IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 30, 2020, accepted June 12, 2020, date of publication June 24, 2020, date of current version July 14, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3004430

Optimal Performance and Application for Firework Algorithm Using a Novel Chaotic Approach

WEILIN LUO[1], HONGBIN JIN[1], HAO LI[1], XI FANG[2], AND RONGHUA ZHOU[1]
[1]People's Liberation Army Air Force Early Warning Academy, Wuhan 430019, China
[2]College of Science, Wuhan University of Technology, Wuhan 430070, China

Corresponding author: Hao Li (afeu_li@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61502522, in part by the Equipment Pre-Research Field Fund under Grant JZX7Y20190253036101, in part by the Equipment Pre-Research Ministry of Education Joint Fund under Grant 6141A02033703, and in part by the Hubei Provincial Natural Science Foundation under Grant 2019CFC897.

ABSTRACT Aimed at the bottleneck of Firework Algorithm's performance and the problem that it is easy to fall into the local extremum, this paper introduces a chaotic map approach and analyzes its statistical characteristics. New hybrid chaotic systems and a chaotic perturbation operator are designed. Based on different chaotic map approaches, a new Chaotic Firework Algorithm (CFWA) is proposed and simulated by typical test functions. The results show that performance of the Chaotic Firework Algorithm is generally better than the original algorithms, and the chaotic hybrid approaches are superior to single chaotic approaches. In order to further verify the application of the proposed algorithm in practical engineering problems, CFWA is applied to Blind Source Separation (BSS) of radar signals. As a result, it does well in sorting observed signals, and has faster convergence and better sorting performance than the traditional algorithms. In this way, the CFWA extends its engineering application.

INDEX TERMS Firework algorithm, chaotic map, chaotic perturbation, hybrid chaotic systems, blind source separation.

I. INTRODUCTION

Metaheuristic algorithms usually solve complex engineering optimization problems in the scientific field. Due to their excellent performance, they have been the focus of research in recent years. There are many kinds of metaheuristic algorithms which inspired mechanisms, they can be roughly divided into two categories: one is to imitate biological processes, such as Genetic Algorithm (GA) [1], Particle Swarm Optimization Algorithm (PSO) [2], Ant Colony Optimization Algorithm (ACO) [3], Fish Swarm Search Algorithm (FSS) [4], Artificial Bee Colony Algorithm (ABC) [5], Firefly Algorithm (FA) [6], Cuckoo Algorithm (CS) [7], Bacterial Foraging Optimization (BFO) [8], Grey Wolf Optimizer (GWO) [9], Fruit Fly Optimization Algorithm (FOA) [10] etc.; and the other is based on the principles of physics, such as Simulated Annealing (SA) [11], Gravitational Search Algorithm (GSA) [12], Biogeography-Based Optimization (BBO) [13], Water Wave Optimization, (WWO) [14], Vortex Search (VS) [15], Harmony search, (HS) [16], etc. In particular, a metaheuristic algorithm based on swarm intelligence, has an incomparable advantage over traditional optimization algorithms in solving optimization problems in engineering.

The Firework Algorithm (FWA) is a new swarm intelligence algorithm proposed by Tan and others [17] in 2010, inspired by the natural phenomenon of firework explosions in the night sky. So the FWA is also a metaheuristic algorithm based on physical principles. Compared with the traditional Genetic Algorithm, Particle Swarm Algorithm, and Differential Evolution Algorithm, FWA shows a good performance in optimization. In addition, FWA has a simple structure, few control parameters, and is easy to implement. It has local search and global search self-adjusting mechanisms. More and more researchers have begun to study the improvement and application of FWA in engineering problems from

The associate editor coordinating the review of this manuscript and approving it for publication was Dongxiao Yu.

different perspectives, such as sensor deployment [18], optimization calculation of distribution grid scheme [19], robot path planning [20], multi-objective optimization problem [21], etc.

Although FWA has been widely used in engineering field, there are still some shortcomings, such as low accuracy of an optimization result, and slow convergence speed, and it's easy to fall into a local optimum. In recent years, scholars have proposed many improved algorithms. Zheng *et al.* [22] analyzed the explosion operator, mutation operator, selection strategy, and map rule mechanism of the firework algorithm in detail, and proposed the Enhanced Firework Algorithm (EFWA). Next, Zheng and Tan [23] dynamically calculated the firework explosion radius based on the dynamic change of the spark fitness value, and proposed a dynamic search Firework Algorithm (dynFWA). Li *et al.* [24] determined the firework explosion radius based on the distance between the optimal individual and a specific individual, and proposed an Adaptive Firework Algorithm (AFWA). Li *et al.* [25] proposed a Reverse Firework Algorithm by combining the backward learning strategy with the FWA to overcome the slow convergence speed and performance bottlenecks in the traditional FWA. Reddy *et al.* [26] proposed the Binary Fireworks Algorithm (BFWA) and applied BFWA to the PBUC (profit based unit commitment) optimization problem. Li *et al.* [27] introduced a novel Guided Spark (GS) in the FWA to enhance the information utilization of the algorithm. Experiments show that GS's ability to explore and develop the FWA has been greatly enhanced. Shen *et al.* [28] adopted a new type of map rule that takes into account the location characteristics and proposed an Enhanced Multimodal Firework Algorithm based on the loser elimination system. In general, the above algorithm improves the search performance of the FWA, but still has the disadvantages of slow convergence and easily falling into a local optimum.

Chaos optimization is a relatively new optimization method, which has unique characteristics such as the inherent randomness and initial value sensitivity. In this paper, two hybrid chaotic systems are designed by analyzing one-dimensional chaotic maps, and an improved Chaotic Firework Algorithm (CFWA) is proposed. Firstly, chaotic maps are used to replace the traditional random initialization of firework positions, making the initial solution distribution more uniform. Then a new adaptive chaotic perturbation operator is designed to accelerate the convergence speed and avoid the algorithm from falling into a local optimum and evolutionary stagnation. Experiments show that the algorithm performance has been greatly improved.

The structure of this paper is as follows: Section II introduces the flow of the basic Firework Algorithm. Section III analyzes the statistical characteristics of six types of one-dimensional chaotic maps and designs two hybrid chaotic systems. Section IV introduces a standard test function with different characteristics and analyzes the test performance of each function. Section V proposes the CFWA and constructs the algorithm flow. Section VI verifies the effectiveness of the CFWA through experiments and compares it with several classic improvement methods. Section VII applies CFWA to the problem of Blind Source Separation of radar signals. The simulation results show advantages of this algorithm in solving practical engineering problems. Section VIII draws a conclusion and summarizes the whole paper.

II. BASIC FIREWORK ALGORITHM

FWA comes from the simulation of the firework explosion process. The execution process of the algorithm is shown in Figure 1. The basic principle of the FWA: If the fitness corresponding to a firework is smaller, the number of sparks generated by the fireworks explosion is smaller and the explosion amplitude is smaller; on the contrary, if the corresponding fitness of fireworks is larger, the number of sparks generated by the firework explosion will be smaller and the explosion amplitude will be larger. During the iterative process of the FWA, the explosion operator, mutation operator, map rule, and selection strategy are used in turn until the termination condition is reached. The specific implementation steps are as follows.

Step1: Expression and initialization of solutions. $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ij}, \ldots, x_{iD})$ is the current position of the i-th firework, which is generated by equation (1).

$$\boldsymbol{x}_i = (x_{max} - x_{min}) \cdot rand(1, D) + x_{min} \tag{1}$$

In Equation (1) ($1 \le i \le N$, $1 \le j \le D$), D is the search space dimension, N is the population size, x_{ij} is the j-th component of the i-th firework, and the value range of the component is $[x_{min}, x_{max}]$. $rand(1, D)$ generates a D-dimensional vector.

Step2: Explosion operator. In FWA, the equation for the number of explosion sparks generated by the i-th firework $\boldsymbol{x}_i$ is as Equation (2):

$$S_i = m \frac{Y_{worst} - f(\boldsymbol{x}_i) + \xi}{\sum_{i=1}^{N} (Y_{worst} - f(\boldsymbol{x}_i)) + \xi} \tag{2}$$

In Equation (2), m is a constant, and Y_{worst} is the worst fitness value in the current population. $f(\boldsymbol{x}_i)$ is the fitness value of $\boldsymbol{x}_i$. ξ is a very small normal number that the computer can represent, and is used to prevent the division by zero error. At the same time, in order to prevent too much or too little S_i from being generated, S_i is corrected according to the equation (3):

$$S_i = \begin{cases} round(am), & if\ S_i < am \\ round(bm), & if\ S_i > bm, a < b < 1 \\ round(S_i), & otherwise \end{cases} \tag{3}$$

In Equation (3), $round(\bullet)$ is a rounding function, a and b are given constants. The equation for calculating the explosion radius of $\boldsymbol{x}_i$ is as follows:

$$A_i = \hat{A} \frac{f(\boldsymbol{x}_i) - Y_{best} + \xi}{\sum_{i=1}^{N} (f(\boldsymbol{x}_i) - Y_{best}) + \xi} \tag{4}$$

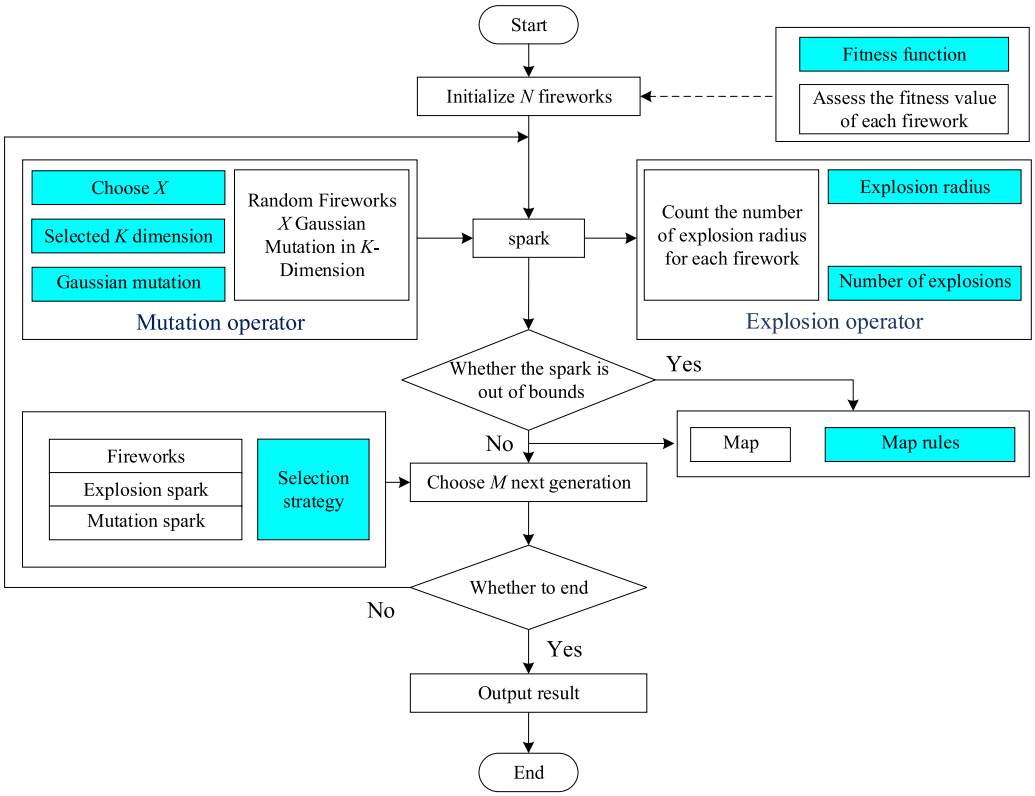


FIGURE 1. Execution process of FWA.

In Equation (4), $\hat{A}$ is the preset maximum explosion radius. Assuming that the objective function to be optimized is a D-dimensional function, the z-dimensional coordinates in the D-dimensional are randomly selected for update. The k-dimensional coordinate update equation of the j-th spark generated by $\boldsymbol{x}_i$ is as follows:

$$x_i^k = x_i^k + A_i rand(-1, 1) \tag{5}$$

Step3: Mutation operator. The mutation operator is used to generate Gaussian sparks in order to increase population diversity. Generate M Gaussian sparks according to preset parameters, and the k-dimensional coordinate update equation of the j-th spark is as follows:

$$x_j^k = x_j^k Gaussian(1, 1) \tag{6}$$

In Equation (6), $j = 1, 2, \ldots, M$ and $k = 1, 2, \ldots, z$.

