proposed a method of non-linearly adjusting the step-size factor. The step-size factor decreases non-linearly with the increase in the number of iterations, which avoids disturbances in the optimal position of the firefly by the random term. Liang et al. [23] proposed an enhanced FA, which achieves a balance between exploration and exploitation ability by dynamically adjusting the step factor α . In the initial stage of the iteration, the value of α is large and the algorithm's exploration ability is strong; in the later stage of the iteration, the value of α is small and the algorithm's exploitation ability is strong. In 2017, Wang et al. [4] proved that when the number of iterations approached infinity and the limit of the step factor α was equal to 0, the algorithm converged, and they proposed a new method for dynamically adjusting the step factor α . The step factor α in this method decreases rapidly with the increase in iteration times and finally approaches zero, thus, ensuring the convergence of FA. In 2018, Banerjee et al. [24] proposed a firefly algorithm (PropFA) based on a new parameter-adjustment mechanism. In this mechanism, all parameters in the FA are dynamically adjusted according to the value of the objective function, striking a balance between exploration and exploitation. Experiments show that Prop FA performs better than other comparison algorithms. In 2019, Zhang et al. [25] proposed a dynamic adjustment in step-size factor α to improve the firefly algorithm. The results show that this method has better performance than other algorithms. In 2020, Amit et al. [26] proposed an improved firefly algorithm that considers the initial brightness β_0 of environmental factors. The initial brightness of the algorithm changes dynamically, thereby achieving a better balance between exploration and exploitation. However, the algorithm does not consider the situation where the attractiveness value β is 0 when the two

fireflies attracting each other in the population are far apart, which affects the performance of the algorithm.

2.2.2. Improved Location-Update Method

Location update is a crucial component in the FA and has a significant impact on how well the algorithm can optimize. The convergence speed and solution accuracy of the algorithm can be increased by optimizing the FA's location-updating procedure. In 2016, Wang [19] and others proposed an improved firefly algorithm that replaces full attraction with random attraction, which greatly reduces the number of attractions and reduces the TC in the algorithm. However, the RAM may cause slow convergence of the algorithm and poor solution quality due to too few attraction times. In 2018, Zhan et al. [27] proposed an improved firefly algorithm, which introduced accelerated attractiveness and evading strategies into the location-update formula of the firefly algorithm. Accelerated attractiveness makes the current firefly converge in the vicinity of the optimal firefly faster, and the evading strategy keeps the current firefly away from the low-brightness firefly. The two operations can improve the convergence speed of the algorithm and reduce the probability of the algorithm falling into local optimum. In 2019, Wang et al. [28] proposed a firefly algorithm based on gender differences and designed two position-update formulas for fireflies of different genders. Male fireflies are attracted to two randomly selected females for global search; the female fireflies move to the vicinity of the best male fireflies for local search, which better balances the exploration and exploitation ability of the algorithm. The problem with the gravity calculation formula of this algorithm is that when the two fireflies are far apart, the value of relative attraction approaches 0, which causes the algorithm's update formula to not work. In 2020, Wu et al. [29] proposed an adaptive logarithmic spiral-Levy FA (ADIFA). ADIFA designed two position-update formulas: the first position update formula introduced Levy flight, which improves the exploration ability of the algorithm; the second position-update formula introduces the logarithmic spiral path, which improves the exploitation capabilities of the algorithm. In addition, an adaptive switch is also designed to realize the algorithm's adaptive switching between exploration and exploitation modes. Experimental results show that the ADIFA algorithm is much better than the other three firefly algorithms.

2.2.3. Improvement in Attraction Model

The attraction model is an important part of the FA, which has a great impact on its performance. In 2008, the standard FA proposed by Yang et al. [4] used the CAM, in which each iteration can attract any brighter firefly in the population. In 2016, Wang et al. [19] proposed a random-attracting firefly algorithm to solve the problem that the algorithm too easily falls into local optimum due to too many times of attraction in the CAM. In each iteration of the random-attraction firefly algorithm, the i -th firefly in the population ($i = 1, 2, \dots, n$, n is the population size) matches the randomly selected j -th ($j = 1, 2, \dots, n, i \neq j$) and compares the brightness of only fireflies. If the brightness of the j randomly selected firefly is higher than that of firefly i , firefly i will be attracted by firefly j . Otherwise, the i -th firefly will not be attracted by the j -th firefly. Compared with the CAM, the attraction times of the RAM are greatly reduced. Too few attraction times will cause the algorithm to converge slowly. In 2017, Wang et al. [30] proposed a firefly algorithm with neighborhood attraction. Each firefly in the population can be attracted by a brighter firefly in its k -neighborhood. The number of attractions in the NAM is relatively moderate, which can improve the convergence speed of the algorithm. Too much brightness comparison leads to high TC in the algorithm. In 2021, Cheng et al. [20] proposed a GAM. The GAM sorted the fireflies in the population according to the fitness value from small to large, and then the fireflies in the population are divided into $n/2$ groups. The fireflies with high brightness in each group attract the fireflies with low brightness. In addition, the best fireflies in the population also attracted bright fireflies in each group. Although the number of attraction times and

the number of brightness comparisons are moderate in the GAM, since each firefly in the population is attracted by a fixed firefly, the algorithm easily falls into a local optimum.

2.2.4. Hybrid Firefly Algorithm

A single intelligence optimization method is constrained by the program's structure or conditions that are associated with it and it easily falls into local optimum, producing subpar solutions. In order to fully exploit various intelligence optimization algorithms, the hybrid optimization algorithm combines two or more intelligence optimization algorithms or optimization ideas. Furthermore, the advantages of optimization ideas can improve the algorithm's performance. In 2013, Huang et al. [31] proposed a hybrid firefly algorithm combining local random search methods, which effectively improved the local search ability of the algorithm (HFA). In addition, HFA is applied to reduce the jacket loss of downhole transmission and good results are achieved. In 2016, Verma et al. [32] proposed a hybrid firefly algorithm based on reverse learning (ODFA). ODFA uses reverse learning to optimize the position of the initial solution and improve the quality of the initial population. In 2017, Dash et al. [33] proposed a hybrid meta-heuristic algorithm that mixes the firefly algorithm and the differential evolution algorithm. The algorithm takes full advantage of the strengths of the firefly algorithm and the differential evolution algorithm and is well balanced, with improved algorithm exploration and exploitation ability. In 2018, Aydilek et al. [34] proposed a hybrid algorithm combining the firefly algorithm and particle swarm algorithm (HFPSO). The algorithm makes use of the fast convergence of PSO for global search and the fine tuning of FA for the local search, to balance the relationship between algorithm exploration and exploitation. The experimental results of the two test function sets, CEC 2015 and CEC 2017, show that HFPSO is significantly better than other algorithms. In 2019, Li et al. [35] proposed a hybrid firefly algorithm that embeds the cross-entropy method into the firefly algorithm. The method uses adaptive smoothing and co-evolution to fully absorb the robustness, ergodicity and adaptability of the cross-entropy method. This hybrid algorithm enhances the global search ability of the algorithm, stops the algorithm from falling into local optimization and improves the convergence speed of the algorithm. In 2020, Wang et al. [36] proposed a hybrid firefly algorithm that introduces a learning strategy containing Cauchy mutation into the firefly algorithm. In each iteration, the best firefly must implement L learning strategies to better balance the relationship between exploration and exploitation. However, the value of L is too large, which greatly increases the TC in the algorithm and will cause the algorithm to fall into a local optimum.

3. Proposed Methods

3.1. Population Initialization

3.1.1. Initial Population Generation Method Based on Square-Root Sequence Method

The good point set theory was first proposed by Hua Luogeng and Wang Yuan in the book "The Application of Number Theory in Approximate Analysis" [37]. The point set generated based on this theory is evenly distributed in the unit space. Therefore, using the good point set theory can produce a uniformly distributed initial population in the search space, which can ensure the diversity of the initial population and reduce the possibility of the algorithm falling into local optimum. The specific steps of the initial population generation method based on the square-root sequence method are as follows:

Step 1: use the square-root sequence method to generate the first good point in the unit cube, that is, $r_1 = (r_1^1, r_1^2, \dots, r_1^D)$, where, $r_1^j = \{\sqrt{p_j}\}$, $j = 1, 2, \dots, D$. $p_1, p_2, \dots, p_D$ is D prime numbers from small to large; $\{\bullet\}$ represents the decimal part.

Step 2: based on r_1 , generate a good point set containing n good points according to Equation (4) $P_n = (r_1, r_2, \dots, r_n)$.

$$r_i^j = \left\{ \sqrt{p_j} \times i \right\}, i = 1, 2, \dots, n; j = 1, 2, \dots, D \quad (4)$$

where n is the population size and p_j is the j -th prime among D prime numbers that are not equal from small to large. D is the dimension of the problem and $\{\bullet\}$ is the decimal part.

Step 3: map the good point set generated by the square-root sequence method to the search space to obtain the initial population X of the population. The method of mapping the good point set to the search space is:

$$X_i = Lb + (Ub - Lb) \otimes r_i \quad (5)$$

where X_i is the i -th individual in the population, Ub and Lb are the upper and lower bounds of the search space, r_i is the i -th good point in the good point set P_n and $\otimes$ is the product of the corresponding elements of the two vectors.

The method of initial population generation is shown in Algorithm 2.

Algorithm 2: Initial population generated based on the square-root sequence good point set method

Input: The population size of the population is n , and the dimension of the variable is D ;

Output: The initial population of the population;

Produce the first good point r_1 :

Generate a good point set according to Equation (4) P_n

for $I = 1$ to n do

 The individual generating the initial population according to Equation (5) X_i :

end for

3.1.2. Compare Different Population Initialization Methods

The commonly used population initialization methods include random initialization method, population initialization method based on chaotic mapping, population initialization method based on reverse learning and population initialization method based on good point set theory. The random initialization method is simple in principle and easy to implement. Many references [19,38] use this method to generate the initial population and achieve good results. Chaotic mapping has the characteristics of randomness, ergodicity and regularity. Many scholars often use chaotic mapping to generate an initial population. In 2005, Yu et al. [39] took the lead in using chaotic mapping to generate an initialization population in GA, which effectively improved the optimization accuracy of the algorithm. Since then, many scholars have used chaotic mapping to generate initialization populations [40–42]. There are more than ten kinds of chaotic mappings. The chaotic mappings commonly used to generate the initial population are logistic mapping and tent mapping. The initial population generated by logistic mapping is mainly distributed at both ends of the search space, and the middle region less so. If the optimal solution is in the middle region of the search space, the convergence speed of the algorithm will be reduced. Tent mapping has better ergodicity and uniformity than logistic mapping, so tent mapping is often used to generate the initial population. In 2005, Tizzoosh et al. [41] proved that both the current solution and the elite individual have a 50% probability that they are far from the optimal solution compared to the reverse solution. In 2008, Rahnamayan et al. [40] used the population initialization method based on reverse learning to generate the initial population and achieved good results. Good point set theory is a point set generation method proposed by Professor Luogeng Hua [37]. It is mathematically proved that the point sets generated by good point set theory are evenly distributed in unit space. There are three major ways to generate point sets in the unit cube: exponential sequence method, circular domain method and square-root sequence method. Because the number of digits after the decimal point that can be displayed by the computer is limited, when the dimension is greater than 35, the decimal part of the number generated by the exponential sequence ek is 0, and the exponential sequence method is invalid. Therefore, the initial population can be generated using the split-circle domain method or the square-root sequence method.

The advantages and disadvantages of the initial population generated by different methods can be judged according to the uniformity of individual distribution in the search

space and the polyphase of the population. In order to clearly describe the uniformity of the initial population generated by different methods in the search space, taking a population size of 100 and a variable dimension of 2 as examples, the initial population is generated in the interval $[0,1]$ by using the random initialization method, the population initialization method based on tent mapping, the population initialization method based on reverse learning and the population initialization method based on the square-root sequence method. The initial population generated by different initialization methods is shown in Figure 1.

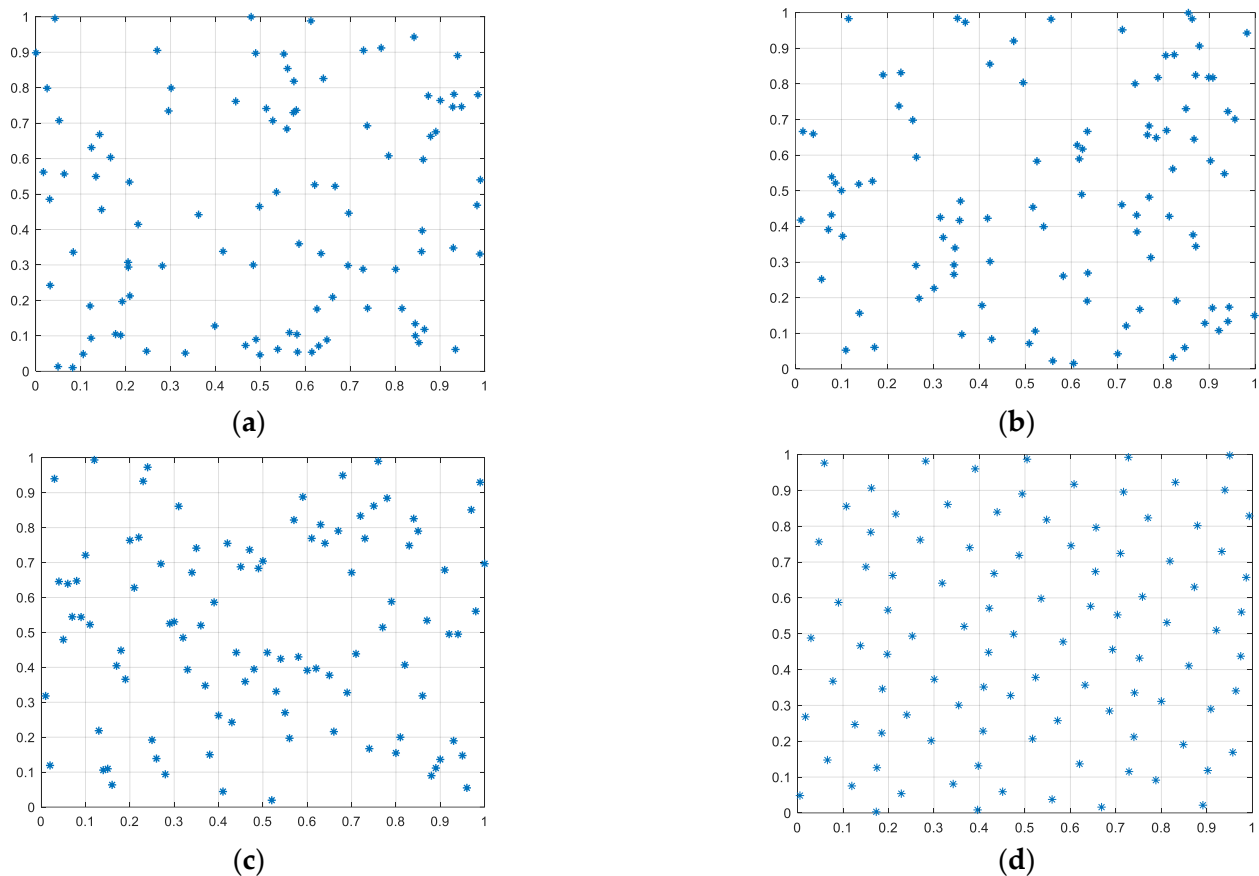


Figure 1. Schematic diagram of initial population generated by different methods. (a) Random initialization method; (b) initialization method based on *Tent* mapping; (c) initialization method based on reverse learning; (d) initialization method based on square-root sequence.

From Figure 1, the initial population generated by the random initialization method, the population initialization method based on tent mapping and the population initialization method based on reverse learning have the phenomenon of uneven distribution and aggregation. If the optimal solution is far away from the aggregation region, the convergence speed of the algorithm will be slow and it easily falls into a local optimum. The initial population generated by the population initialization method based on square-root sequence is dispersed in the whole search space and there is no aggregation phenomenon. Therefore, the initial population is generated based on square-root sequence.

The variety in the population has a significant effect on how well the algorithm performs. Good population variety makes it difficult for the algorithm to fall into premature convergence. A population diversity computation method is provided in order to quan-

tatively compare the diversity in the starting population created by various population initiation methods. The calculation method of population diversity is:

$$Diversity = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^n d(X_i, X_j) \quad (6)$$

where n is the population size, X_i is the i -th firefly, X_j is j -th firefly in the population and $d(X_i, X_j)$ represents the Euclidean distance between the i -th firefly and the j -th firefly. The calculation formula of $d(X_i, X_j)$ is as follows:

$$d(X_i, X_j) = \sqrt{\sum_{k=1}^D (x_{ik} - x_{jk})^2} \quad (7)$$

where D is the dimension of the variable, x_{ik} is the k -th component of the i -th firefly and x_{jk} is the k -th component of the j -th firefly.

In Equation (6), the value of *Diversity* is proportional to the distance between individual fireflies in the population. The greater the distance between individual fireflies, the greater the value of *Diversity*; the smaller the distance between individual fireflies, the smaller the value of *Diversity*. The greater the value of *Diversity*, the more uniform the distribution of individuals in the population, and the less likely it is the algorithm will fall into local optimum; the smaller the value of *Diversity*, the more likely it is that there is individual clustering in the population, and the easier it is for the algorithm to fall into local optimum.

According to Equation (4), the diversity in the initial population generated by the above four methods is calculated, and the calculation results are given in Table 1.

Table 1. Population diversity.

Population Initialization Method	Diversity
Random initialization method	0.48
Based on <i>Tent</i> chaotic mapping method	0.49
Reverse learning method	0.49
Good point set method based on square-root sequence	0.54

In Table 1, the population initialization method based on the square-root sequence method produces the best initial population diversity, followed by the population initialization method based on tent mapping and the population initialization method based on reverse learning, then for the initial population generated by the random initialization method, the population diversity in the population is the worst.

In conclusion, whether it is population diversity or population uniformity in the search space, the initial population produced by the population initialization approach based on the square-root sequence method is the best. Therefore, the square-root-sequence-based population initialization approach is chosen as the population initialization method.

3.2. Probability Attraction Model

3.2.1. Common Attraction Model

In the existing literature, the commonly used attraction models are: complete attraction model (CAM), random attraction model (RAM), neighborhood attraction model (NAM) and grouping attraction model (GAM). In order to facilitate the statistics of the attraction times and brightness comparison times of various attraction models, let the population size of FA be n and the objective function is the minimum.

In the CAM, each firefly is attracted by other brighter fireflies in the population at each iteration. The individuals in the population are sorted according to the OFV from small to large. Since each firefly will only be attracted to other brighter fireflies, the first firefly (the best firefly) is not attracted and the number of attractors is 0; the second firefly is only

attracted by the first firefly, and the number of attractions is 1. By analogy, the last firefly (the worst firefly) in the population will be attracted by all other fireflies, and the number of attractions is $n - 1$. Therefore, the total attraction times T_1 of the CAM are:

$$T_1 = 0 + 1 + 2 + \dots + n - 1 = \frac{n(n-1)}{2} \quad (8)$$

According to Equation (8), the total attraction times T_1 of the CAM $T_1 = n(n-1)/2$, then the average number of attractions $t_1 = T_1/n = (n-1)/2$ for each firefly.

For the FA using the CAM, each firefly must judge whether it is attracted by other fireflies. Therefore, the n fireflies in the population need brightness comparison. When comparing brightness, each firefly needs to compare brightness with another $n - 1$ firefly, then the total brightness comparison times of n fireflies are $T_2 = n(n-1)$, and the average brightness comparison times of each firefly are $t_2 = n - 1$.

In the NAM [30], each firefly is attracted by a brighter firefly in the k -neighborhood during iteration. The k -neighborhood of X_i consists of $2k + 1$ fireflies $\{X_{i-k}, \dots, X_i, \dots, X_{i+k}\}$, where k is an integer and $1 \leq k \leq (n-1)/2$. k in reference [42] is much smaller than $(n-1)/2$. In order to analyze the times of attraction in each iteration, firstly, the fireflies are sorted according to the OFV from small to large. After sorting, the first firefly is the firefly with the highest brightness and the n th firefly is the firefly with the darkest brightness. Secondly, all fireflies are linked to a ring topology according to the indicator order $1, 2, \dots, n$. The total attraction times of all fireflies are $T_1 = 0 + 1 + \dots + k - 1 + k + k + 1 + k + 2 + \dots + 2k - 1 + 2k = kn$, and the average attraction times of each firefly are $t_1 = T_1/n = k$. Each firefly needs to compare its brightness with all other fireflies in its k -neighborhood. The total brightness comparison times are $T_2 = 2kn$, and the brightness comparison times of each firefly are $t_2 = 2k$.

In the RAM [19], each firefly compares its brightness with another randomly selected firefly in each iteration. Only the fireflies move; otherwise, they do not move. Therefore, the maximum number of attraction times for all fireflies is $T_1 = n$, and the average number of attraction times for each firefly is $t_1 = 1$. Since each firefly has to be compared with another randomly selected firefly, the number of times of brightness comparison for each firefly is 1. For the RAM with a population size of n , the total number of comparisons of the brightness of all fireflies is n , and the number of comparisons of the average brightness of each firefly is $t_2 = 1$.

In the GAM [20], there are two fireflies in each group. In each iteration, the fireflies with low brightness in each group are attracted by the fireflies with high brightness, and the fireflies with high brightness in each group are also attracted by the fireflies with the highest brightness in the population. Therefore, the total attraction times of all fireflies are $T_1 = n - 1$, and the average attraction times of each firefly are $t_1 = T_1/n = (n-1)/n$. In the GAM, the brightness of two fireflies in each group should be compared. There are $n/2$ groups, which need to be compared $n/2$ times; the firefly with high brightness in each group needs to be compared with the highest firefly in the population for $(n-1)/2$ times. Therefore, the sorting of n fireflies requires $n - 1$ comparisons, and the average brightness comparison times of each firefly $t_2 = (n-1)/n$.

The population variety is worse, the TC is higher and local optimization is simpler to fall into as there are more times of attraction. Additionally, for different attraction models, there are various brightness comparison times. The algorithm's TC increases with the number of fitness comparisons. For a good attraction model, the attraction times and brightness comparison times should be moderate, and the selection of fireflies with high brightness should be adaptive or have a better guiding effect on the attracted fireflies. A new attraction model known as the probability attraction model is developed in order to make the attraction times and brightness comparison times of the attraction model modest and to make the attraction model adaptive while choosing fireflies with high brightness.

3.2.2. Probability Attraction Model

To facilitate the description of the probability attraction model, suppose the population size is n and the objective function of the optimization problem is to find the minimum. If the objective function is to seek the maximum value, the objective function can be changed to seek the minimum value according to $\max P(X, M) = -\min[-P(X, M)]$. In addition, in order to make the probability attraction model adaptive in choosing to attract fireflies, and to make the attracted fireflies have a better guiding effect on the attracted fireflies, all the fireflies in the population are sorted according to their OFV and the selection probability of each firefly is calculated. When selecting to attract fireflies, the fireflies with a high probability of selection have a high probability of being selected as attracting fireflies. In the GAM, a firefly is used to attract fireflies, and a firefly in the probability attraction model is selected, as attracting fireflies is determined by its selection probability. The firefly with high probability of selection has a greater probability of being selected as an attractive firefly. Therefore, the probability attraction model is adaptive in choosing to attract fireflies. At the same time, the selected attracting fireflies have a better guiding effect on the attracted fireflies. The specific steps in the probability attraction model are as follows:

- (1) Order fireflies according to the OFV from small to large.
- (2) Calculate the selection probability of each firefly after sorting. The calculation formula for selection probability is:

$$P_k = \frac{fit(k)}{\sum_{j=1}^{i-1} fit(j)} \quad (i = 2, 3, \dots, n; k = 1, 2, \dots, i - 1) \quad (9)$$

$$fit(k) = \lambda \times (1 - \lambda)^{k-1} \quad (k = 1, 2, \dots, n - 1) \quad (10)$$

where P_k is the selection probability of the k -th individual in the population, λ is a constant between 0.01 and 0.3, and $\lambda = 0.15$, $fit(k)$ is the fitness value of the k -th firefly.

- (3) According to the selection probability P_k ($k = 1, 2, \dots, n - 1$) of each firefly, calculate the cumulative probability PP_k ($k = 1, 2, \dots, n - 1$) of each firefly. PP_k is calculated as follows:

$$PP_k = \sum_{j=1}^k P_j \quad (k = 1, 2, \dots, n - 1) \quad (11)$$

- (4) Choose to attract fireflies for the n -th firefly in the population. The selection method is to generate a random number rr evenly distributed in the interval $[0, 1]$. If rr satisfies $PP_{k-1} \leq rr \leq PP_k$ ($k = 2, 3, \dots, n - 1$), the k -th firefly is selected as the attracting firefly. Selecting and attracting fireflies for the $n - 1$ firefly also requires generating a random number rr , and if rr satisfies $PP_{k-1} \leq rr \leq PP_k$ ($k = 2, 3, \dots, n - 1$), select the k -th firefly as the attracting firefly. By analogy, the $n - 3, n - 4, \dots, 2$ th fireflies are selected to attract fireflies, and the first firefly is the best. It is not attracted by any fireflies and only moves randomly.

In summary, when the population size is n , the total number of attraction times is $T_1 = n - 1$ and the average number of attraction times for each firefly is $t_1 = (n - 1)/n$. The total number of brightness comparisons $T_2 = n(n - 1)/2$, the average number of brightness comparisons of each firefly $t_2 = (n - 1)/2$. The number of attraction times and the number of brightness comparisons of the five attraction models are shown in Table 2.

In Table 2, the total attraction times of the probability attraction model are less than the CAM and the NAM, which are equal to the GAM. For the RAM, if the randomly selected n fireflies have high brightness, the total attraction times of the probability attraction model are less than that of the RAM; otherwise, the total attraction times of the probability attraction model are greater than or equal to the RAM. For the average attraction times, the comparison result of the average attraction times of each firefly is the same as that of the total attraction times. In addition, the brightness comparison times of the probability attraction model are equal to the GAM, but less than the CAM, RAM and NAM.

Table 2. The number of attractions and brightness comparison of the 5 attraction models.

Attraction Model	T_1	t_1	T_2	t_2
CAM	$n(n-1)/2$	$(n-1)/2$	$n(n-1)$	$n-1$
RAM	$\leq n$	≤ 1	n	1
NAM	kn	k	$2kn$	$2k$
GAM	$n-1$	$(n-1)/n$	$n-1$	$(n-1)/n$
Probability attraction Model	$n-1$	$(n-1)/n$	$n-1$	$(n-1)/n$

To sum up, for the five attraction models, the probability attraction model and GAM have moderate attraction times and the least brightness comparison times. In the GAM, the attraction relationship between fireflies is fixed, that is, the attraction of the next $n/2$ fireflies to the first $n/2$ fireflies is unchanged, and the attraction from the second to the $n/2$ fireflies to the first firefly is also unchanged. In the probability attraction model, whether a firefly with low brightness is attracted by a firefly with high brightness is determined by the selection probability of the firefly with high brightness. When selecting to attract fireflies, it is adaptive and has a great probability to select the firefly with higher brightness, which has a better guiding role for the attracted firefly.

Therefore, the probability attraction model can avoid premature convergence caused by too many times of attraction and can also avoid too little attraction and reduce the convergence speed of the algorithm. When choosing to attract fireflies, according to the selection probability of fireflies, the selection strategy is adaptive and has a better guiding role for attracted fireflies. In addition, because the brightness comparison times of the probability attraction model are less, the TC in the algorithm can be reduced.

