applied sciences

MDPI

Article

Hybrid Algorithm of Improved Beetle Antenna Search and Artificial Fish Swarm

Jian Ni, Jing Tang * and Rui Wang

School of Information and Electrical Engineering, Hebei University of Engineering, Handan 056038, China
* Correspondence: tang1996j@163.com

Abstract: The beetle antenna search algorithm (BAS) converges rapidly and runs in a short time, but it is prone to yielding values corresponding to local extrema when dealing with high-dimensional problems, and its optimization result is unstable. The artificial fish swarm algorithm (AFS) can achieve good convergence in the early stage, but it suffers from slow convergence speed and low optimization accuracy in the later stage. Therefore, this paper combines the two algorithms according to their respective characteristics and proposes a mutation and a multi-step detection strategy to improve the BAS algorithm and raise its optimization accuracy. To verify the performance of the hybrid composed of the AFS and BAS algorithms based on the Mutation and Multi-step detection Strategy (MMSBAS), AFS-MMSBAS is compared with AFS, the Multi-direction Detection Beetle Antenna Search (MDBAS) Algorithm, and the hybrid algorithm composed of the two (AFS-MDBAS). The experimental results show that, with respect to high-dimensional problems: (1) the AFS-MMSBAS algorithm is not only more stable than the MDBAS algorithm, but it is also faster in terms of convergence and operation than the AFS algorithm, and (2) it has a higher optimization capacity than the two algorithms and their hybrid algorithm.

Keywords: artificial fish swarm algorithm; beetle antenna search algorithm; hybrid algorithm; mutation strategy; multistep detection

Citation: Ni, J.; Tang, J.; Wang, R. Hybrid Algorithm of Improved Beetle Antenna Search and Artificial Fish Swarm. *Appl. Sci.* **2022**, *12*, 13044. https://doi.org/10.3390/app122413044

Academic Editors: Sokratis Katsikas and Stylianos Pappas

Received: 12 November 2022
Accepted: 15 December 2022
Published: 19 December 2022

1. Introduction

Optimization refers to the study of problems with multiple feasible solutions and selecting the best solution [1]. Similarly, an optimization algorithm refers to the optimal scheme found in many schemes under certain conditions, such as finding the optimal super parameter in a neural network such that the neural network can achieve optimal performance. Intelligent optimization algorithms are widely used for optimization problems in various fields. For example, the dynamic differential annealed optimization algorithm and an improved moth-flame optimization algorithm were used to solve mathematical and engineering optimization problems [2,3], the sunflower optimization algorithm was proposed for the optimal selection of the parameters of the circuit-based PEMFC model [4], and a novel meta-heuristic equilibrium algorithm was used to find the optimal threshold value for a grayscale image [5]. Intelligent optimization algorithms are mainly divided into four categories, namely, natural simulation optimization algorithms, evolutionary algorithms, plant growth simulation algorithms, and swarm intelligence optimization algorithms, among which swarm intelligence optimization algorithms is the most important [6]. Researchers observed the habits of animals with respect to foraging and communicating with each other, described the characteristics of these animals with algorithms, and, finally, developed the first swarm intelligence optimization algorithm.

Common swarm intelligence optimization algorithms include the Cuckoo Search Algorithm (CSA) [7], the particle swarm optimization algorithm (PSO) [8], the Ant Colony Optimization algorithm (ACO) [9], and the Firefly Algorithm (FA) [10]. Early on, scholars proposed the Artificial Fish Swarm algorithm (AFS) [11], which was inspired by fish

communities' food-seeking, gathering, and following behaviors. The AFS algorithm can achieve good convergence in the early stage, but the algorithm suffers from problems such as a slow convergence speed and low optimization accuracy in the later stage [12]. Accordingly, scholars have been researching a solution to this problem. Zhang et al. [13] dynamically adjusted the visual field and step size of the artificial fish and improved the updating strategy of the position of the artificial fish. Liu et al. [14] used chaotic transformation to initialize the positions of the individual fish in order to render the fish more evenly distributed in a limited area. In addition, a physical transformation model was built based on the relationship between motion and physical fitness. The algorithm effectively improved the speed of convergence and the accuracy of optimization. Li et al. [15] used the steepest descent method to update the artificial fish with the best fitness values, instructed other artificial fish through the exchange of information between the artificial fish, and accelerated the convergence speed of the artificial fish algorithm. Later, a hybrid algorithm was created by Liu et al. [16], who introduced the movement operator of the particle swarm algorithm in order to adjust the movement direction and position of the artificial fish and enhance their ability to escape the local optimum. Li et al. [17] introduced the gene exchange behavior of GA into the AFS algorithm to enhance its ability to escape the local optimum and improve its search efficiency. Inspired by the above methods, it has been found that the advantages of one algorithm can compensate for the disadvantages of another algorithm.

In recent years, many optimization algorithms have been proposed by scholars. Among them, Jiang was inspired by the feeding and mating behaviors of beetles and proposed a new intelligent optimization algorithm based on these animals: the beetle antenna search algorithm (BAS) [18]. Since the time and space complexity of the BAS algorithm is lower and more efficient than that of the swarm intelligence algorithm, it is also widely used in other fields, such as medicine [19], engineering design [20], and image processing [21]. In addition, the BAS algorithm has obvious advantages over other algorithms in terms of convergence speed, which addresses the problem posed by the slow convergence speed of the AFS algorithm in the later period. However, the BAS algorithm, employing a single beetle for optimization, is prone to yielding values corresponding to a local extremum and has poor optimization stability. Therefore, is has been improved by many scholars. Wang et al. [22] expanded a single beetle to a population of beetles, optimized the step size update, and proposed a beetle swarm antenna search algorithm (BSAS) that combines the swarm intelligence algorithm with the feedback-based step-size update strategy. Zhao et al. [23] proposed an algorithm combining the BAS and GA to solve the BAS algorithm's susceptibility to yielding values corresponding to a local extremum in the optimization of multimodal complex functions. Khan [24] used ADAM update rules to adaptively adjust the step size in each iteration to improve the BAS algorithm.

In order to solve the instability and low optimization accuracy at high dimensions of the AFS and BAS algorithms, this paper proposes a hybrid algorithm comprising the artificial fish swarm and beetle search algorithms (AFS-MMSBAS). The algorithm combines the advantages of the BAS and AFS, of which the BAS algorithm is improved by using the mutation of beetles and a multi-step detection strategy to improve optimization accuracy. In the early stage, the AFS algorithm is used for a global search and convergence to a certain extent; then, the improved BAS algorithm is used to accelerate convergence. The simulation results show that the AFS-MMSBAS algorithm has significantly improved stability and optimization compared with the AFS, MDBAS, and AFS-MDBAS algorithms.

2. Hybrid Algorithm of AFS and Improved BAS

To address the above problems, this section proposes the following algorithms:

(1) Improved BAS algorithm: The use of the mutation strategy and the multi-step detection strategy of the BAS are proposed to allow the beetle to escape the local extremum value and accelerate convergence speed.

(2) Hybrid of the AFS algorithm and BAS algorithm: The respective advantages of the AFS and BAS algorithms are combined to improve the optimization stability of the algorithm in high dimensions.

2.1. Mutation Selection Strategy of BAS

The BAS algorithm is prone to yielding values corresponding to a local extremum. In order to make the beetle escape a local extremum, the application of a mutation strategy to the beetle is proposed. In multi-dimensional problems, if each dimension changes at the same time, the following principle applies: the higher the dimension, the lower the probability that the newly generated individual is superior to the current individual. Therefore, this paper proposes a method wherein each dimension changes separately. If the beetle with the dimensional value changed is better than the one without a change, the current beetle will be updated. Otherwise, the mutation operation is continued in the next dimension—and so on—until the last dimension, as shown in Figure 1.

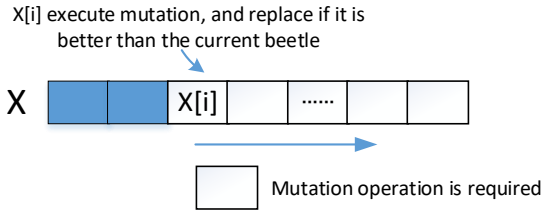


Figure 1. Variation diagram of single beetle in high-dimensional case.

In addition, in order to acquire better results, a single variation is expanded to multiple variations. The beetle with the best fitness from multiple mutant beetles is selected, and the next step is initiated. The variation process is shown in Figure 2.