Step4: Map rules. Map rules are used to correct coordinates that exceed the value range during coordinate update. The map rules using modulo operation are as follows:

$$x_i^k = x_{min}^k + \left| x_i^k \right| mod(x_{max}^k - x_{min}^k) \tag{7}$$

In Equation (7), x_{max}^k and x_{min}^k are the upper and lower bounds of the boundary of the k-th value range of the i-th spark, respectively.

Step5: Selection strategy. In FWA, the elite retention strategy is adopted, and the spark with the best fitness is automatically retained to the next generation. The remaining individuals choose to adopt the roulette method, and the probability of being selected is represented by $p(\boldsymbol{x}_i)$.

$$p\left(\boldsymbol{x}_i\right) = \frac{R(\boldsymbol{x}_i)}{\sum_{j=1}^{K} R(\boldsymbol{x}_j)} \tag{8}$$

In Equation (8), $R(\boldsymbol{x}_i)$ is the sum of the distances between the individual $\boldsymbol{x}_i$ and other individuals. Calculate using the Equation (9).

$$R\left(\boldsymbol{x}_i\right) = \sum_{j=1}^{K} d\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \sum_{j=1}^{K} \left\| \boldsymbol{x}_i, \boldsymbol{x}_j \right\| \tag{9}$$

$d\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$ is the Euclidean distance between individuals $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. K is the total number of sparks generated by the explosion operator and mutation operator.

III. CHAOS OPTIMIZATION ANALYSIS

A. CHAOS INTRODUCTION

Chaos originated from mathematics and physics. Chaos was first proposed by American biologist May [29], which caused a research climax in the 1970s. Chaos is a peculiar evolutionary behavior of nonlinear dynamical systems, which refers to an unpredictable, random-like motion that occurs in deterministic dynamical systems (with certain equations) and is sensitive to initial values [30]. In nature, chaos is

everywhere, such as flags swaying in the wind, rising smoke, flowing streams, clouds moving in the sky, weather changes, the path of lightning, star clusters in the universe and even economic fluctuations, population growth etc.

Order in chaos is different from random processes. There is an essential difference between chaos and random processes. Chaotic motions are caused by deterministic physical laws (deterministic equations), and are caused by the external realization of intrinsic characteristics; while random processes are caused by random factors such as external noise. Chaos is non-periodic but has unique characteristics such as intrinsic randomness and initial value sensitivity, which are not found in random motion.

1) Non-periodic. For some parameters, chaos will generate non-periodic dynamic processes under almost all initial conditions.
2) Initial value sensitivity. Even if the two initial values differ by a factor of 10,000, over time and evolution, the results obtained by the same chaotic system are very different, such as the so-called butterfly effect. Therefore, the evolution of chaos also has long-term unpredictability.
3) Ergodicity. Different from the balance, periodic and quasi-periodic movements of deterministic movements, chaos is a kind of movement with a limited range and never repetitive and complex movement trajectory, which is constantly evolving and never stops.
4) Intrinsic stochasticity. Chaos is an unstable trajectory generated by a deterministic equation, and the system's motion shows an inherently random and complex behavior. Chaos is neither pure disorder nor pure order, but the unity of the two. It is also the unity of certainty and randomness. It has inherent regularity and universality.

It is precisely because chaos has the properties of aperiodicity, initial value sensitivity, ergodicity, and internal randomness that it has become one of the hot spots in current nonlinear scientific research. It is also widely used in many fields such as optimization search, neural network, image compression, nonlinear time series data prediction, pattern recognition, fault diagnosis, secure communication, etc. [31]–[34].

The basic idea of chaos optimization is to map the variables from the chaotic space to the solution space of the optimization problem, and then use the chaotic variables to search and optimize using the characteristics of spatial ergodicity and external randomness. The general steps of the CO method are as follows:

Step1: Use chaotic map to generate a set of chaotic variables with the same number of variables as the problem to be optimized.

Step2: The chaotic variable $cx_d \in (0, 1)$ is mapped to (L_d, U_d) according to Equation (10), that is, the traversal range of the chaotic motion is expanded from $(0, 1)$ to (L_d, U_d), and then the new chaotic variable x_d is used for optimization.

$$x_d = L_d + cx_d\,(U_d - L_d) \tag{10}$$

For most heuristic optimization algorithms, the generation of random variables is usually based on a certain standard probability distribution, such as uniform distribution and Gaussian distribution. Chaos is not only random-like but also has better spatial ergodicity and non-repetition. Therefore, compared with random search based on a certain probability distribution, the population of the algorithm after the chaos is relatively more diverse, and the possibility of jumping out of local extreme points is relatively larger, which in turn allows the algorithm to search at a relatively faster speed. At the same time, there are different types of chaos. Each chaotic system has some different statistical characteristics, which has a greater impact on the initialization of chaos and the generation of new chaotic solutions. Therefore, it is necessary to analyze the statistical characteristics of typical chaotic maps.

B. ANALYSIS OF A TYPICAL ONE-DIMENSIONAL CHAOTIC MAP

To introduce chaos optimization, some one-dimensional, irreversible maps are used here to generate chaotic sequences. As shown in Table 1, six typical one-dimensional chaotic maps are introduced (k in the Table 1 is the number of iterations and x_0 is the initial value).

To visually show the characteristics of the above-mentioned one-dimensional chaotic maps. Figures 2 to 7 respectively show the chaotic numerical graphs and the probability distribution characteristics in the interval (0,1) of 6 chaotic maps such as Logistic map at 100 iterations. Inspired by the literature [35], the probability distribution is estimated by the following methods: iteratively uses chaotic maps to iterate 100,000 times to generate 100,000 chaotic numerical points, divides the interval (0,1) into 100 intervals, and counts the probability of the 100,000 points falling into the 100 equal partitions, as shown on the right of each figure. The initial value of each chaotic map is randomly selected as shown in the title of each figure.

Figure 2 shows the numerical graphs and their probability distributions of the Logistic map when are 0.8 and 0.8001. From Figure 2 (a) and (c), the numerical graphs are chaotic, random, and have no overlapping overlap, reflecting the external randomness and traversal of chaos; the initial values of Figure 1 (a) and (c) are 0.8 and 0.8001 respectively, the difference between them is only one ten thousandth, but comparing the two graphs, it can be seen that after 10 iterations, the Logistic system shows a large difference, and there is no similarity at the end, which fully reflects the initial value sensitivity of chaos; although the differences in Figure 2 (a) and (c) are large, however, comparing Figure 2 (b) and (d), it can be seen that the probability distributions in the two cases show striking similarities, fully reflecting the inherent regularity of chaos.

TABLE 1. Typical one-dimensional chaotic map.

No	Name	Expression	Description
1	Logistic map	$x_{k+1} = a x_k (1 - x_k)$	a generally takes 4,when $x_0 \in (0,1)$ and $x_0 \notin (0, 0.25, 0.5, 0.75, 1)$, $x_k \in (0,1)$
2	Circle map	$x_{k+1} = x_k + b - (\frac{a}{2\pi}) \sin(2\pi x_k) \bmod (1)$	When a=0.5,b=0.2, $x_k \in (0,1)$
3	Sinusoidal iterator	$x_{k+1} = a x_k^2 \sin(\pi x_k)$	$x_k \in (0,1)$,when a=2.3 and $x_0 = 0.7$, x_{k+1} Can be abbreviated as $x_{k+1} = \sin(\pi x_k)$
4	Gauss map	$x_{k+1} = \begin{cases} 0, & x_k = 0 \\ 1/x_k \bmod (1), & x_k \in (0,1) \end{cases}$	$1/x_k \bmod (1) = 1/x_k - \lfloor 1/x_k \rfloor$, $\lfloor \square \rfloor$ means round down, $x_k \in (0,1)$
5	Bernouilli	$x_{k+1} = \begin{cases} \frac{x_k}{1-\lambda} & 0 < x_k \le 1-\lambda \\ \frac{x_k - (1-\lambda)}{\lambda} & 1-\lambda < x_k < 1 \end{cases}$	Special form is $x_{k+1} = 2 x_k \bmod 1$, exist unstable cycle points, such as 0.25, 0.5, 0.75 will iterate to the fixed 0.
`6	Tent map	$x_{k+1} = \begin{cases} x_k / 0.7, & x_k < 0.7 \\ 10/3 x_k (1 - x_k), & otherwise \end{cases}$	$x_k \in (0,1)$

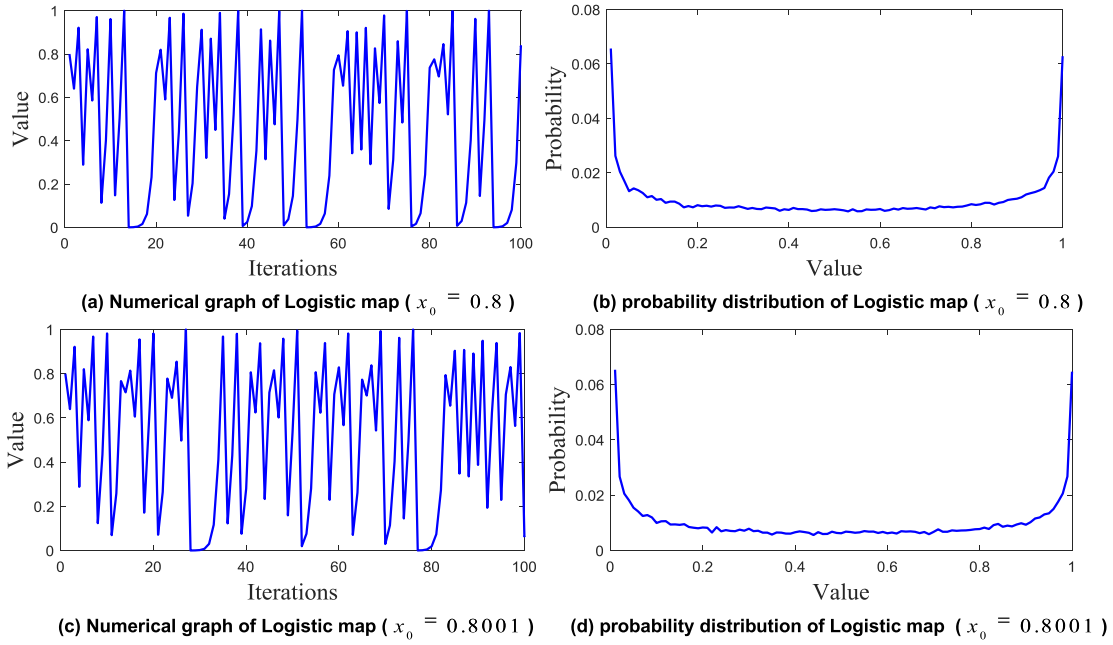
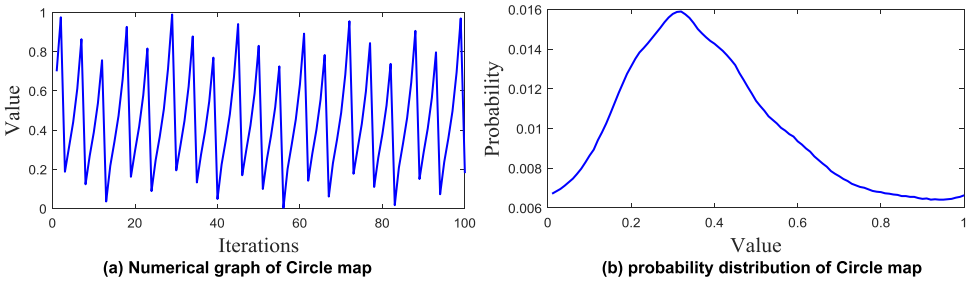


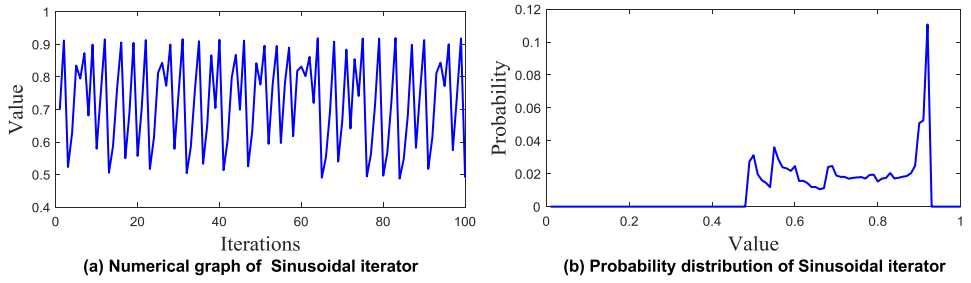
FIGURE 2. Numerical graph and probability distribution of logistic.