3.3. Improved Location-Update Method

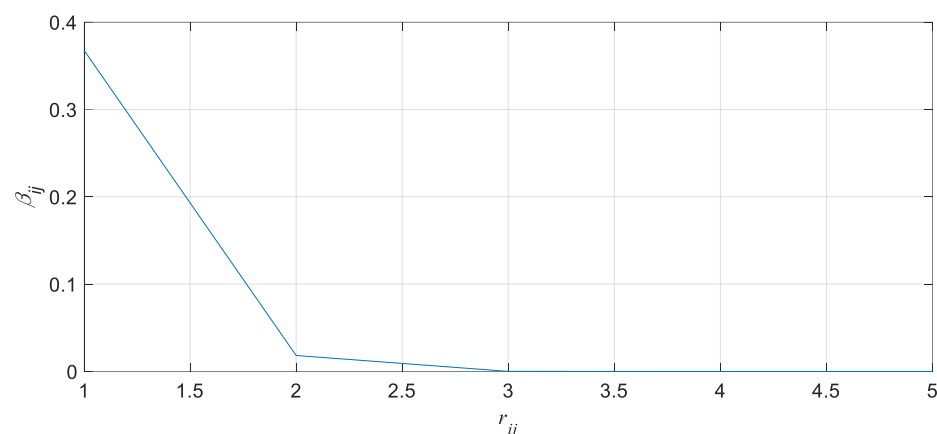
In the firefly algorithm in the existing literature [6,43,44], the position-update formula and the calculation formula of relative attraction are:

$$x_j(t+1) = x_j(t) + \beta_{ij}(\gamma_{ij})(x_i(t) - x_j(t)) + \alpha \varepsilon_j \quad (12)$$

$$\beta_{ij}(r_{ij}) = \beta_0 e^{-\gamma r_{ij}^2} \quad (13)$$

where the brightness of x_i is higher than that of x_j , β_{ij} is the relative attraction between firefly i and firefly j , β_0 is the maximum attraction, γ is the light-absorption coefficient, r_{ij} is the distance between firefly i and firefly j , α is a constant and ε_j is a random number obtained from Gaussian distribution. For most problems, it can take $\beta_0 = 1$, $\gamma \in [0.01, 100]$, $\alpha \in [0, 1]$.

When $\gamma = 1$, the curve of relative attractiveness β_{ij} as the number of iterations increases is shown in Figure 2.

**Figure 2.** Variation curve of relative attraction β_{ij} .

In Figure 2, the relatively attractive force β_{ij} decreases with the distance r_{ij} between firefly i and firefly j increases. When the distance between firefly i and firefly j is $r_{ij} \geq 3$, the value of relative attraction $\beta_{ij}(r_{ij})$ approaches 0. At this point, the value of the attractive term β_{ij} in the position-update formula approaches 0, the β_{ij} has no effect, the position-update formula has no guiding effect and the algorithm degenerates into a random search with a slow convergence speed. At the same time, because of the relatively small value of the perturbation term $\alpha\epsilon_j$, the global search ability of the algorithm is weak, and it is easy to fall into local optimization. In addition, for a given problem to be optimized, if the value range of variables is large (for example, $-100 \leq x_{ik} \leq 100$, $I = 1, 2, \dots, n$, $k = 1, 2, \dots, D$), the distance between fireflies in the population after initialization is relatively far, and the value of relative attraction $\beta_{ij}(r_{ij})$ may approach 0, so the attraction term does not work. As the number of iterations increases, the distance between fireflies in the population decreases gradually and the value of relative attraction $\beta_{ij}(r_{ij})$ increases gradually. When $r_{ij} < 3$, the attraction term works, and with the increase in attraction $\beta_{ij}(r_{ij})$, the attraction term gradually increases, so the guiding effect of the firefly with high brightness in the position-update formula gradually increases and the convergence speed of the algorithm will be gradually improved. To solve the above problems, a new position-update formula is proposed. The new location-update formula is:

- (1) Sort the fireflies according to their OFV from small to large.
- (2) The fireflies in the population update their positions as follows.

$$\begin{cases} X_i(t+1) = X_i(t) + \beta R(X_k(t) - X_i(t)) + (1-R)\text{rand}(1,D) \oplus (X_{best}(t) - X_i(t)) + \alpha(\epsilon - 0.5) \\ X_{best}(t+1) = N\left(X_{best}, \frac{|X_{best} - \bar{X}|^2}{36}\right) \end{cases} \quad (14)$$

$$\bar{X} = \frac{X_1 + X_2 + \dots + X_n}{n} \quad (15)$$

where t is the number of iterations, $X_i(t)$ is the position of the i -th firefly of generation t , $X_i(t+1)$ is the position of the i -th firefly in generation $t+1$, $X_{best}(t)$ is the position of the best firefly in generation t , $\text{rand}(1,D)$ is a random vector uniformly distributed between $[0,1]$, $X_k(t)$ is the position of the firefly selected by probability attraction in generation t , β is the position of the firefly i and the attraction between fireflies X_k , R is the adjustment coefficient, α is the dynamic step and ϵ is a uniformly distributed D -dimensional random vector between $[0,1]$, $\oplus$ indicating that the elements of two vectors at the same position are multiplied.

$\beta R(X_k(t) - X_i(t))$ in Equation (14) is called the attraction term, $(1-R)\text{rand}(1,D) \oplus (X_{best}(t) - X_i(t))$ is called the bootstrap term and $\alpha(\epsilon - 1/2)$ is called the stochastic term. β and R are calculated as follows:

$$\beta = \beta_{\min} + (\beta_{\max} - \beta_{\min})e^{-\gamma r_{ij}^2} \quad (16)$$

$$R = 1 - \frac{\text{runtime}(t)}{\text{Maxtime}} \quad (17)$$

where $\beta_{\max}$ is the maximum attraction, $\beta_{\min}$ is the minimum attraction (both are constants), γ is the light-absorption coefficient, usually taken as 1, runtime is the time of the current iteration and Maxtime is the maximum time for the algorithm to run.

From Equations (14) and (16), it can be seen that Equation (16) can avoid the attraction β being 0 at the beginning of the iteration, which leads to the phenomenon that the attraction term does not work. In addition, from Equation (16), at the beginning of the iteration, the fireflies are far away from each other, so the value of β is small, and with an increase in the number of iterations, the distance of fireflies gradually decreases, and the value of β gradually becomes larger; when the distance of fireflies is 0, the value of β reaches the maximum, at which time $\beta = \beta_{\max}$. As the number of iterations increases, β becomes larger and R becomes smaller, so the trend of βR cannot be determined. To judge the changing trend of βR with the increase in the number of iterations, the C01 function in the CEC 2017

function set was chosen as the test function, and the βR values of the 2nd firefly and the 15th firefly at each iteration were kept, and a change curve of βR with an increase in the number of iterations was drawn, as shown in Figure 3.

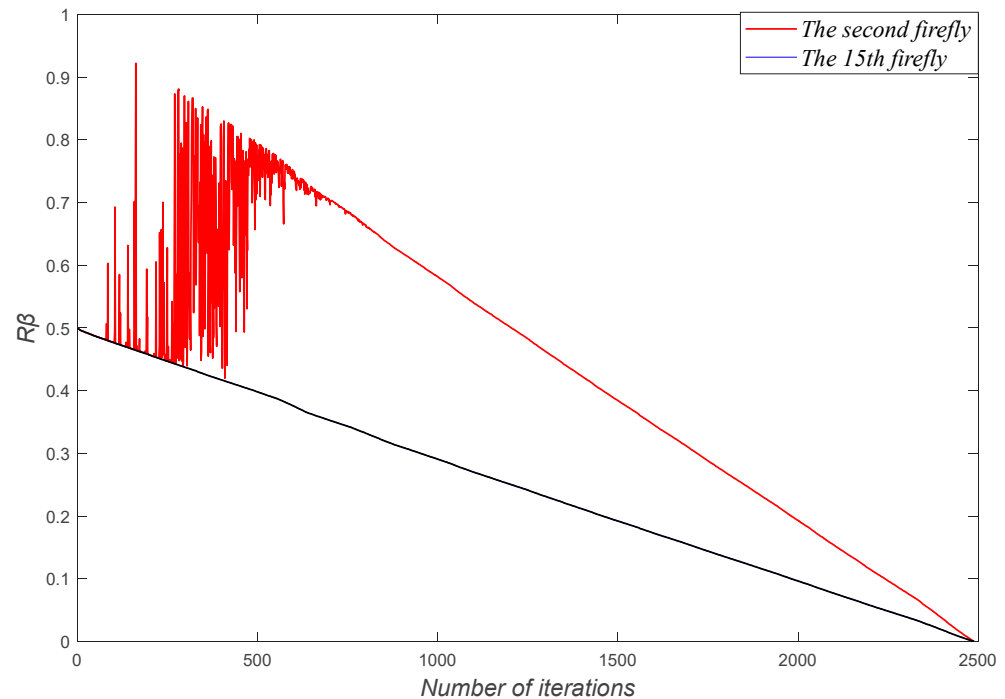


Figure 3. Variation curve of βR .

Figure 3 illustrates that the value of βR decreases with an increase in the number of iterations. At the beginning of the iteration, the value of βR is larger, the distance between fireflies in the population is relatively far, the value of $X_k(t) - X_i(t)$ is larger and the value of $\beta R(X_k(t) - X_i(t))$ is larger. In addition, it can be seen from Equation (17) that at the beginning of the iteration, the value of R is larger, the value of $1 - R$ is smaller, the value of $\text{rand}(1, D)(1 - R)$ is even smaller and the value of $\text{rand}(1, D)(1 - R)(X_{\text{best}}(t) - X_i(t))$ is also smaller. At the beginning of the iteration, the term $\beta R(X_k(t) - X_i(t))$ plays a major role, then the algorithm's global search ability is strong; at the later part of the iteration, the distance between fireflies becomes smaller compared with the beginning of the iteration and the value of $X_k(t) - X_i(t)$ is also decreasing compared with the beginning of the iteration. In addition, in the later stage of the iteration, the value of βR gradually decreases. Hence, in the later stage of the iteration, the value of $\beta R(X_k(t) - X_i(t))$ gradually decreases compared with the initial stage of the iteration, and the $\beta R(X_k(t) - X_i(t))$ term has a weaker influence on the change in firefly position. As the value of R gradually becomes smaller in the later iterations, the value of $1 - R$ gradually becomes larger and the value of $\text{rand}(1, D)(1 - R)(X_{\text{best}}(t) - X_i(t))$ gradually changes in a large manner. Therefore, at the later stage of the iteration, the local search capability of the algorithm is enhanced. The position-update formula of the population can better consider the global search ability and local search ability of the algorithm.

The main content of the firefly population position update is given in Algorithm 3.

Algorithm 3: Improved position update based on probability attraction model**Input:** individual fireflies of the population;**Output:** the updated firefly population;

Calculation of the OFV for fireflies in the current population;

The fireflies in the population are sorted according to their OFV from small to large;

Recording the best firefly X_{best} and its OFV f_{best} ;for $i = 2$ to n do Update $X_i(t)$ and X_{best} according to Equation (14);

End for

3.4. Combined Mutation Operator Based on Selection Probability

FA's exploration and exploitation abilities are determined by the firefly's location-update formula. The firefly location-update formulation given in the existing literature has a strong exploration capability in the early iteration and a strong exploitation capability in the late iteration, which can balance the exploration and exploitation ability of the algorithm to some extent. For a multiple-optimization problem, even if the iteration termination condition is the maximum number of iterations or the maximum running time, it is impossible to judge when it is the early stage of the iteration and when it is the late stage of the iteration. Therefore, estimating the early and late iterations based on the quantity of iterations or running duration is incorrect. According to the maximum number of iterations and the maximum running duration, there are two issues with determining the issues in the early and late iterations: First, the algorithm performs local search when local search is required and global search when local search is required. In addition, the algorithm does not need global search in each iteration in the early stages of iteration, and sometimes it may need local search; in the late stage of iteration, local search is not required for each iteration. If it falls into local optimum, global search is required. The combination mutation operator in the literature [20] selects a single mutation operator according to the same probability. It is possible that when a mutation operator with strong exploration ability is needed, a mutation operator with strong exploitation ability is selected. A combined mutation operator based on selection probability is proposed to solve the above problem.

For the convenience of description, suppose the i -th firefly participating in the mutation operation is $X_i(t)$ ($i = 1, 2, \dots, n$) and the firefly obtained after the mutation is $X_i(t+1)$ ($i = 1, 2, \dots, n$), $b_1, b_2, b_3, b_4, b_5 \in \{1, 2, \dots, n\}$, and $b_1 \neq b_2 \neq b_3 \neq b_4 \neq b_5 \neq i$, $X_{best}(t)$ is the best firefly in the t -th generation population. The four single variance operators selected are:

$$X_i(t+1) = X_{b1}(t) + F(X_{b2}(t) - X_{b3}(t)) \quad (18)$$

$$X_i(t+1) = X_{b1}(t) + F(X_{b2}(t) - X_{b3}(t)) + F(X_{b4}(t) - X_{b5}(t)) \quad (19)$$

$$X_i(t+1) = X_{best}(t) + F(X_{b1}(t) - X_{b2}(t)) \quad (20)$$

$$X_i(t+1) = X_{best}(t) + F(X_{b1}(t) - X_{b2}(t)) + F(X_{b3}(t) - X_{b4}(t)) \quad (21)$$

where F is the step size and the calculation formula of F is:

$$F = 0.4 + 0.6 \times r \quad (22)$$

where r is a random number between $[0,1]$.

The single mutation operators in Equation (18) to Equation (21) are divided into two categories: the first is the single mutation operator with strong exploration ability, including Equations (18) and (19); the second type is a single mutation operator with strong exploitation ability, including Equations (20) and (21). The calculation formula of the selection probability of the two types of mutation operators is:

$$P_1 = \frac{\frac{S_1}{F_1}}{\frac{S_1}{F_1} + \frac{S_2}{F_2}} = \frac{1}{1 + \frac{S_2 F_1}{S_1 F_2}} \quad (23)$$

$$P_2 = 1 - P_1 \quad (24)$$

where P_1 is the selection probability of the first type of mutation operator, S_1 is the number of candidate solutions generated by the first type of mutation operator retained for the next generation, F_1 is the number of candidate solutions generated by the first type of mutation operator that is not kept for the next generation, and S_2 is the number of candidate solutions generated by the second type of mutation operator that is retained for the next generation and F_2 is the number of candidate solutions generated by the second type of mutation operator that cannot be retained to the next generation. Generally, the initial values of S_1 , F_1 , S_2 and F_2 are taken as 1.

First, judge whether to choose the first type of mutation operator or the second type of mutation operator. The judgment method is randomly generating a random number μ between [0,1]. If $\mu < P_1$, select the first type of mutation operator; if $\mu \geq P_1$, select the second type of mutation operator. Secondly, judge whether to choose the first mutation operator or the second mutation operator in a certain class. The method of judgment is: randomly generate a random number λ between [0,1]; if $\lambda < 0.5$, select the first mutation operator in a certain type of mutation operator; otherwise, select the second mutation operator.

Algorithm 4 describes the main steps of combining mutation operators based on selection probability.

Algorithm 4: Combined mutation operation based on selection probability

Input: Firefly individuals in the population;

Output: Firefly individuals in the population after combined mutation operation;

Calculate the value of scaling factor f according to Equation (22);

for $i = 1$ to n

if $rand < P_1$

if $rand \leq 0.5$

Generate a new solution $X_i(t + 1)$ according to Equation (18);

else

Generate a new solution $X_i(t + 1)$ according to Equation (19);

else

Else $rand \leq 0.5$

Generate a new solution $X_i(t + 1)$ according to Equation (20);

else

Generate a new solution $X_i(t + 1)$ according to Equation (21);

End if

End if

End for

Equations (23) and (24) report that, if a global search is required for the t -th iteration, if the second type of mutation operator is selected, the value of S_2 is less than that of F_2 . Since the values of S_1 and F_1 remain unchanged, the value of $(S_2F_1)/(S_1F_2)$ decreases and the value of P_1 increases. The first type of mutation operator will probably be selected in the $t + 1$ iteration. If a global search is required for the t -th iteration, if the first type of mutation operator is selected, the value of S_1 is greater than the value of F_1 . Since the values of S_2 and F_2 remain unchanged, the value of $(S_2F_1)/(S_1F_2)$ decreases and the value of P_1 increases. The first type of mutation operator will be selected with a high probability in the $t + 1$ iteration, and vice versa.

In conclusion, based on the historical contributions of the two types of mutation operators, the combined mutation operator based on the selection probability can choose which type of mutation operator is selected to perform the mutation operation in the following iteration. As a result, regardless of whether the algorithm is in an early or late iteration, the combined mutation operator based on the selection probability can flexibly select a particular type of mutation operator with high exploration or exploitation ability. In addition, due to the strong exploration ability of the first type of mutation operator, the

contribution made at the beginning of the iteration is greater than that of the second type of mutation operator, so the first type of mutation operator is more likely to be selected, which improves the exploration of the algorithm ability. At the end of the iteration, the historical contribution of the second type of mutation operator is greater than that of the first type of mutation operator, so mutation operators of the second type are more likely to be selected, which improves the exploitation ability of the algorithm.

3.5. Remove Similarity Operation

Similar individuals mean that the difference in fitness values between individuals is less than a certain threshold ζ . There will be a large number of similar individuals in the population as iteration times rise. The population's members are generally gathered close to the optimal firefly, especially later in the iteration. For the multi-extremum optimization problem, if the best individual in the population is near a local extremum, the algorithm can easily fall into a local optimum. Therefore, in the iterative process, similar individuals need to be removed from the population to maintain diversity. In order to keep the population size constant, similar individuals that were removed can be generated via the random initialization method. The specific steps in similarity removal are:

- (1) Order the fireflies according to the OFV from small to large;
- (2) Evaluate the similarity of fireflies. The formula for calculating the similarity S between fireflies in the population is:

$$S = \frac{f(0.5n) - f(1) + eps}{f(n) - f(1) + eps} \quad (25)$$

where $f(0.5n)$ is the OFV of the $0.5n$ firefly in the sorted population and $f(1)$ is the OFV of the first firefly in the sorted population, $f(n)$ is the OFV of the last firefly in the sorted population, eps is a small number, $eps = 2.2204 \times 10^{-16}$ and eps is used to avoid the denominator being zero.

- (3) If $S \geq \zeta$, the similarity between fireflies in the population is higher and the population diversity is poor. In this case, the better q individuals in the population are retained, and the remaining $n-q$ individuals are generated by a random initialization method.

To visualize the similarity degree of the population, the population is processed in two dimensions; let the population size be $n = 40$ and the number of dimensions of the variables be $D = 2$. In the iterative process of the algorithm, Figure 4 shows the degree of similarity in the population that does not use a remove-similarity operation, and Figure 5 illustrates the degree of similarity in the population after using a remove-similarity operation.

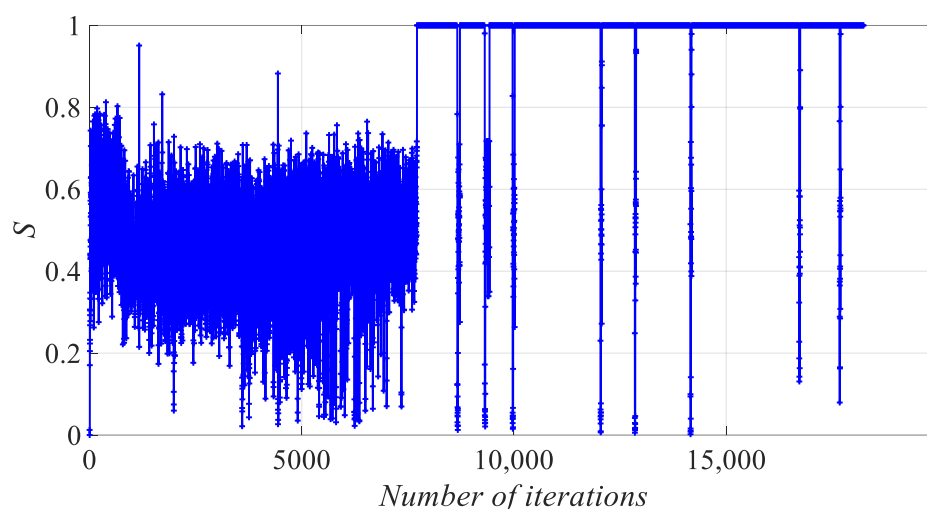


Figure 4. Population similarity without adding removal similarity operation.

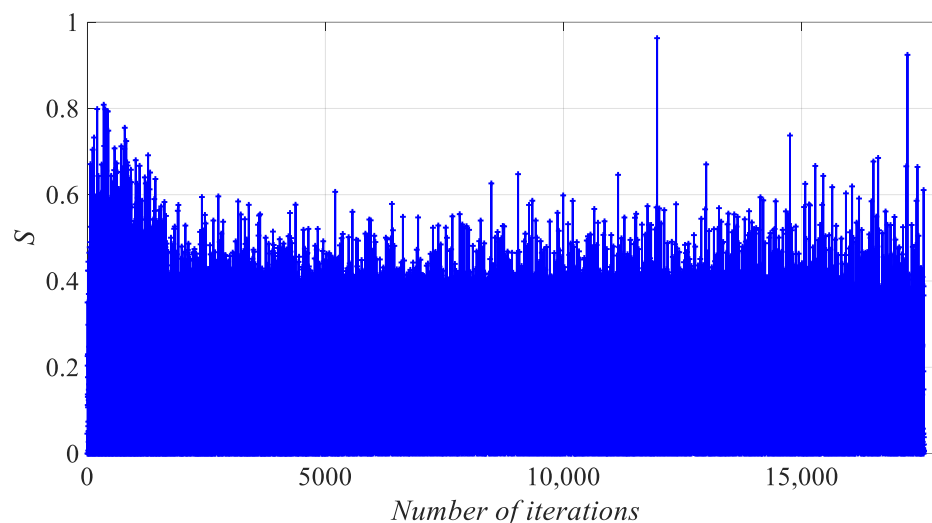


Figure 5. Add the population similarity after removing similarity operation.

In Figure 4, the difference between the OFV of the first firefly and the last firefly is larger in the early iteration, and the difference between the OFV of the $0.5n$ -th firefly and the first firefly is also larger, so the S value is smaller. As the algorithm gradually iterates, the individuals in the population will slowly gather near the optimal individual, and the $0.5n$ firefly and the last firefly in the population also focus on the optimal individual. The difference between the OFV of $f(0.5n)$ and $f(n)$ gradually becomes smaller, and the value of S gradually increases. Due to the introduction of a combinatorial mutation operator in the algorithm, the S value is small occasionally, but it is not enough to jump out of the local optimum, so the value of S increases rapidly. From Figure 5, after adding the remove-similarity operation, the S value of the algorithm is small in most cases and large in occasional cases. When $S \geq \zeta$, the S value decreases rapidly, and the diversity in the population is better because of the increase in performing the remove-similarity operation in the population. Therefore, the addition of the remove-similarity operation increases the global search capability of the algorithm and reduces the probability of the algorithm falling into a local optimum.

3.6. The Evolutionary Strategy of IHFAPA

The evolutionary strategy in the improved hybrid firefly algorithm with probability attraction (IHFAPA) is presented in Algorithm 5. Figure 6 shows a flow chart of IHFAPA, which more clearly displays the specific steps in the algorithm. From Figure 6, we can see that the performance of IHFAPA is related to the location-update method, probability attraction model, combined mutation operator, remove-similarity operation and evolution strategy of fireflies. IHFAPA uses a new location-update method. Under the attraction of the high-brightness fireflies and the best fireflies, the low-brightness fireflies move to the high-brightness fireflies, which better balances the exploration and exploitation ability of the algorithm. Meanwhile, IHFAPA uses the combined mutation operator, which can be based on the historical contributions of the two types of mutation operators, to select a suitable single mutation operator to perform mutation operation adaptively according to needs. Combined mutation operators include two different single mutation operators, and each type of mutation operator has a different selection probability. If a certain type of operator has a greater contribution in the previous iteration process, then this type of operator has a greater probability of selection; on the contrary, this type of mutation operator has a smaller selection probability. In the iterative process, the selection probabilities of the two types of mutation operators are not static. As the historical contributions of the two types of mutation operators change, the selection probability will also change. Therefore, the combined mutation operator based on the selection probability can well balance the exploration and exploitation ability of the algorithm. In addition, a remove-similarity

operation is added to IHFAPA, so that the algorithm can better maintain the diversity in the population in the iterative process and reduce the probability of the algorithm falling into a local optimum. IHFAPA uses an evolutionary strategy of elite retention in its iterative process. The position-update formula, combination mutation operator and similarity-removal operation balance the exploration and exploitation ability of IHFAPA and facilitate the maintenance of population diversity.

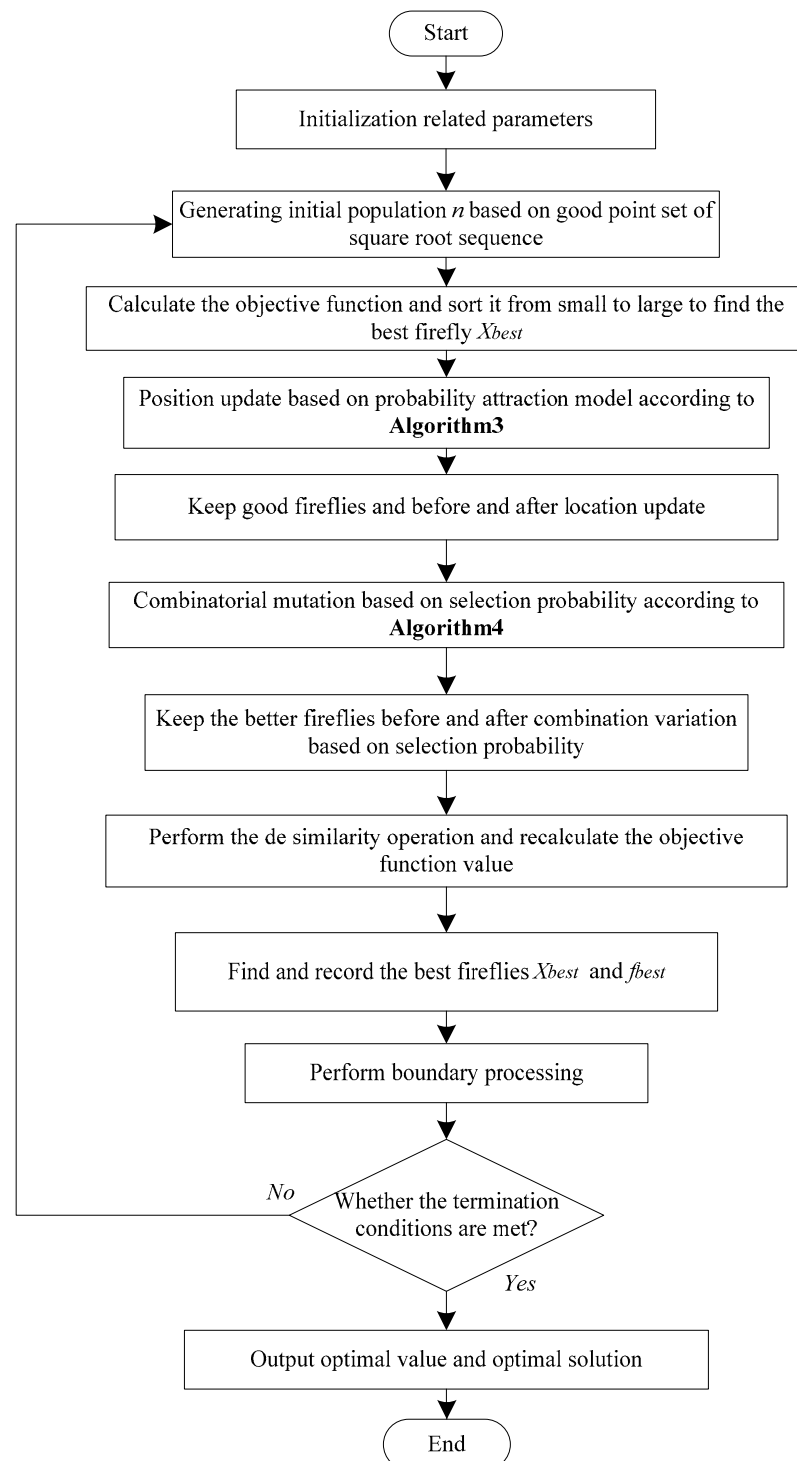


Figure 6. A flow chart of IHFAPA.

Algorithm 5: IHFAPA

Input: population size n , initial values of S_1, F_1, S_2 and F_2 , and step size α_0 , maximum attraction β_{max} , minimum attraction β_{min}

Output: optimal solution x and optimal value $f(x)$;

7. Start timing and make the initial time $RunTime = 0$;
8. Generate an initial population $X = (X_1, X_2, \dots, X_n)$ of population size n using a population initialization method based on a square-root sequence, and calculate the OFV of fireflies.
9. Sort the population of fireflies according to the OFV from small to large, find the best firefly X_{best} ;
10. Let $t = 0$
11. **While** $RunTime \leq Maxtime$
12. $t = t + 1$;
13. The fireflies in the population update their positions according to Algorithm 3, then calculate the OFV of the firefly after the location update, compare the OFV of fireflies before and after update, keep better fireflies and their OFV;
14. The fireflies in the population are combined and mutated according to Algorithm 4, and the OFV of the mutated fireflies are calculated and compared with the OFV before mutation to retain the good fireflies and their OFV.
15. Perform remove-similarity operation, recalculate the OFV of the firefly, renew the best fireflies X_{best} in the population
16. Boundary processing of variables, variables that exceed the value range are randomly generated within the value range;
17. Calculate the OFV of the fireflies in the population, update the best firefly X_{best} in the population;
18. Recording the runtime of the algorithm $RunTime$.
19. End While
20. Output the optimal solution and the optimal value.