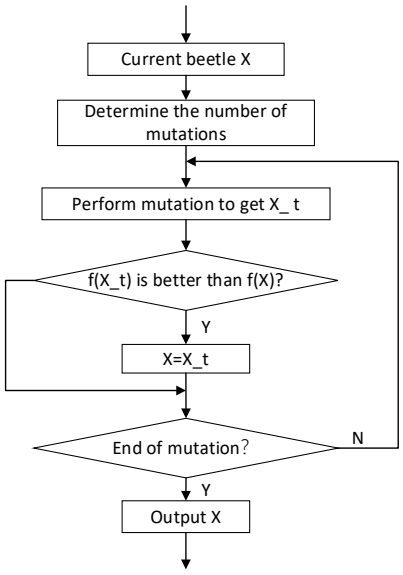


Figure 2. Flow chart of variation.

2.2. Multistep Detection Strategy

The choice of the initial step size has a great impact on the performance of the BAS algorithm. If the initial step size is too large, the algorithm converges slowly. If it is too small, it becomes easy to fall into the local extreme value and difficult to escape. The initial step size is usually determined manually and is then gradually reduced under the effect of a decreasing factor to achieve the goal of approaching the optimal value. Inspired by the multi-directional detection strategy [23], this paper proposes its own multi-step detection

strategy. This strategy dynamically increases or decreases the step size in the optimization process and reduces the impact of the initial step size on the algorithm's performance. The basic idea is to expand and reduce the current step size by a certain multiple, move forward according to different steps, and select the step size with the best fitness value for an update. This method can accelerate the convergence speed and increase the possibility of escaping local extremum. Figure 3 is a schematic diagram showing the multiple-step detection of the beetles.

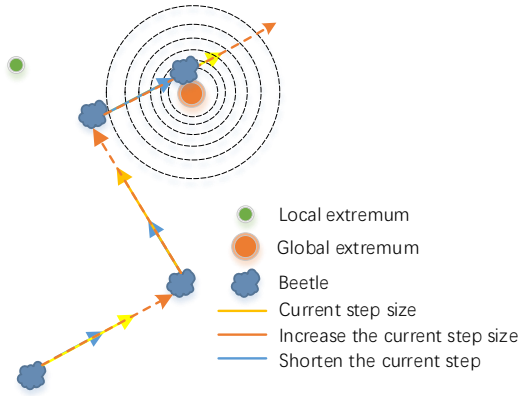


Figure 3. Schematic diagram of multi-step detection.

Set the current iteration as i, the step size as $step_i$, and scale the step size according to Formula (1), where $step_i^{(-)}$ represents the shortened step size, φ^s is the reduction factor, $step_i^{(+)}$ represents the extended step size, and φ^e is the amplification factor.

$$\begin{cases} step_i^{(-)} = \varphi^s * step_i \\ step_i^{(+)} = \varphi^e * step_i \end{cases} \tag{1}$$

$f(x_i, s)$ is the fitness of the current beetle x_i after a step length of s in a certain direction. Execute $f(x_i, step)$, $f\left(x_i, step_i^{(-)}\right)$, and $f\left(x_i, step_i^{(+)}\right)$ at the same time, and then select the step size corresponding to the optimal fitness $step_b$ to update it according to Formula (2), where δ is the step-decreasing factor

$$step_{i+1} = step_b * \delta \tag{2}$$

Pseudocode of multi-step detection strategy is shown in Algorithm 1.

Algorithm 1. Multi-step detection pseudo code

1. **Inputs**: $\text{step}_{\text{list}}$ = narrow_step, *step*, enlarge_step
2. **for each** $step_i$ of the $\text{step}_{\text{list}}$ do
3. $fit = f(x, step_i)$
4. **end for**
5. select $step_{best}$ of corresponding to the best fit
6. do Formula (2)
7. **return** a new *step*

In addition, this paper initializes the step size of the beetle according to Formula (3) to avoid manual interference

$$step = \text{mean}(\text{abs}(x))/3 + \varphi \tag{3}$$

wherein x represents the position of the initial beetle, δ represents the step-decreasing factor, and φ is a constant to prevent the initial step from equaling 0.

2.3. Hybrid Strategy of FAS-MMSBAS Algorithm

The FAS-MMSBAS algorithm is composed of the BAS algorithm and the AFS algorithm combined through a simple hybrid strategy. In the early stage, the AFS algorithm is used for global optimization and fast convergence, and in the later stage, the improved BAS algorithm is used to replace it and continue optimization. The hybrid algorithm reduces the instability of the BAS algorithm caused by individual random initialization to a certain extent, accelerates the convergence speed, and improves the optimization accuracy. The most important consideration regarding the hybrid algorithm is to determine when to end the AFS algorithm. If it is too early, the global optimization ability of the AFS algorithm will be too small. If it is too late, it will converge too slowly and waste time. Therefore, according to the characteristics of the AFS algorithm, this paper proposes the termination of the AFS algorithm when the optimal value remains unchanged or the convergence speed becomes significantly slow. The optimal individual obtained by the AFS algorithm is taken as the initial individual of the BAS algorithm. The interruption rule of the AFS algorithm is shown in Algorithm 2:

Algorithm 2. FAS algorithm termination rule pseudo code

```
Input: parameters p and q
if the repeat times of best individual == q, or maximum iterations of AFS > q then
    initial individual of the BAS = The optimal individual of AFS
end if
do Improve BAS
```

The parameters p and q are manually set values, where p represents the number of iterations of the best value, and q represents the maximum number of iterations of the AFS algorithm in the hybrid algorithm. When the fish swarm iterates p times and the optimal position remains unchanged, the AFS algorithm ends. This means that the current fish population may find the best value or fall into a local extreme value. In addition, because the AFS algorithm converges slowly in the later period, if condition p is not met under certain conditions, q is set as the termination condition of the AFS algorithm to avoid wasting time. The implementation steps of FAS-MMSBAS algorithm are as follows:

Step 1. Set the initial parameters of the algorithm, including the populations, visual properties, attempt number, crowding factor, steps of the fish, antenna number, antenna length, and variation rate.

Step 2. Use the AFS algorithm to continuously optimize until the AFS iteration termination condition is met.

Step 3. Take the best individual in the optimization stage of the AFS algorithm as the initial position for the optimization of the beetle.

Step 4. Judge whether there is variation. If there is, go to step 5; if not, go to step 6.

Step 5. Allow the dimension to be mutated, until the end of the mutation.

Step 6. Perform multi-directional and multistep detection on the beetle; then, select the best beetle to continue to move forward and update the current beetle step length and whisker length.

Step 7. If the termination conditions are met, the algorithm is ended; if not, go to step 4.

Figure 4 provides a flow chart of the AFS-MMSBAS algorithm depicted according to the above steps.

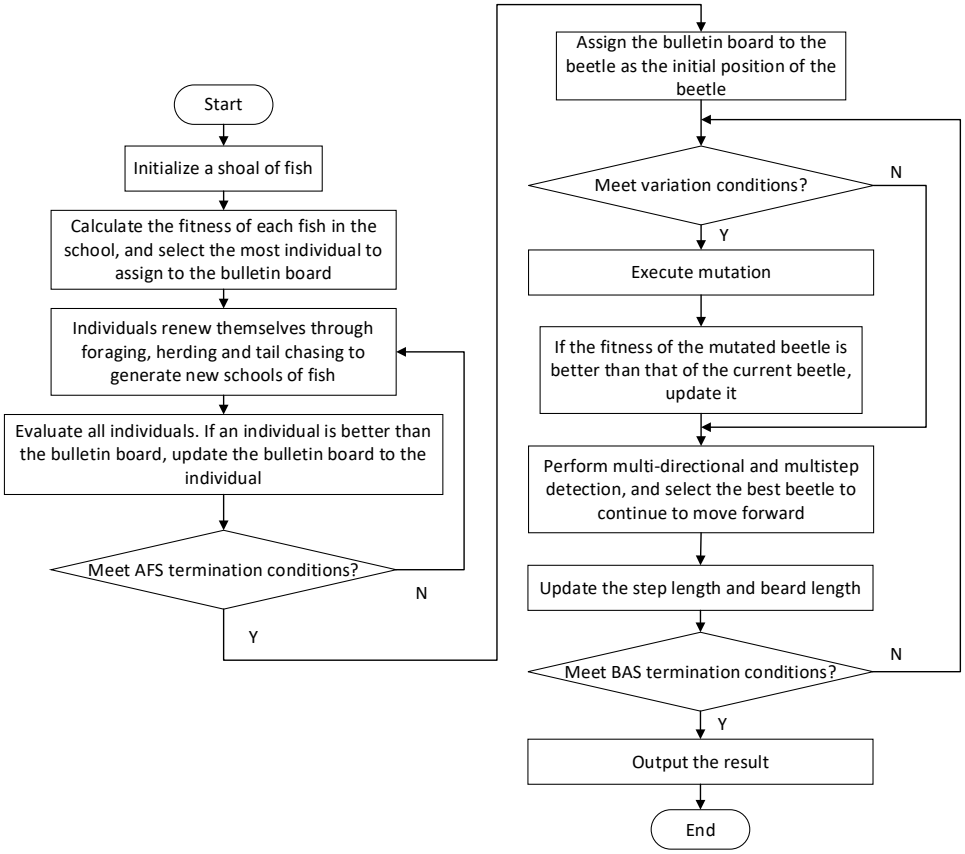


Figure 4. Flow chart of FAS-MMSBAS algorithm.