The other five chaotic maps such as the Tent map also show similar chaotic properties as Logistic map, which will not be detailed here. Here we only briefly analyze the probability distribution of each chaotic map. As shown in Figures 2 (b) and (d), the distribution characteristics of the chaotic sequence of the Logistic map are relatively even in the middle, but the probability of the values at both ends (0,0.1) and (0.9,1) is particularly high, which is the middle distribution About 10 times. Therefore, when chaotic optimization is performed using the Logistic map, a large number of searches are performed at both ends of the variable value interval. If the global optimal solution of the optimization problem is not distributed at both ends but in the middle, a large number of invalid searches will occur. It is very bad for global optimization. What is interesting is that, as shown in Figure 3 (b), the probability distribution of the Circle map just shows a trend of high in the middle and low in both ends. Therefore, inspired by the above two chaotic



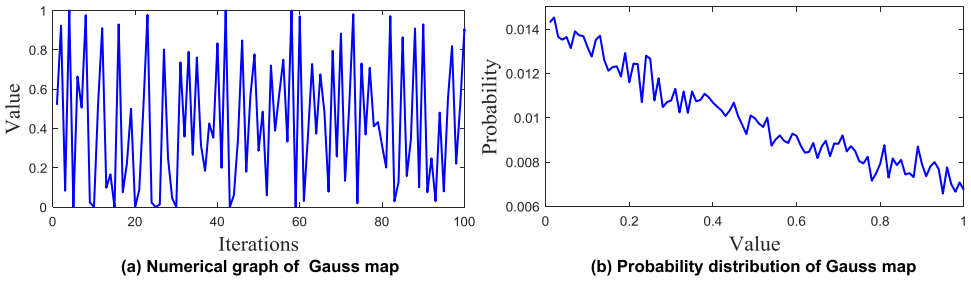
(a) Numerical graph of Circle map (b) probability distribution of Circle map

FIGURE 3. Numerical graph and probability distribution of Circle map ($x_0 = 0.7$).



(a) Numerical graph of Sinusoidal iterator (b) Probability distribution of Sinusoidal iterator

FIGURE 4. Numerical graph and probability distribution of Sinusoidal iterator($x_0 = 0.7$).



(a) Numerical graph of Gauss map (b) Probability distribution of Gauss map

FIGURE 5. Numerical graph and probability distribution of Gauss map($x_0 = 0.52$).

maps, the author attempts to combine Logistic map and Circle map by some method to obtain an iterative numerical sequence that is never repeated, has inherent randomness, and is relatively uniformly distributed. This can facilitate the initialization of intelligent optimization algorithms, and non-repetitive heuristic search.

Similarly, as shown in Figures 4 (b) and 5 (b), the sinusoidal iterator and Gauss map also show roughly complementary probability distribution. The Sinusoidal iterator shows a low front-end and high-back end distribution, while the Gauss map is just the opposite. The probability distribution shows a decreasing trend. Therefore, it is also possible to combine the Sinusoidal iterator and Gauss map to design a chaotic hybrid optimization approach to improve FWA to increase the algorithm's optimization performance.

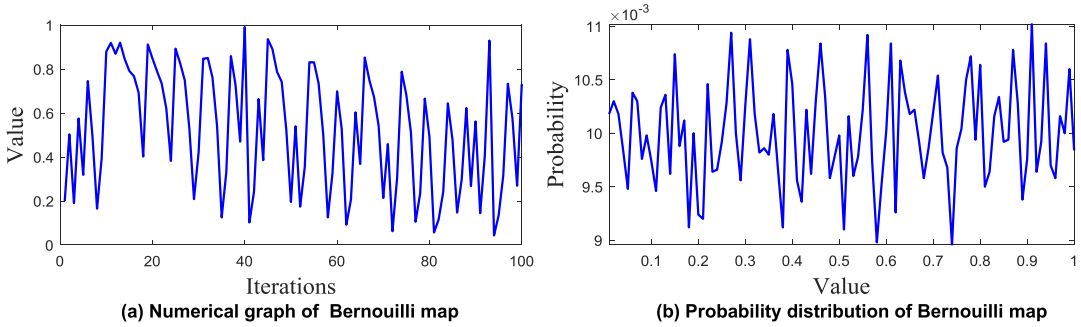
Bernouilli map has unstable periodic points, such as 0.25, 0.5, 0.75 will iterate to fixed point 0. Therefore, the method in [36] is introduced into the Bernouilli map to form an improved Bernouilli map: if $x_k = 0, 0.25, 0.5, 0.75$, Bernouilli map re-enters the chaos state under perturbation. The improved Bernouilli map is shown in Equation (11).

$$x_k = 2\left(x_k + 0.1 \times rand\left(0, 1\right)\right) \bmod 1 \tag{11}$$

Figures 6 and 7 show the numerical graphs and probability distribution of Bernouilli map and Tent map. It can be seen intuitively that the chaotic sequences of the two are evenly distributed in the (0,1) interval. This distribution can avoid excessive search in some local areas (such as a large number of Logistic map are concentrated at both ends of the interval), thereby reducing the incompatibility between the distribution characteristics of the chaotic sequence and the global optimal solution position of the optimization problem causes adverse effects on the optimization algorithm.

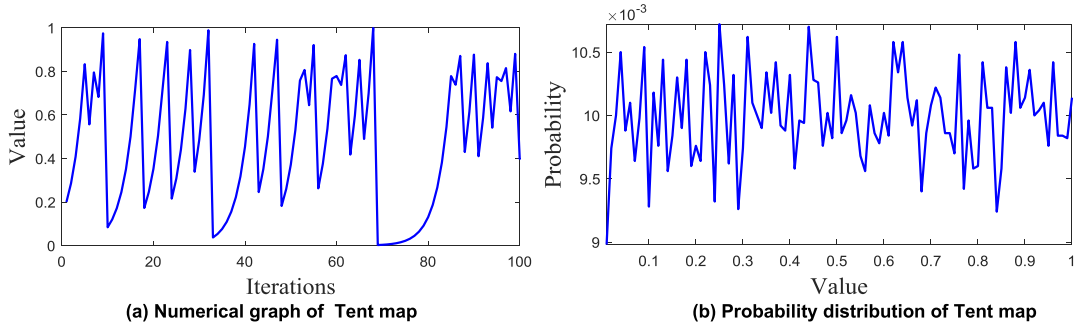
C. HYBRID CHAOTIC SYSTEM

Based on the above-mentioned analysis of the probability distribution/statistical characteristics of the Logistic map and the Circle map, the Sinusoidal iterator and the Gauss map, two pairs of "complementary" statistical characteristics are obtained. Here we try to combine them in pairs to obtain a



(a) Numerical graph of Bernouilli map (b) Probability distribution of Bernouilli map

FIGURE 6. Numerical graph and probability distribution of Bernouilli map ($x_0 = 0.2$).



(a) Numerical graph of Tent map (b) Probability distribution of Tent map

FIGURE 7. Numerical graph and probability distribution of Tent map($x_0 = 0.2$).

''perfect'' sequence that is never repeated and has a relatively uniform distribution of statistical characteristics. Traverse the entire solution space. A hybrid chaotic system is proposed here, which combines the variables in two chaotic maps with ''complementary'' statistical characteristics. The pseudo code of the hybrid chaotic map of the Logistic map and the Circle map (abbreviated as L-C map) is shown in figure 8. The pseudo code of the hybrid chaotic map of Sinusoidal iterator and Gauss map (abbreviated as S-G map) is shown in figure 9.

IV. CHARACTERISTICS OF FUNCTION OPTIMIZATION PROBLEMS AND STANDARD TEST FUNCTIONS

A. CHARACTERISTICS OF FUNCTION OPTIMIZATION PROBLEMS

A large number of optimization problems can be transformed into functional optimization problems through mathematical modeling, such as PID parameter optimization, mechanical parameter optimization, aerodynamic parameter identification, aircraft trajectory optimization, etc. [37], [38]. However, optimization problems abstracted by these realistic problems have different mathematical characteristics such as high-dimensional, multimodal, nonlinear and non-separable.

1) High-dimensional. The solution space of high-dimensional complex functions is very large, which makes the difficulty of algorithm optimization increase sharply. Many algorithms that work well for low-dimensional functions are available may perform poorly for high-dimensional complex functions [39].
2) Multimodal. Multimodal functions are relative to Unimodal functions. Unimodal function means that the function has only global optimums and no local optimums in the domain; and the multimodal function means that it has multiple local extremums, sometimes even infinite (such as Ackley function, Griewank function, etc.). Therefore, the multimodal function is more complicated than the unimodal function. It is difficult for ordinary optimization algorithms to find the global optimal value, and it is easy to fall into local extreme values or oscillate between local extreme values [40].
3) Nonlinear. The nonlinear function solution space is extremely complicated, and there is a lot of misleading gradient information, which easily causes the algorithm to sink into the local optimum, making optimization difficult.
4) Non-separable. If a function with N variables can be represented by the sum of N univariate functions, the function is separable, otherwise it is a non-separable function. Because relationships between non-separable function variables are complex, solving such function optimization problems is relatively more difficult.

Therefore, this paper introduces four types of complex functions: unimodal separable, unimodal non-separable, multimodal separable, and multimodal non-separable.

B. STANDARD TEST FUNCTIONS

To test the performance of various swarm intelligence algorithms, several groups of Benchmark functions with different

Code : Hybrid chaotic map combining Circle and Logistic generates chaotic variable sequences
Input : D, N ;
Output: Cx
for $i = 1, 2, \cdots, N$ do
if $0 < \delta < 0.1$ or $0.8 < \delta < 1$ // δ is a uniformly distributed random number on (0,1)
Use Logistic map to generate a chaotic variable sequence Cx_i of length D;
else
Use Circle map to generate a chaotic variable sequence Cx_i of length D;
endif
endfor

FIGURE 8. The pseudo-code of chaotic variable sequence generated by the hybrid chaotic map of Circle and Logistic.

Code : Hybrid chaotic map combining Gauss and Sinusoidal generates chaotic variable sequences
Input : D, N ;
Output: Cx
for $i = 1, 2, \cdots, N$ do
if $0 < \delta < 0.5$ // δ is a uniformly distributed random number on (0,1)
Using Gauss map to generate chaotic variable sequence Cx_i ;
else
Using Sinusoidal iterator to generate chaotic variable sequence Cx_i ;
endif
endfor

FIGURE 9. The pseudo-code of chaotic variable sequence generated by the hybrid chaotic map of Sinusoidal and Gauss.

characteristics are cited for testing. The author has selected 15 test functions with different mathematical characteristics and scales as shown in Table 2 to facilitate the comparison of subsequent algorithms.

Specifically, the feature 'U' in the Table 2 indicates that the function is a Unimodal function, 'M' indicates Multimodal, 'S' indicates Separable, and 'N' indicates Non-Separable. As shown in Table 2, the 15 standard test functions selected include various complex mathematical features, including 4 unimodal inseparable functions, 2 unimodal separable functions, and 3 multimodal separable functions. Multimodal inseparable functions are the most, with a total of 6. Besides, the set part in the feasible solution space is the domain of the independent variable, and its superscript is the dimension. For example, for the Ackley function, $D = 200$, and $[-32, 32]^D$ is the optimal space. The bold figure in the global extremum indicates that the Bridge function finds the global maximum, and all other complex functions solve the global minimum.