4. Numerical Experiment and Result Analysis

To ensure the fairness of the experiment, all experiments were conducted on the same computer. The computer's operating system is Windows 10, the processor is AMD Ryzen 9 3900 12-Core, the main frequency is 3.09 GHz and the RAM is 32 GB. All algorithms are developed in MATLAB R2019b programming language.

4.1. Selection of Test Function

The CEC2017 benchmark test function set, which is currently common internationally, is selected to verify the performance of the IHFAPA. Table 3 provides some brief information regarding the CEC 2017 test function set (see reference [41] for details). The 28 test functions in CEC 2017 are constrained optimization problems, and the objective function is to find the minimum value.

Table 3. The specifics of the 28 CEC 2017 optimization questions.

Function	Search Range	Type of Objective	Number of Constraints	
			E	I
C01	$[-100,100]^D$	Non-Separable	0	1 Separable
C02	$[-100,100]^D$	Non-Separable Rotated	0	1 Non-Separable, Rotated
C03	$[-100,100]^D$	Non-Separable	1 Separable	1 Separable
C04	$[-10,10]^D$	Separable	0	2 Separable

Table 3. Cont.

Function	Search Range	Type of Objective	Number of Constraints	
			<i>E</i>	<i>I</i>
C05	$[-10,10]^D$	Non-Separable	0	2 Non-Separable, Rotated
C06	$[-20,20]^D$	Separable	6	0 Separable
C07	$[-20,20]^D$	Separable	2 Separable	0
C08	$[-100,100]^D$	Separable	2 Non-Separable	0
C09	$[-10,10]^D$	Separable	2 Non-Separable	0
C10	$[-100,100]^D$	Separable	2 No Separable	0
C11	$[-100,100]^D$	Separable	1 Separable	1 Non Separable
C12	$[-100,100]^D$	Separable	0	2 Separable
C13	$[-100,100]^D$	Non-Separable	0	3 Separable
C14	$[-100,100]^D$	Non-Separable	1 Separable	1 Separable
C15	$[-100,100]^D$	Separable	1	1
C16	$[-100,100]^D$	Separable	1 Non-Separable	1 Separable
C17	$[-100,100]^D$	Non-Separable	1	1 Separable
C18	$[-100,100]^D$	Separable	0	2 Non-Separable
C19	$[-50,50]^D$	Separable	0	2 Non-Separable
C20	$[-100,100]^D$	Non-Separable	0	2
C21	$[-100,100]^D$	Rotated	0	2 Rotated
C22	$[-100,100]^D$	Rotated	1 Rotated	3 Rotated
C23	$[-100,100]^D$	Rotated	1 Rotated	1 Rotated
C24	$[-100,100]^D$	Rotated	1 Rotated	1 Rotated
C25	$[-100,100]^D$	Rotated	1 Rotated	1 Rotated
C26	$[-100,100]^D$	Rotated	1 Rotated	1 Rotated
C27	$[-100,100]^D$	Rotated	1 Rotated	2 Rotated
C28	$[-50,50]^D$	Rotated	0	2 Rotated

In Table 3, D is the number of decision variables, I is the number of inequality constraints and E is the number of equality constraints.

4.2. Evaluation Method of Algorithm Performance

4.2.1. Algorithm Performance Evaluation Indicators

To compare the performance of IHFAPA and various comparison algorithms, the mean value, standard deviation, $w/t/l$, Friedman rank ranking [45] and Holm's procedure [46] are selected as performance evaluation indicators.

- Mean value

The mean value is the average value of the optimal value of the test function obtained by the algorithm in R independent runs, noted as *Mean*. The calculation formula of *Mean* is:

$$Mean = \frac{\sum_{i=1}^R f_i}{R} \quad (26)$$

where R is the total number of independent runs in the algorithm. f_i is the optimal value of the test function obtained by the algorithm in the i -th independent operation.

- Standard deviation

The standard deviation is the deviation between the mean value of the optimal value obtained by the algorithm in R runs and the respective optimal value, noted as *Std*. The calculation formula for standard deviation is:

$$Std = \sqrt{\frac{\sum_{i=1}^R (f_i - Mean)^2}{R}} \quad (27)$$

where R is the total number of algorithm runs and f_i is the optimal value of the test function obtained by the algorithm in the i -th independent operation. *Mean* is the average of R optimal values.

- $w/t/l$ indicators

To better measure the performance of IHFAPA and the various algorithms participating in the comparison, compare the average value of the optimal value of the objective function of IHFAPA with other comparison algorithms and the comparison result is recorded as $w/t/l$. For a certain test function, if the performance of the IHFAPA is better than the other algorithm (denoted as *Alg*), then the value of w is recorded as 1; if the performance of the IHFAPA algorithm is the same as the performance of another algorithm taking part in the comparison, the value of t is recorded as 1; if the performance of another algorithm participating in the comparison is better than IHFAPA, then the value of l is recorded as 1. For all test functions, add the value of w from the algorithm to obtain the value of w . In the same way, the values of t and l can be calculated.

- Friedman rank ranking

Friedman rank ranking is a nonparametric statistical method, which can rank the performance of participating comparison algorithms. Suppose there are m algorithms involved in the comparison and k test functions are selected. Then, the specific steps in Friedman rank sorting are (take the minimization problem as an example):

1. Each algorithm is run R times independently on each test function, and the optimal value of each run is retained.
2. Record the optimal value obtained from R runs and calculate the average value of R optimal values according to the following formula:

$$meanf_i^j = \frac{\sum_{l=1}^R f_i^j(l)}{R} \quad (i = 1, 2, \dots, m; j = 1, 2, \dots, k) \quad (28)$$

where m is the number of algorithms participating in the comparison, k is the number of test functions and R is the number of independent runs. $meanf_i^j$ represents the average value of the optimal value obtained by the i -th algorithm independently running on the j -th test function for R times.

3. For each test function, all the algorithms participating in the comparison are sorted in the order of $meanf_i^j$ from small to large and give the algorithm rank ranking $rank_i^j$ ($i = 1, 2, \dots, m; j = 1, 2, \dots, k$). If the average value of the optimal value of the comparison algorithm is the same, then take the average of the ranking position as the rank ranking. To explain the calculation method of ranking and rank ranking, suppose there are five algorithms involved in the comparison. For a certain test function, if

the average value of the optimal value obtained by the algorithm participating in the comparison is 1, 3, 3, 2 and 4, respectively, since the average of the optimal values found by the second and third algorithms are the same, then the ranking of the two algorithms are 3 and 4, respectively. Take the average of the ranking positions of these two algorithms $((3 + 4) / 2 = 3.5)$ as the rank ranking of the algorithm. Therefore, the rank rankings corresponding to the five algorithms are 1, 3.5, 3.5, 2 and 5. The results of the rank ranking are shown in Table 4.

4. Calculate the average of the rank ranking of each algorithm $Averank_i$.

$$Averank_i = \frac{1}{m} \sum_{j=1}^k rank_i^j (i = 1, 2, \dots, m) \quad (29)$$

where m is the number of participating comparison algorithms and k is the number of test functions.

5. Rank according to the average value $Averank_i$ of the rank ranking of each algorithm from small to large, the result of the sorting is the final ranking $Finalrank_i$ of various algorithms.

Table 4. Results of rank ranking.

The j -th Test Function	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4	Algorithm 5
Average of optimal values	1	3	3	2	4
Ranking position	1	3	4	2	5
Ranking $rank_i^j$	1	3.5	3.5	2	5

4.2.2. Algorithm Performance Difference Significance Test

The Friedman test was first proposed by Friedman in 1945. It is a nonparametric test and is used to determine whether there are significant differences between the comparison algorithms. Suppose there are m comparison algorithms, that is, there are m samples. Each sample contains the average of the optimal value of the objective function obtained by k corresponding algorithms. Then, the Friedman test steps are as follows:

1. The original hypothesis, opposite hypothesis and significance level of Friedman test are given α .

H_0 : There is no significant difference in the performance of the m algorithms participating in the comparison;

H_1 : There are obvious differences in the performance of the m algorithms participating in the comparison.

2. Calculate the rank of each algorithm corresponding to each test function $rank_i^j (1 \leq I \leq m, 1 \leq j \leq k)$.
3. Calculate the sum of rank ranking of each test function corresponding to each algorithm $sunrank_i$; the $sunrank_i$ calculation formula is $sunrank_i$:

$$sunrank_i = \sum_{j=1}^k rank_i^j (i = 1, 2, \dots, m) \quad (30)$$

4. Calculate the Friedman test value χ^2 . The calculation formula of χ^2 is:

$$\chi^2 = \frac{12}{km(m+1)} \sum_{i=1}^m sunrank_i^2 - 3k(m+1) \quad (31)$$

5. According to the pre-determined significance level α and degrees of freedom $(m - 1)$. Critical values can be obtained from the table of critical values of the Chi-square test $\chi^2_{\alpha[m-1]}$, if

$$\chi^2 \geq \chi_{\alpha(m-1)}^2 \quad (32)$$

Then, reject the original hypothesis H_0 . It shows that there are obvious differences in the performance of the m algorithms participating in the comparison. Otherwise, accept the null hypothesis H_0 . It shows that there is no significant difference in the performance of the m algorithms involved in the comparison.

4.3. Obtain the Optimal Parameter Combination through Orthogonal Experiments

The best parameter combination can be found via orthogonal experiments because the threshold size and population size of the similarity-removal operation have a significant impact on algorithm performance. There are two factors in the orthogonal experiment, that is, the threshold ζ for the degree of similarity between individuals in the population and the population size n . Design three levels for each factor. The three levels of the threshold ζ are 0.3, 0.4 and 0.5; the three levels of population size are 0.95, 0.97 and 0.98. Therefore, the orthogonal experiment includes two factors and three levels. Tables 5 and 6 show the orthogonal experimental design.

Table 5. Factors and level of orthogonal experiment.

Value	Factor A(ζ)	Factor B(n)
Level 1	0.3	0.95
Level 2	0.4	0.97
Level 3	0.5	0.98

Table 6. Orthogonal array L9 (32) for the orthogonal experiment.

Experiment Number	A(ζ)	B(n)
E1	Level 1	Level 1
E2	Level 1	Level 2
E3	Level 1	Level 3
E4	Level 2	Level 1
E5	Level 2	Level 2
E6	Level 2	Level 3
E7	Level 3	Level 1
E8	Level 3	Level 2
E9	Level 3	Level 3

To obtain the optimal combination of parameters, the 30-dimensional CEC 2017 test function is selected. In addition, to ensure the fairness of the comparison results, suppose the population size $n = 40$, the dimension of the variable $D = 30$, maximum running time $Maxtime = 20$ s, the penalty factor $M = 10^8$, statistics times $tjcs = 20$. Tables 7 and 8 and Figure 7 report the results of orthogonal experiments. The test results obtained are compared in Table 7. Table 8 exhibits the Friedman test results of the orthogonal experiment, and Figure 7 shows Friedman mean rank and final rank ranking.

In Table 8, $\chi^2 = 160.54$, $\chi^2 \propto [k - 1] = 15.51$. Since $\chi^2 \geq \chi^2 \propto [k - 1]$, the original hypothesis is rejected. It shows that there are significant differences between the comparative test combinations. In addition, since E5 ranks first in all trial combinations (*mean rank* = 2.68), it shows that E5 is significantly better than other test combinations. Therefore, the optimal parameter combination is $\zeta = 0.4$, $P = 0.97$.

Table 7. The results of orthogonal experiment.

Function	Evaluation Indicator	E1	E2	E3	E4	E5	E6	E7	E8	E9
C01	Mean	7.05×10^{-13}	1.50×10^{-15}	5.22×10^{27}	6.72×10^{-14}	6.17×10^{-15}	2.51×10^{27}	3.30×10^{-13}	4.04×10^{-13}	1.24×10^{27}
	Std.	1.30×10^{-12}	1.86×10^{-15}	8.43×10^{27}	1.01×10^{-13}	8.21×10^{-15}	2.88×10^{27}	7.05×10^{-13}	3.53×10^{-13}	1.84×10^{27}
C02	Mean	3.80×10^{-13}	8.11×10^{-14}	2.29×10^{27}	4.18×10^{-13}	2.52×10^{-14}	3.76×10^{27}	3.88×10^{-12}	1.27×10^{-11}	9.05×10^{26}
	Std.	5.10×10^{-13}	2.12×10^{-13}	2.59×10^{27}	5.53×10^{-13}	5.99×10^{-14}	6.54×10^{27}	6.59×10^{-12}	2.48×10^{-11}	1.31×10^{27}
C03	Mean	6.93×10^9	9.14×10^9	4.43×10^{27}	2.29×10^9	9.86×10^9	3.13×10^{27}	8.24×10^{10}	5.33×10^{10}	2.25×10^{27}
	Std.	2.19×10^{10}	3.67×10^{10}	6.09×10^{27}	7.23×10^9	2.60×10^{10}	7.30×10^{27}	2.17×10^{11}	1.68×10^{11}	2.87×10^{27}
C04	Mean	4.11×10^2	4.09×10^2	1.89×10^{18}	3.62×10^2	3.77×10^2	2.51×10^{20}	4.42×10^2	4.10×10^2	1.14×10^{19}
	Std.	7.13×10^1	6.05×10^1	5.91×10^{18}	4.03×10^1	6.23×10^1	6.87×10^{20}	9.32×10^1	6.26×10^1	2.34×10^{19}
C05	Mean	2.14×10^1	1.89×10^1	3.65×10^{23}	2.60×10^1	1.88×10^1	2.96×10^{23}	2.81×10^1	2.39×10^1	1.35×10^{23}
	Std.	5.65×10^1	5.44×10^1	4.85×10^{23}	1.56×10^1	4.17×10^1	5.70×10^{23}	1.95×10^1	5.46×10^1	2.02×10^{23}
C06	Mean	4.11×10^{18}	9.90×10^{12}	8.61×10^{21}	6.17×10^3	5.58×10^3	1.70×10^{22}	5.27×10^3	4.44×10^3	7.76×10^{21}
	Std.	1.23×10^{19}	4.43×10^{13}	6.56×10^{21}	1.55×10^3	1.03×10^3	1.23×10^{22}	7.52×10^2	8.20×10^2	6.83×10^{21}
C07	Mean	3.69×10^{21}	2.08×10^{21}	1.71×10^{25}	4.31×10^{21}	-1.48×10^2	1.32×10^{25}	1.27×10^{21}	1.22×10^{21}	1.15×10^{25}
	Std.	5.18×10^{21}	4.23×10^{21}	3.71×10^{24}	1.03×10^{22}	2.78×10^{21}	5.07×10^{24}	2.10×10^{21}	1.92×10^{21}	7.45×10^{24}
C08	Mean	7.43×10^{-4}	2.99×10^{-4}	3.98×10^{30}	4.15×10^{-4}	4.49×10^{-4}	4.53×10^{30}	4.91×10^{-4}	4.11×10^{-4}	2.36×10^{30}
	Std.	3.68×10^{-4}	1.77×10^{-4}	6.99×10^{30}	1.37×10^{-4}	2.76×10^{-4}	5.74×10^{30}	2.23×10^{-4}	2.45×10^{-4}	1.86×10^{30}
C09	Mean	5.65×10^1	4.38×10^1	1.23×10^{39}	7.76×10^1	5.45×10^1	7.66×10^{38}	8.71×10^1	7.24×10^1	3.00×10^{36}
	Std.	3.41×10^1	3.09×10^1	3.88×10^{39}	2.58×10^1	3.34×10^1	2.42×10^{39}	4.49×10^1	3.64×10^1	7.15×10^{36}
C10	Mean	2.71×10^{-4}	2.88×10^{-4}	5.62×10^{31}	2.55×10^{-4}	2.22×10^{-4}	1.13×10^{32}	2.40×10^{-4}	2.71×10^{-4}	2.62×10^{31}
	Std.	9.89×10^{-5}	1.09×10^{-4}	6.58×10^{31}	6.51×10^{-5}	1.16×10^{-4}	2.65×10^{32}	6.58×10^{-5}	8.95×10^{-5}	2.35×10^{31}
C11	Mean	1.58×10^{20}	2.08×10^{20}	1.43×10^{119}	6.95×10^{19}	8.64×10^{19}	1.29×10^{124}	6.65×10^{18}	1.01×10^{20}	1.40×10^{126}
	Std.	2.03×10^{20}	5.87×10^{20}	4.51×10^{119}	8.82×10^{19}	1.92×10^{20}	4.08×10^{124}	1.34×10^{19}	1.70×10^{20}	4.41×10^{126}
C12	Mean	2.11×10^1	1.73×10^1	3.81×10^{28}	2.50×10^1	4.00×10^{-1}	3.78×10^{28}	1.50×10^1	1.25×10^1	3.71×10^{28}
	Std.	1.32×10^1	1.02×10^1	1.50×10^{28}	1.59×10^1	1.21×10^1	1.66×10^{28}	9.72×10^1	4.47×10^1	1.34×10^{28}
C13	Mean	6.51×10^{22}	4.21×10^{22}	2.99×10^{28}	7.12×10^{22}	4.64×10^{22}	3.48×10^{28}	6.64×10^{22}	6.41×10^{22}	3.43×10^{28}
	Std.	4.74×10^{22}	4.26×10^{22}	1.09×10^{28}	6.24×10^{22}	3.88×10^{22}	1.56×10^{28}	3.85×10^{22}	3.48×10^{22}	1.70×10^{28}
C14	Mean	1.41×10^1	1.42×10^1	7.02×10^{28}	1.41×10^1	1.41×10^1	8.18×10^{28}	1.42×10^1	1.41×10^1	5.92×10^{28}
	Std.	2.24×10^{-13}	3.18×10^{-2}	2.69×10^{28}	1.26×10^{-13}	3.77×10^{-14}	3.25×10^{28}	2.75×10^{-2}	9.62×10^{-13}	1.87×10^{28}
C15	Mean	2.03×10^1	2.06×10^1	3.27×10^{28}	2.21×10^1	1.96×10^1	3.48×10^{28}	2.25×10^1	2.12×10^1	3.28×10^{28}
	Std.	2.98×10^1	2.19×10^1	1.41×10^{28}	3.33×10^1	3.75×10^1	1.09×10^{28}	3.69×10^1	4.91×10^1	9.69×10^{28}
C16	Mean	1.84×10^2	1.61×10^2	3.71×10^{28}	1.82×10^2	1.65×10^2	3.25×10^{28}	1.89×10^2	1.81×10^2	2.74×10^{28}
	Std.	1.62×10^1	2.11×10^1	1.57×10^{28}	1.84×10^1	1.99×10^1	1.25×10^{28}	1.00×10^1	1.86×10^1	1.00×10^{28}
C17	Mean	9.61×10^{20}	9.61×10^{20}	3.99×10^{28}	9.61×10^{20}	9.61×10^{20}	4.05×10^{28}	9.61×10^{20}	9.61×10^{20}	3.09×10^{28}
	Std.	0	0	1.69×10^{28}	0	0	1.31×10^{28}	0	0	1.27×10^{28}

Table 7. Cont.

Function	Evaluation Indicator	E1	E2	E3	E4	E5	E6	E7	E8	E9
C18	Mean	4.91×10^{22}	5.92×10^{22}	9.95×10^{40}	4.86×10^{22}	3.65×10^1	6.05×10^{40}	8.93×10^{22}	5.57×10^{22}	6.46×10^{40}
	Std.	1.21×10^{23}	1.24×10^{23}	4.81×10^{40}	1.33×10^{23}	1.24×10^{23}	4.54×10^{40}	1.69×10^{23}	1.32×10^{23}	5.26×10^{40}
C19	Mean	1.84×10^{27}	1.84×10^{27}	1.85×10^{27}	1.84×10^{27}	1.84×10^{27}	1.85×10^{27}	1.84×10^{27}	1.84×10^{27}	1.85×10^{27}
	Std.	1.67×10^{24}	1.96×10^{24}	1.47×10^{23}	1.77×10^{24}	1.72×10^{24}	5.70×10^{23}	1.29×10^{24}	8.66×10^{23}	3.52×10^{23}
C20	Mean	2.98×10^1	2.51×10^1	3.19×10^{17}	2.64×10^1	2.48×10^1	1.80×10^{17}	2.70×10^1	2.77×10^1	1.20×10^1
	Std.	5.29×10^1	3.78×10^{-1}	6.85×10^{17}	3.78×10^{-1}	2.76×10^{-1}	5.69×10^{17}	3.73×10^{-1}	5.49×10^{-1}	1.72×10^1
C21	Mean	1.71×10^1	2.16×10^1	2.29×10^{28}	2.46×10^1	1.45×10^1	3.00×10^{28}	1.86×10^1	2.11×10^1	2.41×10^{28}
	Std.	7.92×10^1	1.24×10^1	6.66×10^{27}	1.29×10^1	9.78×10^1	1.25×10^{28}	1.16×10^1	1.32×10^1	1.17×10^{28}
C22	Mean	4.67×10^{20}	4.88×10^{20}	2.60×10^{28}	5.54×10^{20}	7.41×10^{20}	3.96×10^{28}	4.55×10^{20}	4.29×10^{20}	2.02×10^{28}
	Std.	3.00×10^{20}	4.12×10^{20}	1.71×10^{28}	6.33×10^{20}	5.04×10^{20}	1.13×10^{28}	3.76×10^{20}	2.70×10^{20}	7.96×10^{27}
C23	Mean	1.41×10^1	1.41×10^1	4.62×10^{28}	1.41×10^1	1.41×10^1	5.08×10^{28}	1.41×10^1	1.42×10^1	5.59×10^{28}
	Std.	7.60×10^{-13}	1.94×10^{-2}	1.38×10^{28}	9.67×10^{-13}	1.94×10^{-2}	2.344×10^{28}	1.21×10^{-11}	2.75×10^{-2}	1.54×10^{28}
C24	Mean	2.06×10^1	2.01×10^1	2.374×10^{28}	2.18×10^1	1.95×10^1	2.564×10^{28}	2.12×10^1	2.06×10^1	2.104×10^{28}
	Std.	1.99×10^1	2.55×10^1	1.05×10^{28}	2.89×10^1	2.38×10^1	1.01×10^{28}	2.57×10^1	1.99×10^1	1.01×10^{28}
C25	Mean	1.82×10^2	1.77×10^2	2.65×10^{28}	1.91×10^2	1.66×10^2	2.07×10^{28}	1.92×10^2	1.83×10^2	2.27×10^{28}
	Std.	2.45×10^1	1.34×10^1	1.26×10^{28}	2.24×10^1	1.49×10^1	8.76×10^{27}	1.67×10^1	1.26×10^1	8.05×10^{27}
C26	Mean	9.61×10^{20}	9.61×10^{20}	2.60×10^{28}	9.61×10^{22}	9.61×10^{20}	2.24×10^{28}	9.61×10^{20}	9.61×10^{20}	2.49×10^{28}
	Std.	0	0	1.29×10^{28}	0	0	3.77×10^{27}	0	0	1.04×10^{28}
C27	Mean	6.55×10^{23}	5.70×10^{22}	5.51×10^{40}	9.31×10^{22}	3.65×10^1	6.38×10^{40}	7.71×10^{22}	1.15×10^{23}	6.07×10^{40}
	Std.	1.08×10^{23}	1.19×10^{23}	3.45×10^{40}	1.77×10^{23}	1.60×10^{23}	4.76×10^{40}	1.18×10^{23}	1.70×10^{23}	3.61×10^{40}
C28	Mean	1.85×10^{27}	1.85×10^{27}	1.85×10^{27}	1.85×10^{27}	1.85×10^{27}	1.85×10^{27}	1.85×10^{27}	1.85×10^{27}	1.85×10^{27}
	Std.	1.49×10^{24}	1.70×10^{23}	4.75×10^{23}	6.63×10^{23}	1.63×10^{24}	3.01×10^{23}	1.14×10^{24}	1.19×10^{24}	5.28×10^{23}

Table 8. Friedman test results of orthogonal experiment.

Dimension	Significant Level	k	χ^2	$\chi^2_{\alpha[k-1]}$	p -Value	Null Hypothesis	Alternative Hypothesis
$D = 30$	$A = 0.05$	9	160.54	15.51	1.23487×10^{-30}	Reject	Accept

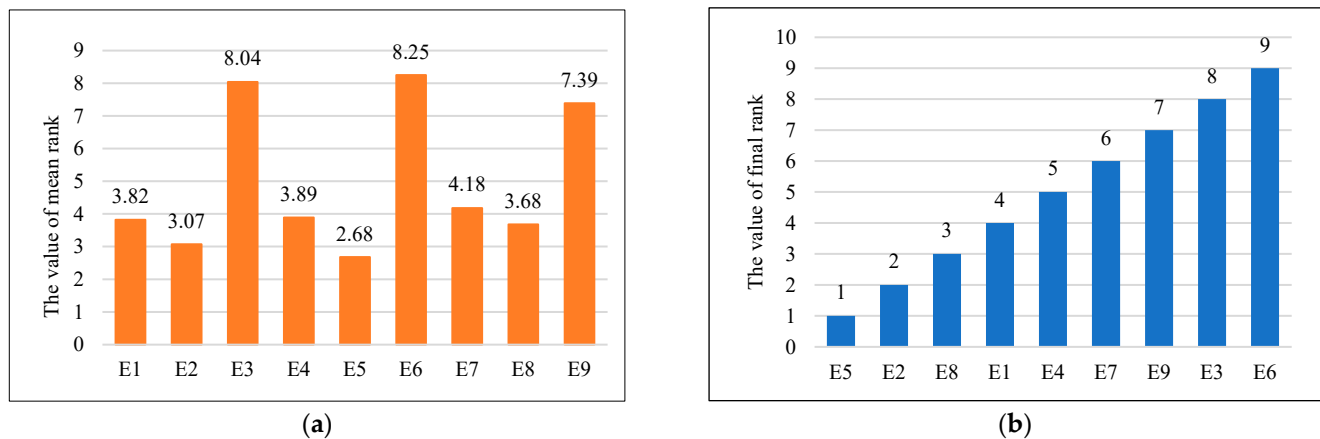


Figure 7. Results of the Friedman mean ranking test of various experiment number. (a) Mean rank ranking; (b) final rank ranking.

4.4. Termination Condition of Algorithm Iteration

There are two commonly used iteration termination conditions at present. One is taking the maximum number of iterations as the termination condition and the other is taking the maximum evaluation times as the termination condition. However, the two iteration termination conditions are not fair to other algorithms participating in the comparison. The specific reasons are as follows:

1. Taking the maximum number of iterations as the iteration termination condition, it is unfair to the algorithms that participate in the comparison. Let the time required for an iteration of algorithm A be t_1 , the time required for one iteration of algorithm B is t_2 , and $t_1 > t_2$. Let $t_1 = 1.2t_2$, $t_2 = 0.005$ s, the maximum number of iterations is 1000. When both algorithm A and algorithm B reach the maximum number of iterations, algorithm A needs 6 s and algorithm B needs 5 s. The running time of A is 1 s longer than the running time of B . If the solution quality of A is better than that of B , it cannot be said that the performance of algorithm A is better than that of algorithm B . Because the running time of A is long, if algorithm B runs for another 1 s, the quality of the solution from algorithm B is not necessarily worse than that of algorithm A .
2. Taking the maximum evaluation times of the objective function as the termination condition of the algorithm iteration, it is also unfair to the algorithms involved in the comparison. The main reasons are: some algorithms in an iteration process, although the evaluation times of the objective function are few, the running time of the program is very long; some algorithms in an iteration process, although there are many evaluation times of the objective function, but the running time of the program is very short. Therefore, taking the maximum evaluation times of the objective function as the iteration termination condition is unfair to some algorithms.