3. Results and Analysis

3.1. Test Function and Evaluation Indicator

In order to verify the performance of the hybrid algorithm proposed in this paper, 11 typical test functions are selected for simulation experiments, as shown in Table 1, including the function name, search range, optimal position, and optimal value. In addition, Table 2 is a supplement to Table 1, which is used to show function expressions. f_1~f_4 are unimodal functions, which are used to test the optimization ability of the algorithm for a single extreme value function. f_5~f_{10} are multimodal functions, which are used to test the global search ability of the algorithm under multiple local extreme values and the ability to solve complex optimization problems. f_2 and f_{10} are non-partitioned functions with a control relationship between dimensions; f_4 is a noise function with random interference; and f_{11} is a step function.

Table 1. Benchmark test functions.

Function	Function Name	Search Range	Optimal Position	Optimum Value
f_1	Sphere	$x_i\epsilon[-100,100]$	(0,0, . . . ,0)	0
f_2	Rosenbrock	$x_i\epsilon[-50,50]$	(1,1, . . . ,1)	0
f_3	SchwefelP222	$x_i\epsilon[-10,10]$	(0,0, . . . ,0)	0
f_4	Quartic	$x_i\epsilon[-1.28,1.28]$	(0,0, . . . ,0)	0
f_5	Rastrigin	$x_i\epsilon[-5.12,5.12]$	(0,0, . . . ,0)	0
f_6	Griewank	$x_i\epsilon[-600,600]$	(0,0, . . . ,0)	0
f_7	Ackley	$x_i\epsilon[-32,32]$	(0,0, . . . ,0)	0
f_8	Levy and Montalvo 2	$x_i\epsilon[-5,5]$	(1,1, . . . ,1)	0
f_9	Schwefel226	$x_i\epsilon[-500,500]$	(420.968746, . . . , 420.968746)	$-418.98288 \times$ D
f_{10}	Schaffer	$x_i\epsilon[-100,100]$	(0,0, . . . ,0)	0
f_{11}	Step	$x_i\epsilon[-100,100]$	(0,0, . . . ,0)	0

Table 2. Expressions of benchmark test function; this table is a supplement to Table 1.

Function Name	Expression
Sphere	$f_1 = \sum_{i=1}^{D} x_i^2$
Rosenbrock	$f_2 = \sum_{i=1}^{D-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$
SchwefelP222	$f_3 = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$
Quartic	$f_4 = \sum_{i=1}^{D} \left(i x_i^4\right) + random[0,1)$
Rastrigin	$f_5 = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$
Griewank	$f_6 = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \left[cos\left(\frac{x_i}{\sqrt{i}}\right)\right] + 1$
Ackley	$f_7 = -20exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - exp\left(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi x_i)\right) + 20 + e$
Levy and Montalvo 2	$f_8 = 0.1\left\{sin^2(3\pi x_1) + \sum_{i=1}^{D} (x_i - 1)^2\left[1 + sin^2(3\pi x_i + 1)\right] + (x_D - 1)^2\left[1 + sin^2(2\pi x_D)\right]\right\}$
Schwefel226	$f_9 = -\sum_{i=1}^{D} x_i sin\left(\sqrt{\lvert x_i \rvert}\right)$
Schaffer	$f_{11} = -\sum_{i=1}^{D-1} \left(x_i^2 + x_{i+1}^2\right)^{0.25} \left[sin^2\left(50\left(x_i^2 + x_{i+1}^2\right)^{0.1}\right) + 1\right]$
Step	$f_{10} = \sum_{i=1}^{D} (floor(x_i + 0.5))^2$

Three indicators are selected in the experiment to evaluate the performance of the algorithm:

(1) The mean, which is the average value of the algorithm run many times, reflecting the quality of the solution. The closer the average value is to the optimal value, the better the algorithm's results.

(2) The standard deviation, which is the standard deviation of the optimal value and can reflect the stability of the algorithm. The smaller the standard deviation, the better the stability of the algorithm.

(3) The running time, in seconds, which is the average time of multiple runs of the algorithm under the same environment and parameters, reflecting the running efficiency of the algorithm.

The calculation formulas of the three evaluation indicators are shown in Formulas (4)–(6):

$$\text{Mean} = \left| \frac{1}{N} \sum_{i=1}^{n} f(x_i) \right| \tag{4}$$

$$\text{Standard} = \sqrt{\frac{1}{N} \sum_{i}^{n} (f(x_i) - Mean)^2} \tag{5}$$

$$\text{Time} = \frac{1}{N} \sum_{i}^{n} t_i \tag{6}$$

3.2. Computational Complexity Analysis of FAS-MMSBAS Algorithm

The main improvement to the algorithms in this paper is that after the convergence of the AFS algorithm becomes slow, the improved BAS algorithm is used to continue to search for the optimal position. In the later stage, the optimization is changed from population optimization to individual optimization, which reduces the complexity of the algorithm. When the optimization problem dimension is D and the population number is N, the

complexity analysis of the FAS-MMSBAS algorithm is as follows: the number of iterations of the AFS algorithm is t_1, the number of iterations of the MMSBAS algorithm is t_2, and the total number of iterations is $T = t_1 + t_2$. In the optimization stage of the AFS algorithm, the computational complexity of initializing the artificial fish group is O(DN). In the iteration process, the computational complexity of clustering behavior is $O(DN^2)$, and that of chasing behavior is $O(DN^2)$; thus, the total complexity of this stage is $O(DN) + O(2DN^2)t_1$. In the iteration process of the MMSBAS algorithm, the number of multi-directional detection is c_1, the number of multi-step detection is 3, the computational complexity of the multi-directional multistep detection strategy is $O(3c_1D)$, the computational complexity of the mutation strategy is O(D), and the total computational complexity is $O(c_2D)t2$, c1 < c2. Combined with the two stages, the total computational complexity of the algorithm can be approximated as $O(DN^2)t_1 + O(c_2D)t_2$.

It can be seen from Table 3 that when the number of iterations is the same, the order of algorithm complexity from low to high is BAS < AFS-DBAS < AFS-MMBAS < AFS.

Table 3. Algorithm complexity.

Algorithm	Calculation Complexity	Algorithm	Calculation Complexity
AFS	$O(DN^2)$ T	AFS-DBAS	$O(DN^2)t_1 + O(c_1D)t_2$
BAS	O(D) T	AFS-MMBAS	$O(DN^2)t_1 + O(c_2D)t_2$

3.3. Simulation Experiment and Result Analysis

The algorithms' comparison and analysis are arranged as follows: (1) first, compare AFS-MMSBAS algorithm with two basic algorithms and the hybrid algorithm of the basic algorithm based on different dimensions; (2) further analyze the stability and convergence speed of the AFS-MMSBAS algorithm in high dimensions; and (3) compare the performance of the AFS-MMSBAS algorithm and other related algorithms in high dimensions.

3.3.1. Parameter Setting

The relevant parameter settings with which to avoid the influence of different parameters on the algorithm are shown in Table 4. All experiments in this paper are based on the PyCharm platform and were run on the Window10 (64 bit) operating system with the processor configured as Inter (R) Core (TM) i5-5200U CPU @ 2.20GHz 2.20 GHz with a memory of 12 GB.

Table 4. Initial parameters of AFS-MMSBAS algorithm.

Parameter	Value	Parameter	Value
population	30	antenna number	10
visual	6	antenna length	step/5
attempt number	20	δ	0.95
crowding factor	0.618	variation rate	0.05
step of fish	0.6	p	40
iterations	800	q	0.65 × iterations

3.3.2. Performance Comparison in Different Dimensions

The AFS-MMSBAS algorithm is compared with the MDBAS, AFS, and AFS-MDBAS algorithms under the same parameters. The performance of the algorithms is tested with respect to the dimensions D = 10, D = 100, and D = 200 of the basic test functions. In order to prevent the experimental results from being affected by randomness, each test function is independently run 50 times. The other parameters are the same as those in 3.2.1. Finally, the average value, standard deviation, and running time of the optimization results are obtained. The experimental results are shown in Table 5, and the optimal results are shown in bold.

By analyzing Table 2, it can be seen that in low dimensions (D = 10), the MDBAS, AFS, AFS-MDBAS, and AFS-MMSBAS algorithms have good optimization ability and stability for most test functions. Among them, the MDBAS algorithm obtains the optimal mean and standard deviation for f_5, the AFS algorithm obtains the optimal mean and standard deviation for f_2 and f_{11}, the AFS-MDBAS algorithm obtains the optimal mean and standard deviation for f_3 and f_{10}, and the AFS-MMSBAS algorithm obtains the optimal mean and standard deviation for f_1, f_4, f_8, and f_9. At low dimensions, each algorithm has certain advantages over different functions, and the gap is not obvious.