The functions in Table 2 can also be divided into two categories based on whether the dimensions can be freely set. The five functions F5, F6, F9, F14, and F15 are free-dimensional functions, their function dimensions can be set freely. The higher the dimension, the more complicated the solution space, and the more difficult it is to find the optimal function. The other 10 functions are all fixed-dimensional functions and the solution space dimensions of the functions have been set.

For an intuitive understanding and convenient analysis of function properties, Figure 10 gives three-dimensional plots of 14 test functions except for Colville (Colville functions are 4-dimensional fixed-dimensional functions, which cannot be represented by a single three-dimensional plot). It can be seen from Figure 10 that each function has its characteristics. For example, the function F1 presents a cone space and has a large gradient. The optimization algorithm can easily miss the global best. Therefore, it is very difficult for the algorithm to achieve high optimization accuracy; F3 is a banana-like unimodal function, whose optimal value is hidden in a long and narrow parabolic channel. It is also difficult to find the optimal solution; the number of local optimal points of F5, F6 and F9 increases with the increase of the dimension, which makes the difficulty of algorithm optimization sharply increased; F11 is a derivable function with a total of 2 global minimum points $(-0.0898, 0.7126)$, $(0.0898, -0.7126)$ and 6 local extremum points, each extremum point is independent of each other. This makes it difficult for the algorithm to obtain the heuristic information of the function, making it difficult to find the optimal solution for the algorithm; F14 is an inseparable complex multimodal function. Its domain $[-600, 600]^D$ can build a very large search space for optimization, especially when the value of the dimension D is large. Besides, a large number of local extreme values and high obstacles surround the global optimal value, which is very deceptive, making the optimization algorithm easily fall into the local extreme value and difficult to jump out; F15 is a complex multimodal inseparable function formed by the superposition of exponential function and cosine function. Countless local extreme points are distributed in the solution space of F15, which can cause a lot of misleading information to the optimization algorithm. In addition, its global

TABLE 2. Standard test functions.

No.	Function	Expression	Feature	Solution space	Global extremum
1	Easom	$f(\boldsymbol{X}) = -\cos(x_1)\cos(x_2) \times \exp\ -(x_1-\pi)^2-(x_2-\pi)^2$	UN	$[-10,10]^2$	-1
2	Matyas	$f(\boldsymbol{X}) = 0.26\ x_1^2+x_2^2\ -0.48x_1x_2$	UN	$[-10,10]^2$	0
3	Rosenbrock	$f(\boldsymbol{X}) = 100(x_2-x_1^2)^2+(1-x_1)^2$	UN	$[-2.048, 2.048]^2$	0
4	Colville	$f(\boldsymbol{X}) = 100\ x_1^2-x_2\ ^2+\ x_1-1\ ^2+\ x_3-1\ ^2+90\ x_3^2-x_4\ ^2$ $+10.1\ x_2-1\ ^2+\ x_4-1\ ^2+19.8\ x_2-1\ \ x_4-1$	UN	$[-10,10]^4$	0
5	Sumsquares	$f(\boldsymbol{X}) = \sum_{i=1}^{D} ix_i^2$	US	$[-10,10]^D$	0
6	Sphere	$f(\boldsymbol{X}) = \sum_{i=1}^{D} x_i^2$	US	$[-100,100]^D$	0
7	Booth	$f(\boldsymbol{X}) =\ x_1+2x_2-7\ ^2+\ 2x_1+x_2-5\ ^2$	MS	$[-10,10]^2$	0
8	Bohachevsky1	$f(\boldsymbol{X}) = x_1^2+2x_2^2-0.3\cos(3\pi x_1)-0.4\cos(4\pi x_2)+0.7$	MS	$[-100,100]^2$	0
9	Quadric	$f(\boldsymbol{X}) = \sum_{i=1}^{D}\left(\sum_{k=1}^{i} x_k\right)^2$	MS	$[-30,30]^D$	0
10	Eggcrate	$f(\boldsymbol{X}) = x_1^2+x_2^2+25(\sin^2 x_1+\sin^2 x_2)$	MN	$[-2\pi, 2\pi]^2$	0
11	Six Hump Camel Back	$f(\boldsymbol{X}) = 4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	MN	$[-5,5]^2$	-1.0316
12	Bohachevsky3	$f(\boldsymbol{X}) = x_1^2+2x_2^2-0.3\cos(3\pi x_1+4\pi x_2)+0.3$	MN	$[-100,100]^2$	0
13	Bridge	$f(\boldsymbol{X}) = \frac{\sin\sqrt{x_1^2+x_2^2}}{\sqrt{x_1^2+x_2^2}}+\exp(\frac{\cos 2\pi x_1+\cos 2\pi x_2}{2})-0.7129$	MN	$[-1.5,1.5]^2$	**3.0054**
14	Griewank	$f(\boldsymbol{X}) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2-\prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}})+1$	MN	$[-600,600]^D$	0
15	Ackley	$f(\boldsymbol{X}) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2})-\exp(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i)+20+e$	MN	$[-32,32]^D$	0

extremum is hidden in a deep, narrow cone, making it difficult for the algorithm to find the global extremum; F4, F6, F8, F12, F14, and F15 all have infinitely many local optimal values, and there is only one global minimum. It is very difficult for ordinary optimization algorithms to converge to the global extreme.

In summary, the use of the above-mentioned functions with different characteristics for simulation experiments is conducive to evaluating the adaptability of different optimization algorithms to different function optimization problems.

V. CHAOTIC FIREWORK ALGORITHM

To improve the performance of FWA, the external randomness, initial value sensitivity, and spatial ergodicity of the chaotic system described in Section III are fully used here. The above chaotic map is used to initialize the firework position instead of the random initialization of the basic FWA. This method makes the initial population distribution more uniform and diverse. At the same time, to make full use of chaotic characteristics, the author designed a chaotic perturbation operator. When the algorithm is trapped in local extremum and evolutionary stagnation occurs, it helps the algorithm jump out of the local optimum and accelerate convergence.

A. CHAOTIC FIREWORK ALGORITHM INITIALIZATION

Step1: Using a chaotic map equation or a chaotic hybrid system to generate N chaotic variable sequences $\boldsymbol{Cx}$ of length D, $\boldsymbol{Cx} = \{c\boldsymbol{x}_1, c\boldsymbol{x}_2, \ldots, c\boldsymbol{x}_N\}$, $c\boldsymbol{x}_i = \{cx_{i1}, cx_{i2}, \ldots, cx_{id}, \ldots, cx_{iD}\}$, $i = 1, 2, \ldots, N$, $d = 1, 2, \ldots, D$.

Step2: Map each chaotic variable from (0, 1) to (L_d, U_d) according to equation (10), so as to generate an initial firework $\boldsymbol{x} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ of scale N. Where $\boldsymbol{x}_i = \{x_{i1}, x_{i2}, \ldots, x_{id}, \ldots, x_{iD}\}$ and (L_d, U_d) are variable optimization intervals for the optimization problem.

B. CHAOTIC PERTURBATION OPERATOR

Generally speaking, swarm intelligence algorithms tend to fall into local extremes in the later stages of evolution, resulting in stagnant evolution. The chaotic perturbation operator is

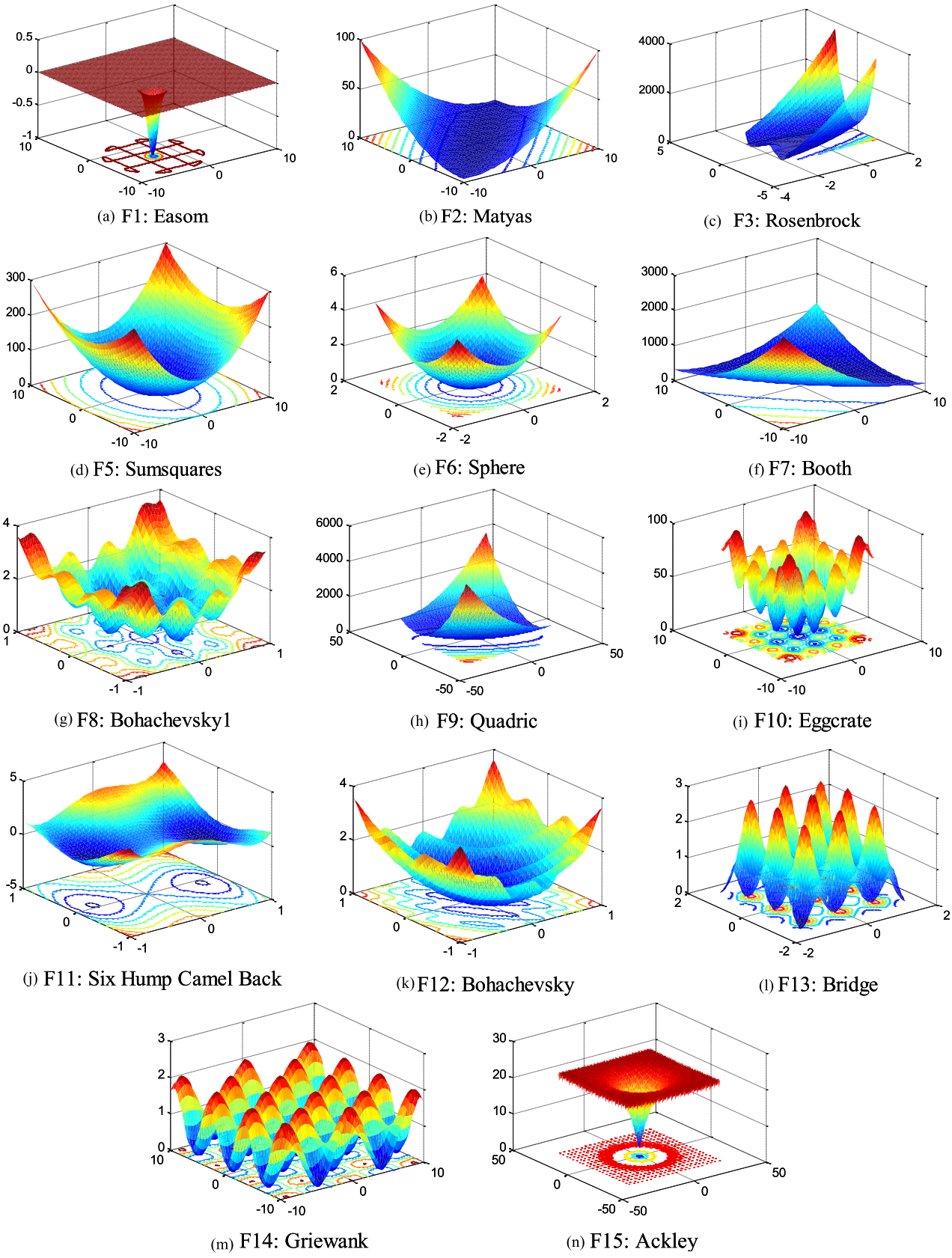


FIGURE 10. 3D plot of test functions.

one of the important means to use the ergodicity of the chaotic system to enhance the ability to explore new high-quality new solutions in the later stages of algorithm evolution. This operator helps the algorithm to continue to explore more high-quality new solutions near high-quality individuals in the population and improve the locality Optimizing ability and overall quality [41].

Combined with the chaotic map in Section III, a chaotic perturbation operator is proposed that can effectively improve the local optimization ability of the algorithm. The specific steps are as follows:

Step1: Set the number of chaotic disturbances R, perform multiple iterations according to the chaotic map to generate a sufficient number of chaotic sequences, and randomly take a number of R values to form a chaotic sequence $\boldsymbol{C} = \{c_1, c_2, \ldots, c_k, \ldots, c_R\}$.