In summary, to fairly compare various algorithms, the maximum running time *Maxtime* can be used as the iteration termination condition of the algorithm. Therefore, *Maxtime* is taken as the termination condition of algorithm iteration in this paper. When the program's running time reaches *Maxtime*, the algorithm stops iterating and outputs the optimal solution and value. The advantages of taking the maximum running time *Maxtime* as the termination condition of algorithm iteration are: regardless of whether the time and space complexity of different algorithms are the same, it is fair to the algorithms participating in the comparison.

4.5. Performance Comparison of Different Attraction Models

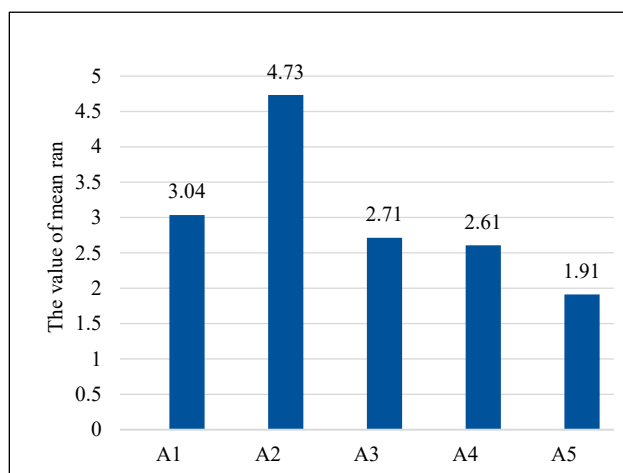
To verify that the performance of the probability attraction model is better than other attraction models, the CAM, RAM, NAM and GAM were selected to compare their performance with the probability attraction model. In addition, to ensure the fairness of the comparison, IHFAPA uses a different model of attraction and the rest is the same. Suppose population size $n = 40$, dimension of variable $D = 30$, maximum running time $Maxtime = 20$ s, maximum attraction $\beta_{max} = 1$, minimum attraction $\beta_{min} = 0.5$, light-absorption coefficient $\gamma = 1$, penalty factors corresponding to equality constraints and inequality constraints $M_1 = M_2 = 10^8$. The test function is CEC 2017; IHFAPA with different attractors was independently run 20 times. Run 20 times independently to obtain the mean value of the optimal value of the test function and the standard deviation of the optimal value, as shown in Table 9. The results of Friedman's mean rank and final rank ordering for different attraction models are given in Figure 8. The Friedman test results of different attraction models are shown in Table 10.

Table 9. Statistical results of IHFAPA with different attraction models.

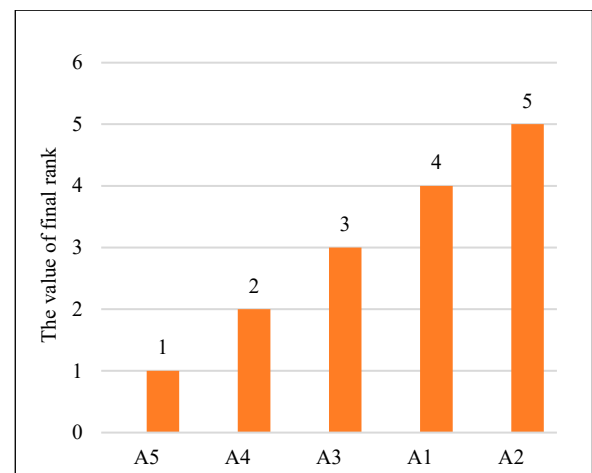
Test Functions	Performance Indicators	Complete Attraction Model	Random Attraction Model	Neighborhood Attraction Model	Grouping Attraction Model	Probability Attraction Model
C01	Mean	7.10×10^1	9.49×10^3	1.98×10^{-2}	1.08×10^1	6.17×10^{-15}
	Std	5.79×10^3	4.47×10^4	5.63×10^{-2}	2.06×10^2	8.21×10^{-15}
C02	Mean	2.36×10^1	9.52×10^3	2.58×10^{-2}	3.29×10^1	2.52×10^{-14}
	Std	3.03×10^3	2.58×10^4	1.07×10^{-1}	8.15×10^1	5.99×10^{-14}
C03	Mean	9.34×10^4	2.67×10^6	3.38×10^5	9.99×10^5	9.86×10^9
	Std	3.41×10^6	5.89×10^6	1.90×10^{10}	4.80×10^6	2.60×10^{10}
C04	Mean	2.34×10^2	4.07×10^2	1.21×10^2	5.33×10^2	3.77×10^2
	Std	5.88×10^2	6.63×10^2	2.22×10^2	6.62×10^2	6.23×10^1
C05	Mean	1.85×10^1	2.54×10^2	2.18×10^1	1.55×10^1	1.88×10^1
	Std	7.52×10^1	5.40×10^3	1.01×10^2	2.30×10^1	1.17×10^1
C06	Mean	2.42×10^{12}	3.44×10^3	4.36×10^{13}	3.91×10^3	5.58×10^3
	Std	9.93×10^{12}	3.20×10^{11}	1.56×10^{18}	2.26×10^{20}	1.03×10^3
C07	Mean	6.17×10^{21}	1.20×10^{21}	-1.43×10^2	-1.93×10^2	-1.48×10^2
	Std	7.58×10^{20}	1.01×10^{24}	9.70×10^{22}	9.17×10^{23}	2.78×10^{21}
C08	MEan	7.02×10^{-4}	1.87×10^{25}	2.58×10^{13}	5.22×10^{16}	4.49×10^{-4}
	Std	4.80×10^{20}	3.27×10^{26}	3.63×10^{14}	3.65×10^{18}	2.76×10^{-4}
C09	Mean	2.28×10^1	3.79×10^1	6.65×10^1	-2.65×10^{-3}	5.45×10^1
	Std	1.50×10^1	1.99×10^{18}	1.91×10^1	1.40×10^1	1.34×10^1
C10	Mean	3.50×10^{-4}	2.73×10^{26}	4.64×10^{12}	2.61×10^{-4}	2.22×10^{-4}
	Std	6.75×10^{-4}	4.06×10^{27}	3.60×10^{13}	8.60×10^{-4}	1.16×10^{-4}
C11	Mean	6.74×10^{15}	1.01×10^{26}	1.04×10^{16}	5.62×10^{19}	8.64×10^{19}
	Std	1.26×10^{25}	3.51×10^{27}	8.81×10^{20}	7.25×10^{22}	1.92×10^{20}
C12	Mean	4.00×10^1	2.73×10^{23}	1.69×10^2	4.00×10^1	4.00×10^1
	Std	3.97×10^1	2.65×10^{25}	2.50×10^2	3.97×10^1	1.21×10^1
C13	Mean	2.71×10^{23}	4.27×10^{25}	4.85×10^2	8.35×10^{22}	4.64×10^{22}
	Std	2.37×10^{24}	2.73×10^{26}	7.45×10^{21}	1.59×10^{24}	3.88×10^{22}
C14	Mean	1.41×10^1	1.69×10^{24}	1.90×10^1	1.41×10^1	1.41×10^1
	Std	1.50×10^1	7.77×10^{25}	2.32×10^1	1.50×10^1	3.77×10^{-4}
C15	Mean	1.81×10^1	1.81×10^1	1.49×10^1	1.81×10^1	1.96×10^1
	Std	3.06×10^1	6.63×10^{24}	1.81×10^1	3.06×10^1	1.75×10^1
C16	Mean	1.76×10^2	2.20×10^2	6.91×10^1	2.07×10^2	1.65×10^2
	Std	2.40×10^2	1.63×10^{25}	1.19×10^2	2.47×10^2	1.99×10^1
C17	Mean	9.61×10^{20}	2.76×10^{24}	9.61×10^{20}	9.61×10^{20}	9.61×10^{20}
	Std	9.61×10^{20}	4.46×10^{26}	9.61×10^{20}	9.61×10^{20}	0
C18	Mean	3.65×10^1	1.01×10^{27}	5.59×10^{14}	3.65×10^1	3.65×10^1
	Std	2.75×10^{25}	6.34×10^{35}	2.57×10^{25}	2.39×10^{26}	1.24×10^{27}

Table 9. Cont.

Test Functions	Performance Indicators	Complete Attraction Model	Random Attraction Model	Neighborhood Attraction Model	Grouping Attraction Model	Probability Attraction Model
C19	Mean	1.84×10^{27}	1.85×10^{27}	1.84×10^{27}	1.85×10^{27}	1.84×10^{27}
	Std	1.85×10^{27}	1.85×10^{27}	1.85×10^{27}	1.85×10^{27}	1.72×10^{24}
C20	Mean	2.05×10^1	2.59×10^1	2.48×10^1	2.07×10^1	2.01×10^1
	Std	3.97×10^1	5.18×10^1	4.97×10^1	3.40×10^1	2.76×10^{-1}
C21	Mean	4.00×10^1	1.69×10^{24}	1.81×10^2	4.00×10^1	1.45×10^1
	Std	3.97×10^1	6.98×10^{25}	3.00×10^2	3.97×10^2	9.78×10^1
C22	Mean	3.56×10^{23}	5.42×10^{25}	2.91×10^4	5.63×10^{22}	7.41×10^{20}
	Std	3.02×10^{24}	3.17×10^{26}	5.11×10^{21}	1.33×10^{24}	5.04×10^{22}
C23	Mean	1.41×10^1	4.65×10^{24}	1.95×10^1	1.41×10^1	1.41×10^1
	Std	1.50×10^1	1.22×10^{26}	2.33×10^1	1.41×10^1	1.94×10^{-2}
C24	Mean	1.81×10^1	2.12×10^1	1.18×10^1	1.81×10^1	1.95×10^1
	Std	2.75×10^1	3.95×10^{25}	1.81×10^1	3.06×10^1	2.38×10^1
C25	Mean	1.88×10^2	2.26×10^2	7.54×10^1	1.95×10^2	1.66×10^2
	Std	2.45×10^2	1.40×10^{25}	1.45×10^2	2.51×10^2	1.49×10^1
C26	Mean	9.61×10^{20}	5.52×10^{27}	9.61×10^{20}	9.61×10^{20}	9.61×10^{20}
	Std	9.61×10^{20}	2.71×10^{26}	9.61×10^{20}	9.61×10^{20}	0
C27	Mean	3.66×10^1	1.20×10^{33}	3.83×10^{15}	3.65×10^1	3.65×10^1
	Std	3.35×10^{26}	5.39×10^{35}	3.08×10^{25}	3.09×10^{26}	1.60×10^{23}
C28	Mean	1.84×10^{27}	1.85×10^{27}	1.84×10^{27}	1.85×10^{27}	1.85×10^{27}
	Std	1.85×10^{27}	1.85×10^{27}	1.85×10^{27}	1.85×10^{27}	1.63×10^{25}
<i>w/t/l</i>		25/0/3	23/0/5	26/0/2	25/0/3	-



(a)



(b)

Figure 8. Friedman rank ranking results of various attraction models ($D = 30$). (a) Mean rank ranking; (b) final rank ranking. A1 = Complete attraction model; A2 = Random attraction model; A3 = Neighborhood attraction model; A4 = Grouping attraction model; A5 = Probability based attraction model.

Table 10. Friedman test results of various attraction models.

Dimension	Significance Level	k	χ^2	$\chi^2_{\alpha[k-1]}$	p -Value	Null Hypothesis	Alternative Hypothesis
$D = 30$	$\alpha = 0.05$	5	26.07	9.49	3.05747×10^{-5}	Reject	Accept

In Table 9, *Mean* is the average value of the optimal value of the test function obtained when each test function runs 20 times and *Std* is the standard deviation for the optimal value obtained. Table 9 indicates that the probability attraction model performs better than

the other attraction models in regard to winning frequency. As a result, the probability attraction model performs significantly better than other attraction models. Figure 8 demonstrates that the probability attraction model performs better than the other attraction models that were tested by placing first in both the mean rank and final rank among all the attraction models that were compared. In addition, Friedman test was selected to verify whether there were noticeable differences in the performance of the various attraction models involved in the comparison. According to Table 10, when $k = 5$, $\alpha = 0.05$, $\chi^2_{\alpha[4]}$ the degree of freedom $k - 1 = 4$ corresponds to $\chi^2_{\alpha[4]} = 9.49$, $\chi^2 = 26.07$, $\chi^2 > \chi^2_{\alpha[4]}$. Therefore, rejecting the original hypothesis shows that there are obvious differences in the performance of various attraction models involved in the comparison.

In conclusion, for the 28 30-D test functions in CEC 2017, the performance of the probabilistic attraction model is better than the other four models, which verifies the validity of the probabilistic attraction model.

4.6. Performance Comparison between IHFAPA and Other FAs

To verify the superiority of the IHFAPA algorithm, five different FA algorithms were selected for comparison. The selected algorithm is:

1. Standard firefly algorithm (SFA) [6];
2. Random-attracting firefly algorithm (RaFA) [19];
3. Neighborhood-attracting firefly algorithm (NaFA) [42];
4. Gender-difference firefly algorithm (GDFA) [47];
5. An adaptive Log Spiral levy firefly algorithm (ADIFA) [29];
6. Cauchy mutation of the Yin and Yang firefly algorithm (YYFA) [36];
7. Group-attraction hybrid firefly algorithm (GAHFA) [20].

4.6.1. Parameter Settings

To compare the performance of various algorithms fairly, suppose the population size of all algorithms is $n = 40$, maximum running time $Maxtime = 20$ s, penalty factor $M = 10^8$, the allowable error value of the equality constraint for constrained optimization problem $\theta = 10^{-4}$ and number of statistics $tjcs = 20$. The other parameters in the comparison algorithm adopt the values in the original literature, and the specific parameter values are shown in Table 11.

Table 11. Parameter settings of various FAs participating in the comparison.

Algorithm	Reference	Years	Parameters
SFA	[6]	2008	$\alpha = 0.2, \beta_0 = 1, \gamma = 1.0$
RaFA	[19]	2016	$\alpha \in [0,1], \beta_0 = 1, \gamma = 1$
NaFA	[42]	2017	$\alpha = 0.5, \gamma = 1, \beta_{min} = 0.2, \beta_0 = 1.0, k = 3$
GDFA	[47]	2019	$\beta_0 = 1, \gamma = 1$
ADIFA	[29]	2020	$a(0) = 0.2, \beta_0 = 1, \gamma = 1$
YYFA	[36]	2020	$a(0) = 0.2, \beta_{min} = 0.2, \beta_0 = 1, \gamma = 1, L = 800$
GAHFA	[20]	2021	$\alpha_0 = 0.3, \varphi_0 = 0.9, p_m = 0.7, F_{max} = 0.9, F_{min} = 0.1$
IHFAPA	-	-	$N = 40, \alpha_0 = 0.1, \phi_0 = 0.9, \beta_{max} = 1, \beta_{min} = 0.5, \gamma = 1, P_m = 1$

4.6.2. Statistical Results and Analysis

1. Statistical results

Calculation results of 28 benchmark functions in CEC 2017 of various firefly algorithms $D = 30$ and $D = 50$ are given in Tables 12 and 13, and the Friedman test results are shown in Tables 14 and 15. Holm's procedure was used to further verify the algorithm performance. The results of all pairwise comparisons are summarized in Tables 16 and 17. When $D = 30$ and $D = 50$, the ranking results of the Friedman average rank and final rank of various FA algorithms are shown in Figure 9.

Table 12. Statistical results of various FAs ($D = 30$).

Test Functions	Performance Indicators	SFA	RaFA	NaFA	GDFA	ADIFA	YYFA	GAHFA	IHFAPA
C01	Mean	6.48×10^5	9.23×10^2	5.12×10^5	2.90×10^4	5.12×10^2	2.59×10^3	3.00×10^{-5}	6.17×10^{-15}
	Std	0	1.92×10^2	4.25×10^5	3.95×10^3	1.70×10^2	2.24×10^3	6.73×10^{-6}	8.21×10^{-15}
C02	Mean	3.18×10^5	1.06×10^3	9.08×10^4	2.61×10^4	3.58×10^2	2.44×10^3	3.01×10^{-5}	2.52×10^{-14}
	Std	1.25×10^5	3.95×10^2	5.26×10^4	3.20×10^3	9.42×10^1	2.24×10^3	5.66×10^{-6}	5.99×10^{-14}
C03	Mean	1.08×10^{24}	9.18×10^5	3.83×10^6	1.34×10^5	4.86×10^5	1.70×10^6	3.70×10^4	9.86×10^9
	Std	2.77×10^{23}	6.53×10^5	1.36×10^6	5.94×10^6	2.37×10^5	1.30×10^6	1.04×10^4	2.60×10^{10}
C04	Mean	1.11×10^3	5.20×10^2	8.00×10^2	3.09×10^2	2.48×10^2	5.57×10^2	6.23×10^2	3.77×10^2
	Std	8.90×10^1	8.15×10^1	8.61×10^1	2.90×10^1	1.89×10^1	8.22×10^1	6.63×10^1	6.23×10^1
C05	Mean	6.71×10^6	1.73×10^5	1.32×10^6	1.98×10^5	4.02×10^1	4.94×10^3	7.71×10^1	1.88×10^1
	Std	1.13×10^6	1.02×10^5	3.01×10^5	6.21×10^4	1.45×10^1	7.65×10^3	4.83×10^1	4.17×10^1
C06	Mean	4.36×10^{24}	9.76×10^9	6.85×10^{10}	7.15×10^9	4.01×10^8	1.72×10^{10}	5.42×10^3	5.58×10^3
	Std	4.90×10^{23}	1.84×10^9	5.01×10^{10}	3.28×10^9	1.75×10^8	7.55×10^{10}	1.35×10^3	1.03×10^3
C07	Mean	2.00×10^{24}	6.66×10^{12}	5.97×10^{14}	9.80×10^{12}	8.85×10^4	9.03×10^{12}	1.85×10^{12}	-1.48×10^2
	Std	3.86×10^{18}	6.73×10^{12}	1.08×10^{14}	1.84×10^{13}	1.29×10^4	2.28×10^{12}	4.73×10^{12}	2.78×10^{21}
C08	Mean	2.00×10^{24}	3.30×10^{13}	1.58×10^{18}	2.10×10^{14}	4.69×10^9	4.76×10^{14}	1.16×10^3	4.49×10^{-4}
	Std	0	1.32×10^{13}	1.99×10^{18}	9.03×10^{13}	7.47×10^8	7.02×10^{14}	3.07×10^{-4}	2.76×10^{-4}
C09	Mean	1.00×10^{24}	1.71×10^{12}	3.00×10^{15}	1.25×10^6	1.29×10^5	6.95×10^1	1.62×10^6	5.45×10^1
	Std	1.02×10^{20}	1.37×10^{12}	1.89×10^{15}	1.14×10^6	5.76×10^4	6.25×10^{-1}	8.10×10^6	3.34×10^1
C10	Mean	2.02×10^{24}	2.20×10^{13}	1.32×10^{19}	1.14×10^{15}	2.08×10^6	3.22×10^{15}	8.66×10^{-4}	2.22×10^{-4}
	Std	5.20×10^{21}	6.18×10^{12}	1.95×10^{19}	5.77×10^{14}	4.96×10^6	9.17×10^{15}	1.42×10^{-4}	1.16×10^{-4}
C11	Mean	1.00×10^{24}	4.20×10^{13}	9.66×10^{17}	7.20×10^{14}	1.70×10^{11}	6.03×10^{14}	1.32×10^6	8.64×10^{19}
	Std	5.74×10^{20}	4.09×10^{13}	2.92×10^{17}	2.55×10^{14}	7.07×10^{10}	1.64×10^{15}	6.24×10^6	1.92×10^{20}
C12	Mean	1.00×10^{24}	7.96×10^{11}	3.26×10^{17}	5.10×10^{13}	2.05×10^2	5.23×10^{13}	1.89×10^1	1.00×10^1
	Std	2.26×10^{20}	1.54×10^{11}	8.73×10^{16}	1.95×10^{13}	9.29×10^{-1}	2.45×10^{14}	1.25×10^1	1.21×10^1
C13	Mean	1.32×10^{24}	1.11×10^{13}	3.67×10^{17}	7.08×10^{15}	1.08×10^{12}	3.54×10^{11}	3.10×10^{15}	4.64×10^{22}
	Std	4.76×10^{23}	4.05×10^{12}	7.39×10^{16}	1.87×10^{15}	1.70×10^{11}	5.19×10^{12}	1.91×10^{15}	3.88×10^{22}
C14	Mean	2.00×10^{24}	1.32×10^{12}	6.63×10^{17}	1.07×10^{14}	2.12×10^1	4.52×10^{13}	1.41×10^1	1.40×10^1
	Std	6.45×10^{20}	3.79×10^{11}	1.67×10^{17}	6.01×10^{13}	5.04×10^{-1}	1.72×10^{13}	1.74×10^{-1}	3.77×10^{-14}
C15	Mean	1.00×10^{24}	2.55×10^1	2.82×10^{17}	1.93×10^1	2.39×10^1	3.13×10^1	2.47×10^1	1.96×10^1
	Std	3.32×10^{20}	3.77×10^1	6.66×10^{17}	4.22×10^1	1.50×10^1	8.75×10^1	3.99×10^1	3.75×10^1
C16	Mean	1.00×10^{24}	2.34×10^2	3.17×10^{17}	1.70×10^2	1.56×10^2	3.16×10^{13}	2.39×10^2	1.65×10^2
	Std	1.84×10^{20}	1.00×10^1	6.46×10^{16}	2.34×10^1	1.86×10^1	1.58×10^{14}	1.10×10^1	1.99×10^1
C17	Mean	2.00×10^{24}	9.69×10^{10}	3.53×10^{17}	4.89×10^{15}	9.61×10^{10}	3.34×10^{12}	9.61×10^{10}	9.61×10^{20}
	Std	2.41×10^{20}	1.28×10^9	9.45×10^{16}	1.28×10^{15}	3.58×10^{-2}	1.52×10^{13}	4.75×10^{-2}	0

Table 12. Cont.

Test Functions	Performance Indicators	SFA	RaFA	NaFA	GDFA	ADIFA	YYFA	GAHFA	IHFAPA
C18	Mean	1.14×10^{30}	9.73×10^{19}	5.06×10^{28}	3.84×10^{24}	6.95×10^{11}	3.21×10^{17}	2.55×10^{11}	3.65×10^1
	Std	4.99×10^{29}	2.05×10^{20}	1.92×10^{28}	2.38×10^{24}	9.18×10^{10}	1.47×10^{18}	1.00×10^{12}	1.24×10^{24}
C19	Mean	1.00×10^{24}	1.84×10^{17}	1.85×10^{17}	1.85×10^{17}	1.84×10^{17}	1.84×10^{17}	1.85×10^{17}	1.84×10^{27}
	Std	6.60×10^{16}	7.59×10^{13}	3.76×10^{13}	3.92×10^{13}	4.58×10^{13}	6.94×10^{13}	1.06×10^{13}	1.72×10^{24}
C20	Mean	8.40×10^1	8.87×10^1	3.03×10^1	6.04×10^1	3.63×10^1	9.18×10^1	2.36×10^1	2.48×10^1
	Std	4.67×10^{-1}	2.44×10^{-1}	5.45×10^{-1}	1.09×10^1	1.27×10^{-1}	4.96×10^{-1}	3.41×10^{-1}	2.76×10^{-1}
C21	Mean	1.00×10^{24}	6.03×10^{11}	1.62×10^{17}	3.82×10^{13}	2.06×10^2	9.40×10^{13}	1.61×10^1	1.45×10^1
	Std	2.11×10^{20}	1.97×10^{11}	3.75×10^{16}	2.03×10^{12}	1.39×10^1	4.35×10^{13}	1.14×10^1	9.78×10^1
C22	Mean	1.00×10^{24}	1.06×10^{13}	1.91×10^{17}	8.16×10^{15}	1.53×10^{13}	5.92×10^{12}	2.44×10^{15}	7.41×10^{22}
	Std	2.92×10^{20}	3.56×10^{13}	5.36×10^{16}	1.92×10^{13}	3.34×10^{12}	8.28×10^{12}	8.79×10^{14}	5.04×10^{22}
C23	Mean	2.00×10^{24}	1.39×10^{12}	2.85×10^{17}	6.03×10^{14}	2.07×10^1	5.48×10^1	1.41×10^1	1.40×10^1
	Std	6.22×10^{20}	4.88×10^{11}	7.02×10^{16}	2.94×10^{14}	5.75×10^{-2}	1.31×10^{14}	1.74×10^{-2}	1.94×10^{-2}
C24	Mean	1.00×10^{24}	1.44×10^2	1.50×10^{17}	2.08×10^1	1.78×10^1	2.02×10^1	2.32×10^1	1.95×10^1
	Std	2.61×10^{20}	9.23×10^1	3.83×10^{16}	2.91×10^1	9.63×10^{-1}	3.23×10^1	3.94×10^1	2.38×10^1
C25	Mean	1.00×10^{24}	2.57×10^2	1.51×10^{17}	1.91×10^2	2.11×10^2	1.33×10^2	2.40×10^2	1.66×10^2
	Std	2.11×10^{20}	4.09×10^1	3.32×10^{16}	1.89×10^1	1.23×10^1	3.02×10^1	1.52×10^1	1.49×10^1
C26	Mean	2.00×10^{24}	9.63×10^{10}	1.58×10^{17}	5.19×10^{14}	9.61×10^{14}	2.39×10^{12}	9.61×10^{10}	9.61×10^{20}
	Std	2.15×10^{20}	2.60×10^8	4.21×10^{16}	1.56×10^{14}	3.05×10^{-1}	1.14×10^{13}	9.87×10^{-3}	0
C27	Mean	5.19×10^{29}	6.63×10^{19}	1.75×10^{28}	9.08×10^{24}	9.06×10^{11}	5.25×10^{19}	7.26×10^{11}	3.65×10^1
	Std	2.22×10^{29}	1.41×10^{20}	9.03×10^{27}	7.87×10^{24}	1.31×10^{12}	2.08×10^{20}	1.61×10^{12}	1.60×10^{24}
C28	Mean	1.00×10^{24}	1.85×10^{17}	1.85×10^{17}	1.85×10^{17}	1.85×10^{17}	1.85×10^{17}	1.85×10^{17}	1.85×10^{27}
	Std	6.50×10^{16}	1.73×10^{14}	6.35×10^{13}	3.81×10^{13}	5.57×10^{14}	1.22×10^{14}	1.61×10^{14}	1.63×10^{24}
<i>w/t/l</i>		28/0/0	28/0/0	28/0/0	27/0/1	16/0/12	25/0/3	20/0/10	-

Table 13. Calculation results of various algorithms ($D = 50$).