Table 5. Comparisons of four algorithms regarding the optimization of test functions f_1~f_{11} .

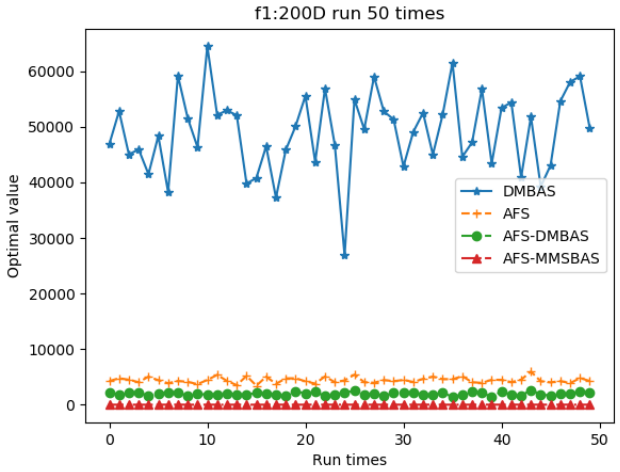
Func.	Algorithm	D = 10			D = 100			D = 200		
		Mean	**Standard**	**Time**	**Mean**	**Standard**	**Time**	**Mean**	**Standard**	**Time**
f_1	MDBAS	4.5×10^{-32}	1.91×10^{-32}	1.008	9.16×10^{2}	3.98×10^{2}	3.481	4.91×10^{4}	7.16×10^{3}	7.045
	AFS	0.627	0.154	94.87	3.08×10^{2}	51.4	149.8	4.43×10^{3}	5.14×10^{2}	213.4
	AFS-MDBAS	8.37×10^{-28}	2.28×10^{-27}	13.25	7.51	3.64	68.92	2.0×10^{3}	2.99×10^{2}	82.17
	AFS-MMSBAS	$\mathbf{4.5 \times 10^{-106}}$	$\mathbf{4.99 \times 10^{-105}}$	17.27	**0.00814**	**0.00454**	72.97	**19.1**	**4.67**	81.85
f_2	MDBAS	4.62×10^{2}	9.69×10^{2}	1.341	3.92×10^{6}	4.23×10^{6}	4.911	5.21×10^{8}	1.58×10^{8}	7.65
	AFS	**67.3**	**18.6**	105.5	3.57×10^{4}	2.29×10^{4}	166.4	1.84×10^{6}	3.99×10^{5}	253.7
	AFS-MDBAS	1.25×10^{2}	3.12×10^{2}	4.46	1.74×10^{3}	1.23×10^{3}	83.17	1.58×10^{6}	6.05×10^{5}	130.7
	AFS-MMSBAS	90.2	3.22×10^{2}	8.071	$\mathbf{4.08 \times 10^{2}}$	$\mathbf{3.99 \times 10^{2}}$	86.12	$\mathbf{5.18 \times 10^{3}}$	$\mathbf{1.69 \times 10^{3}}$	133.7
f_3	MDBAS	0.0659	0.454	1.333	3.19×10^{9}	2.25×10^{10}	4.979	1.94×10^{37}	7.36×10^{37}	7.355
	AFS	2.01	0.345	131.9	1.32×10^{2}	3.56	130.1	2.67×10^{2}	5.52	180.0
	AFS-MDBAS	$\mathbf{4.99 \times 10^{-14}}$	$\mathbf{1.26 \times 10^{-13}}$	15.71	38.1	4.93	23.63	1.41×10^{2}	9.85	32.26
	AFS-MMSBAS	0.0925	0.0823	29.39	**2.74**	**0.92**	32.33	**10.3**	**2.19**	42.11
f_4	MDBAS	0.195	0.118	1.561	23.3	4.8	4.958	2.59×10^{2}	48.0	7.638
	AFS	0.388	0.142	134.5	71.1	18.5	176.7	3.21×10^{2}	81.9	317.8
	AFS-MDBAS	0.196	0.0986	9.239	13.6	3.47	15.24	64.1	11.5	28.59
	AFS-MMSBAS	**0.0361**	**0.0244**	13.04	**1.18**	**0.311**	20.11	**3.43**	**0.7**	42.35
f_5	MDBAS	$\mathbf{2.56 \times 10^{-15}}$	$\mathbf{2.07 \times 10^{-15}}$	1.241	2.38	1.33	4.92	1.27×10^{2}	19.4	7.058
	AFS	0.544	0.141	126.2	1.81×10^{2}	22.2	189.4	6.46×10^{2}	50.9	332.4
	AFS-MDBAS	2.98×10^{-15}	2.34×10^{-15}	19.6	1.87	0.97	43.93	1.2×10^{2}	17.5	99.94
	AFS-MMSBAS	2.66×10^{-15}	3.17×10^{-15}	28.56	$\mathbf{3.45 \times 10^{-8}}$	$\mathbf{6.32 \times 10^{-8}}$	47.26	**0.00236**	**0.00541**	108.7
f_6	MDBAS	**0.0565**	0.0391	1.865	8.98	4.35	4.228	4.58×10^{2}	62.1	8.403
	AFS	0.182	**0.0285**	64.83	1.43×10^{2}	1.0×10^{2}	41.34	1.48×10^{3}	79.3	60.52
	AFS-MDBAS	0.104	0.0695	20.43	7.16	2.95	25.74	4.48×10^{2}	61.8	40.41
	AFS-MMSBAS	0.0982	0.0579	28.48	**0.335**	**0.0928**	27.9	**2.2**	**0.601**	47.5
f_7	MDBAS	0.279	0.666	1.663	18.3	0.339	4.346	19.1	**0.18**	8.062
	AFS	2.37	**0.234**	153.1	12.6	**0.327**	248.0	17.9	0.276	321.0
	AFS-MDBAS	**0.25**	0.612	19.04	11.1	0.902	74.87	18.1	0.529	92.61
	AFS-MMSBAS	0.48	0.594	23.02	**4.22**	0.778	86.45	**4.24**	0.7	109.0
f_8	MDBAS	0.0793	0.148	1.615	2.14	0.919	5.304	11.6	1.74	8.444
	AFS	0.194	0.0669	104.6	31.4	1.1	137.7	78.4	2.25	241.9
	AFS- MDBAS	0.0222	0.0392	12.69	16.5	12.7	21.91	75.5	7.33	28.76
	AFS-MMSBAS	**0.00154**	**0.0026**	16.62	**0.00294**	**0.00365**	29.29	**0.0111**	**0.00735**	48.42
f_9	MDBAS	-2.35×10^{3}	3.84×10^{2}	1.136	-2.03×10^{4}	1.15×10^{3}	4.789	-3.57×10^{4}	1.67×10^{3}	7.777
	AFS	-3.11×10^{3}	2.59×10^{2}	119.6	-2.45×10^{4}	$\mathbf{4.59 \times 10^{2}}$	201.1	-4.66×10^{4}	$\mathbf{9.21 \times 10^{2}}$	311.3
	AFS- MDBAS	-2.4×10^{3}	3.64×10^{2}	13.19	-2.51×10^{4}	7.1×10^{2}	96.28	-4.68×10^{4}	9.3×10^{2}	169.5
	AFS-MMSBAS	$\mathbf{-4.18 \times 10^{3}}$	**20.8**	15.8	$\mathbf{-4.12 \times 10^{4}}$	5.55×10^{2}	104.99	$\mathbf{-8.01 \times 10^{4}}$	1.84×10^{3}	177.2
f_{10}	MDBAS	0.0778	0.0691	3.582	0.172	0.164	11.01	0.279	0.162	19.64
	AFS	0.00246	1.33×10^{-10}	253.9	0.00246	2.02×10^{-10}	591.5	0.00246	1.75×10^{-10}	888.0
	AFS- MDBAS	**0.00246**	$\mathbf{2.85 \times 10^{-17}}$	24.41	**0.00246**	$\mathbf{2.76 \times 10^{-17}}$	89.42	**0.00246**	$\mathbf{2.98 \times 10^{-17}}$	109.7
	AFS-MMSBAS	0.00246	4.56×10^{-9}	27.2	0.00246	1.71×10^{-10}	95.4	0.00246	4.87×10^{-11}	149.5
f_{11}	MDBAS	3.68	2.61	2.869	1.37×10^{4}	3.45×10^{3}	13.19	1.11×10^{5}	1.55×10^{4}	28.22
	AFS	**0.0**	**0.0**	236.7	**28.8**	**24.2**	397.2	3.31×10^{3}	$\mathbf{4.71 \times 10^{2}}$	617.4
	AFS- MDBAS	0.48	0.707	18.45	5.48×10^{2}	1.45×10^{2}	54.51	9.63×10^{3}	9.6×10^{2}	225.1
	AFS-MMSBAS	0.06	0.24	21.12	1.87×10^{2}	57.7	61.78	$\mathbf{2.08 \times 10^{3}}$	7.05×10^{2}	271.2

At high dimensions (D = 100, 200), the averages of the MDBAS algorithm and the AFS algorithm are far greater than the optimal value, and the standard deviation is also large, indicating that the optimization ability and stability of these two algorithms are poor. For most functions, the averages and standard deviations of the AFS-MDBAS hybrid algorithm are better than those of the separate MDBAS and AFS algorithms, which shows that the hybrid algorithm improves the optimization ability and stability of the MDBAS and AFS algorithms. The AFS-MMSBAS algorithm is an improvement of the AFS-MDBAS algorithm. Although the algorithm is not the best at optimizing the functions f7, f9, f10, and f11, it is not far behind the algorithm with the best performance. In general, the AFS-MMSBAS algorithm proposed in this paper has better optimization ability and stability with respect to high-dimensional problems.