Step2: Suppose the current firework is $\{\boldsymbol{x}_i, i = 1, 2, \ldots, N\}$, the i-th firework is $\boldsymbol{x}_i = \{x_{i1}, x_{i2}, \ldots, x_{ij}, \ldots, x_{iD}\}$, and the optimal firework position is $\boldsymbol{gx} = \{gx_1^*, gx_2^*, \ldots, gx_j^*, \ldots, gx_D^*\}$. After chaotic perturbation for R times, get R chaotic fireworks in the neighborhood of optimal firework position $\{\boldsymbol{gx}_k, k = 1, 2, \ldots, R\}$. The position of a single firework is as follows:

$$\boldsymbol{gx}_k = \{gx_1^k, gx_2^k, \ldots, gx_j^k, \ldots, gx_M^k\} \tag{12}$$

$$\boldsymbol{gx}_j^k = gx_j^* + \omega \cdot c_k \cdot r_j \tag{13}$$

Step3: In Equation (12), $\boldsymbol{gx}_j^k$ is the placement of the k-th firework in the j-th variable obtained by performing neighborhood disturbance on the optimal fireworks, r_j is the radius of the perturbation in the j-dimensional variable space, and ω is calculated from Equation (14):

$$\omega = \begin{cases} 1, & if\ random \geq 1/2 \\ -1, & if\ random < 1/2 \end{cases} \tag{14}$$

In Equation (14), $random \in (0, 1)$. The perturbation radius in equation (13) is important when the algorithm is a chaotic perturbation because if the perturbation radius is too large, it will affect the algorithm's ability to mine in the vicinity of the optimal solution; if the disturbance radius is too small, it will affect the ability of the algorithm to perform wide-area mining near the optimal solution. Besides, because the scope of different variable spaces is different, the value range of each variable is different, so disturbance radius needs to be calculated in different dimensions. the radius of the disturbance in the j-dimensional variable space is as Equation (15):

$$r_j = \left| \frac{1}{N} \sum_{i=1}^{N} x_{ij} - gx_j^* \right| \tag{15}$$

Step4: Sort the newly generated R chaotic fireworks with the current N fireworks, make a greedy decision, select the best N fireworks from the $R + N$ fireworks and enter the next iteration.

C. CFWA FLOW

Based on the differences between the two non-uniform maps: Circle map and Gauss map, two uniform maps: Bernouilli map and Tent map, and two Hybrid maps: Logistic map and Circle map, Sinousoidal iterator and Gauss map, the CFWA is recorded as CFWA_C, CFWA_G, CFWA_B, CFWA_T, CFWA_L_C, CFWA_S_G, although the map is different, the algorithm flow is basically same. The specific process of obtaining the algorithm is as follows:

Step1: Initialization parameters. the initial setting algorithm of the firework scale N, Gaussian operator M, maximum number of iterations g_{max}, constant m, parameters a, b, explosion radius $\hat{A}$, chaotic disturbance number R.

Step2: Initialize the fireworks position. Select the corresponding chaotic map, use the chaotic initialization method to initialize the firework position $\{\boldsymbol{x}_i, i = 1, 2, \ldots, N\}$, and calculate the fitness value of each initial firework.

Step3: Generate explosion sparks and Gaussian sparks. Calculate the number of explosion sparks and explosion radius according to equations (2) and (4), and generate explosion sparks and Gaussian sparks through N initial fireworks.

Step4: Generate the next generation of preparatory primitive fireworks. Calculate the fitness value of the newly generated explosion sparks and Gaussian sparks, and select N fireworks from the original fireworks, explosion sparks and Gaussian sparks according to the selection strategy.

Step5: Chaotic perturbation. Use chaotic perturbation operator to perform neighborhood perturbation near the optimal solution

Step6: Make greedy decisions. Pick best N fireworks from $R + N$ fireworks for the next iteration.

Step7: Determine the number of iterations. Determine whether the maximum iterations has been reached. If not, go to Step 3; if yes, stop iterating and output the best fireworks and its fitness value.

VI. EXPERIMENT AND ANALYSIS

A. EVALUATION STANDARDS FOR OPTIMIZATION RESULTS

For the algorithm simulation results, inspired by [42], the algorithm performance can be evaluated from four aspects: accuracy, convergence, robustness, and computational cost. The accuracy of the solution is mainly measured by the two indexes of the best value (BEST) and the average value (MEAN) obtained by the solution. The convergence of the solution is measured by the solution success rate (SR). The solution success is a relative concept, that is, the result of the solution satisfies Equation (16) is considered to be successful once:

$$\begin{cases} \frac{|Y - Y^*|}{Y} < \varepsilon, & Y \neq 0 \\ |Y - Y^*| < \varepsilon, & Y = 0 \end{cases} \tag{16}$$

In Equation (16), Y is the optimal value obtained by the solution, and Y^* is the reference optimal value of the function, ε is set to 10e-6. Then the number of successful optimizations divided by the total number is the solution success rate, which is used to measure the algorithm's solution convergence.

The robustness of the algorithm is expressed by the variance (STD) and worst-case value (WORST). It mainly examines the ability of the algorithm to overcome random interference and obtain high-precision global extreme values. The computational cost of the algorithm is measured by the average number of iterations (AIN) required to achieve a certain accuracy relative to the optimal solution [43]. In summary, we use BEST, MEAN, WORST, STD, SR, AIN to evaluate the performance of different algorithms.

The average iterations AIN is required by each algorithm to obtain a result with better accuracy than $\sigma = 10\text{e-}6$. When a certain calculation does not reach a higher accuracy than $\sigma = 10\text{e-}6$, the iterations is regarded as the preset maximum iterations as $g_{max} = 2000$.

B. CONTRAST ALGORITHMS AND PARAMETER SETTINGS

The six CFWAs (CFWA_C, CFWA_G, CFWA_B, CFWA_T, CFWA_L-C, CFWA_S-G) designed in this paper are compared with three improved algorithms (EFWA [22], dynFWA [23], AFWA [24]) and FWA.

The common parameters are set as follows: the maximum number of iterations is $g_{max} = 2000$, the population size is $N = 20$, the constant $m = 50$ to adjust the explosion sparks, parameters $a = 0.8, b = 0.04$, and the explosion radius $\hat{A} = 40$. The algorithms perform 20 independent operations on 15 test functions and calculate the average value. The internal parameters settings of CFWA and dynFWA, EFWA, AFWA are shown in Table 3.

C. SIMULATIONS

In addition, to visually display the average fitness curve of each algorithm for solving complex functions, the numbers less than 10e-30 are regarded as 0, shown on the graph is that some curves will be iterated to a certain algebraic termination. At the same time, considering the length of the paper, the author selected three fixed-dimensional functions F3 ($D = 2$), F4 ($D = 4$), F7 ($D = 2$), and all high-dimensional complex functions F5 ($D = 150$), F6 ($D = 200$), F9 (($D = 100$)), F14 ($D = 100$), F15 ($D = 200$). The average fitness curves of these 8 functions are plotted, as shown in Figure 11. Table 4 is the simulation data of all algorithms for all functions. The simulations employed Lenovo personal computer with Core^{TM} i7-6500 CPU operating under a clock frequency of 2.60 GHz, and 8 GB of memory. The average fitness curves of these 8 functions are plotted, as shown in Figure 11. Table 4 is the simulation data of ten algorithms for all functions.

For some test functions, the performance of FWA is better than the three improved algorithms of dynFWA, EFWA and AFWA. This is because the FWA map rules and mutation operators enhance the search near the origin [22], so the algorithm is easy to solve near optimal value if the optimal value of the test function is far from the origin, FWA shows poor performance.

For low-dimensional complex functions, the six types of CFWA performance better than FWA, dynFWA, AFWA, and EFWA when solving functions F1, F2, F3, F4, F8, F10, F11, F12, and F13, showing better BEST value, MEAN value, WORST value, and smaller STD and average number of AIN. For example, when solving functions F8, F10, and F12, the BEST, MEAN, WOSRT, and STD values obtained by the six CFWAs are all zero. However, when solving the function F7, the performance of the six CFWAs is not as good as AFWA. AFWA can continue to evolve and achieve higher accuracy, but at the beginning of evolution, the iteration speed is slower than CFWAs. A careful comparison of Table 4 and Figure 11, it can be found that among the six CFWAs, CFWA_L_C and CFWA_S_G are better than CFWA_C, CFWA_G, CFWA_T, CFWA_B. Hybrid chaotic maps are better than single maps. The hybrid chaotic map can improve the convergence speed and accuracy.

For high-dimensional complex functions, dynFWA, AFWA, and EFWA perform poorly. One possible reason is that as the algorithm iteratively evolves, the diversity of the population gradually disappears, which ultimately results in poor results when obtaining high-dimensional complex functions. Six CFWAs perform well, for the 150-dimensional complex function F5, the optimal value of the CFWA_S_G reaches 0, and the CFWA_L_C reaches 5.85e-268, even the relatively poorly performing CFWA_G also reaches 1.68e-240; for the 200-dimensional F6, CFWA_S_G reaches the global extreme value 0; for the 100-dimensional F9, six CFWAs obtain the BEST value of 0; For the 100-dimensional F14, the BEST, WORST, and STD values obtained by the six CFWAs are all 0; even for the 200-dimensional function F15, the six CFWAs obtained the BEST value, MEAN value, and WORST value are all 8.88e-16, showing the amazing ability to solve high-dimensional complex functions. Therefore, the six CFWAs have better solving ability and stability for high-dimensional complex functions. At the same time, it can be found that when solving high-dimensional functions, it still shows a similar pattern to low-dimensional functions, hybrid maps are better than single maps. From the simulation results, CFWA_L_C is better for most low-dimensional functions, CFWA_S_G is better at solving high-dimensional complex functions.

CFWA adds chaotic initialization and perturbation operators to FWA. Chaotic initialization has little effect on the time complexity, so only the time complexity of chaotic perturbation operator needs to be analyzed. The time complexity of chaotic perturbation operator focuses on the sorting process after chaotic perturbation. For different sorting methods, the corresponding time complexity is also different. In the worst case, the time complexity is $O(n^2)$. However, using methods such as quick sorting and merge sorting, the average time complexity is $O(n\log_2 n)$.

Limitations of CFWA are mainly manifested in two aspects: ň Not every chaotic perturbation operator will improve performance better, for example, the CFWA_C is worse than AFWA in F3 (Rosenbrock function) and all CFWAs is worse than AFWA in F7 (Booth function); Due to

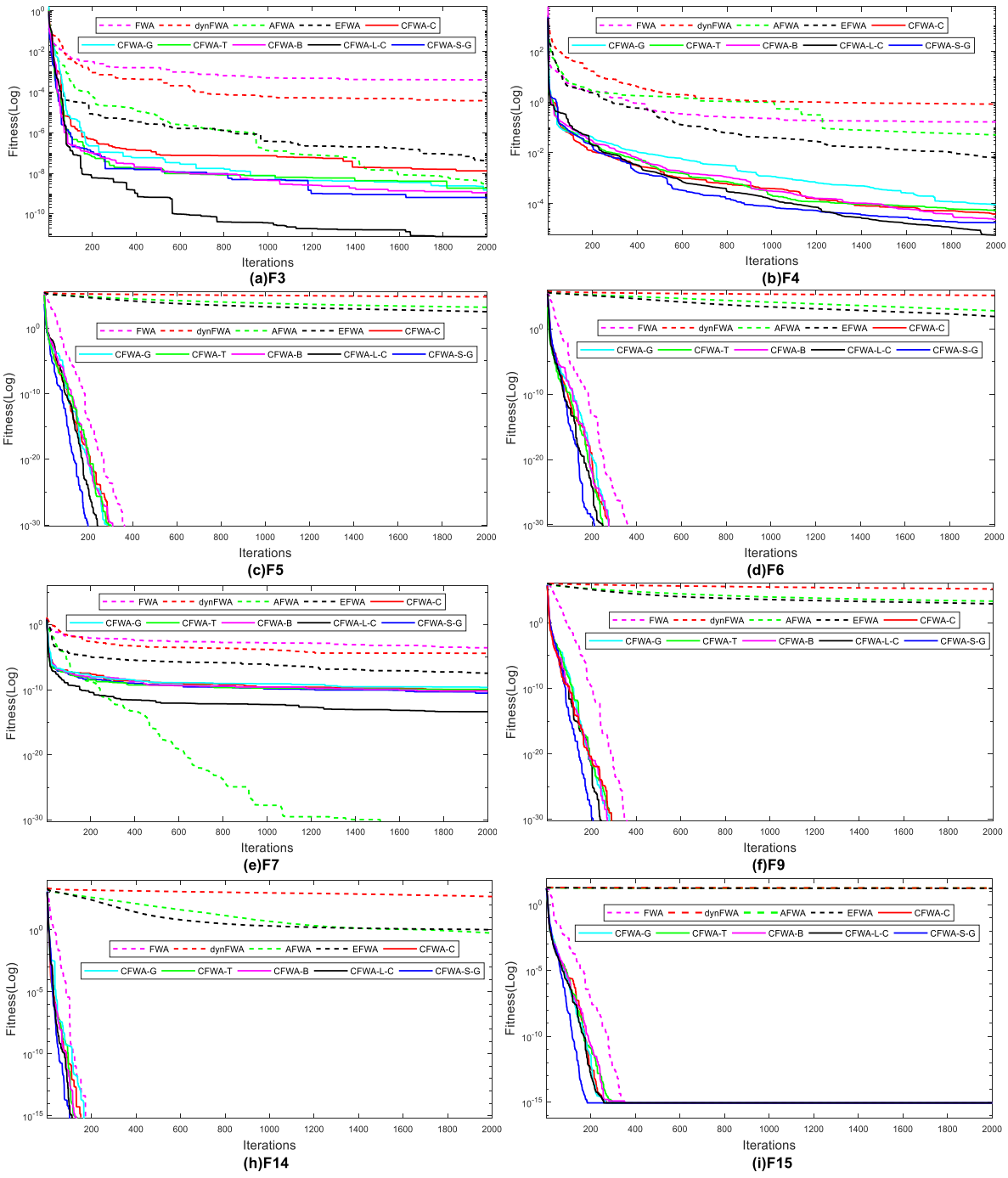


FIGURE 11. Average fitness curve of 10 swarm intelligence algorithms.

the addition of chaotic perturbation operator, it will slightly increase the time complexity of FWA.