Test Functions	Performance Indicators	SFA	RaFA	NaFA	GDFA	ADIFA	YYFA	GAHFA	IHFAPA
C01	Mean	1.52×10^6	1.69×10^4	1.02×10^6	1.07×10^5	1.77×10^4	9.27×10^3	3.68×10^{-4}	1.21×10^{-5}
	Std	0	3.24×10^3	8.28×10^5	1.15×10^4	1.18×10^3	4.17×10^3	$5.81E \times 10^{-5}$	1.10×10^{-5}
C02	Mean	1.18×10^6	1.66×10^4	4.65×10^5	1.09×10^5	1.67×10^4	7.38×10^3	4.31×10^{-4}	6.35×10^{-5}
	Std	8.20×10^5	3.43×10^3	2.00×10^5	1.73×10^4	2.54×10^3	3.29×10^3	8.41×10^{-5}	5.02×10^{-5}
C03	Mean	1.16×10^{24}	1.69×10^6	1.32×10^7	5.52×10^5	1.05×10^6	8.78×10^6	1.31×10^5	2.15×10^5
	Std	3.74×10^{23}	9.10×10^5	1.50×10^7	2.79×10^5	4.44×10^5	8.51×10^6	6.00×10^4	4.47×10^4
C04	Mean	2.13×10^3	9.81×10^2	1.33×10^3	8.34×10^2	5.69×10^2	1.31×10^3	1.07×10^3	8.71×10^2
	Std	1.21×10^2	6.675×10^1	8.11×10^1	5.305×10^1	1.625×10^1	8.135×10^1	1.83×10^2	1.36×10^2
C05	Mean	1.76×10^7	4.26×10^3	3.37×10^6	1.66×10^6	3.40×10^5	4.38×10^4	4.22×10^1	5.345×10^1
	Std	1.95×10^6	3.72×10^3	3.08×10^5	2.87×10^5	1.68×10^6	4.78×10^4	1.655×10^1	2.845×10^1
C06	Mean	4.52×10^{24}	8.67×10^3	8.11×10^{10}	1.30×10^{10}	9.97×10^8	3.32×10^{10}	9.12×10^3	7.85×10^6
	Std	5.10×10^{23}	1.96×10^3	4.19×10^{10}	5.32×10^9	4.53×10^8	1.37×10^{10}	2.29×10^3	3.92×10^7
C07	Mean	2.00×10^{24}	3.05×10^{16}	2.32×10^{15}	1.36×10^{15}	9.01×10^7	4.58×10^{13}	2.54×10^{13}	5.41×10^{11}
	Std	5.94×10^{18}	7.59×10^{10}	2.28×10^{14}	1.94×10^{14}	1.18×10^8	1.16×10^{13}	4.33×10^{13}	5.91×10^{11}
C08	Mean	2.01×10^{24}	4.96×10^{15}	6.94×10^{18}	1.56×10^{16}	8.44×10^{13}	4.33×10^{15}	4.49×10^{-1}	2.31×10^{-3}
	Std	4.72×10^{21}	1.56×10^{15}	4.88×10^{18}	3.59×10^{15}	1.12×10^{13}	4.57×10^{15}	8.67×10^{-1}	3.72×10^{-4}
C09	Mean	1.00×10^{24}	4.77×10^1	1.90×10^{16}	2.27×10^{10}	8.42×10^6	1.47×10^7	3.03×10^1	2.32×10^1
	Std	1.90×10^{20}	1.77×10^1	1.06×10^{16}	1.29×10^{10}	2.16×10^6	2.20×10^7	1.40×10^1	8.08×10^1
C10	Mean	2.03×10^{24}	1.91×10^{14}	2.75×10^{19}	7.96×10^{16}	6.96×10^{10}	1.88×10^{16}	1.21×10^{-1}	6.4×10^{-4}
	Std	6.03×10^{21}	8.75×10^{13}	1.38×10^{19}	1.57×10^{16}	1.05×10^{10}	3.76×10^{16}	4.27×10^{-1}	1.29×10^{-4}
C11	Mean	1.01×10^{24}	3.64×10^{14}	3.81×10^{18}	1.22×10^{16}	2.58×10^{12}	3.53×10^{15}	1.24×10^{10}	2.33×10^{12}
	Std	1.01×10^{21}	1.70×10^{14}	7.93×10^{17}	3.73×10^{15}	5.14×10^{11}	1.13×10^{16}	2.36×10^{10}	3.14×10^{12}
C12	Mean	1.00×10^{24}	1.47×10^2	1.09×10^{18}	1.50×10^{15}	3.36×10^2	3.90×10^{14}	1.65×10^1	1.62×10^1
	Std	3.64×10^{20}	3.87×10^1	1.66×10^{17}	4.34×10^{14}	1.64×10^1	1.24×10^{15}	1.11×10^1	1.14×10^1
C13	Mean	1.17×10^{24}	9.25×10^{13}	1.07×10^{18}	1.46×10^{17}	1.86×10^{13}	2.34×10^{14}	9.83×10^{15}	8.65×10^{13}
	Std	3.74×10^{23}	4.85×10^{13}	2.13×10^{17}	3.51×10^{16}	1.44×10^{12}	5.75×10^{14}	4.23×10^{15}	4.13×10^{13}
C14	Mean	2.01×10^{24}	1.17×10^1	2.16×10^{18}	3.05×10^{15}	2.21×10^5	8.66×10^{13}	1.11×10^1	1.10×10^1
	Std	8.08×10^{20}	6.23×10^{-2}	3.09×10^{17}	1.14×10^{15}	3.33×10^5	1.91×10^{14}	7.98×10^{-6}	8.51×10^{-9}
C15	Mean	1.00×10^{24}	2.46×10^1	9.37×10^{17}	5.53×10^6	2.64×10^1	3.96×10^1	2.69×10^1	1.22×10^1
	Std	4.52×10^{20}	3.00×10^1	1.46×10^{17}	2.76×10^7	1.71×10^1	9.58×10^1	4.71×10^1	1.68×10^1
C16	Mean	1.00×10^{24}	3.94×10^2	9.97×10^{17}	3.92×10^2	2.78×10^2	2.22×10^2	3.98×10^2	2.83×10^2
	Std	3.16×10^{20}	2.28×10^1	2.13×10^{17}	4.82×10^1	1.38×10^1	3.78×10^1	1.66×10^1	2.62×10^1
C17	Mean	2.00×10^{24}	2.64×10^{11}	1.10×10^{18}	2.23×10^{17}	2.60×10^{11}	3.49×10^{14}	2.60×10^{11}	2.60×10^{11}
	Std	2.91×10^{20}	2.20×10^{-3}	1.52×10^{17}	3.43×10^{16}	1.44×10^{-2}	8.21×10^{14}	4.73×10^{-3}	1.11×10^{-2}

Table 13. Cont.

Test Functions	Performance Indicators	SFA	RaFA	NaFA	GDFA	ADIFA	YYFA	GAHFA	IHFAPA
C18	Mean	4.25×10^{30}	5.38×10^{18}	1.54×10^{29}	2.53×10^{27}	1.95×10^{13}	3.49×10^{19}	4.00×10^{12}	5.43×10^1
	Std	1.55×10^{30}	1.81×10^{19}	4.31×10^{28}	9.33×10^{26}	2.96×10^{12}	1.16×10^{20}	1.23×10^{13}	5.36×10^1
C19	Mean	1.00×10^{24}	5.27×10^{17}	5.28×10^{17}	5.28×10^{17}	5.28×10^{17}	5.28×10^{17}	5.28×10^{17}	5.27×10^{17}
	Std	6.48×10^{16}	2.99×10^{14}	1.10×10^{14}	5.24×10^{13}	9.26×10^{13}	1.64×10^{15}	3.06×10^{14}	4.02×10^1
C20	Mean	1.65×10^1	3.84×10^1	1.89×10^1	1.63×10^1	7.73×10^1	7.75×10^1	1.19×10^1	5.34×10^1
	Std	4.28×10^{-1}	5.03×10^{-1}	1.02×10^1	5.83×10^{-1}	1.47×10^1	6.47×10^{-1}	7.98×10^{-1}	2.07×10^{-1}
C21	Mean	1.00×10^{24}	3.49×10^6	7.12×10^{17}	3.49×10^{16}	3.35×10^2	1.12×10^{14}	1.98×10^1	1.61×10^1
	Std	3.93×10^{20}	1.74×10^7	1.20×10^{17}	7.00×10^{15}	1.21×10^1	3.95×10^{14}	1.18×10^1	1.14×10^1
C22	Mean	1.00×10^{24}	8.59×10^{13}	7.49×10^{17}	1.44×10^{17}	1.01×10^{14}	1.88×10^{14}	8.99×10^{15}	8.50×10^{13}
	Std	4.63×10^{20}	1.85×10^{13}	1.36×10^{17}	2.44×10^{16}	1.02×10^{13}	4.51×10^{14}	4.19×10^{15}	4.86×10^{13}
C23	Mean	2.01×10^{24}	3.64×10^8	1.43×10^{17}	7.53×10^{16}	1.86×10^1	1.67×10^{14}	1.10×10^1	1.10×10^1
	Std	8.48×10^{20}	1.05×10^9	2.46×10^{17}	1.43×10^{16}	7.82×10^{-1}	3.57×10^{14}	6.74×10^{-6}	2.48×10^{-8}
C24	Mean	1.00×10^{24}	2.39×10^1	6.44×10^{17}	2.01×10^2	1.96×10^1	2.33×10^1	2.52×10^1	2.30×10^1
	Std	3.90×10^{20}	2.54×10^1	1.44×10^{17}	3.93×10^2	1.53×10^1	3.60×10^1	3.33×10^1	3.28×10^1
C25	Mean	1.00×10^{24}	3.96×10^2	6.57×10^{17}	1.19×10^3	3.64×10^2	3.09×10^2	3.99×10^2	2.94×10^2
	Std	3.87×10^{20}	1.62×10^1	1.28×10^{17}	1.33×10^3	9.95×10^1	3.47×10^1	1.39×10^2	2.10×10^1
C26	Mean	2.00×10^{24}	2.60×10^{11}	7.07×10^{17}	2.16×10^{17}	2.61×10^{11}	6.59×10^{13}	2.616×10^{11}	2.60×10^{11}
	Std	2.94×10^{20}	1.73×10^{-3}	1.33×10^{17}	3.92×10^{16}	9.79×10^{-1}	1.86×10^{14}	4.78×10^{-3}	2.40×10^{-3}
C27	Mean	2.47×10^{30}	3.42×10^{20}	1.05×10^{29}	2.63×10^{27}	2.69×10^{13}	1.40×10^{22}	4.83×10^{10}	2.48×10^{12}
	Std	1.01×10^{30}	6.20×10^{20}	3.46×10^{28}	8.19×10^{26}	4.92×10^{12}	5.71×10^{22}	1.34×10^{11}	1.23×10^{13}
C28	Mean	1.00×10^{24}	5.28×10^{17}	5.28×10^{17}	5.28×10^{17}	5.28×10^{17}	5.28×10^{17}	5.28×10^{17}	5.28×10^{17}
	Std	6.31×10^{16}	2.56×10^{14}	1.37×10^{14}	7.90×10^{13}	7.17×10^{13}	2.19×10^{14}	3.17×10^{14}	2.90×10^{14}
<i>w/t/l</i>		28/0/0	28/0/0	28/0/0	27/0/1	25/0/3	27/0/1	16/0/12	-

Table 14. Friedman test results of various FAs ($D = 30$).

Dimension	Significance Level	k	χ^2	$\chi^2_{\alpha[k-1]}$	p -Value	Null Hypothesis	Alternative Hypothesis
$D = 30$	$\alpha = 0.05$	8	148.47	14.07	8.51889×10^{-29}	Reject	Accept

Table 15. Friedman test results of various FAs ($D = 50$).

Dimension	Significance Level	k	χ^2	$\chi^2_{\alpha[k-1]}$	p -Value	Null Hypothesis	Alternative Hypothesis
$D = 50$	$\alpha = 0.05$	8	151.31	14.07	2.15923×10^{-29}	Reject	Accept

Table 16. The unadjusted and adjusted p -values for IHFAPA and various FAs ($D = 30$).

Comparison	Unadjusted p -Value	Adjusted p -Value
IHFAPA vs. SFA	3.28×10^{-7}	4.69×10^{-8}
IHFAPA vs. RaFA	1.92×10^{-2}	9.60×10^{-3}
IHFAPA vs. NaFA	1.00×10^{-3}	2.00×10^{-4}
IHFAPA vs. GDFA	8.00×10^{-3}	2.67×10^{-3}
IHFAPA vs. ADIFA	1.00×10^{-5}	1.67×10^{-6}
IHFAPA vs. YYFA	1.60×10^{-3}	4.00×10^{-4}
IHFAPA vs. GAHFA	8.43×10^{-2}	8.43×10^{-2}

Table 17. The unadjusted and adjusted p -values for IHFAPA and various FAs ($D = 50$).

Comparison	Unadjusted p -Value	Adjusted p -Value
IHFAPA vs. SFA	4.06×10^{-14}	5.80×10^{-15}
IHFAPA vs. RaFA	3.02×10^{-3}	7.55×10^{-4}
IHFAPA vs. NaFA	8.38×10^{-13}	1.40×10^{-13}
IHFAPA vs. GDFA	6.01×10^{-9}	1.20×10^{-9}
IHFAPA vs. ADIFA	1.36×10^{-2}	4.53×10^{-3}
IHFAPA vs. YYFA	4.82×10^{-2}	2.41×10^{-2}
IHFAPA vs. GAHFA	9.34×10^{-2}	9.34×10^{-2}

2. Result analysis

(1) Analysis of the calculation results of the 30- D test function

Figure 9 reports that, when using the eight firefly algorithms to solve the C03 benchmark test function, GDFA produces the highest-quality solutions; similarly, YYFA produces the highest-quality results when addressing the two C13 and C25 benchmark test functions. The highest quality of the ADIFA solutions is achieved when solving the six test functions C07, C15, C16, C17, C24 and C26; when solving the other 19 benchmark functions, IHFAPA has the highest solution quality. According to the $w/t/l$ values in Table 12, IHFAPA has better performance than the seven other firefly algorithms. Specifically, IHFA-PA has 28 test functions with a mean value better than SFA, 27 test functions with a mean value better than RaFA, 28 test functions with a mean value better than NaFA, 28 test functions with a mean value better than NDFA and for 27 test functions, the mean is better than GDFA, the mean of 25 test functions is better than YYFA and the mean of 22 test functions is better than AD-IFA. From the Friedman ranking results of various FAs with $D = 30$ in (a) and (b) of Figure 9, IHFAPA ranks first among all the algorithms, indicating that IHFAPA's performance is superior to the comparison algorithm in this paper. In addition, the Friedman test was used to verify whether there were obvious differences in the performance of the various algorithms involved in the comparison. In Table 13, $k = 8$, $\alpha = 0.05$, $k - 1 = 7$, the critical value $\chi^2_{\alpha[7]} = 14.07$, inspection value $\chi^2 = 148.47$ and the inspection value χ^2 is

greater than the critical value $\chi^2_{\alpha[7]}$, so the null hypothesis is rejected, indicating that there are obvious differences in the performance of the eight algorithms. From Table 14, the results obtained from Holm's procedure show that the IHFAPA did not show a significant difference from the GAHFA for the adjusted p -value and unadjusted p -value, but IHFAPA is significantly due to GAHFA in other indicators. Compared to NaFA, RaFA, NDFA, GDFA, AD-IFA and YYFA, IHFAPA verified the algorithm's superior performance. The statistical test shows that the IHFAPA was significantly different from the other comparison algorithms, with all p -values less than 0.05 at a 95% confidence level. The analyses above demonstrate the outstanding performance of the IHFAPA.

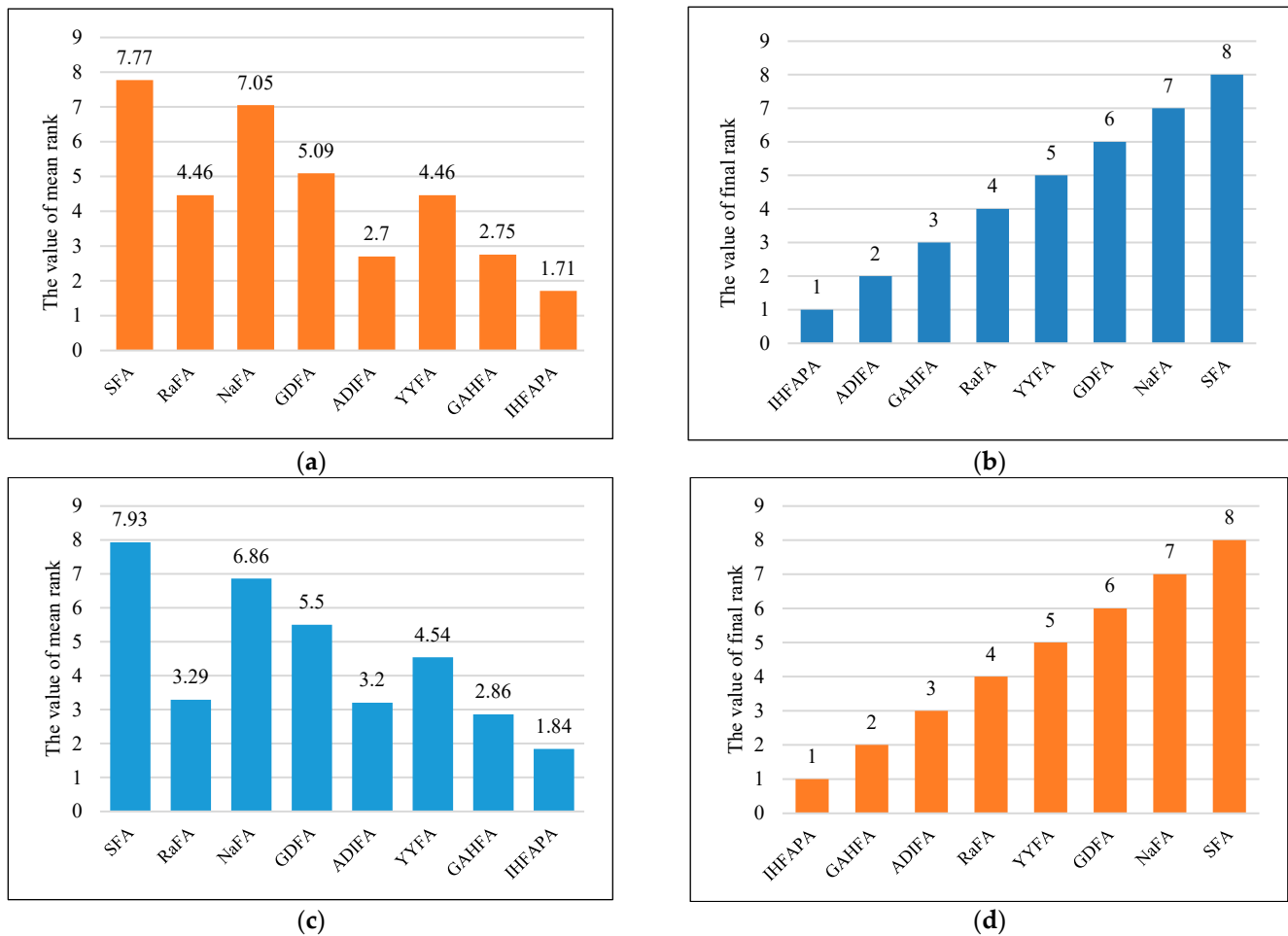


Figure 9. Friedman rank ranking results of various FAs. (a) Mean rank ranking ($D = 30$); (b) final rank ranking ($D = 30$); (c) mean rank ranking ($D = 50$); (d) final rank ranking ($D = 50$).

In summary, the experimental results of 28 30- D test functions in CEC 2017 show that the outperformance of IHFAPA over the other seven FAs verifies the effectiveness of IHFAPA.

(2) Analysis of calculation results of 50- D test function

Table 15 demonstrates that RaFA has the highest solution quality when eight firefly algorithms are used to solve the C06 benchmark function, GDFA has the highest solution quality when to solve the C03 and C25 benchmark functions, ADIFA has the highest solution quality when five test functions—C13, C15, C16, C24 and C26—are solved and IHFAPA has the highest solution quality when solving the remaining 20 benchmark functions. Figure 9's (c,d) results show that IHFAPA performs better than the other seven firefly algorithms. Specifically, IHFA-PA has 28 test functions with a mean value better than SFA, 25 test functions with a mean value better than RaFA, 28 test functions with a mean value better than NaFA, 28 test functions with a mean value better than NDFA, 26 test functions with an

average value better than GDFA, the average value of 28 test functions is better than YYFA and the average value of 22 test functions is better than AD-IFA. It can be seen from Table 15 that IHFAPA ranks first among all algorithms, indicating that IHFAPA's performance is superior to other algorithms involved in the comparison. In addition, Friedman test is used to verify whether there is a significant difference in the performance of the eight algorithms. According to Table 16, inspection value $\chi^2 = 151.31$, when $k = 8$, $\alpha = 0.05$, $k - 1 = 7$, the critical value $\chi^2_{\alpha[7]} = 14.07$. Because the inspection value χ^2 is greater than the critical value $\chi^2_{\alpha[7]}$, the original hypothesis is rejected, showing that there are significant differences in the performance of the eight algorithms. From Table 17, the results obtained from Holm's procedure show that the IHFAPA did not show a significant difference from the GAHFA for the adjusted p -value and unadjusted p -value, but IHFAPA is significantly superior to GAHFA in other indicators. IHFAPA confirmed the algorithm's superior performance when measured against the other comparative algorithms. The statistical test demonstrates that the IHFAPA differed from the other comparison algorithms in a significant way.

In summary, the experimental results of 28 50-D test functions in CEC 2017 show that the performance of IHFAPA is better than the other seven firefly algorithms, thus, verifying the effectiveness of IHFAPA.

4.6.3. Convergence Curve of Firefly Algorithm

To verify that IHFAPA's performance is superior to other FAs, four 30-D and 50-D test functions in CEC 2017 are selected and draw the convergence curves of eight FAs. The convergence curve of FA for solving four 30-D test functions is shown in Figure 10, and the convergence curve of FA for solving four 50-D test functions is shown in Figure 11.

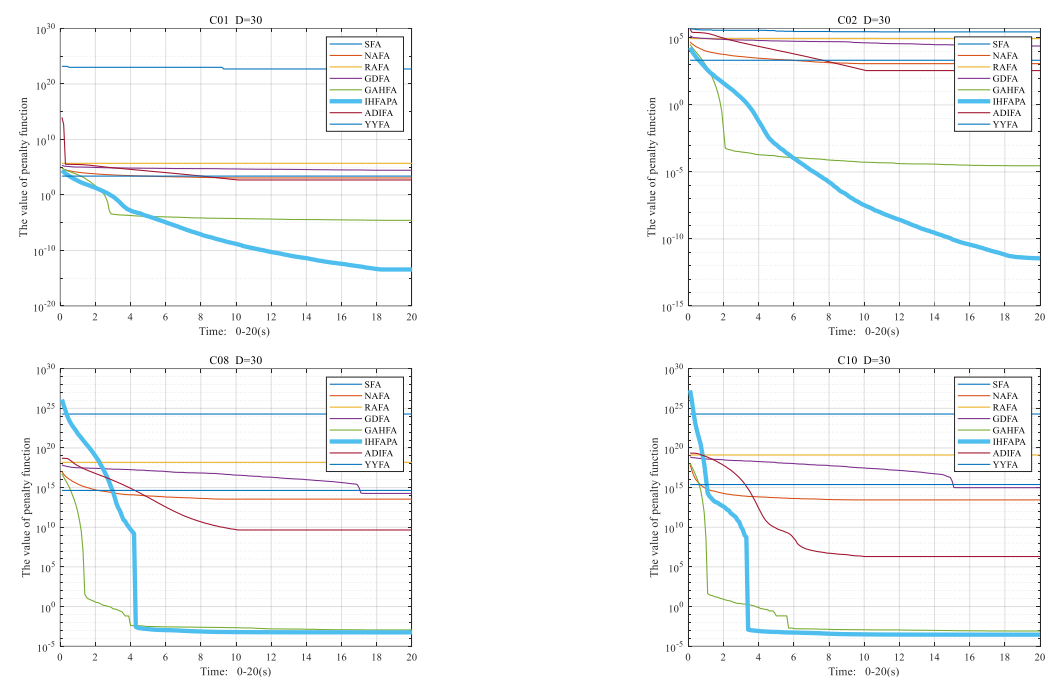


Figure 10. Convergence curves of FA for solving four 30-D test functions in CEC 2017.

According to Figure 10, for C01 and C02, IHFAPA converges to the global optimal solution much more quickly than the other six algorithms, with the exception of GAHFA, at the beginning of the iteration. At the end of the iteration, IHFAPA uses the combined mutation operator and adds the de-similarity operation, giving it a strong exploration ability. Therefore, IHFAPA has better solving quality than the other seven algorithms. For C08 and C10, the convergence speed of IHFAPA and GAHFA is obviously better than the other six algorithms. At the end of the iteration, IHFAPA and GAHFA rapidly converge to

the global extreme point, while the other six algorithms fall into local optimum. In addition, IHFAPA's solution accuracy is superior to GAHFA due to its strong exploitation ability.

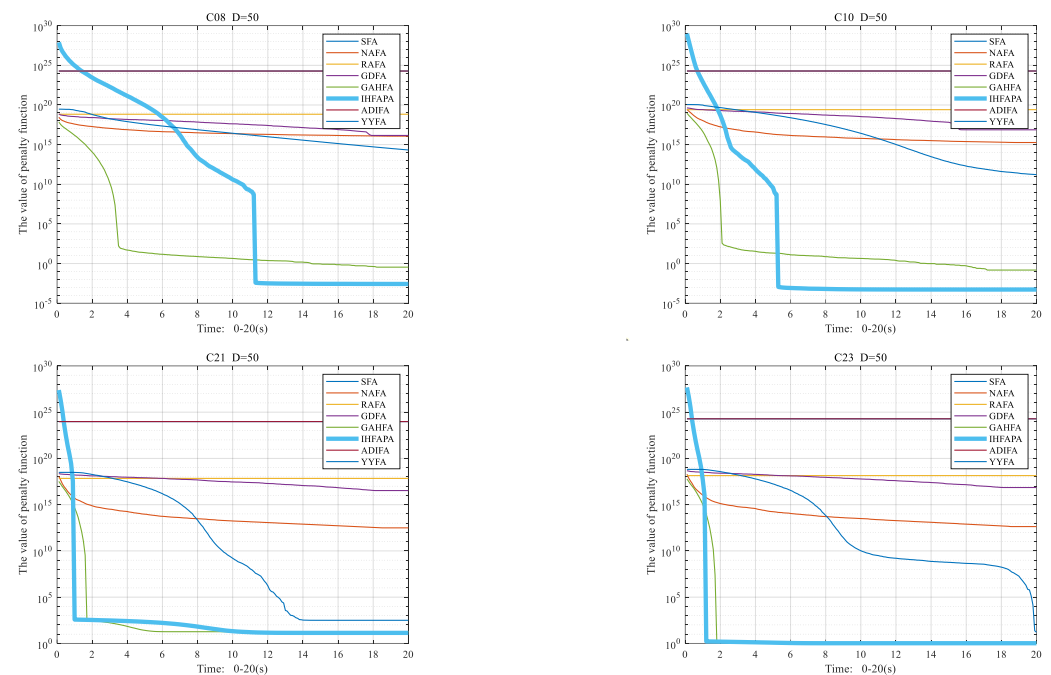


Figure 11. Convergence curves of FA for solving four 50-D test functions in CEC 2017.

Figure 11 illustrates, for C08 and C10, that IHFAPA and GAHFA converge faster than the other six algorithms in the early iteration; in late iterations, IHFAPA uses a combined mutation operator and adds the remove-similarity operation, so IHFAPA has strong exploration ability and can quickly move closer to the global optimal advantage, while the other seven algorithms have relatively weak exploration ability and quickly fall into local optimum. Therefore, IHFAPA's convergence speed and solving quality are better than those of the other seven algorithms. For C21 and C23, the convergence speed of IHFAPA and GAHFA is better than the other six algorithms in the early iteration stage; at the end of the iteration, except IHFAPA and GAHFA, the other six algorithms quickly fall into local optimum, while IHFAPA and GAHFA approached the global optimum quickly. In summary, IHFAPA is superior to other algorithms in convergence speed and solution quality.

4.7. Comparison of IHFAPA and Other Improved Algorithms

To further verify the performance of the IHFAPA, we selected six other improved algorithms for comparison. The six other improved algorithms selected are as follows:

1. Adaptive differential evolution algorithm (HDE) [47];
2. Improved sine and cosine algorithm with crossover operator (ISCA) [48];
3. Hybrid chicken swarm algorithm based on differential mutation (DMCSO) [49];
4. Particle swarm optimization based on oppositional group decision learning (OBLP-SOGD) [50];
5. Hybrid algorithm of improved bat algorithm and differential evolution algorithm (MBADE) [51];
6. Firefly single-objective genetic optimization algorithm based on partition and unity (MFAGA) [52];
7. Single-objective real-parameter optimization: Algorithm Jso (JSO) [53];

Select 28 test functions in CEC 2017 to compare the performance of IHFAPA, HDE, ISCA, DMCSO, OBLPSOGD, MBADE, MFAGA and JSO.

4.7.1. Parameter Setting

To compare the performance of various algorithms fairly, the population size of all algorithms $n = 40$, the maximum running time $Maxtime = 20$ s, the penalty factor $M_1 = M_2 = 10^8$, the allowable error of equality constraints in constraint optimization problem $\theta = 10^{-4}$ and the statistical times $tjcs = 20$. The values in the original literature were used for the other parameters of the comparison algorithm, and the specific parameter values are shown in Table 18.

Table 18. Other parameter settings of various algorithms participating in the comparison.