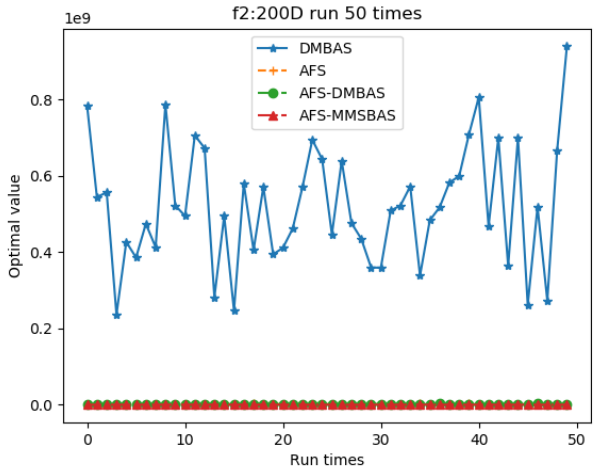
Finally, under the same number of iterations, it can be seen that the running time of the algorithm increases with the increase in dimensions. Moreover, whether at low or high dimensions, the hybrid algorithm greatly shortens the running time compared with the AFS algorithm. However, compared with the BAS algorithm, it greatly increases the running time. The hybrid algorithm trades time for optimization capability and stability. The running time relationship between algorithms is as follows: T (MDBAS) < T (AFS-MDBAS) < T (AFS-MMSBAS) < T (AFS).

3.3.3. Stability Analysis of Algorithms

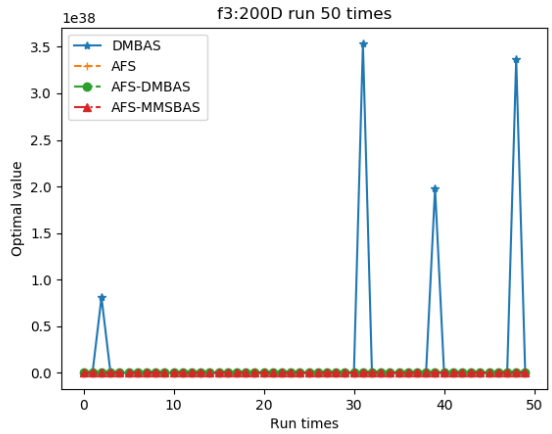
In order to observe and compare the stability of each algorithm at higher dimensions more intuitively, functions $f_1 \sim f_{11}$ are run 50 times in order to develop line graphs (D = 200), as shown in Figure 5a–k. The more stable the curve, the better the stability of the algorithm. The smaller the value of the curve, the better the optimization ability of the algorithm.



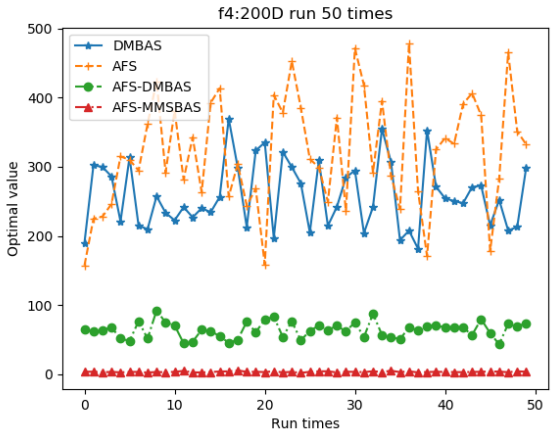
(**a**)



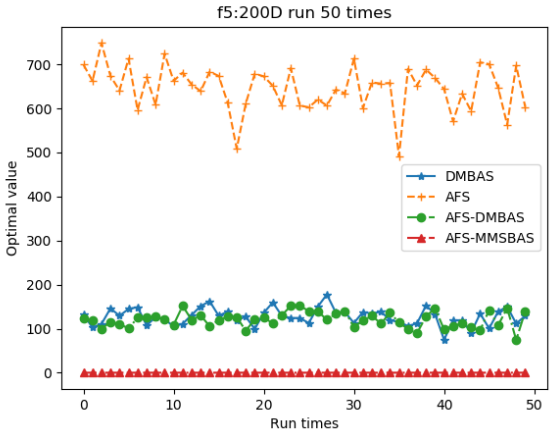
(**b**)


f3:200D run 50 times
1e38
DMBAS
AFS
AFS-DMBAS
AFS-MMSBAS
Optimal value
Run times

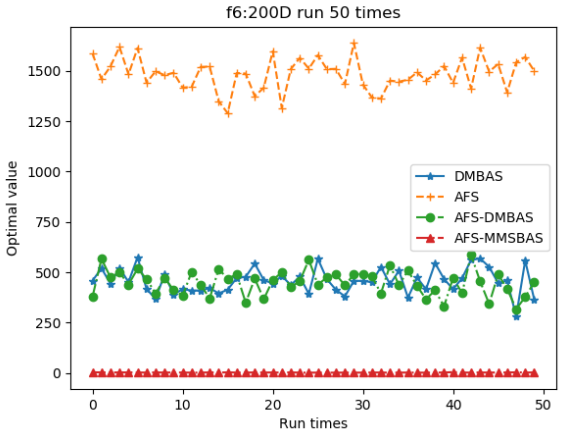
(**c**)


f4:200D run 50 times
DMBAS
AFS
AFS-DMBAS
AFS-MMSBAS
Optimal value
Run times

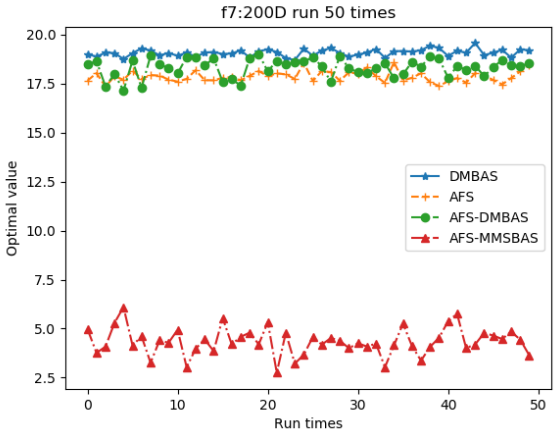
(**d**)


f5:200D run 50 times
DMBAS
AFS
AFS-DMBAS
AFS-MMSBAS
Optimal value
Run times

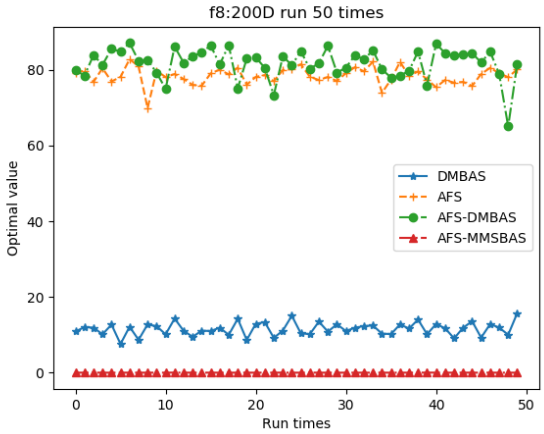
(**e**)


f6:200D run 50 times
DMBAS
AFS
AFS-DMBAS
AFS-MMSBAS
Optimal value
Run times

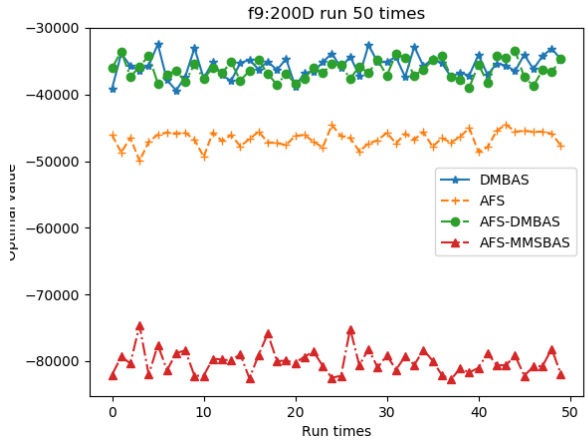
(**f**)