VII. APPLICATION OF CFWA IN BLIND SOURCE SEPARATION

Blind Source Separation (BSS) refers to the process of estimating the source signal only from the observed signal when the transmission channel is unknown [44]. In BSS, Independent Component Analysis [45] (ICA) is a major method. ICA means that under the condition that the source signals are independent of each other, the objective function is established using the signal's probability density function and related information theory knowledge, and the non-Gaussian nature of the estimated signal is maximized by an optimization algorithm. Therefore, CFWA_L_C and CFWA_S_G which perform best in test functions are used as an optimization algorithm in BSS to verify the performance of CFWA in practical engineering problems.

TABLE 3. Internal parameter settings of dynFWA, EFWA, AFWA and CFWA.

Algorithm	Internal parameters
dynFWA	Reduction factor $C_r = 0.9$, Amplification factor $C_a = 1.2$
AFWA	Explosion radius adjustment factor $\lambda = 1.3$
EFWA	Maximum radius factor $S_{\max} = 0.02$, Minimum radius factor $S_{\min} = 0.001$
CFWA	Chaotic disturbance number $R = 100$

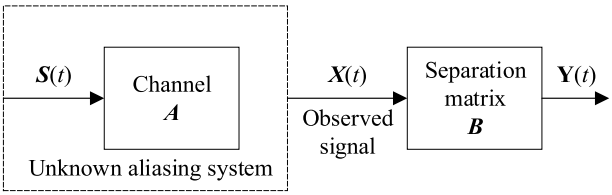


FIGURE 12. Typical schematic of BSS.

A. BASIC CONCEPT OF BSS

A typical BSS schematic is shown below:

n independent source signals $\boldsymbol{S}(t)=[s_1(t), s_2(t), \ldots, s_n(t)]^{\mathrm{T}}$ are mixed to $\boldsymbol{X}(t) = [x_1(t), x_2(t), \ldots, x_m(t)]^{\mathrm{T}}$ through an unknown system, $\boldsymbol{X}(t)$ is m observation signals, after ignoring the effects of transmission delay and noise, it can be obtained from equation (17):

$$\boldsymbol{X}(t) = \boldsymbol{A}\boldsymbol{S}(t) \tag{17}$$

In Equation (17), $\boldsymbol{A}$ is a n-dimensional full-rank reversible mixed matrix. BSS is when only the observation signal (mixed signal) $\boldsymbol{X}(t)$ is known, a full-rank separation matrix $\boldsymbol{B}$ is obtained through an optimization algorithm, and then the separation signal $\boldsymbol{Y}(t)$ is obtained from the observation signal as shown in equation (18).

$$\boldsymbol{Y}(t) = \boldsymbol{B}\boldsymbol{X}(t) \tag{18}$$

In Equation (18), $\boldsymbol{Y}(t) = [y_1(t), y_2(t), \ldots, y_n(t)]^{\mathrm{T}}$. The BSS algorithm is the process of determining the separation matrix $\boldsymbol{B}$ based on the statistical independence between $y_i(t)$.

B. SEPARATION PRINCIPLE AND OBJECTIVE FUNCTION

Kurtosis and negative entropy are two common indicators for measuring the statistical independence of signals, but the kurtosis is easily affected by outliers and the separation performance is unstable. In information theory, negative entropy is a more robust criterion. The negative entropy of a random signal can be expressed as Equation (19).

$$J(y) = H_G(y) - H(y) \tag{19}$$

In Equation (19), $H(y)$ is the differential entropy of the random variable y. According to [46], the negative entropy of a multivariate can be approximated as (20).

$$J(y) = \frac{1}{48}[4K_3^2 + K_4^2 + 7K_4^3 - 6K_3^2K_4] \tag{20}$$

In the Equation (20), K_3 is the third-order cumulant of the signal; K_4 is the fourth-order cumulant of the signal. When the signal's probability density distribution is symmetrical,

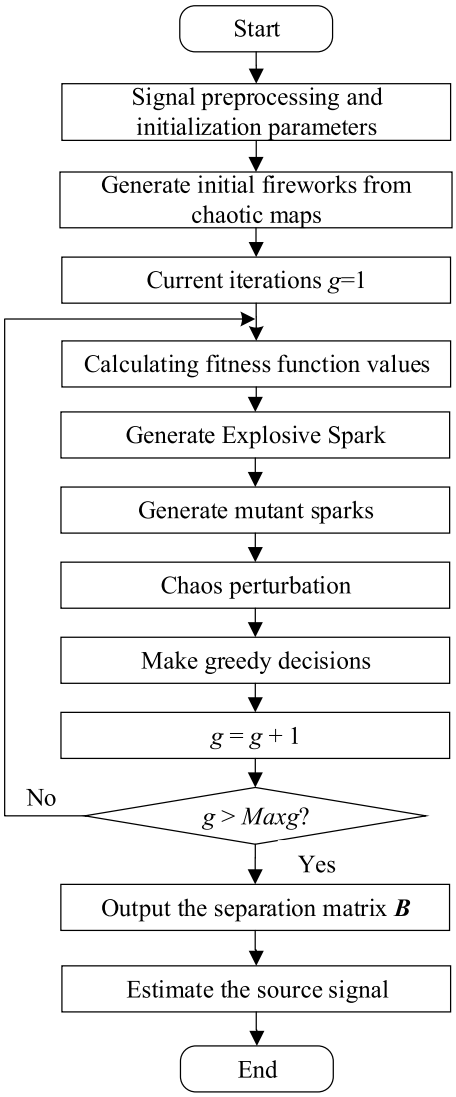


FIGURE 13. BSS flow based on CFWA.

TABLE 4. Comparative analysis of performance of 10 swarm intelligence algorithms.

Fun	Evaluation standard	FWA	dynFWA	AFWA	EFWA	CFWA_C	CFWA_G	CFWA_B	CFWA_T	CFWA_LC	CFWA_SG
F1	BEST	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	MEAN	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	WORST	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	STD	5.90e-06	4.19e-07	0	1.48e-11	1.91e-11	2.79e-11	7.84-12	1.12e-10	2.07e-15	7.44e-12
	SR	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	AIN	1074	282	24	85	14	17	11	12	7	10
F2	BEST	2.74e-238	5.76e-08	7.40e-167	3.51e-12	0	2.55e-304	0	5.67e-296	4.90e-324	0
	MEAN	1.05e-230	1.23e-06	1.68e-120	7.60e-11	2.60e-258	7.54e-238	4.43e-253	2.68e-256	4.64e-295	0
	WORST	4.05e-227	9.85e-06	3.34e-119	3.03e-10	2.60e-257	7.54e-237	1.77e-252	2.68e-255	1.85e-294	0
	STD	0	2.22e-06	7.47e-120	1.07e-10	0	0	0	0	0	0
	SR	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	AIN	26	450	52	38	5	5	5	6	4	5
F3	BEST	2.11e-06	5.12e-07	8.33e-30	5.02e-10	8.28e-11	3.61e-11	2.22e-10	2.34e-10	2.43e-14	5.01e-11
	MEAN	3.98e-04	5.74e-05	2.56e-09	3.28e-08	1.32e-08	1.10e-09	1.50e-09	1.86e-09	7.66e-12	6.49e-10
	WORST	0.0010	1.90e-04	1.66e-08	2.41e-07	5.17e-08	3.56e-09	6.66e-09	6.46e-09	6.41e-11	2.33e-09
	STD	3.464e-04	7.89e-05	5.66e-09	4.28e-08	1.78e-08	1.29e-09	2.04e-09	2.03e-09	1.99e-11	6.54e-10
	SR	10%	40%	100%	100%	100%	100%	100%	100%	100%	100%
	AIN	2000	2000	444	186	75	72	68	71	67	69
F4	BEST	0.0077	0.0298	2.21e-05	5.89e-05	2.71e-07	2.13e-06	4.43e-07	2.19e-07	3.10e-08	1.54e-06
	MEAN	0.1644	0.8218	0.0506	0.0064	3.82e-05	9.11e-05	2.37e-05	5.42e-05	5.62e-06	1.76e-05
	WORST	0.3981	2.2877	0.1353	0.0277	1.44e-04	2.45e-04	8.80e-05	1.49e-04	2.47e-05	9.55e-05
	STD	0.1572	0.6898	0.0460	0.0080	4.36e-05	8.60e-05	2.60e-05	5.89e-05	7.37e-06	2.76e-05
	SR	0	0	0	0	30%	20%	30%	30%	80%	50%
	AIN	2000	2000	2000	2000	2000	2000	2000	2000	1855	2000
F5	BEST	8.00e-238	4.53e+04	718.76	2.29e+02	7.52e-252	1.68e-240	1.06e-260	8.18e-260	5.85e-268	0
	MEAN	1.25e-198	6.14e+04	1.17e+03	3.13e+02	1.00e-212	1.33e-200	1.94e-205	1.05e-210	8.44e-218	6.70e-320
	WORST	6.63e-194	8.01e+04	1.77e+03	4.60e+02	1.00e-211	1.33e-199	1.94e-204	1.05e-209	8.44e-217	6.70e-319
	STD	0	9.34e+03	260.08	1.28e+02	0	0	0	0	0	0
	SR	100%	0	0	0	100%	100%	100%	100%	100%	100%
	AIN	108	2000	2000	2000	83	76	64	61	50	36
F6	BEST	2.73e-231	1.16e+05	4.48e+02	66.330	1.71e-260	8.23e-245	1.30e-257	1.32e-244	2.40e-244	0
	MEAN	3.86e-201	1.35e+05	6.02e+02	80.3776	6.88e-205	4.12e-205	7.46e-209	3.30e-205	2.39e-220	0
	WORST	3.86e-199	1.60e+05	8.58e+02	1.01e+02	6.88e-204	4.12e-208	7.46e-200	3.3e-204	9.60e-220	0
	STD	0	1.34e+04	1.57e+02	39.836	0	0	0	0	0	0
	SR	100%	0	0	0	100%	100%	100%	100%	100%	100%
	AIN	79	2000	2000	2000	50	46	45	33	35	28
F7	BEST	5.90e-06	8.19e-07	0	1.41e-09	7.98e-12	1.13e-12	1.23e-11	3.86e-11	2.06e-16	3.23e-12
	MEAN	2.98e-04	6.88e-05	2.24e-30	3.63e-08	8.44e-11	2.39e-10	6.51e-11	1.12e-10	4.38e-14	3.12e-11
	WORST	9.49e-04	3.15e-04	2.52e-29	1.41e-07	3.32e-10	1.61e-09	2.11e-10	2.31e-10	2.04e-13	8.10e-11
	STD	3.59e-04	8.87e-05	5.99e-30	4.33e-08	9.81e-11	4.88e-10	6.63e-11	7.45e-11	7.19e-14	3.02e-11
	SR	10%	15%	100%	100%	100%	100%	100%	100%	100%	100%
	AIN	2000	2000	90	220	15	20	15	12	11	15
F8	BEST	0	7.14e-06	0	9.48e-07	0	0	0	0	0	0
	MEAN	0	1.96e-04	0	1.090e-06	0	0	0	0	0	0
	WORST	0	7.89e-04	0	2.98e-06	0	0	0	0	0	0
	STD	0	5.10e-04	0	9.48e-07	0	0	0	0	0	0
	SR	100%	15%	100%	100%	100%	100%	100%	100%	100%	100%
	AIN	35	2000	62	1484	16	18	16	16	14	16
F9	BEST	2.43e-238	1.28e+05	8.79e+02	2.30e+02	2.78e-261	7.07e-260	3.90e-256	4.91e-246	2.87e-254	0
	MEAN	1.28e-198	1.63e+05	2.07e+03	7.39e+02	1.05e-201	2.55e-204	5.65e-218	5.78e-212	6.55e-222	0
	WORST	1.28e-194	2.13e+05	3.78e+03	1.15e+03	1.05e-200	2.55e-203	5.65e-211	5.78e-211	6.55e-214	0
	STD	0	2.49e+04	7.45e+02	1.78e+02	0	0	0	0	0	0
	SR	100%	0	0	0%	100%	100%	100%	100%	100%	100%
	AIN	98	2000	2000	2000	53	70	44	68	44	40