Algorithm	Reference	Years	Parameters
JSO	[53]	2017	$MF = 0.5, MCR = 0.8.$
HDE	[47]	2018	$F = [0.7, 1], CR = [0.4, 1].$
ISCA	[48]	2019	$CR = 0.3.$
DMCSO	[49]	2019	$R_N = 0.2n, H_N = 0.6n, C_N = 0.2n, M_N = 0.1n, F_L \in [0.4, 1], G = 10.$
OBLPSOGD	[50]	2019	$P_0 = 0.3, \alpha = 3.2, k = 15, \sigma = 0.3, w_{min} = 0.4, w_{max} = 0.9.$
MBADE	[51]	2020	$A_0 = 0.9, r_0 = 0.5, f_{max} = 2, f_{min} = 0, \alpha = \gamma = 0.9.$
MFAGA	[52]	2021	$\alpha = 4, \beta_0 = 1, \gamma = 2, w = 0.7.$
IHFAPA	-	-	$n = 40, \alpha_0 = 0.1, \phi_0 = 0.9, \beta_{max} = 1, \beta_{min} = 0.5, \gamma = 1, P_m = 1.$

4.7.2. Statistical Results and Analysis

1. Statistical results

When the variable dimension $D = 30$ and $D = 50$ of the 28 benchmark test functions in CEC 2017, the statistical results of the seven meta-heuristic algorithms are reported in Tables 19 and 20, the results of Friedman rank ranking are shown in (a) and (b) in Figure 12 and the results of Friedman test are shown in Tables 21 and 22. Holm's post hoc pairwise comparison results are in Tables 23 and 24.

2. Results analysis

(1) Analysis of the results of 30-D test functions

From Table 19, JSO obtains the optimal solution in the three test functions of C10, C11 and C22. When solving the benchmark functions C03, C06 and C28, DMCSO has the highest solution quality, whereas ISCA has the highest solution quality for C09 and C13. With regard to the five benchmark functions C07, C15, C16, C24 and C25, OBLPSOGD has the best solution quality. When solving the C19 benchmark function, MBADE has the highest solution quality. The solution quality of IHFAPA is the highest when the fourteen test functions C01, C02, C05, C08, C12, C14, C17, C18, C20, C21, C23, C26 and C27 are solved. Figure 12a,b show that IHFAPA ranks first among the seven meta-heuristic algorithms, indicating that IHFAPA's performance is better than the other seven meta-heuristic algorithms. In addition, Friedman test is used to verify whether there is a significant difference in the performance of the seven algorithms. According to Table 20, $k = 8, \alpha = 0.05, k - 1 = 7$, critical value $\chi^2_{\alpha[6]} = 14.07$ and inspection value $\chi^2 = 118.68$. Because the test value is greater than the critical value, the null hypothesis is rejected, showing that there are obvious differences in the performance of the seven algorithms. Table 21 reports that Holm's experimental results show that IHFAPA only shows no significant difference with JSO in the adjusted p -value and the unadjusted p -value, but IHFAPA is significantly better than JSO in other indicators. Statistical tests show that IHFAPA is significantly different from other comparison algorithms.

Table 19. Statistical results of various meta-heuristic algorithms ($D = 30$).

Test Functions	Performance Indicators	JSO	HDE	ISCA	DMCSO	OBLPSOGD	MBADE	MFAGA	IHFAPA
C01	Mean	3.77×10^1	4.41×10^3	5.97×10^3	5.49×10^3	1.51×10^2	3.05×10^3	7.94×10^1	6.17×10^{-15}
	Std	9.50×10^1	1.60×10^3	2.08×10^3	2.22×10^3	9.09×10^1	6.61×10^{-4}	1.80×10^1	8.21×10^{-15}
C02	Mean	2.01×10^1	3.67×10^3	4.86×10^3	3.81×10^3	3.30×10^2	2.03×10^3	8.06×10^1	2.52×10^{-14}
	Std	5.64×10^1	1.47×10^3	1.35×10^3	1.69×10^3	1.19×10^2	2.32×10^{-4}	2.28×10^1	5.99×10^{-14}
C03	Mean	5.39×10^5	3.22×10^5	8.41×10^4	5.00×10^4	6.92×10^4	9.86×10^4	8.33×10^7	1.56×10^5
	Std	3.73×10^5	5.13×10^5	2.66×10^4	1.25×10^4	4.05×10^4	2.18×10^4	4.15×10^7	1.74×10^5
C04	Mean	1.46×10^2	7.04×10^1	6.04×10^1	3.97×10^2	5.45×10^2	5.37×10^1	3.66×10^2	4.84×10^2
	Std	6.81×10^1	1.98×10^1	1.12×10^1	6.04×10^1	6.27×10^1	1.42×10^1	1.83×10^1	6.23×10^1
C05	Mean	1.18×10^1	3.09×10^4	2.04×10^2	3.82×10^4	3.49×10^2	5.89×10^2	7.23×10^3	7.64×10^1
	Std	7.78×10^1	5.61×10^4	1.80×10^2	4.30×10^4	3.29×10^2	1.53×10^1	1.83×10^3	4.17×10^1
C06	Mean	1.81×10^{10}	4.96×10^9	4.40×10^7	3.62×10^3	1.14×10^9	5.05×10^9	8.76×10^9	4.10×10^3
	Std	7.48×10^9	2.28×10^9	4.93×10^7	9.95×10^2	2.55×10^9	1.04×10^3	7.01×10^9	1.03×10^3
C07	Mean	1.55×10^{12}	8.79×10^9	4.48×10^2	2.18×10^9	-1.77×10^2	4.85×10^{11}	7.51×10^{12}	-1.48×10^2
	Std	4.76×10^{11}	1.46×10^{10}	1.03×10^3	5.96×10^9	1.61×10^2	1.12×10^9	5.23×10^{12}	2.78×10^{11}
C08	Mean	1.98×10^1	7.44×10^{15}	1.43×10^{15}	1.75×10^{13}	3.58×10^{11}	1.88×10^{15}	2.69×10^{11}	3.15×10^{-3}
	Std	5.20×10^1	4.92×10^{15}	1.20×10^{15}	6.57×10^{12}	3.47×10^{11}	1.43×10^{-3}	1.57×10^{11}	2.76×10^{-4}
C09	Mean	4.86×10^1	4.09×10^{11}	1.23×10^1	1.04×10^{10}	1.89×10^{11}	6.10×10^1	6.37×10^9	9.16×10^1
	Std	1.50×10^1	1.64×10^{12}	1.56×10^1	2.68×10^{10}	4.12×10^{11}	2.61×10^1	6.40×10^9	3.34×10^1
C10	Mean	6.25×10^{-4}	3.43×10^{16}	3.28×10^{15}	4.49×10^{14}	2.88×10^{12}	3.19×10^{15}	1.05×10^{12}	7.23×10^{-4}
	Std	4.33×10^{-4}	5.05×10^{16}	4.40×10^{15}	1.40×10^{14}	8.26×10^{12}	4.51×10^{-4}	2.81×10^{11}	1.16×10^{-4}
C11	Mean	2.49×10^{11}	8.86×10^{14}	2.78×10^{13}	3.65×10^{13}	2.94×10^{16}	3.79×10^{14}	2.05×10^{12}	4.64×10^{11}
	Std	4.21×10^{11}	1.39×10^{15}	5.57×10^{13}	3.77×10^{13}	2.05×10^{16}	6.97×10^{17}	1.57×10^{12}	1.92×10^{14}
C12	Mean	2.15×10^2	1.69×10^{14}	1.26×10^{12}	2.61×10^{13}	5.4710^{10}	2.37×10^2	2.77×10^{10}	4.00×10^1
	Std	1.95×10^1	1.58×10^{14}	2.57×10^{12}	1.81×10^{13}	7.4710^{10}	1.58×10^1	9.19×10^9	1.21×10^1
C13	Mean	5.6010^{10}	3.71×10^{14}	2.81×10^{12}	2.28×10^{13}	3.66×10^{12}	1.86×10^{13}	3.59×10^{12}	5.06×10^{12}
	Std	1.04×10^{11}	4.53×10^{14}	6.41×10^{12}	1.47×10^{13}	3.39×10^{12}	9.18×10^{13}	9.56×10^{11}	3.88×10^{12}
C14	Mean	2.17×10^1	6.85×10^{14}	3.64×10^{13}	3.86×10^{13}	8.0410^{10}	2.94×10^7	7.23×10^{10}	1.41×10^1
	Std	7.04×10^{-2}	8.40×10^{14}	1.09×10^{14}	2.01×10^{13}	1.09×10^{11}	2.06×10^{-7}	2.30×10^{10}	3.77×10^{-4}
C15	Mean	1.71×10^1	8.08×10^{13}	2.40×10^1	1.54×10^{13}	1.40×10^1	2.12×10^1	1.78×10^1	1.49×10^1
	Std	2.59×10^1	3.20×10^{14}	5.04×10^1	1.42×10^{13}	1.65×10^1	5.10×10^1	1.93×10^1	3.75×10^1
C16	Mean	1.65×10^2	3.08×10^{12}	1.74×10^2	2.15×10^{13}	1.35×10^2	2.16×10^2	1.57×10^2	1.90×10^2
	Std	1.38×10^1	1.00×10^{13}	1.25×10^1	1.49×10^{13}	9.81×10^1	1.74×10^1	2.06×10^1	1.99×10^1
C17	Mean	9.6110^{10}	4.12×10^{14}	1.58×10^{14}	2.42×10^{13}	9.6110^{10}	9.61×10^{10}	9.61×10^{10}	9.6110^{10}
	Std	3.53×10^1	6.49×10^{14}	6.57×10^{14}	1.56×10^{13}	2.12×10^{-2}	5.53×10^{-2}	3.14×10^3	0

Table 19. Cont.

Test Functions	Performance Indicators	JSO	HDE	ISCA	DMCSO	OBLPSOGD	MBADE	MFAGA	IHFAPA
C18	Mean	5.10×10^{10}	4.07×10^{25}	1.19×10^{20}	1.84×10^{24}	7.19×10^{17}	8.77×10^{14}	2.98×10^{16}	3.65×10^1
	Std	1.61×10^{11}	1.05×10^{26}	2.46×10^{20}	1.77×10^{24}	1.68×10^{18}	1.24×10^{11}	2.28×10^{16}	1.24×10^{11}
C19	Mean	1.83×10^{17}	1.85×10^{17}	1.83×10^{17}	9.21×10^{13}	1.84×10^{17}	1.83×10^{17}	1.85×10^{17}	1.84×10^{17}
	Std	2.27×10^3	1.01×10^{14}	1.59×10^{14}	8.29×10^{10}	1.98×10^{14}	6.26×10^{13}	1.67×10^{14}	1.72×10^{14}
C20	Mean	8.45×10^1	8.65×10^1	4.18×10^1	6.31×10^1	8.08×10^1	7.82×10^1	7.90×10^1	3.54×10^1
	Std	7.25×10^{-1}	5.34×10^{-1}	2.77×10^{-1}	7.15×10^{-1}	4.30×10^{-1}	1.35×10^1	6.00×10^{-1}	2.76×10^{-1}
C21	Mean	2.13×10^2	9.32×10^{13}	1.00×10^{12}	1.32×10^{13}	1.16×10^{12}	3.30×10^2	3.2410^{10}	1.46×10^1
	Std	2.59×10^1	1.08×10^{14}	1.88×10^{12}	7.07×10^{12}	2.72×10^{12}	3.08×10^1	1.010^{10}	9.78×10^1
C22	Mean	4.51×10^{11}	3.43×10^{14}	9.88×10^{12}	9.44×10^{12}	1.88×10^{13}	1.45×10^{13}	3.67×10^{12}	1.25×10^{12}
	Std	2.01×10^{11}	5.12×10^{14}	9.30×10^{12}	4.02×10^{12}	1.34×10^{13}	4.78×10^{13}	1.32×10^{12}	5.04×10^{12}
C23	Mean	2.01×10^1	3.19×10^{14}	2.73×10^{12}	2.52×10^{13}	3.47×10^{12}	8.25×10^1	7.56×10^{12}	1.41×10^1
	Std	7.06×10^{-2}	3.18×10^{14}	5.63×10^{12}	1.24×10^{13}	6.38×10^{12}	1.40×10^{-1}	2.68×10^{12}	1.94×10^{-2}
C24	Mean	1.82×10^1	7.27×10^{12}	1.71×10^1	1.03×10^{13}	1.54×10^1	1.81×10^1	1.85×10^1	1.81×10^1
	Std	1.54×10^1	3.21×10^{13}	1.79×10^1	5.70×10^{12}	1.91×10^1	2.67×10^1	1.64×10^1	2.38×10^1
C25	Mean	1.72×10^2	2.78×10^{13}	1.95×10^2	9.87×10^{12}	1.42×10^2	1.95×10^2	1.47×10^2	1.82×10^2
	Std	1.50×10^1	9.57×10^{13}	1.78×10^1	8.68×10^{12}	1.25×10^1	2.21×10^1	1.78×10^1	1.49×10^1
C26	Mean	9.61×10^{10}	1.52×10^{14}	1.39×10^{12}	1.30×10^{13}	1.45×10^{11}	9.61×10^{12}	9.61×10^{12}	9.61×10^{12}
	Std	2.76×10^1	2.54×10^{14}	3.54×10^{12}	6.46×10^{12}	2.24×10^{11}	4.06×10^{-3}	1.43×10^3	0.00×10^1
C27	Mean	4.50×10^{13}	3.57×10^{22}	1.61×10^{18}	2.22×10^{23}	6.85×10^{17}	1.66×10^{16}	1.32×10^{16}	3.65×10^1
	Std	1.10×10^{14}	8.85×10^{22}	3.86×10^{18}	3.66×10^{23}	1.59×10^{18}	2.06×10^{13}	9.40×10^{15}	1.60×10^{13}
C28	Mean	1.84×10^{17}	1.85×10^{17}	1.84×10^{17}	9.23×10^{13}	1.84×10^{17}	1.84×10^{17}	1.85×10^{17}	1.85×10^{17}
	Std	4.59×10^{14}	6.40×10^{13}	1.49×10^{14}	4.69×10^{12}	3.39×10^{14}	1.88×10^{14}	2.05×10^{14}	1.63×10^{14}
<i>w/t/l</i>		25/0/3	28/0/0	26/0/2	25/0/3	23/0/5	26/0/2	25/0/3	-

Table 20. Calculation results of various meta-heuristic algorithms ($D = 50$).

Test Functions	Performance Indicators	JSO	HDE	ISCA	DMCSO	OBLPSOGD	MBADE	MFAGA	IHFAPA
C01	Mean	2.37×10^2	1.86×10^{16}	1.61×10^{16}	9.82×10^{13}	7.46×10^{13}	2.81×10^9	4.65×10^{12}	1.21×10^{-5}
	Std	3.43×10^2	1.16×10^{13}	7.94×10^7	1.91×10^{11}	1.72×10^{12}	9.51×10^1	2.91×10^{11}	1.10×10^{-5}
C02	Mean	1.62×10^3	3.51×10^{13}	2.04×10^8	2.00×10^{11}	2.90×10^{12}	3.82×10^1	2.47×10^{11}	6.35×10^{-5}
	Std	2.41×10^3	1.80×10^{17}	4.40×10^{17}	2.80×10^{15}	1.08×10^{14}	6.11×10^2	1.52×10^{13}	5.02×10^{-5}
C03	Mean	1.31×10^6	1.05×10^{17}	1.83×10^{17}	9.29×10^{14}	1.83×10^{14}	1.63×10^3	4.89×10^{12}	1.15×10^3
	Std	7.10×10^5	6.43×10^{15}	1.89×10^{15}	2.61×10^{14}	1.91×10^{17}	1.45×10^{10}	2.50×10^{13}	4.47×10^4
C04	Mean	3.20×10^2	7.11×10^{15}	2.04×10^{15}	1.70×10^{14}	7.90×10^{16}	1.55×10^{10}	1.51×10^{13}	8.71×10^2
	Std	1.16×10^2	1.83×10^{15}	1.15×10^{15}	1.67×10^{14}	1.24×10^{13}	1.28×10^2	3.60×10^{11}	1.36×10^2
C05	Mean	8.23×10^1	1.86×10^{15}	1.35×10^{15}	5.62×10^{13}	1.38×10^{13}	5.16×10^1	8.08×10^{10}	5.34×10^1
	Std	4.57×10^1	2.02×10^{15}	3.48×10^{15}	1.25×10^{14}	1.78×10^{14}	1.65×10^{15}	2.70×10^{13}	2.84×10^1
C06	Mean	4.69×10^1	1.96×10^{15}	4.30×10^{15}	5.42×10^{13}	1.13×10^{14}	1.30×10^{15}	3.42×10^{12}	7.85×10^6
	Std	1.13×10^1	3.16×10^{15}	2.45×10^{15}	3.68×10^{14}	2.05×10^{13}	1.20×10^1	7.87×10^{11}	3.92×10^7
C07	Mean	6.69×10^{12}	2.29×10^{15}	2.01×10^{15}	9.99×10^{13}	1.53×10^{13}	8.69×10^{-2}	1.22×10^{11}	5.41×10^{11}
	Std	9.39×10^{11}	4.60×10^{12}	2.17×10^{13}	1.37×10^{14}	1.93×10^1	3.24×10^1	2.11×10^1	5.91×10^{11}
C08	Mean	7.74×10^{14}	1.35×10^{13}	9.70×10^{13}	5.41×10^{13}	2.44×10^1	5.24×10^1	2.54×10^1	2.31×10^{-3}
	Std	1.50×10^{15}	8.01×10^{13}	4.33×10^{13}	1.45×10^{14}	2.78×10^2	3.44×10^2	2.82×10^2	3.72×10^{-4}
C09	Mean	6.23×10^9	2.56×10^{14}	1.90×10^{14}	4.02×10^{13}	1.37×10^1	2.16×10^1	2.81×10^1	1.32×10^1
	Std	1.95×10^{10}	2.48×10^{15}	1.82×10^{15}	1.98×10^{14}	1.51×10^{13}	2.60×10^{11}	2.60×10^{11}	8.08×10^1
C10	Mean	2.26×10^{10}	2.56×10^{15}	2.20×10^{15}	5.12×10^{13}	1.29×10^{13}	4.54×10^{-2}	1.04×10^4	6.47×10^{-4}
	Std	7.14×10^{10}	3.24×10^{26}	1.39×10^{25}	1.64×10^{25}	6.38×10^{20}	1.60×10^{11}	1.23×10^{18}	1.29×10^{-4}
C11	Mean	8.68×10^{12}	5.47×10^{26}	2.30×10^{25}	7.03×10^{24}	8.68×10^{20}	2.01×10^{11}	1.88×10^{18}	2.33×10^{12}
	Std	1.63×10^{13}	5.28×10^{17}	5.25×10^{17}	2.64×10^{14}	5.26×10^{17}	5.22×10^{17}	5.27×10^{17}	3.14×10^{12}
C12	Mean	3.07×10^2	2.94×10^{14}	2.95×10^{14}	1.32×10^{11}	2.78×10^{14}	1.28×10^{14}	3.18×10^{14}	1.62×10^1
	Std	3.70×10^1	1.75×10^1	1.03×10^1	1.41×10^1	1.59×10^1	1.29×10^1	1.59×10^1	1.14×10^1
C13	Mean	1.73×10^{12}	5.59×10^{-1}	6.98×10^{-1}	1.19×10^1	5.07×10^{-1}	3.00×10^1	6.79×10^{-1}	8.65×10^{13}
	Std	9.82×10^{11}	1.63×10^{15}	9.51×10^{14}	1.15×10^{14}	2.39×10^{13}	2.80×10^2	4.05×10^{11}	4.13×10^{13}
C14	Mean	6.40×10^4	1.56×10^{15}	8.48×10^{14}	3.03×10^{13}	1.94×10^{13}	4.08×10^1	1.13×10^{11}	1.10×10^1
	Std	1.46×10^5	2.07×10^{15}	1.45×10^{15}	1.18×10^{14}	3.96×10^{14}	3.91×10^{15}	2.62×10^{13}	8.51×10^{-9}
C15	Mean	2.01×10^1	1.72×10^{15}	9.71×10^{14}	4.11×10^{13}	1.88×10^{14}	3.24×10^{15}	6.20×10^{12}	2.22×10^1
	Std	1.31×10^1	3.07×10^{15}	2.04×10^{15}	2.49×10^{14}	5.46×10^{13}	1.54×10^1	7.97×10^{11}	2.68×10^1
C16	Mean	3.22×10^2	2.78×10^{15}	1.98×10^{15}	6.78×10^{13}	8.09×10^{13}	6.49×10^{-2}	1.75×10^{11}	2.83×10^2
	Std	1.51×10^1	1.55×10^{14}	6.46×10^{11}	9.51×10^{13}	1.82×10^1	2.65×10^1	2.04×10^1	2.62×10^1
C17	Mean	2.60×10^{11}	4.61×10^{14}	2.89×10^{12}	3.07×10^{13}	1.43×10^1	4.34×10^1	1.68×10^1	2.60×10^{11}
	Std	4.11×10^1	2.70×10^{13}	2.77×10^{12}	9.08×10^{13}	2.89×10^2	3.88×10^2	2.91×10^2	1.11×10^{-2}

Table 20. Cont.

Test Functions	Performance Indicators	JSO	HDE	ISCA	DMCSO	OBLPSOGD	MBADE	MFAGA	IHFAPA
C18	Mean	3.05×10^{15}	9.14×10^{13}	8.70×10^{12}	3.33×10^{13}	1.36×10^1	2.42×10^1	2.66×10^1	5.43×10^1
	Std	9.65×10^{15}	1.07×10^{15}	1.35×10^{15}	1.24×10^{14}	8.67×10^{14}	2.60×10^{11}	2.60×10^{11}	5.36×10^1
C19	Mean	5.22×10^{17}	7.57×10^{14}	1.71×10^{15}	3.88×10^{13}	4.27×10^{14}	2.40×10^{-3}	1.19×10^4	5.27×10^{17}
	Std	1.44×10^{13}	7.56×10^{23}	6.36×10^{23}	5.72×10^{24}	2.50×10^{21}	9.11×10^{10}	5.27×10^{18}	4.02×10^{14}
C20	Mean	1.72×10^1	1.68×10^{24}	1.08×10^{24}	4.52×10^{24}	2.94×10^{21}	1.96×10^{11}	4.46×10^{18}	5.34×10^1
	Std	6.51×10^{-1}	5.28×10^{17}	5.27×10^{17}	2.64×10^{14}	5.27×10^{17}	5.26×10^{17}	5.28×10^{17}	2.07×10^1
C21	Mean	3.33×10^2	1.28×10^{14}	4.47×10^{14}	7.44×10^{10}	2.05×10^{14}	4.11×10^{14}	3.46×10^{14}	1.61×10^1
	Std	2.66×10^1	1.86×10^{16}	1.61×10^{16}	9.82×10^{13}	7.46×10^{13}	2.81×10^9	4.65×10^{12}	1.14×10^1
C22	Mean	8.80×10^{12}	1.16×10^{13}	7.94×10^7	1.91×10^{11}	1.72×10^{12}	9.51×10^1	2.91×10^{11}	8.50×10^{13}
	Std	4.02×10^{12}	3.51×10^{13}	2.04×10^8	2.00×10^{11}	2.90×10^{12}	3.82×10^1	2.47×10^{11}	4.86×10^{13}
C23	Mean	1.63×10^1	1.80×10^{17}	4.40×10^{17}	2.80×10^{15}	1.08×10^{14}	6.11×10^2	1.52×10^{13}	1.10×10^1
	Std	6.93×10^{-2}	1.05×10^{17}	1.83×10^{17}	9.29×10^{14}	1.83×10^{14}	1.63×10^3	4.89×10^{12}	2.48×10^{-8}
C24	Mean	2.23×10^1	6.43×10^{15}	1.89×10^{15}	2.61×10^{14}	1.91×10^{17}	1.45×10^{10}	2.50×10^{13}	2.30×10^1
	Std	1.31×10^1	7.11×10^{15}	2.04×10^{15}	1.70×10^{14}	7.90×10^{16}	1.55×10^{10}	1.51×10^{13}	3.28×10^1
C25	Mean	3.15×10^2	1.83×10^{15}	1.15×10^{15}	1.67×10^{14}	1.24×10^{13}	1.28×10^2	3.60×10^{11}	2.94×10^2
	Std	6.68×10^1	1.86×10^{15}	1.35×10^{15}	5.62×10^{13}	1.38×10^{13}	5.16×10^1	8.08×10^{10}	2.10×10^1
C26	Mean	2.60×10^{11}	2.02×10^{15}	3.48×10^{15}	1.25×10^{14}	1.78×10^{14}	1.65×10^{15}	2.70×10^{13}	2.60×10^{11}
	Std	9.40×10^1	1.96×10^{15}	4.30×10^{15}	5.42×10^{13}	1.13×10^{14}	1.30×10^{15}	3.42×10^{12}	2.40×10^{-3}
C27	Mean	6.21×10^{12}	3.16×10^{15}	2.45×10^{15}	3.68×10^{14}	2.05×10^{13}	1.20×10^1	7.87×10^{11}	2.48×10^{12}
	Std	1.37×10^{13}	2.29×10^{15}	2.01×10^{15}	9.99×10^{13}	1.53×10^{13}	8.69×10^{-2}	1.22×10^{11}	1.23×10^{13}
C28	Mean	5.25×10^{17}	4.60×10^{12}	2.17×10^{13}	1.37×10^{14}	1.93×10^1	3.24×10^1	2.11×10^1	5.28×10^{17}
	Std	1.38×10^{15}	1.35×10^{13}	9.70×10^{13}	5.41×10^{13}	2.44×10^1	5.24×10^1	2.54×10^1	2.90×10^{14}
<i>w/t/l</i>		25/0/3	28/0/0	28/0/0	28/0/0	23/0/5	18/0/10	26/0/2	-

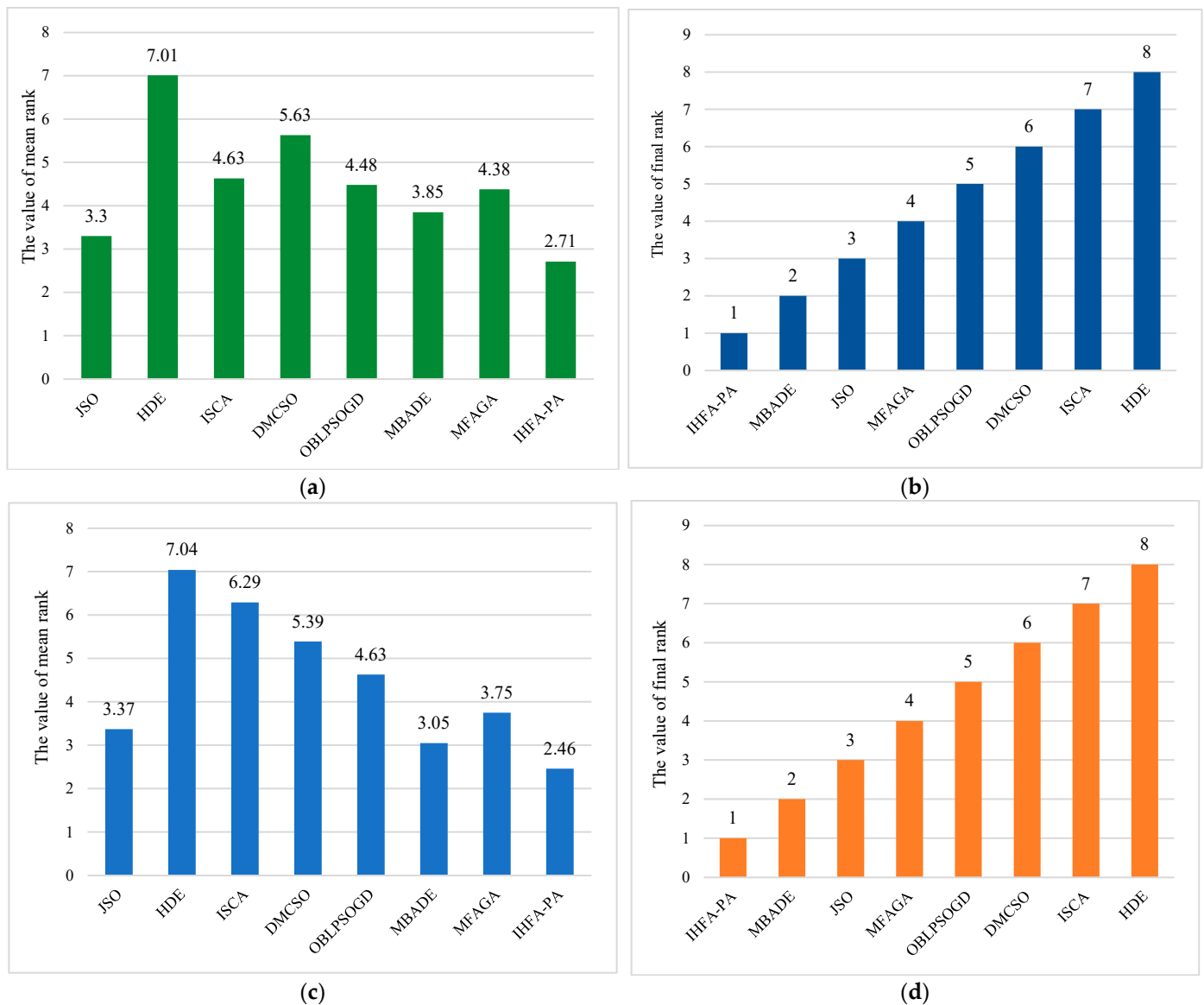


Figure 12. Friedman rank ranking results of different meta-heuristic algorithms. (a) Mean rank ranking ($D = 30$); (b) final rank ranking ($D = 30$); (c) mean rank ranking ($D = 50$); (d) final rank ranking ($D = 50$).