f7:200D run 50 times
DMBAS
AFS
AFS-DMBAS
AFS-MMSBAS
Optimal value
Run times

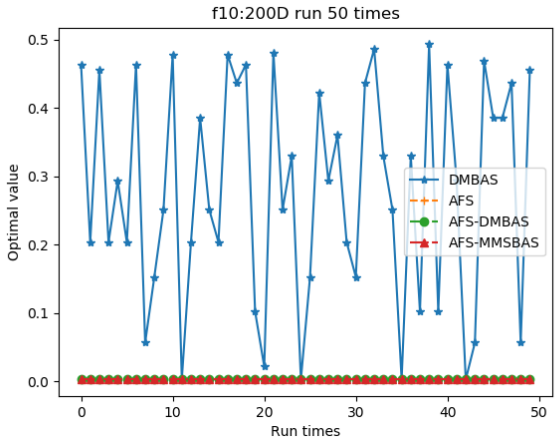
(**g**)


f8:200D run 50 times
DMBAS
AFS
AFS-DMBAS
AFS-MMSBAS
Optimal value
Run times

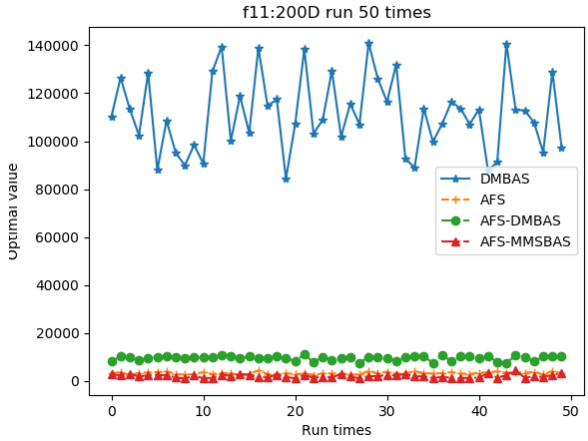
(**h**)



(**i**)



(**j**)



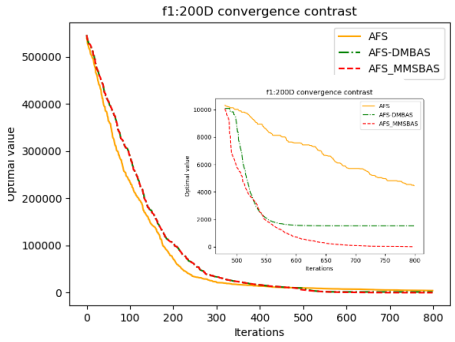
(**k**)

Figure 5. D = 200; comparison of stability of functions $f_1 \sim f_{11}$. (**a**) Comparison of stability of four algorithms for function f_1; (**b**) Comparison of stability of four algorithms for function f_2; (**c**) Comparison of stability of four algorithms for function f_3; (**d**) Comparison of stability of four algorithms for function f_4; (**e**) Comparison of stability of four algorithms for function f_5; (**f**) Comparison of stability of four algorithms for function f_6; (**g**) Comparison of stability of four algorithms for function f_7; (**h**) Comparison of stability of four algorithms for function f_8; (**i**) Comparison of stability of four algorithms for function f_9; (**j**) Comparison of stability of four algorithms for function f_{10}; (**k**) Comparison of stability of four algorithms for function f_{11}.

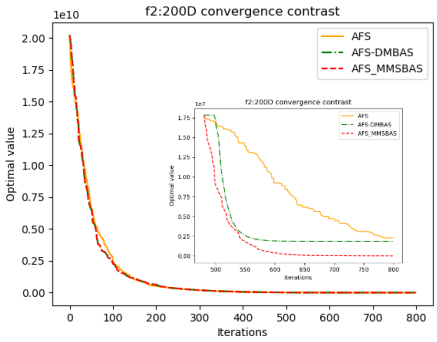
For unimodal functions $f_1 \sim f_4$, the fluctuation of the MDBAS algorithm (blue) and the AFS algorithm (orange) is large, and the curve is far from the optimal value. Combined with Table 5, it can be seen that the stability and optimization accuracy of the AFS-MDBAS (green) and AFS-MMSBAS algorithm (red) are better than those of the individual algorithms. This shows that the hybrid algorithm composed of the AFS and BAS algorithms achieves superior performance for unimodal functions. For the multimodal functions $f_5 \sim f_{10}$, the advantages of the AFS-MDBAS algorithm in terms of stability and optimization accuracy are not obvious compared with the AFS and MDBAS algorithms, but the advantages of the AFS-MMSBAS algorithm are quite obvious. The curve is the most stable and is below all the other curves, which further demonstrates that the AFS-MMSBAS algorithm has the best stability and optimization ability.

3.3.4. Analysis of the Convergence of the Algorithm

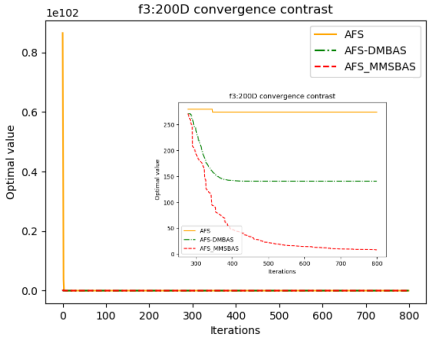
To observe the convergence of the improved BAS algorithm at high dimensions, the convergence curve of functions $f_1 \sim f_{11}$ when D = 200 is developed, as shown in Figure 6a–k. It can be seen from the figure that the convergence curves of the three algorithms are basically coincident in the early stage. Since the AFS algorithm is used for optimization at this stage, the degree of convergence is basically the same. In the later stage, except for functions f_{10} and f_{11}, the convergence curve of the AFS-MMSBAS algorithm drops faster than that of the AFS-DMBAS and AFS algorithms. This shows that the improved BAS algorithm has better convergence speed than the MDBAS algorithm. In addition, the convergence curve of the AFS-MMSBAS algorithm is lower than those of the other algorithms, which also shows that the optimization ability of the algorithm is better than that of the AFS-MDBAS and AFS algorithms.



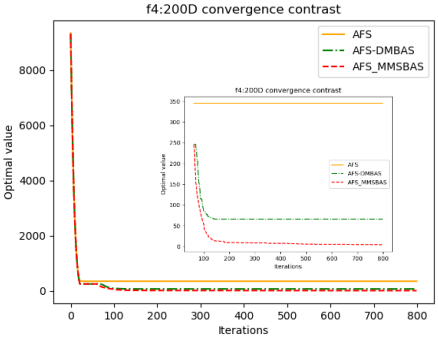
(**a**)



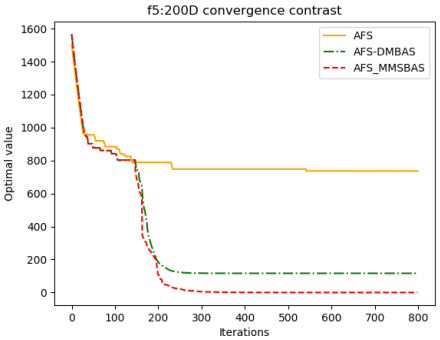
(**b**)



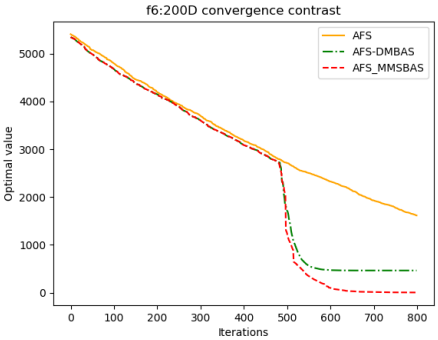
(**c**)



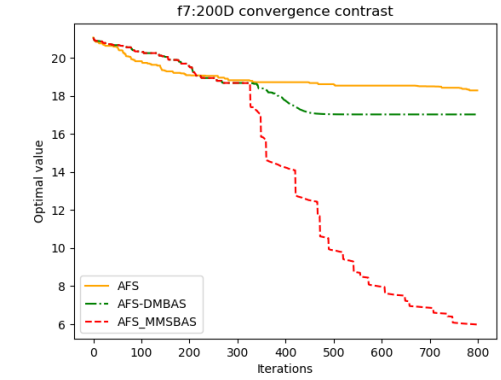
(**d**)