$K_3 = 0$, then Equation (20) can be written as:

$$J(y) = \frac{1}{48}K_4^2 \tag{21}$$

The fourth-order cumulant K_4 can be expressed as:

$$K_4 = E\left[y^4\right] - 3(E[y]^2) \tag{22}$$

So the objective function can be set as:

$$fit(y) = \frac{1}{J(y) + \varepsilon} \tag{23}$$

In Equation (23) ε is an extremely small amount that prevents division by zero. The bigger $J(y)$ is, the smaller $fit(y)$ is, and the better the separation performance is.When using the negative entropy to measure the non-Gaussian nature of the separated signal, the observation signal needs to be

TABLE 4. *(Continued)* **Comparative analysis of performance of 10 swarm intelligence algorithms.**

Fun	Evaluation standard	FWA	dynFWA	AFWA	EFWA	CFWA_C	CFWA_G	CFWA_B	CFWA_T	CFWA_LC	CFWA_SG
F10	BEST	0	1.68e-07	1.17e-191	3.04e-10	0	0	0	0	0	0
	MEAN	0	7.37e-06	5.53e-150	3.90e-09	0	0	0	0	0	0
	WORST	0	3.20e-05	2.26e-149	2.13e-08	0	0	0	0	0	0
	STD	0	7.94e-06	7.14e-150	6.34e-09	0	0	0	0	0	0
	SR	100%	65%	100%	100%	100%	100%	100%	100%	100%	100%
	AIN	32	1782	43	173	17	12	10	11	9	10
F11	BEST	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316
	MEAN	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316
	WORST	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316
	STD	2.47e-06	1.17e-06	1.01e-16	2.86e-08	1.37e-9	1.72e-10	4.36e-10	1.36e-10	4.91e-12	1.67e-10
	SR	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	AIN	519	403	50	77	11	10	10	9	8	9
F12	BEST	0	1.93e-06	0	2.65e-07	0	0	0	0	0	0
	MEAN	0	0.007	3.88e-17	1.86e-05	0	0	0	0	0	0
	WORST	0	0.042	2.22e-16	5.82e-05	0	0	0	0	0	0
	STD	0	0.010	5.72e-17	2.11e-05	0	0	0	0	0	0
	SR	100%	5%	100%	60%	100%	100%	100%	100%	100%	100%
	AIN	54	2000	125	2000	19	18	15	13	12	13
F13	BEST	3.0054	3.0054	3.0054	3.0054	3.0054	3.0054	3.0054	3.0054	3.0054	3.0054
	MEAN	3.0054	3.0054	3.0054	3.0054	3.0054	3.0054	3.0054	3.0054	3.0054	3.0054
	WORST	3.0054	3.0054	3.0054	3.0054	3.0054	3.0054	3.0054	3.0054	3.0054	3.0054
	STD	4.68e-16	3.26e-07	9.11e-16	1.17e-09	4.68e-16	4.68e-16	4.68e-16	4.68e-16	4.68e-16	4.68e-16
	SR	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	AIN	27	87	138	80	16	16	14	14	9	14
F14	BEST	0	3.07e+02	0.323	0.9415	0	0	0	0	0	0
	MEAN	0	5.01e+02	0.507	0.9878	0	0	0	0	0	0
	WORST	0	7.10e+02	0.690	1.0513	0	0	0	0	0	0
	STD	0	93.46	0.108	0.0351	0	0	0	0	0	0
	SR	100%	0	0	0	100%	100%	100%	100%	100%	100%
	AIN	61	2000	2000	2000	38	38	32	34	31	26
F15	BEST	8.88e-16	18.21	17.78	16.0260	8.88e-16	8.88e-16	8.88e-16	8.88e-16	8.88e-16	8.88e-16
	MEAN	8.88e-16	18.66	18.25	17.7084	8.88e-16	8.88e-16	8.88e-16	8.88e-16	8.88e-16	8.88e-16
	WORST	8.88e-16	19.07	18.78	18.3497	8.88e-16	8.88e-16	8.88e-16	8.88e-16	8.88e-16	8.88e-16
	STD	0	0.241	0.306	0.6863	0	0	0	0	0	0
	SR	100%	0	0	0	100%	100%	100%	100%	100%	100%
	AIN	128	2000	2000	2000	90	91	86	82	73	63

pre-centered and whitened. After these two pre-processing, the statistical independence of the signal will be stronger.

C. ALGORITHM STEPS AND PROCESSES

The separation matrix $\boldsymbol{B}$ is an orthogonal matrix. literature [47] shows that an orthogonal matrix can be represented as a product of a series of rotation matrixs. There are three source signals, and the separation matrix $\boldsymbol{B}$ can be expressed as equation (24):

$$\boldsymbol{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_1 & \sin\theta_1 \\ 0 & \sin\theta_1 & \cos\theta_1 \end{bmatrix} \bullet \begin{bmatrix} \cos\theta_2 & 0 & -\sin\theta_2 \\ 0 & 1 & 0 \\ \sin\theta_2 & 0 & \cos\theta_2 \end{bmatrix} \bullet \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 \\ \sin\theta_3 & \cos\theta_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (24)$$

The steps of BSS based on CFWA are shown below:

Step1: Collect and sample observation signals, and perform centralization and whitening preprocessing.

Step2: The rotation angle $\theta = [\theta_1, \theta_2, \theta_3]$ of the rotation matrix is used as the fireworks position $\boldsymbol{x} = [x_1, x_2, x_3]$. Within the solution space $[0, 2\pi]$. Initialize CFWA parameters and firework position according to the chaotic maps.

Step3: Calculate and record the objective function value of the fireworks.

Step4: Use Equation (4) to calculate the explosion radius of the fireworks and generate explosion sparks.

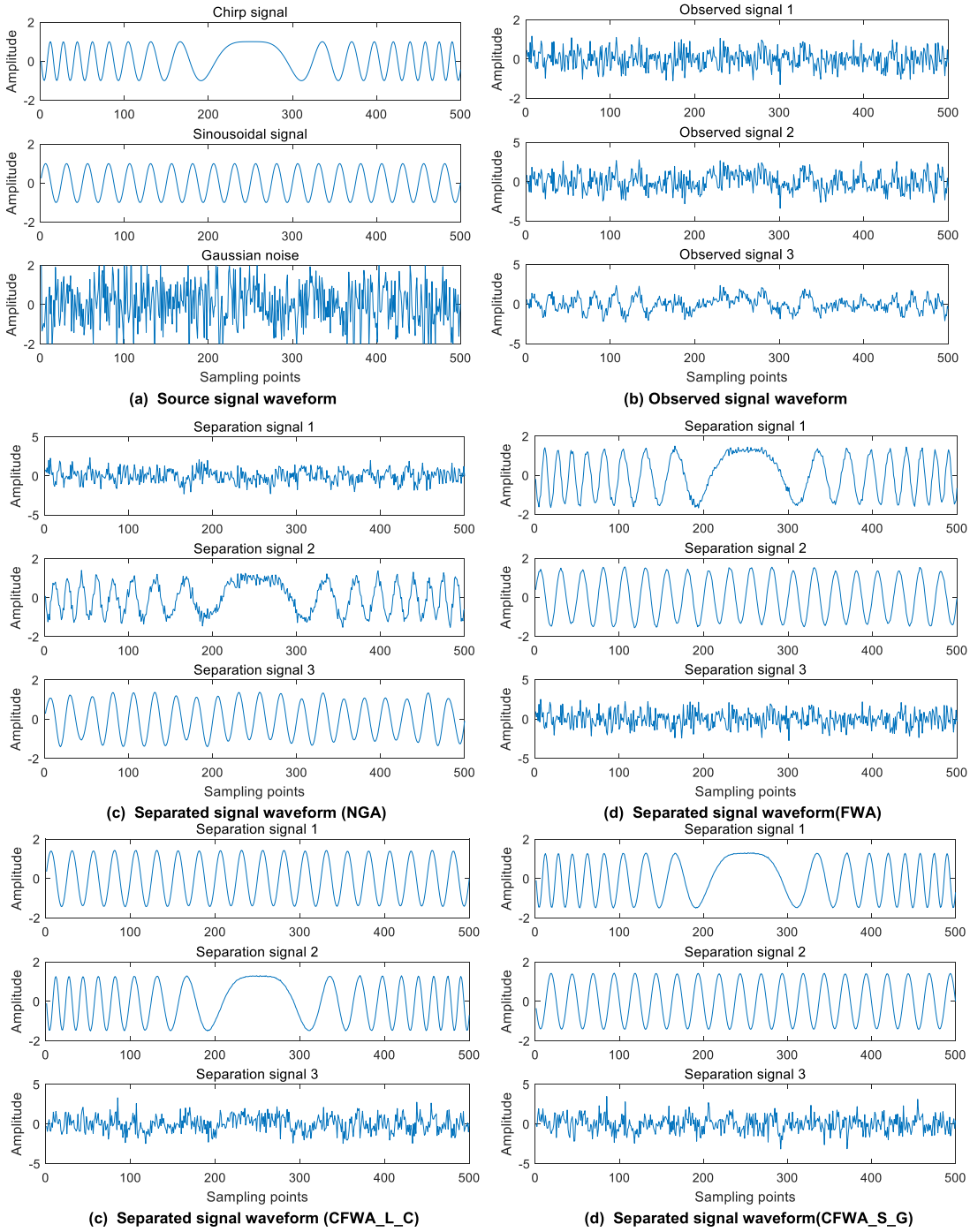


FIGURE 14. Simulation results.

Step5: Use Equation (6) to generate Gaussian mutation sparks.

Step6: The out-of-range spark is mapped into the range by Equation (7).

Step7: Perform chaotic perturbation and greedy decision, choose the best fireworks to enter the next iteration.

Step8: $g = g + 1$, return to Step3 until the maximum number of iterations is reached or the best firework is found, and the loop ends.

Step9: Output the optimal solution $\boldsymbol{x}^*$ and calculate the separation matrix $\boldsymbol{B}$ by equation (24).

Step10: Using separation matrix $\boldsymbol{B}$, the source signal is estimated by equation (18).