Table 21. The results of Friedman test for various meta-heuristic algorithms ($D = 30$).

Dimension	Significance Level	k	χ^2	$\chi^2_{\alpha[k-1]}$	p -Value	Null Hypothesis	Alternative Hypothesis
$D = 30$	$\alpha = 0.05$	8	118.68	14.07	1.4438×10^{-22}	Reject	Accept

Table 22. The results of Friedman test for various meta-heuristic algorithms ($D = 50$).

Dimension	Significance Level	k	χ^2	$\chi^2_{\alpha[k-1]}$	p -Value	Null Hypothesis	Alternative Hypothesis
$D = 50$	$\alpha = 0.05$	8	173.61	14.07	4.3532×10^{-34}	Reject	Accept

Table 23. The results of Friedman test for various meta-heuristic algorithms ($D = 30$).

Comparison	Unadjusted p -Value	Adjusted p -Value
IHFAPA vs. JSO	3.56×10^{-1}	3.56×10^{-1}
IHFAPA vs. HDE	5.23×10^{-10}	7.47×10^{-11}
IHFAPA vs. ISCA	4.99×10^{-5}	9.98×10^{-6}
IHFAPA vs. DMCSO	7.45×10^{-8}	1.24×10^{-8}
IHFAPA vs. OBLPSOGD	9.39×10^{-4}	3.13×10^{-4}
IHFAPA vs. MBADE	3.67×10^{-2}	1.84×10^{-2}
IHFAPA vs. MFAGA	6.85×10^{-4}	1.71×10^{-4}

Table 24. The results of Friedman test for various meta-heuristic algorithms ($D = 50$).

Comparison	Unadjusted p -Value	Adjusted p -Value
IHFAPA vs. JSO	3.70×10^{-3}	7.47×10^{-11}
IHFAPA vs. HDE	1.34×10^{-14}	1.91×10^{-15}
IHFAPA vs. ISCA	1.23×10^{-13}	2.04×10^{-14}
IHFAPA vs. DMCSO	4.45×10^{-12}	8.89×10^{-13}
IHFAPA vs. OBLPSOGD	6.23×10^{-8}	1.56×10^{-8}
IHFAPA vs. MBADE	1.49×10^{-1}	1.49×10^{-1}
IHFAPA vs. MFAGA	2.55×10^{-5}	8.50×10^{-6}

In conclusion, the experimental results of 28 30- D test functions in CEC 2017 show that IHFAPA performs better than the other seven meta-heuristic algorithms, thus, verifying the effectiveness of IHFAPA.

(2) Analysis of the results of 50- D test functions

Table 22 reports that JSO has the highest solution quality for C04, C15 and C18 benchmark functions; OBLPSOGD has the highest solution quality when solving C17 and C18 benchmark functions with seven meta-heuristic algorithms; MBADE has the highest solution quality when solving the eight benchmark functions C03, C05, C07, C16, C19, C22, C25 and C27; when solving the C28 benchmark test function, MFAGA has the highest solution quality; IHFAPA's C01, C02, C04, C08, C09, C10, C12, C14, C20, C21, C23, C24 and C26 benchmark functions have the highest solving quality. According to the $w/t/l$ values in Table 22, IHFAPA's performance is better than the other meta-heuristic algorithms. Specifically, IHFAPA has 25 test functions with a mean value better than JSO, 28 test functions with a mean value better than HDE, 28 test functions with a mean value better than ISCA, 28 test functions with a mean value better than DMCSO, 23 test functions with a mean value better than OBLPSOGD and 18 test functions with a mean value better than MBADE; the average of 26 test functions is better than MFAGA. In Figure 12, (c) and (d) show that IHFAPA ranks first among the seven meta-heuristic algorithms, indicating that IHFAPA's performance is better than the other six meta-heuristic algorithms. In addition, Friedman test is used to verify whether there is a significant difference in the performance of the seven algorithms. According to Table 23, $k = 7$, $\alpha = 0.05$, $k - 1 = 6$, critical value $\chi^2_{\alpha[6]} = 12.59$ and inspection value $\chi^2 = 75.75$. Since the test value is greater than the critical value, the null hypothesis is rejected, showing that there are obvious differences in the performance of the seven algorithms. From Table 24, Holm's experimental results show that IHFAPA only shows no significant difference with MBADE in the adjusted p -value and the unadjusted p -value, but IHFAPA is significantly better than MBADE in other indicators. Statistical tests show that IHFAPA is significantly different from other comparison algorithms.

In conclusion, the experimental results of 28 50- D test functions in CEC 2017 show that IHFAPA performs better than the other seven meta-heuristic algorithms, thus, verifying the effectiveness of IHFAPA.

4.7.3. Convergence Curve of the Proposed Algorithm

The results of IHFAPA's better performance than the other six meta-heuristic algorithms were visualized. Four 30-*D* and 50-*D* test functions of CEC 2017 are selected to draw convergence curves of eight FAs. Convergence curves of FA solving four 30-*D* test functions are shown in Figure 13, and convergence curves of FA solving four 50-*D* test functions are shown in Figure 14.

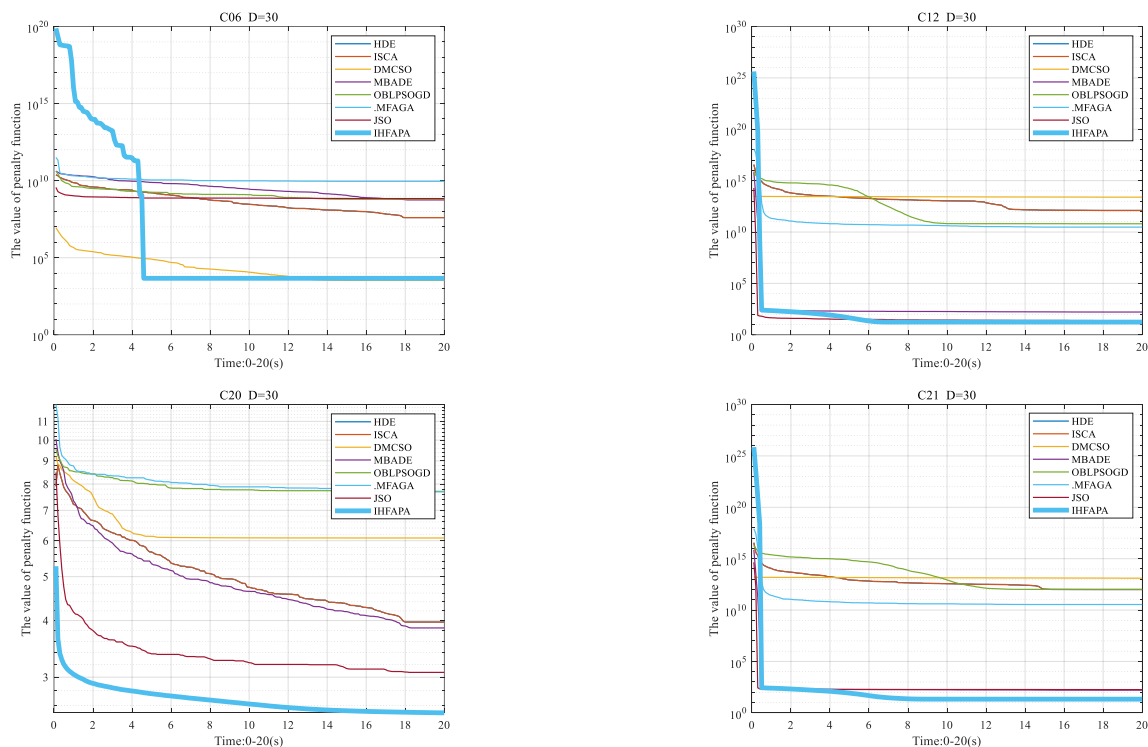


Figure 13. Convergence curves of CEC 2017 30-*D* test functions.

In Figure 13, for C06, the convergence speed of IHFAPA is like that of DMCSO, and the solution accuracy of the two is better than other proposed comparison algorithms. For C12 and C21, the convergence speed of IHFAPA and MBADE is faster than that of other proposed algorithms in the early stage of iteration; however, due to the usage of a combined mutation operator and the addition of a removing-similarity operation, IHFAPA has great exploration ability in the late stage of iteration and can jump out of the local optimal value and swiftly converge to the global optimal value. IHFAPA, therefore, handles quality problems better than the other six algorithms. IHFAPA clearly outperforms other algorithms in terms of solution quality for C20 and has a rapid convergence speed in the initial iteration.

Figure 14 illustrates that when calculating the 50-*D* CEC2017 test function, C01 test function converges to the optimal value of the solution faster than the other seven functions in the early stage, showing that IHFAPA has strong global search ability. In the C12 test function, IHFAPA has a fast convergence speed in the early stage of search, similar to MBADE and ILTD-ABC, but the solution accuracy of IHFAPA is higher than that of the other seven algorithms in the late stage of the search process. For C12 and C21 test functions, IHFAPA's convergence speed is relatively fast in the early stage of the search for the two test functions, but for the C21 test function, IHFAPA's solution accuracy is better in the later stage of the search and other algorithms fall into local optimum, which shows that IHFAPA has a strong ability to jump out of local optimum.

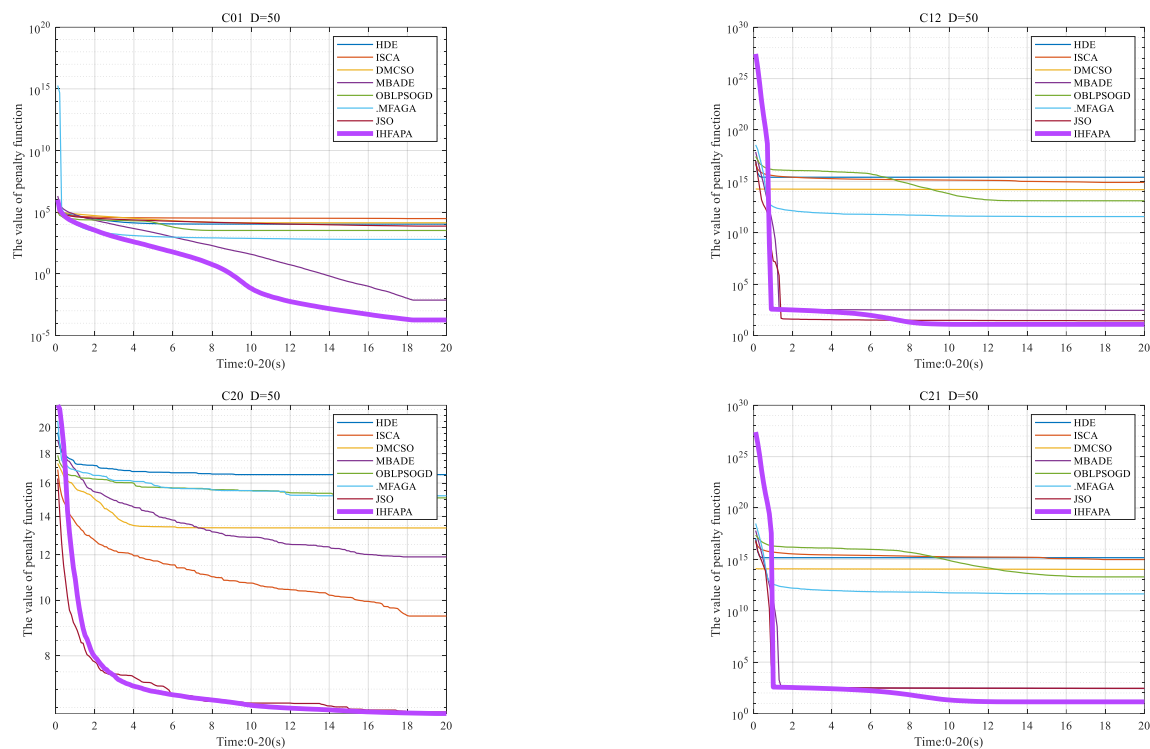


Figure 14. Convergence curves of CEC 2017 50-D test functions.

5. Application to Engineering Problems

To further verify the effectiveness of IHFAPA, 14 intelligence optimization algorithms in the literature were selected, namely, SFA [6], RaFA [19], NaFA [43], GDFA [49], AD-IFA [29], YYFA [36], GAHFA [20], HDE [47], ISCA [48], DMCSO [49], OBLPSOGD [50], MRADE [51], MFAGA [52] and JSO [53]. Then, 14 meta-heuristic algorithms in IHFAPA and the literature were used to solve the optimization design problems for the cantilever beam, welded beam, piston rod and three-bar truss, and the results of 14 intelligence optimization algorithms were counted and compared.

Set the population size $n = 40$, the maximum running time $Maxtime = 20$ s, the penalty factor $M_1 = M_2 = 10^8$ and the allowable error values for equality constraint in constrained optimization problems $\theta = 10^{-4}$ in order to accurately compare the performance of various algorithms. For each engineering optimization problem, each algorithm is run 20 times independently, the results are solved statistically and the results are compared and analyzed to verify the effectiveness and feasibility of IHFAPA in practice.

5.1. Optimization Design of Cantilever Beam [28]

The aim of the cantilever beam design is to minimize the weight of the cantilever beam. Taking the optimization design of the cantilever beam given in the literature [28] as an example, the cantilever beam structure is shown in Figure 15.

The mathematical model for the optimal design of the cantilever beam is as follows:

$$\begin{aligned} \min f(X) &= 0.6224(x_1 + x_2 + x_3 + x_4 + x_5) \\ \text{s.t.} \quad &\begin{cases} g(X) = \frac{61}{x_1^3} + \frac{27}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \leq 0 \\ 0.01 \leq x_1, x_2, x_3, x_4, x_5 \leq 100 \end{cases} \end{aligned} \quad (33)$$

Table 25 shows that 14 intelligence optimization algorithms are used to solve Equation (13) and the best result of various algorithms solving Problem 1. To better evaluate the performance of the algorithm, the minimum optimal value (Best), maximum optimal value (Worst), average optimal value (Mean) and standard deviation of optimal value

(Std) are taken as statistical indicators. Table 26 gives the statistical indicator values of the algorithms.

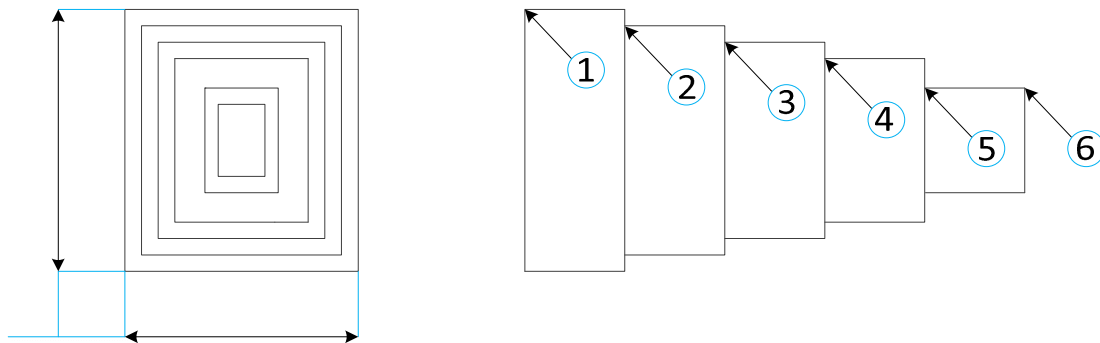


Figure 15. Structure design of cantilever beam.

Table 25. The best result of various algorithms solving Problem 1.

Algorithm	x_1	x_2	x_3	x_4	x_5	$f(X)$
SFA	5.378221	5.196736	5.352442	33.838475	1.509515	31.913801
RaFA	27.867726	19.660745	11.522870	4.807659	8.096651	44.785197
NaFA	5.822165	4.833300	4.296618	3.704691	2.380659	13.093698
GDFA	5.995532	4.712419	4.524987	3.592656	2.135814	13.046380
ADIFA	5.812717	4.894457	4.637111	3.486921	2.130079	13.046305
YYFA	5.974240	4.872583	4.427257	3.513410	2.164964	13.040808
GAHFA	6.013024	4.733404	4.514108	3.525513	2.164763	13.039888
JSO	5.978223	4.876190	4.466096	3.479479	2.139142	13.032515
HDE	5.980548	4.884228	4.459534	3.482497	2.132448	13.032592
ISCA	5.980177	4.873386	4.469500	3.477099	2.138981	13.032522
DMCSO	5.976299	4.877723	4.466438	3.478975	2.139698	13.032516
MBADE	5.978223	4.876190	4.466096	3.479479	2.139142	13.032514
MFAGA	5.978223	4.876190	4.466096	3.479479	2.139142	13.032514
IHFAPA	5.978223	4.876190	4.466096	3.479479	2.139142	13.032514

Table 26. Statistical indicator values of 14 algorithms.

Algorithm	Best	Worst	Mean	Std
SFA	31.91380143	95.41133362	72.02906616	1.73×10^1
RaFA	44.78519656	98.22406344	78.05979901	1.30×10^1
NaFA	13.09369793	14.01238482	13.46412795	2.55×10^{-1}
GDFA	13.04638006	13.137194	13.08140472	2.80×10^{-2}
ADIFA	13.04630463	23.70457178	15.15767351	2.59
YYFA	13.04080799	13.20624313	13.11404444	3.81×10^{-2}
GAHFA	13.03988751	14.23777192	13.53016911	4.17×10^{-1}
JSO	13.03251551	13.03251551	13.03251551	2.19×10^{-15}
HDE	13.03259225	13.03320678	13.0327743	1.77×10^{-4}
ISCA	13.03252229	14.2417099	13.21728098	2.62×10^{-1}
DMCSO	13.03251606	13.03256694	13.03253796	1.43×10^{-5}
MBADE	13.03251427	13.03251427	13.03251427	5.88×10^{-15}
MFAGA	13.03251422	13.03251422	13.03251422	1.82×10^{-15}
IHFAPA	13.03251422	13.03251422	13.03251422	1.82×10^{-15}

In Table 25, the optimal value obtained by MFAGA and IHFAPA is the smallest among the 14 algorithms. Table 26 illustrates that the Best, Worst, Mean and Std of IHFAPA and MFAGA are the same and better than the other 11 algorithms. Therefore, the quality of IHFAPA is not inferior to the other algorithms in solving the cantilever optimization design problem, which verifies the effectiveness of IHFAPA.

5.2. Optimization Design of Welded Beam [54]

One well-known nonlinear optimization design problem in the realm of engineering is the optimization of welded beams. The objective of this problem is to minimize the manufacturing cost of welded beams under the constraints of shear stress (τ), beam end deflection (δ), bar buckling load (P_c) and bending stress (σ). The design variable for the optimization design of the welded beam is $X = [x_1, x_2, x_3, x_4]$. The structure of the welded beam is shown in Figure 16.

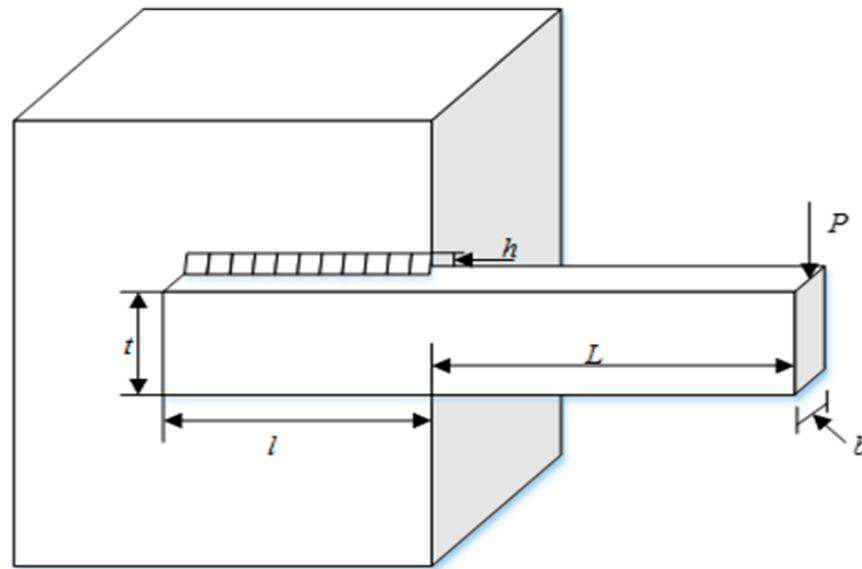


Figure 16. The structure of welded beam.

The mathematical model for the optimization design of the welded beam is:

$$\begin{aligned} \min f(X) &= 1.10471x_1^2x_2 \\ \text{s.t.} \quad &\begin{cases} g_1(X) = \tau(\vec{x}) - \tau_{\max} \leq 0 \\ g_2(X) = \sigma(\vec{x}) - \sigma_{\max} \leq 0 \\ g_3(X) = \delta(\vec{x}) - \delta_{\max} \leq 0 \\ g_4(X) = x_1 - x_4 \leq 0 \\ g_5(X) = P - P_c(\vec{x}) \leq 0 \\ g_6(X) = 0.125 - x_1 \leq 0 \\ g_7(X) = 1.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \leq 0 \end{cases} \end{aligned} \quad (34)$$

$$\tau(X) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2} \quad (35)$$

$$\tau' = \frac{P}{\sqrt{2}x_1x_2} \quad (36)$$

$$\tau'' = \frac{MR}{J} \quad (37)$$

$$M = P\left(L + \frac{x_2}{2}\right) \quad (38)$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2} \quad (39)$$

$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\} \quad (40)$$

$$\sigma(X) = \frac{6PL}{x_4 x_3^2} \quad (41)$$

$$\delta(X) = \frac{6PL^3}{Ex_3^2 x_4} \quad (42)$$

$$P_c(\vec{x}) = \frac{4.013E\sqrt{\frac{x_3^2 x_4^6}{36}}}{L^2} \left(1 - \frac{x_3}{2L} \sqrt{\frac{E}{4G}} \right) \quad (43)$$

where $P = 6000$ lb, $\tau_{max} = 13600$ psi, $\delta_{max} = 0.25$ in, $L = 14$ in, $E = 30 \times 10^6$ psi, $G = 12 \times 10^6$ psi and $\sigma_{max} = 30,000$ psi.

The value range of design variables in the optimization design of welded beams is $0.1 \leq x_i (i = 1, 4) \leq 2$, $0.1 \leq x_i (i = 2, 3) \leq 10$. Table 27 reports intelligence optimization algorithms are used to solve Equation (34) and the best result of various algorithms solving Problem 2. To better evaluate the performance of the algorithm, the Best, Worst, Mean and Std are taken as statistical indicators. The statistical indicator values of the 14 algorithms are given in Table 28.

Table 27. The best result of various algorithms solving Problem 2.

Algorithm	x_1	x_2	x_3	x_4	$f(X)$
SFA	0.205142	3.286062	9.040616	0.205725	1.699504
RaFA	0.152123	4.864129	9.337794	0.217702	1.969270
NaFA	0.343650	2.812506	7.196487	0.426302	2.848371
GDFA	0.203523	3.278857	9.114596	0.205344	1.705893
ADIFA	0.207075	3.229328	9.036616	0.205730	1.702817
YYFA	0.206911	3.232201	9.036624	0.205730	1.819633
GAHFA	0.252491	2.581204	9.036624	0.205730	1.695247
JSO	0.205730	3.253120	9.036624	0.205730	1.695247
HDE	0.327076	2.308069	7.180879	0.326335	2.111363
ISCA	0.205180	3.262910	9.036821	0.205736	1.695851
DMCSO	0.187914	4.153360	8.537442	0.230493	1.880627
MBADE	0.205730	3.253120	9.036624	0.205730	1.695247
MFAGA	0.127434	5.184727	9.958296	0.298460	1.820290
IHFAPA	0.205730	3.253109	9.036624	0.205730	1.695247

Table 28. Statistical indicator values of 14 algorithms.

Algorithm	Best	Worst	Mean	Std
SFA	1.699504131	1.703915182	1.701907472	1.76×10^{-3}
RaFA	1.969269943	2.360207438	2.183403881	1.70×10^{-1}
NaFA	2.848370826	6.543512806	4.44958705	1.04
GDFA	1.705892603	1.794671332	1.750032772	2.62×10^{-2}
ADIFA	1.702817329	1.710019138	1.705453849	3.97×10^{-3}
YYFA	1.819633196	2.040470288	1.917570624	1.13×10^{-1}
GAHFA	1.695247121	1.695247121	1.695247121	2.22×10^{-16}
JSO	1.695247547	1.695247547	1.695247547	1.00×10^{-11}
HDE	2.111363368	3.326935976	2.647398239	4.80×10^{-1}
ISCA	1.695850898	2.062965308	1.828526064	1.76×10^{-1}
DMCSO	1.880626824	2.659265752	2.288024928	3.04×10^{-1}
MBADE	1.695247165	1.695247165	1.695247165	2.72×10^{-16}
MFAGA	1.820289796	2.261054896	2.008538645	1.08×10^{-1}
IHFAPA	1.695246726	1.695246726	1.695246726	1.70×10^{-11}

Table 27 reports that Best of IHFAPA is the smallest among the 14 algorithms. Table 28 reports IHFAPA's Best, Worst, Mean and Std are better than those of other algorithms. Therefore, IHFAPA is superior to the other algorithms in solving the optimization design of welded beams, thus, verifying the effectiveness of IHFAPA.

5.3. Optimization Design of Piston Rod [55]

The piston rod structure is also a common optimization structure in practical engineering, and its structure is shown in Figure 17.

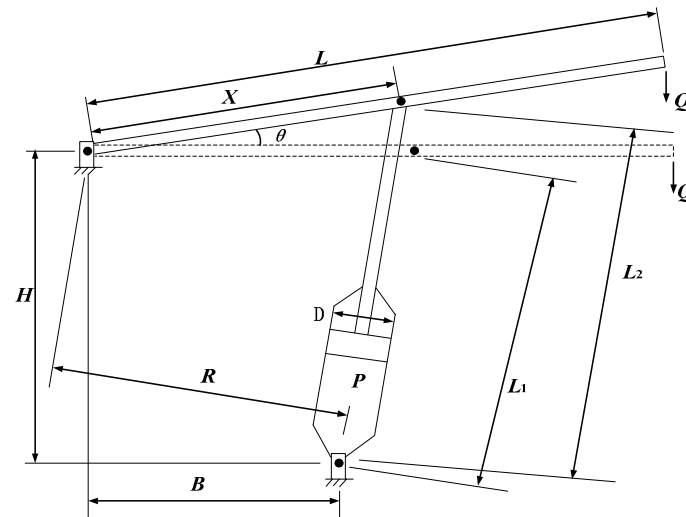


Figure 17. The structure of the piston lever.

The primary objective of the piston rod optimization problem is to position piston components H , B , D and X by minimizing the amount of oil when the piston rod moves from 0 to 45° .

The design variable of the piston rod optimization problem is $X = [H, B, D, X] = [x_1, x_2, x_3, x_4]$ and the mathematical model is:

$$\begin{aligned} \min f(X) &= \frac{1}{4}\pi x_4^2(L_2 - L_1) \\ \text{s.t.} \quad &\begin{cases} g_1(X) = QL \cos \theta - RF \leq 0 \\ g_2(X) = Q(L - x_4) - M_{\max} \leq 0 \\ g_3(X) = 1.2(L_2 - L_1) - L_1 \leq 0 \\ g_4(X) = x_3/2 - x_2 \end{cases} \end{aligned} \quad (44)$$

$$R = \frac{|-x_4(x_4 \sin \theta + x_1) + x_1(x_2 - x_4 \cos \theta)|}{\sqrt{(x_4 - x_2)^2 + x_1^2}} \quad (45)$$

$$F = \frac{\pi P x_3^2}{4} \quad (46)$$

$$L_2 = \sqrt{(x_4 \sin \theta + x_1)^2 + (x_2 - x_4 \cos \theta)^2} \quad (47)$$

$$L_1 = \sqrt{(x_4 - x_2)^2 + x_1^2} \quad (48)$$

where $Q = 10,000$ lbs, $\theta = 45^\circ$, $L = 240$ in, oil pressure P take 1500 psi and maximum allowable bending moment of lever $M_{\max} = 1.8 \times 10^6$ lbs in.