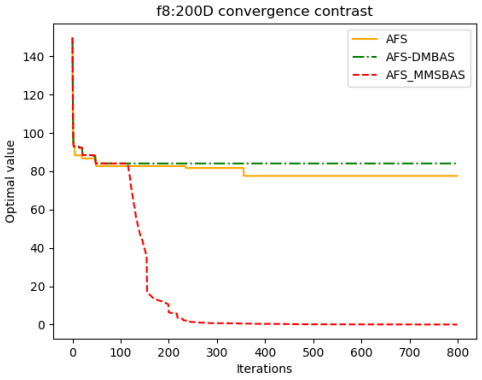
(**e**)



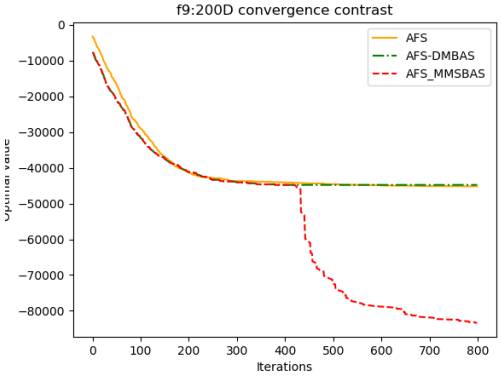
(**f**)



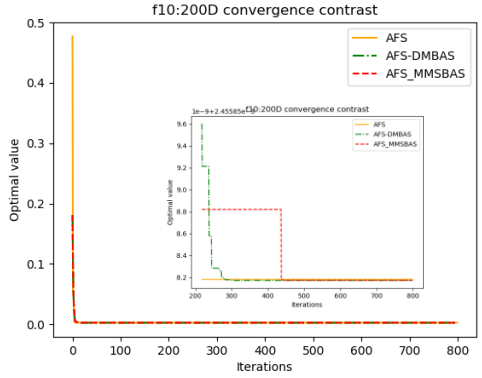
(**g**)



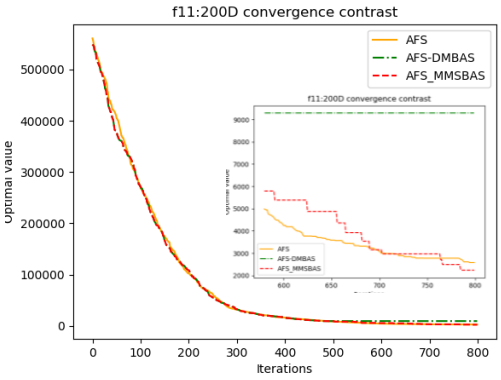
(**h**)



(**i**)



(**j**)



(**k**)

Figure 6. D = 200, functions $f_1 \sim f_{11}$ convergence performance. (**a**) Comparison of convergence of three algorithms for function f_1; (**b**) Comparison of convergence of three algorithms for function f_2; (**c**) Comparison of convergence of three algorithms for function f_3; (**d**) Comparison of convergence of three algorithms for function f_4; (**e**) Comparison of convergence of three algorithms for function f_5; (**f**) Comparison of convergence of three algorithms for function f_6; (**g**) Comparison of convergence of three algorithms for function f_7; (**h**) Comparison of convergence of three algorithms for function f_8; (**i**) Comparison of convergence of three algorithms for function f_9; (**j**) Comparison of convergence of three algorithms for function f_{10}; (**k**) Comparison of convergence of three algorithms for function f_{11}.

3.3.5. Comparison with Other Algorithms

The above experiments verify that the stability and optimization ability of the AFS-MMSBAS algorithm are better than the two basic algorithms with respect to high-dimensional

problems. This section compares the AFS-MMSBAS algorithm with other related algorithms, an Adaptive Dual-Strategy AFS algorithm based on the PSO algorithm (ADSAFS-PSO) [16], and an Adaptive AFSA utilizing Gene Exchange (AAFSA-GE) [17]. The unimodal function f_1, noise function f_4, multimodal functions f_5 and f_8, non-partitioned function f_{10}, and step function f_{11} are selected as test functions. The parameter settings are as follows: population 50, crowding factor 0.75, attempt number = 5, dimensions D = 10 and D = 200, and iterations 800. The other relevant parameters of the AFS-MMABAS algorithm are the same as those in Section 3.3.1, and the other parameters of the PSOEM-FSA and AAFSA-GE algorithms are the same as those in the references. Each algorithm runs independently 10 times. The experimental results are shown in Table 6, including the average, standard deviation, the maximum and minimum of the optimization results, and the average running time of the algorithm.

Table 6. Comparison between AFS-MMSBAS algorithm and other similar algorithms.

Func.	Algorithm	D = 10					D = 200				
		Mean	Standard	Min	Max	Time	Mean	Standard	Min	Max	Time
f_1	ADSAFS-PSO	8.88×10^{-17}	2.15×10^{-16}	**0**	6.66×10^{-16}	248.2	3.14×10^{3}	7.39×10^{3}	10.4	2.29×10^{4}	589.4
	AAFSA-GE	$\mathbf{1.81 \times 10^{-147}}$	$\mathbf{3.54 \times 10^{-147}}$	6.81×10^{-148}	$\mathbf{1.81 \times 10^{-146}}$	186.4	1.9×10^{3}	2.44×10^{3}	**8.5**	8.31×10^{3}	428.1
	AFS-MMSBAS	2.73×10^{-108}	8.26×10^{-108}	3.17×10^{-118}	2.62×10^{-107}	**65.97**	**25.5**	**7.64**	14.2	**35.7**	**268.7**
f_4	ADSAFS-PSO	9.25×10^{-5}	1.21×10^{-4}	6.27×10^{-6}	4.21×10^{-6}	300.3	42.0	98.8	3.05	3.2×10^{2}	499.4
	AAFSA-GE	$\mathbf{5.08 \times 10^{-63}}$	$\mathbf{1.02 \times 10^{-61}}$	$\mathbf{1.8 \times 10^{-64}}$	$\mathbf{6.52 \times 10^{-61}}$	242.3	5.14	2.47	**0.41**	10.4	314.2
	AFS-MMSBAS	0.0463	0.0381	0.00754	0.134	**39.39**	**3.66**	**0.603**	2.74	**4.51**	**88.24**
f_5	ADSAFS-PSO	3.55×10^{-16}	1.12×10^{-15}	0	3.55×10^{-15}	293.0	63.7	39.6	16.0	1.32×10^{2}	462.2
	AAFSA-GE	**0**	**0**	**0**	**0**	184.3	3.2×10^{-2}	3.74×10^{-2}	4.12×10^{-3}	9.04×10^{-2}	341.5
	AFS-MMSBAS	2.31×10^{-15}	1.88×10^{-15}	0	5.33×10^{-15}	**57.49**	$\mathbf{8.01 \times 10^{-4}}$	$\mathbf{8.37 \times 10^{-4}}$	$\mathbf{1.89 \times 10^{-4}}$	$\mathbf{3.04 \times 10^{-3}}$	**190.6**
f_8	ADSAFS-PSO	0.0845	0.123	$\mathbf{1.55 \times 10^{-15}}$	0.274	323.9	32.4	9.2	21.7	47.3	515.2
	AAFSA-GE	$\mathbf{2.48 \times 10^{-10}}$	$\mathbf{1.28 \times 10^{-6}}$	4.9×10^{-11}	$\mathbf{4.9 \times 10^{-9}}$	203.1	3. 25	2.4	1.65	6.7	396.2
	AFS-MMSBAS	0.00231	0.00297	6.7×10^{-7}	0.00599	**46.65**	$\mathbf{1.28 \times 10^{-2}}$	$\mathbf{1.34 \times 10^{-2}}$	$\mathbf{5.17 \times 10^{-3}}$	$\mathbf{4.29 \times 10^{-2}}$	**121.2**
f_{10}	ADSAFS-PSO	0.00246	2.46×10^{-19}	0.00246	0.00246	275.2	0.00246	2.46×10^{-19}	0.00246	0.00246	613.6
	AAFSA-GE	**0.00246**	$\mathbf{2.46 \times 10^{-26}}$	**0.00246**	**0.00246**	105.4	**0.00246**	$\mathbf{2.46 \times 10^{-24}}$	**0.00246**	**0.00246**	401.2
	AFS-MMSBAS	0.00246	4.56×10^{-9}	0.00246	0.00246	**52.92**	0.00246	6.98×10^{-11}	0.00246	0.00246	**166.2**
f_{11}	ADSAFS-PSO	0.2	0.632	0	2.0	280.4	4.66×10^{4}	1.47×10^{5}	20.0	4.66×10^{5}	593.2
	AAFSA-GE	0	0	0	0	112.3	4.41×10^{3}	5.3×10^{3}	**7.0**	6.27×10^{3}	387.4
	AFS-MMSBAS	**0**	**0**	**0**	**0**	**50.35**	$\mathbf{3.19 \times 10^{3}}$	$\mathbf{1.23 \times 10^{3}}$	1.78×10^{3}	$\mathbf{5.34 \times 10^{3}}$	**249.1**

As seen from the table, at low dimensions, the three algorithms can converge to the minimum value for functions f_5 and f_{11}, but the ADSAFS-PSO algorithm is more stable. Moreover, the average value and standard deviation index obtained by the ADSAFS-PSO algorithm are also better than the other two algorithms. In general, the AAFSA-GE algorithm has the best optimization accuracy and stability at low dimensions, while the AFS-MMSBAS and ADSAFS-PSO algorithms perform similarly. However, at high dimensions, the average, standard deviation, and maximum index obtained by the AFS-MMSBAS algorithm are superior to the other two algorithms. Therefore, the AFS-MMSBAS algorithm has more advantages than the other algorithms. However, although the AFS-MMSBAS algorithm has good optimization performance for most test functions, it has poor optimization performance for f11 functions. This shows that the algorithm is not suitable for the optimization of step functions.

In addition, the running time of the AFS-MMSBAS algorithm is the shortest.

4. Conclusions

In view of the advantages and disadvantages of the AFS algorithm and BAS algorithm, this paper proposes a hybrid algorithm, AFS-MMSBAS, which combines the artificial fish swarm and improved beetle antenna search algorithms. This hybrid algorithm involves the use of a mutation strategy to increase the probability of a beetle escaping the local extreme value and altering its course to a better direction. It also proposes a multi-step detection strategy to improve the convergence speed of the algorithm by adaptively changing the step size. Finally, a simple method is used to combine the AFS algorithm with the improved BAS algorithm. In order to test that the AFS-MMSBAS algorithm is superior to its two constituent algorithms, the AFS-MMSBAS algorithm is compared with the AFS, BAS, and AFS-BAS algorithms at different dimensions. In order to further verify the advantages of

this algorithm in dealing with high-dimensional problems, the AFS-MMSBAS algorithm is compared with similar algorithms, namely, AAFSA-GE and ADSAFS-PSO. The experimental results show that the AFS-MMSBAS algorithm can solve the problems of the poor optimization and instability of the AFS and BAS algorithms at high dimensions, and that it has a faster convergence speed during the later period. In a word, the AFS-MMSBAS algorithm performs well in high-dimensional problem processing.

The algorithm proposed in this paper performs well for high-dimensional problem optimization but performs poorly for some function problems, such as step functions. In addition, the algorithm has poor ability to solve low-dimensional problems and has great room for improvement. Therefore, in the future work, the authors plan to improve the algorithm, specifically with respect to its ability to solve high-dimensional partial function optimization and deal with low-dimensional problems. Further, it will be applied to engineering practices.

Author Contributions: J.N. and J.T. proposed the idea of the paper. J.T. and R.W. helped manage the annotation group and helped clean the raw annotations. J.T. conducted all experiments and wrote the manuscript. J.N., J.T. and R.W. revised and improved the text. J.N. and J.T. are the people in charge of this project. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Handan Science and Technology Research and Development Program (19422091008-35) and the Hebei Science and Technology Program (21350101D).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Rahkar Farshi, T. Battle Royale Optimization Algorithm. *Neural Comput. Appl.* **2021**, *33*, 1139–1157. [CrossRef]
2. Ghafil, H.N.; Jármai, K. Dynamic Differential Annealed Optimization: New Metaheuristic Optimization Algorithm for Engineering Applications. *Appl. Soft Comput.* **2020**, *93*, 106392. [CrossRef]
3. Nadimi-Shahraki, M.H.; Fatahi, A.; Zamani, H.; Mirjalili, S.; Abualigah, L. An Improved Moth-Flame Optimization Algorithm with Adaptation Mechanism to Solve Numerical and Mechanical Engineering Problems. *Entropy* **2021**, *23*, 1637. [CrossRef] [PubMed]
4. Yuan, Z.; Wang, W.; Wang, H.; Razmjooy, N. A New Technique for Optimal Estimation of the Circuit-Based PEMFCs Using Developed Sunflower Optimization Algorithm. *Energy Rep.* **2020**, *6*, 662–671. [CrossRef]
5. Abdel-Basset, M.; Chang, V.; Mohamed, R. A Novel Equilibrium Optimization Algorithm for Multi-Thresholding Image Segmentation Problems. *Neural Comput. Appl.* **2021**, *33*, 10685–10718. [CrossRef]
6. Gao, Y.; Yang, Q.; Wang, X.; Li, J.; Song, Y. Overview of new swarm intelligence optimization algorithms. *J. Zhengzhou Univ. (Eng. Ed.)* **2022**, *43*, 21–30.
7. Yang, X.-S.; Deb, S. Cuckoo Search via Lévy Flights. In Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, India, 9–11 December 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 210–214. [CrossRef]
8. Wang, D.; Tan, D.; Liu, L. Particle Swarm Optimization Algorithm: An Overview. *Soft Comput.* **2018**, *22*, 387–408. [CrossRef]
9. Xia, X.; Zhou, Y. Research progress of ant colony optimization algorithm. *J. Intell. Syst.* **2016**, *11*, 10.
10. Cheng, M.; Ni, Z.; Zhu, X. Overview of Firefly Optimization Algorithm. *Theory computer science.* **2015**, *42*, 19–24.
11. Bastos Filho, C.J.A.; de Lima Neto, F.B.; Lins, A.J.C.C.; Nascimento, A.I.S.; Lima, M.P. A Novel Search Algorithm Based on Fish School Behavior. In Proceedings of the 2008 IEEE International Conference on Systems, Man and Cybernetics, Singapore, 12–15 October 2008; pp. 2646–2651.
12. Zhang, L.; Fu, M.; Fei, T.; Li, H. The Artificial Fish Swarm Algorithm Improved by Fireworks Algorithm. *Autom. Control. Comput. Sci.* **2022**, *56*, 11–323.
13. Zhang, C.; Zhang, F.; Li, F.; Wu, H. Improved Artificial Fish Swarm Algorithm. In Proceedings of the 2014 9th IEEE Conference on Industrial Electronics and Applications, Hangzhou, China, 9–11 June 2014; pp. 748–753.
14. Liu, D.; Li, L. A novel improved artificial fish swarm algorithm. *Comput. Sci.* **2017**, *44*, 281–287.
15. Li, J.; Liang, X.M. Improved artificial fish swarm algorithm for elite acceleration. *Comput. Appl. Res.* **2018**, *35*, 1960–1964+1981.
16. Liu, Z.; Shu, Z.; Xu, Y.; Yang, S.; Shen, W. Artificial fish swarm algorithm based on PSO adaptive dual strategy. *Comput. Mod.* **2022**, *5*, 46–53.
17. Li, Z.; Zhou, K.; Ou, Y.; Ding, L. Adaptive artificial fish swarm algorithm based on gene exchange. *Comput. Appl.* **2022**, *42*, 701–707.

18. Jiang, X.; Li, S. BAS: Beetle Antennae Search Algorithm for Optimization Problems. *Int. J. Robot. Control* **2017**, *1*, 1. [CrossRef]
19. Zivkovic, M.; Bacanin, N.; Venkatachalam, K.; Nayyar, A.; Djordjevic, A.; Strumberger, I.; Al-Turjman, F. COVID-19 Cases Prediction by Using Hybrid Machine Learning and Beetle Antennae Search Approach. *Sustain. Cities Soc.* **2021**, *66*, 102669. [CrossRef] [PubMed]
20. Huang, J.; Duan, T.; Zhang, Y.; Liu, J.; Zhang, J.; Lei, Y. Predicting the Permeability of Pervious Concrete Based on the Beetle Antennae Search Algorithm and Random Forest Model. *Adv. Civ. Eng.* **2020**, *2020*, 8863181. [CrossRef]
21. Xiang, Q.; Zhu, P. Image Denoising Using a Deep Auto-Encoder Approach Based on Beetle Antennae Search Algorithm. In *Computer and Communication Engineering*; Neri, F., Du, K.-L., Varadarajan, V.K., Angel-Antonio, S.-B., Jiang, Z., Eds.; Springer International Publishing: Cham, Switzerland, 2022; Volume 1630, pp. 75–84.
22. Wang, J.; Chen, H. BSAS: Beetle Swarm Antennae Search Algorithm for Optimization Problems. *arXiv* **2018**, arXiv:1807.10470.
23. Zhao, Y.; Qian, Q.; Zhou, T. Fuyun sends A hybrid algorithm of longicorn beetle whisker search and genetic algorithm. *Minicomput. Sys.* **2020**, *41*, 8.
24. Khan, A.H.; Cao, X.; Li, S.; Katsikis, V.N.; Liao, L. BAS-ADAM: An ADAM Based Approach to Improve the Performance of Beetle Antennae Search Optimizer. *IEEE/CAA J. Autom. Sin.* **2020**, *7*, 461–471. [CrossRef]