The design variable of the piston lever optimization design problem ranges from $0.05 \leq x_i (i = 1, 2, 3) \leq 500$ to $0.05 \leq x_4 \leq 120$. Thus, 13 intelligence optimization algorithms are used to solve Equation (44) and the best result of various algorithms solving Problem 3 is reported in Table 29. In order to better evaluate the performance of the algorithm, Best, Worst, Mean and Std are taken as statistical indicators. The statistical indicator values of the 13 algorithms are given in Table 30.

Table 29. The best result of various algorithms solving Problem 3.

Algorithm	x_1	x_2	x_3	x_4	$f(X)$
SFA	323.248363	445.241380	0.769432	100.504978	33.017971
RaFA	495.290074	499.715164	2.217472	60.012207	168.239426
NaFA	104.240204	86.759338	0.482788	61.045700	8.485409
GDFA	0.575958	4.047767	4.526363	98.406522	18.808499
ADIFA	472.995123	498.160365	2.257819	60.584553	174.903047
YYFA	0.050070	2.146217	4.091046	119.970691	8.862098
GAHFA	500.000000	500.000000	2.211110	60.000000	8.412698
JSO	5.834864	8.515579	5.181083	82.183156	122.517700
HDE	350.423154	499.875433	2.435342	60.161549	189.607904
ISCA	0.050000	2.041532	4.083032	120.000000	8.412792
DMCSO	0.088350	2.800421	5.317896	68.916930	20.295798
MBADE	255.964175	499.721801	2.719571	60.145636	8.412698
MFAGA	0.071040	2.571204	4.486615	98.392004	12.985521
IHFAPA	0.050000	2.041514	4.083027	120.000000	8.412698

Table 30. Statistical indicator values of 14 algorithms.

Algorithm	Best	Worst	Mean	Std
SFA	33.0179706	655683.391233	174576.900399	2.80×10^5
RaFA	168.2394257	171.464577	169.316744	1.46
NaFA	8.4854092	63459.882063	24002.590167	2.44×10^4
GDFA	18.8084987	203.436288	108.714651	1.01×10^2
ADIFA	174.9030473	237.515228	210.815755	2.39×10^1
YYFA	8.8620984	173.613805	63.925947	9.50×10^1
GAHFA	8.4126983	167.472730	135.660724	7.11×10^1
JSO	122.517700	167.472700	164.607900	1.02×10^1
HDE	189.6079044	22916.006641	3369.843019	7.93×10^3
ISCA	8.4127921	8.413036	8.412932	1.14×10^{-4}
DMCSO	20.2957982	290.999882	176.624502	9.77×10^1
MBADE	8.4126981	167.472730	135.660724	7.11×10^1
MFAGA	12.9855210	543.823923	228.069099	1.54×10^2
IHFAPA	8.4126981	167.472730	91.123915	8.11×10^1

Table 29 shows that Best of IHFAPA is the smallest among the 14 algorithms. Table 30 reports that IHFAPA's Best is better than other 13 algorithms, and ISCA's Worst, Mean and Std are better than other algorithms.

5.4. Optimization Design of Three-Bar Truss [55]

The three-bar truss structure is shown in Figure 18.

The objective of the optimization design of the three-bar truss is to minimize the weight of the three-bar truss under the constraints of stress, deflection and buckling. The design variable of the optimization design of the three-bar truss is $[B_1, B_2] = [x_1, x_2]$. The mathematical model of the optimization problem is:

$$\begin{cases} \min f(X) = 100(2\sqrt{2}x_1 + x_2) \\ \text{s.t.} \begin{cases} g_1(x) = 2(\sqrt{2}x_1 + x_2) / (\sqrt{2}x_1^2 + 2x_1x_2) - 2 \leq 0 \\ g_2(x) = 2x_2 / (\sqrt{2}x_1^2 + 2x_1x_2) - 2 \leq 0 \\ g_3(x) = 2 / (\sqrt{2}x_2 + x_1) - 2 \leq 0 \end{cases} \end{cases} \quad (49)$$

The value range of design variables for the optimization of the three-bar truss design is as follows: $0 \leq x_1 \leq 1$ and $0 \leq x_2 \leq 1$. Table 31 reports that 14 intelligence optimization algorithms are used to solve Equation (49) and the best result of various algorithms solving Problem 4. To better evaluate the performance of the algorithm, Best, Worst, Mean and Std are taken as statistical indicators. The statistical indicator values of the 14 algorithms are given in Table 32.

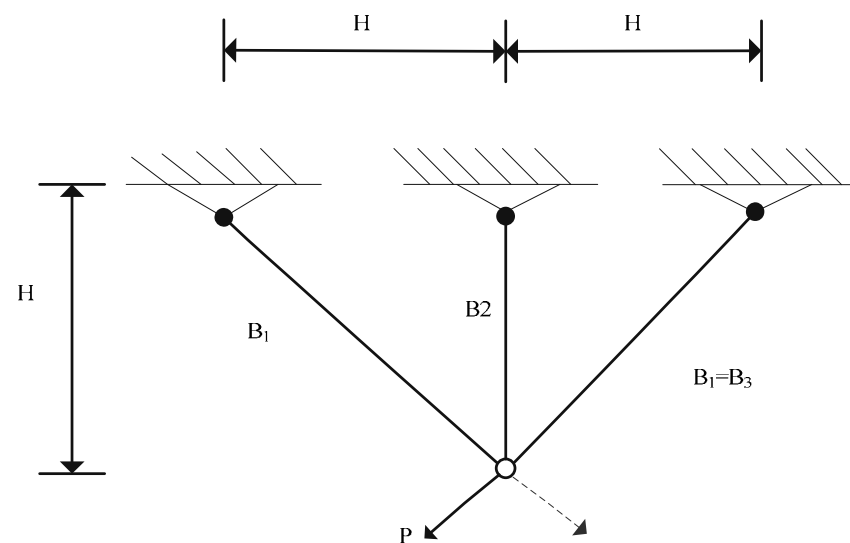


Figure 18. The structure of three-bar truss.

Table 31. The best result of various algorithms solving Problem 3.

Algorithm	x_1	x_2	$f(x)$
SFA	0.787926	0.410374	263.896446
RaFA	0.785019	0.419141	263.950917
NaFA	0.786150	0.416532	264.010146
GDFA	0.788675	0.408248	263.895843
ADIFA	0.788664	0.408279	263.895844
YYFA	0.788660	0.408292	263.895844
GAHFA	0.788675	0.408248	263.895843
JSO	0.788675	0.408248	263.895843
HDE	0.783659	0.422625	263.914661
ISCA	0.788465	0.408844	263.895903
DMCSO	0.788675	0.408248	263.895843
MBADE	0.788675	0.408248	263.895843
MFAGA	0.787865	0.410557	263.897624
IHFAPA	0.788675	0.408248	263.895843

According to Table 31, the Best of IHFAPA is the smallest among 14 algorithms. From Table 32, the Best, Worst, Mean and Std of IHFAPA are better than the other algorithms. Therefore, IHFAPA is superior to the other algorithms in solving the optimization design problem of the three-bar truss, thus, verifying the effectiveness of IHFAPA.

In summary, the optimization results of the four engineering optimization problems show that the minimum optimal value obtained by IHFAPA is better than other algorithms when solving piston rod optimization design problems, and the standard deviation of maximum optimal value, average optimal value and optimal value obtained by IHFAPA is not worse than that of the other algorithms. When the 14 algorithms solve the cantilever beam optimization design problem, the welded beam optimization design problem and the three-bar truss optimization design problem, all the statistical indicator values are not inferior to the other algorithms. Therefore, the performance of IHFAPA is significantly better than other algorithms.

Table 32. Statistical indicator values of 14 algorithms.

Algorithm	Best	Worst	Mean	Std
SFA	263.8964457	263.8999999	263.8979033	9.28×10^{-4}
RaFA	263.9509171	265.2151988	264.3419885	3.17×10^{-1}
NaFA	264.0101463	269.4611471	265.7919397	1.55
GDFA	263.8958434	263.896638	263.8958931	1.76×10^{-4}
ADIFA	263.8958435	263.895876	263.8958504	7.81×10^{-6}
YYFA	263.8958436	267.4844861	264.4581455	9.68×10^{-1}
GAHFA	263.8958434	263.8958434	263.8958434	4.02×10^{-14}
JSO	263.8958534	263.8958434	263.8958544	7.20×10^{-410}
HDE	263.9146614	265.2142127	264.251843	4.03×10^{-1}
ISCA	263.8959028	263.9135898	263.8975107	3.85×10^{-3}
DMCSO	263.8958434	264.2691155	263.9251508	8.45×10^{-2}
MBADE	263.8958434	263.8958434	263.8958434	4.02×10^{-14}
MFAGA	263.8976237	263.9835232	263.9164881	2.33×10^{-2}
IHFAPA	263.8958434	263.8958434	263.8958434	4.02×10^{-14}

5.5. Discussion

The diversity in the population has a great impact on the exploration and exploitation ability of the algorithm. With an increase in the number of iterations, the fireflies in the population will focus on the fireflies near the local extreme value or the global extreme value point, and the similarity degree S between individuals in the population increases. When the similarity between individuals in the population exceeds the threshold value ζ , it is considered that the individuals in the species are so similar that some individuals in the population need to be re-initialized. Let the ratio of the number of re-initialized individuals to the population size be P ; the size of P has an impact on the performance of FA and the diversity in the population. To give an optimal combination of threshold values ζ and P , a two-factor and three-level orthogonal experiment was designed. Orthogonal experiment results show that FA has the best performance when $\zeta = 0.4$ and $P = 0.97$. In addition, the orthogonal experiment results also verify that similarity removal has a great influence on the performance of FA.

CEC 2017 is selected to verify the superiority of the probabilistic attracting model. CAM, RAM, NAM and GAM are selected to design an experiment to verify the effect of different attraction models on algorithm performance. In the experiment, IHFAPA was the same except for the different attraction model. Experimental results show that IHFAPA using the probability attraction model has better performance than IHFA using other attraction models. In addition, the experimental results verify that the attraction model has a significant influence on the performance of the algorithm and the superiority of the probability attraction model.

To verify that the performance of the IHFAPA algorithm is better than other FAs and other intelligence optimization algorithms, 28 30- D and 50- D test functions in CEC 2017 are selected and four practical engineering optimization problems are selected. Two groups of experiments were designed to verify the superiority of the IHFAPA algorithm performance. The experiment that uses 28 30- D and 50- D test functions in CEC 2017 to verify the superiority of IHFAPA performance is called the first combination experiment. The experiments to verify the performance advantage of the algorithm using four practical optimization problems commonly used in engineering are called the second set of experiments. The results of the first combination experiment show that the solution quality of IHFAPA is significantly better than seven other improved FAs and seven other intelligence optimization algorithms, which verifies the effectiveness and superiority of IHFAPA. The second set of experimental results show that the solution quality of IHFAPA is better than the other 14 comparison algorithms, and the effectiveness and performance advantages of the algorithm in practical problems are verified.

6. Conclusions

FA features some deficiencies, such as premature convergence and poor solution quality, when solving high-dimensional and multi-extremum constrained optimization problems. To solve these problems, an IHFAPA is proposed. IHFAPA was improved in four aspects: probability attraction model, position updating formula, combined mutation operator based on selection probability and similarity-removal operation.

The optimal point set method based on square-root sequence is selected to generate the initial population to make the initial population evenly distributed in the whole search space and have good population diversity. An adaptive probability attraction model based on firefly brightness level is proposed to minimize the number of brightness comparisons and make the number of attractions moderate. The probability attraction model can not only avoid the oscillation or fluctuation phenomenon in the search process caused by too many times of attraction but can also avoid the problem of slow convergence speed and poor solution quality caused by too few times of attraction. In addition, the probability attraction model can effectively reduce the TC in the algorithm.

For the firefly location-update formula in literature [6], when the distance between two fireflies is large, the relative attraction may approach 0. At this time, the algorithm is equivalent to random search, resulting in slow convergence speed and poor solution quality. For the firefly position-update formula in the literature [20], when the distance between two fireflies is large, although the problem of relative attraction approaching 0 is avoided, the value of relative attraction is small and the attraction term plays a minimal role. In addition, a new position-update formula is proposed to solve the problem that the value of the guide factor decreases, and the role of the guide term gradually decreases with an increase in the number of iterations. The situation where the relative suction force tends to zero at the start of the iteration is not shown by the new position-update formula. The enhanced position-updating formula takes into account both the guiding influence of optimal fireflies in the population on fireflies undergoing position updating as well as the influence of high-brightness fireflies on fireflies undergoing position updating. On this foundation, a dynamic and adaptive method to change the parameters in the position-update formula is proposed, making the position-update method have strong global search ability at the start of iteration and strong local search ability at the end, so that both the global and local search ability of the algorithm are taken into better account. In addition, in the initial iteration of the new position-update formula, the attraction term plays a larger role than the guide term and the algorithm has strong exploration ability. After iteration, the attraction term plays a smaller role, while the guide term plays a larger role, and the algorithm has strong exploitation ability.

FA's exploration and exploitation ability is determined by firefly's position-update formula, but the position-update formula proposed in the existing literature needs to judge when it is the early iteration and the late iteration; however, sometimes the judgment is not accurate. Therefore, the algorithm may perform local search when global search is needed. When local search is needed, the algorithm performs global search. A combined mutation operator based on selection probability is added to the revised FA to address this flaw. The combined mutation operator based on selection probability that can adaptively select a certain kind of mutation operator with strong exploration ability or exploitation ability does not need to judge whether the algorithm is in the early iteration or the late iteration, so as to better balance the exploration and exploitation ability of the algorithm.

There will be a large number of similar individuals in the population as iteration times rise. For the multi-extremum optimization problem, most of the individuals in the population will focus near the local extremum point in the late iteration, and the diversity in the population is poor, which leads to the algorithm easily falling into a local optimum. In order to solve this issue, a similarity-removal operation is added to the FA. This effectively lowers the population's number of similar individuals, aids in maintaining the population's diversity, lowers the likelihood that the algorithm will reach a local optimum and improves the algorithm's solution quality.

Finally, to verify the superiority of IHFAPA performance, the CEC 2017 constraint optimization problem and four practical engineering optimization problems are used to conduct experiments. The experimental results show that IHFAPA can effectively improve the solution quality, thus, verifying the effectiveness of IHFAPA.

In the future, there are still many fields worth our research and exploration. In addition to the position-update formula, we can also explore the combination of FA and other optimization algorithms to give full play to the advantages of different algorithms. We can also investigate the use of FA in the identification of rice pests and other fields, as well as the use of FA to address mixed-integer and pure-integer programming issues.

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

Funding: This research was funded by the National Social Science Fund of China, grant number 21BGL174.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors thank the anonymous reviewers for their valuable and constructive comments that greatly helped improve the quality and completeness of this paper.

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

References

1. Bhanu, B.; Lee, S.; Ming, J. Adaptive image segmentation using a genetic algorithm. *IEEE Trans. Syst. Man Cybern.* **1995**, *25*, 1543–1567. [\[CrossRef\]](#)
2. Storn, R.; Price, K. Differential Evolution—A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [\[CrossRef\]](#)
3. Gao, W.F.; Liu, S.Y.; Huang, L.L. Particle swarm optimization with chaotic opposition-based population initialization and stochastic search technique. *Commun. Nonlinear Sci. Numer. Simul.* **2012**, *17*, 4316–4327. [\[CrossRef\]](#)
4. Wang, H.; Zhou, X.Y.; Sun, H.; Yu, X.; Zhao, J.; Zhang, H.; Cui, L.Z. Firefly algorithm with adaptive control parameters. *Soft Comput.* **2017**, *21*, 5091–5102. [\[CrossRef\]](#)
5. Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *J. Glob. Optim.* **2007**, *39*, 459–471. [\[CrossRef\]](#)
6. Yang, X.S. *Nature-Inspired Metaheuristic Algorithms*; Luniver Press: Bristol, UK, 2008.
7. Mishra, S.P.; Dash, P.K. Short-term prediction of wind power using a hybrid pseudo-inverse Legendre neural network and adaptive firefly algorithm. *Neural Comput. Appl.* **2019**, *31*, 2243–2268. [\[CrossRef\]](#)
8. Huang, H.C.; Lin, S.K. A Hybrid Metaheuristic Embedded System for Intelligent Vehicles Using Hypermutated Firefly Algorithm Optimized Radial Basis Function Neural Network. *IEEE Trans. Ind. Inform.* **2019**, *15*, 1062–1069. [\[CrossRef\]](#)
9. Dhal, K.G.; Das, A.; Ray, S.; Galvez, J. Randomly Attracted Rough Firefly Algorithm for histogram based fuzzy image clustering. *Knowl. Based Syst.* **2021**, *216*, 106814. [\[CrossRef\]](#)
10. Agarwal, V.; Bhanot, S. Radial basis function neural network-based face recognition using firefly algorithm. *Neural Comput. Appl.* **2018**, *30*, 2643–2660. [\[CrossRef\]](#)
11. Kaya, S.; Gumuscu, A.; Aydilek, I.B.; Karacizmeli, I.H.; Tenekeci, M.E. Solution for flow shop scheduling problems using chaotic hybrid firefly and particle swarm optimization algorithm with improved local search. *Soft Comput.* **2021**, *25*, 7143–7154. [\[CrossRef\]](#)
12. Ewees, A.A.; Al-qaness, M.A.A.; Abd Elaziz, M. Enhanced salp swarm algorithm based on firefly algorithm for unrelated parallel machine scheduling with setup times. *Appl. Math. Model.* **2021**, *94*, 285–305. [\[CrossRef\]](#)
13. Zhang, Y.; Zhou, J.; Sun, L.; Mao, J.; Sun, J. A Novel Firefly Algorithm for Scheduling Bag-of-Tasks Applications Under Budget Constraints on Hybrid Clouds. *IEEE Access* **2019**, *7*, 151888–151901. [\[CrossRef\]](#)
14. He, L.F.; Huang, S.W. Modified firefly algorithm based multilevel thresholding for color image segmentation. *Neurocomputing* **2017**, *240*, 152–174. [\[CrossRef\]](#)

15. Pitchaimanickam, B.; Murugaboopathi, G. A hybrid firefly algorithm with particle swarm optimization for energy efficient optimal cluster head selection in wireless sensor networks. *Neural Comput. Appl.* **2020**, *32*, 7709–7723. [\[CrossRef\]](#)
16. Yogarajan, G.; Revathi, T. Nature inspired discrete firefly algorithm for optimal mobile data gathering in wireless sensor networks. *Wirel. Netw.* **2018**, *24*, 2993–3007. [\[CrossRef\]](#)
17. Pakdel, H.; Fotuhi, R. A firefly algorithm for power management in wireless sensor networks (WSNs). *J. Supercomput.* **2021**, *77*, 9411–9432. [\[CrossRef\]](#)
18. Wang, H.; Wang, W.J.; Cui, L.Z.; Sun, H.; Zhao, J.; Wang, Y.; Xue, Y. A hybrid multi-objective firefly algorithm for big data optimization. *Appl. Soft Comput.* **2018**, *69*, 806–815. [\[CrossRef\]](#)
19. Wang, H.; Wang, W.J.; Sun, H.; Rahnamayan, S. Firefly algorithm with random attraction. *Int. J. Bio-Inspired Comput.* **2016**, *8*, 33–41. [\[CrossRef\]](#)
20. Cheng, Z.W.; Song, H.H.; Wang, J.Q.; Zhang, H.Y.; Chang, T.Z.; Zhang, M.X. Hybrid firefly algorithm with grouping attraction for constrained optimization problem. *Knowl.-Based Syst.* **2021**, *220*, 30. [\[CrossRef\]](#)
21. Coelho, L.D.; Mariani, V.C. Firefly algorithm approach based on chaotic Tinkerbell map applied to multivariable PID controller tuning. *Comput. Math. Appl.* **2012**, *64*, 2371–2382. [\[CrossRef\]](#)
22. Rizk-Allah, R.M.; Zaki, E.M.; El-Sawy, A.A. Hybridizing ant colony optimization with firefly algorithm for unconstrained optimization problems. *Appl. Math. Comput.* **2013**, *224*, 473–483. [\[CrossRef\]](#)
23. Liang, R.H.; Wang, J.C.; Chen, Y.T.; Tseng, W.T. An enhanced firefly algorithm to multi-objective optimal active/reactive power dispatch with uncertainties consideration. *Int. J. Electr. Power Energy Syst.* **2015**, *64*, 1088–1097. [\[CrossRef\]](#)
24. Banerjee, A.; Ghosh, D.; Das, S. Modified firefly algorithm for area estimation and tracking of fast expanding oil spills. *Appl. Soft Comput.* **2018**, *73*, 829–847. [\[CrossRef\]](#)
25. Zhang, J.; Teng, Y.F.; Chen, W. Support vector regression with modified firefly algorithm for stock price forecasting. *Appl. Intell.* **2019**, *49*, 1658–1674. [\[CrossRef\]](#)
26. Ball, A.K.; Roy, S.S.; Kisku, D.R.; Murmu, N.C.; Coelho, L.d.S. Optimization of drop ejection frequency in EHD inkjet printing system using an improved Firefly Algorithm. *Appl. Soft Comput.* **2020**, *94*, 106438. [\[CrossRef\]](#)
27. Zhang, L.; Srisukham, W.; Neoh, S.C.; Lim, C.P.; Pandit, D. Classifier ensemble reduction using a modified firefly algorithm: An empirical evaluation. *Expert Syst. Appl.* **2018**, *93*, 395–422. [\[CrossRef\]](#)
28. Wang, C.F.; Song, W.X. A novel firefly algorithm based on gender difference and its convergence. *Appl. Soft Comput.* **2019**, *80*, 107–124. [\[CrossRef\]](#)
29. Wu, J.R.; Wang, Y.G.; Burrage, K.; Tian, Y.C.; Lawson, B.; Ding, Z. An improved firefly algorithm for global continuous optimization problems. *Expert Syst. Appl.* **2020**, *149*, 113340. [\[CrossRef\]](#)
30. Chen, K.; Zhou, Y.; Zhang, Z.; Dai, M.; Chao, Y.; Shi, J. Multilevel Image Segmentation Based on an Improved Firefly Algorithm. *Math. Probl. Eng.* **2016**, *2016*, 1–12. [\[CrossRef\]](#)
31. Huang, S.J.; Liu, X.Z.; Su, W.F.; Yang, S.H. Application of Hybrid Firefly Algorithm for Sheath Loss Reduction of Underground Transmission Systems. *IEEE Trans. Power Deliv.* **2013**, *28*, 2085–2092. [\[CrossRef\]](#)
32. Verma, O.P.; Aggarwal, D.; Patodi, T. Opposition and dimensional based modified firefly algorithm. *Expert Syst. Appl.* **2016**, *44*, 168–176. [\[CrossRef\]](#)
33. Dash, J.; Dam, B.; Swain, R. Design of multipurpose digital FIR double-band filter using hybrid firefly differential evolution algorithm. *Appl. Soft Comput.* **2017**, *59*, 529–545. [\[CrossRef\]](#)
34. Aydılek, I.B. A Hybrid Firefly and Particle Swarm Optimization Algorithm for Computationally Expensive Numerical Problems. *Appl. Soft Comput.* **2018**, *66*, 232–249. [\[CrossRef\]](#)
35. Li, G.C.; Liu, P.; Le, C.Y.; Zhou, B.D. A Novel Hybrid Meta-Heuristic Algorithm Based on the Cross-Entropy Method and Firefly Algorithm for Global Optimization. *Entropy* **2019**, *21*, 494. [\[CrossRef\]](#)
36. Wang, W.C.; Xu, L.; Chau, K.W.; Xu, D.M. Yin-Yang firefly algorithm based on dimensionally Cauchy mutation. *Expert Syst. Appl.* **2020**, *150*, 18. [\[CrossRef\]](#)
37. Hua, L.K.; Wang, Y.J.S.B.H. *Applications of Number Theory to Numerical Analysis*; Springer: Berlin/Heidelberg, Germany, 1972.
38. Sayadi, M.K.; Hafezalkotob, A.; Naini, S.G.J. Firefly-inspired algorithm for discrete optimization problems: An application to manufacturing cell formation. *J. Manuf. Syst.* **2013**, *32*, 78–84. [\[CrossRef\]](#)
39. Yu, Y.; Liu, Y.; Liu, K.; Chen, Y. Chaos Pseudo Parallel Genetic Algorithm and Its Application on Fire Distribution Optimization. *J. Beijing Inst. Technol.* **2005**, *25*, 1047–1051.
40. Rahnamayan, S.; Tizhoosh, H.R.; Salama, M.M.A. Opposition versus randomness in soft computing techniques. *Appl. Soft Comput.* **2008**, *8*, 906–918. [\[CrossRef\]](#)
41. Tizhoosh, H.R. Opposition-Based Learning: A New Scheme for Machine Intelligence. In Proceedings of the International Conference on International Conference on Computational Intelligence for Modelling, Control & Automation, Vienna, Austria, 28–30 November 2005; pp. 695–701.
42. Wang, H.; Wang, W.; Zhou, X.; Sun, H.; Zhao, J.; Yu, X.; Cui, Z. Firefly algorithm with neighborhood attraction. *Inf. Sci.* **2016**, *382–383*, 374–387. [\[CrossRef\]](#)
43. Mishra, A.; Agarwal, C.; Sharma, A.; Bedi, P. Optimized gray-scale image watermarking using DWT-SVD and Firefly Algorithm. *Expert Syst. Appl.* **2014**, *41*, 7858–7867. [\[CrossRef\]](#)

44. Zhu, L.; Zhang, Z.; Wang, Y. A Pareto firefly algorithm for multi-objective disassembly line balancing problems with hazard evaluation. *Int. J. Prod. Res.* **2018**, *56*, 7354–7374. [[CrossRef](#)]
45. Wang, J.; Zhang, M.; Song, H.; Cheng, Z.; Chang, T.; Bi, Y.; Sun, K. Improvement and Application of Hybrid Firefly Algorithm. *IEEE Access* **2019**, *7*, 165458–165477. [[CrossRef](#)]
46. Holm, S. A simple sequentially rejective multiple test procedure. *Scand. J. Stat.* **1979**, *6*, 65–70.
47. Anh, H.P.H.; Son, N.N.; Van Kien, C.; Ho-Huu, V. Parameter identification using adaptive differential evolution algorithm applied to robust control of uncertain nonlinear systems. *Appl. Soft Comput.* **2018**, *71*, 672–684. [[CrossRef](#)]
48. Gupta, S.; Deep, K. Improved sine cosine algorithm with crossover scheme for global optimization. *Knowl.-Based Syst.* **2019**, *165*, 374–406. [[CrossRef](#)]
49. Han, M. Hybrid chicken swarm algorithm with dissipative structure and differential mutation. *J. Zhejiang Univ. (Sci. Ed.)* **2018**, *45*, 272–283. [[CrossRef](#)]
50. Wang, S.J.; Gao, X.Z. A survey of research on firefly algorithm. *Microcomput. Its Appl.* **2015**, *34*, 8–11.
51. Ylidizdan, G.; Baykan, O.K. A novel modified bat algorithm hybridizing by differential evolution algorithm. *Expert Syst. Appl.* **2020**, *141*, 19. [[CrossRef](#)]
52. Gupta, D.; Dhar, A.R.; Roy, S.S. A partition cum unification based genetic- firefly algorithm for single objective optimization. *Sādhanā* **2021**, *46*, 121. [[CrossRef](#)]
53. Brest, J.; Maucec, M.S.; Boskovic, B. Single objective real-parameter optimization: Algorithm jSO. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), Donostia, Spain, 5–8 June 2017.
54. Cheng, Z.; Song, H.; Chang, T.; Wang, J. An improved mixed-coded hybrid firefly algorithm for the mixed-discrete SSCGR problem. *Expert Syst. Appl.* **2022**, *188*, 116050. [[CrossRef](#)]
55. 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](#)]

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.