D. SIMULATION ANALYSIS

Two classic radar signals and one Gaussian noise are used as source signals. Two radar signals are Chirp signal $s_1(t)$ and Sinousoidal signal $s_2(t)$, and Gaussian noise is $s_3(t)$. The

bandwidth of the chirp signal is $B = 70\text{MHZ}$, the pulse width is $T = 10\text{us}$, and the chirp frequency is $\rho = B/T$, expressions are (25) (26) (27).

$$s_1(t) = rect(\frac{t}{T})\exp(j\pi\rho t^2) \tag{25}$$

$$s_2(t) = sin(2\pi 300t/5000) \tag{26}$$

$$s_3(t) = n(t) \tag{27}$$

In order to quantitatively analyze the separation performance of the algorithm, the Similarity Coefficient and Performance Index (PI) are used to measure the separation performance. The Similarity Coefficient ξ_{ij} and Performance index PI are expressed as Equation (28)(29).

$$\xi_{ij} = (s_i, y_j) = \left| \frac{\sum_{t-1}^{N} y_j(t)s_i(t)}{\sqrt{\sum_{t-1}^{N} y_j^2(t) \sum_{t-1}^{N} s_i^2(t)}} \right| \tag{28}$$

$$PI = \sum_{i=1}^{n} \left(\sum_{j=1}^{n} \frac{|\mu_{ij}|}{\max_k |\mu_{ij}|} - 1 \right) + \sum_{i=1}^{n} \left(\sum_{j=1}^{n} \frac{|\mu_{ji}|}{\max_k |\mu_{ki}|} - 1 \right) \tag{29}$$

In equation (28), s_i is the source signal of the i-th path, and y_j is separation signal of the j-th path. The closer ξ_{ij} is to 1, the higher similarity between the sorting signal and the source signal. In equation (28), μ_{ij} is the element in the i-th row and j-th column of the matrix $\boldsymbol{G}$, $\boldsymbol{G} = \boldsymbol{BA}$. The closer PI is to 0, the higher the separation accuracy of the algorithm. The number of signal sampling points is 500, and the maximum iterations of the algorithm is $g_{max} = 100$. Equation (30) is a randomly generated mixing matrix $\boldsymbol{A}$. At the same time, we use the traditional Natural Gradient Algorithm (NGA) in BSS for comparison. The simulation results are shown in Figure 13. Table 5 is Similarity Coefficient and PI of FWA and CFWA_L_C, for a more intuitive comparison, the author has drawn a histogram of the Similarity Coefficient between the separated signal and the source signal, as shown in Figure 14.

$$\boldsymbol{A} = \begin{bmatrix} 0.063 & 0.117 & 0.397 \\ 0.881 & 0.003 & 0.863 \\ 0.856 & 0.647 & 0.450 \end{bmatrix} \tag{30}$$

Comparing with Figure 14, the Chirp signal and the Sinusoidal signal separated by NGA and FWA algorithm are deformed. The signal separated by the CFWA_L_C and CFWA_S_G has a good agreement with the source signal, and the separation effect is ideal.

As can be seen from Table 5 and Figure 15, CFWA_L_C and CFWA_S_G is superior to NGA and FWA in both the Similarity Coefficient and PI. Compared with traditional

TABLE 5. Data of algorithm performance evaluation index.

Algorithm	NGA	FWA	CFWA_L_C	CFWA_S_G
Similarity Coefficient	0.9669	0.9906	0.9999	0.9999
	0.9898	0.9872	1.0000	0.9999
	0.9346	0.9372	0.9632	0.9625
PI	0.9234	0.4584	0.0143	0.0148
Number of convergence	346	92	11	12

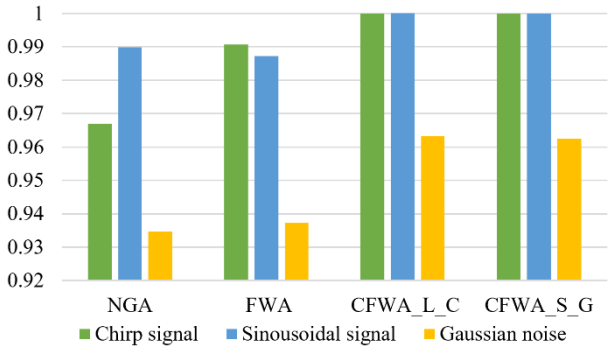


FIGURE 15. Histogram of Similarity Coefficients.

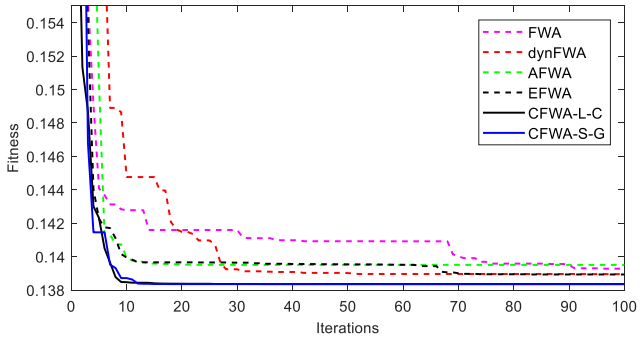


FIGURE 16. Fitness curve of objective function.

NGA algorithm, swarm intelligence algorithms have fewer iterations and higher separation accuracy. Figure 16 is the fitness curve of the objective function. CFWA_L_C has smaller fitness and faster convergence than FWA, dynFWA, AFWA, and EFWA.

VIII. CONCLUSION

This paper has presented the Chaotic Fireworks Algorithm (CFWA), a significant improvement of the Fireworks Algorithm (FWA). In order to improve the performance of conventional FWA, we performed a comprehensive study on the basic FWA and presented several improvements: by analyzing the classic chaotic maps, two hybrid chaotic systems are designed and applied to make the initial fireworks more uniform and increase the probability of finding the optimal solution. At the same time, a new chaotic perturbation operator is proposed, which can increase the speed of the algorithm convergence, to avoid the algorithm falling into a

local optimum. From the result of our experimental evaluation we conclude the following observations:

1. The proposed CFWA algorithm significantly improves the results of FWA, and it performs better than EFWA, dynFWA and AFWA in all test functions except Booth function. In Booth function, AFWA performs the best, but it's speed of reaching 10e-6 accuracy is slower than CFWA.
2. In CFWA, two hybrid chaotic systems proposed in this paper are better than two uniform chaotic maps and two non-uniform chaotic maps.
3. In the BSS problem, the Similarity Coefficient and PI of the CFWA separated signal and the source signal are smaller than FWA, and the convergence speed and convergence accuracy are also better than FWA, EFWA, dynFWA, and AFWA.

REFERENCES

[1] A. S. Fraser, "Simulation of genetic systems by automatic digital computers ii. effects of linkage on rates of advance under selection," *Austral. J. Biol. Sci.*, vol. 10, no. 4, pp. 492–500, 1957.
[2] H. Li, H. Jin, H. Wang, and Y. Ma, "Improved adaptive holonic particle swarm optimization," *Math. Problems Eng.*, vol. 2019, pp. 1–22, Dec. 2019.
[3] C. Rajendran and H. Ziegler, "Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs," *Eur. J. Oper. Res.*, vol. 155, no. 2, pp. 426–438, Jun. 2004.
[4] C. J. A. B. FilhoFernando, B. de Lima NetoAnthony, J. C. C. LinsAntônio, I. S Nascimento, and M. P. Lima, "Fish school search," in *Studies in Computational Intelligence*, vol. 193. 2009, pp. 261–277.
[5] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *J. Global Optim.*, vol. 39, no. 3, pp. 459–471, Oct. 2007.
[6] S. Łukasik, S. Żak, "Firefly algorithm for continuous constrained optimization tasks," in *Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems* (Lecture Notes in Computer Science) vol. 5796. 2009, pp. 97–106.
[7] X. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proc. World Congr. Nature Biol. Inspired Comput. (NaBIC)*, Dec. 2009, pp. 210–214.
[8] K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *IEEE Control Syst. Mag.*, vol. 22, no. 3, pp. 52–67, Mar. 2002.
[9] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014.
[10] W. Pan, "A new fruit fly optimization algorithm: Taking the financial distress model as an example," *Knowl.-Based Syst.*, vol. 26, pp. 69–74, Feb. 2012.
[11] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1982.
[12] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "GSA: A gravitational search algorithm," in *Proc. Int. Conf. Comput. Knowl. Eng.*, 2012.
[13] D. Simon, "Biogeography-based optimization," *IEEE Trans. Evol. Comput.*, vol. 12, no. 6, pp. 702–713, Dec. 2008.
[14] Y. J. Zheng, "Water wave optimization: A new nature-inspired metaheuristic," *Comput. Oper. Res.*, vol. 55, no. 3, pp. 1–11, Mar. 2015.
[15] B. Doğan and T. Ölmez, "A new Metaheuristic for numerical function optimization: Vortex search algorithm," *Inf. Sci.*, vol. 293, pp. 125–145, Feb. 2015.
[16] Z. Woo Geem, J. Hoon Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: Harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, Feb. 2001.
[17] Y. Tan and Y. C. Zhu, "Fireworks algorithm for optimization," in *Proc. Int. Conf. Swarm Intell.*, 2010, pp. 355–364.
[18] T. Guilin, L. C. yun, and G. Jiale, "Multi-sensor optimal disposition model based on firework algorithm," *Syst. Eng. Electron.*, vol. 41, no. 8, pp. 1742–1748, 2019.
[19] A. Mohamed Imran and M. Kowsalya, "A new power system reconfiguration scheme for power loss minimization and voltage profile enhancement using fireworks algorithm," *Int. J. Electr. Power Energy Syst.*, vol. 62, pp. 312–322, Nov. 2014.
[20] X. Yu-Ying, "Path planning and smoothing based on quantum-behaved fireworks algorithm for mobile robot," *Control Theory Appl.*, vol. 36, no. 9, pp. 1398–1408, 2019.
[21] J. Li and Y. Tan, "Loser-out tournament-based fireworks algorithm for multimodal function optimization," *IEEE Trans. Evol. Comput.*, vol. 22, no. 5, pp. 679–691, Oct. 2018.
[22] S. Zheng, A. Janecek, and Y. Tan, "Enhanced fireworks algorithm," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2013, pp. 2069–2077.
[23] S. Zheng, A. Janecek, J. Li, and Y. Tan, "Dynamic search in fireworks algorithm," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2014, pp. 3222–3229.
[24] J. Li, S. Zheng, and Y. Tan, "Adaptive fireworks algorithm," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2014, pp. 3214–3221.
[25] L. Hao, B. Peng, Z. Hui, H. Jin, and J. Xue, "Backward fireworks algorithm and application research," *J. Xian Jiaotong Univ.*, vol. 49, no. 11, pp. 82–88, 2015.
[26] K. Srikanth Reddy, L. K. Panwar, R. Kumar, and B. K. Panigrahi, "Binary fireworks algorithm for profit based unit commitment (PBUC) problem," *Int. J. Electr. Power Energy Syst.*, vol. 83, pp. 270–282, Dec. 2016.
[27] J. Li, S. Zheng, and Y. Tan, "The effect of information utilization: Introducing a novel guiding spark in the fireworks algorithm," *IEEE Trans. Evol. Comput.*, vol. 21, no. 1, pp. 153–166, Feb. 2017.
[28] S. Xiaoning, W. Qian, H. Yao, and Y. Xuan, "An enhanced multi-modal function optimization fireworks algorithm base on loser-out tournament," *J. Syst. Simul.*, vol. 32, no. 1, pp. 9–19, 2020.
[29] R. M. May, "Simple mathematical models with very complicated dynamics," *Nature*, vol. 261, no. 5560, pp. 459–467, 1976.
[30] X. Yuan, J. Zhao, Y. Yang, and Y. Wang, "Hybrid parallel chaos optimization algorithm with harmony search algorithm," *Appl. Soft Comput.*, vol. 17, pp. 12–22, Apr. 2014.
[31] X. Zhang and T. Feng, "Chaotic bean optimization algorithm," *Soft Comput.*, vol. 45, no. 8, pp. 67–77, 2016.
[32] G. Xu, F. Liu, C. Xiu, L. Sun, and C. Liu, "Optimization of hysteretic chaotic neural network based on fuzzy sliding mode control," *Neurocomputing*, vol. 189, pp. 72–79, May 2016.
[33] H. T. Yau, S. Y. Wu, C. L. Chen, and Y. C. Li, "Fractional-order chaotic self-synchronization-based tracking faults diagnosis of ball bearing systems," *IEEE Trans. Ind. Electron.*, vol. 63, no. 6, pp. 3824–3833, Jun. 2016.
[34] M. Mobini and M. R. Zahabi, "Chaos-based vehicle secret communication using master-slave lorenz synchronization system," *Int. J. Mechtron. Manuf. Syst.*, vol. 6, no. 19, pp. 2760–2766, 2016.
[35] M. Sakawa and K. Kato, "Genetic algorithms with double strings for 0–1 programming problems," *Eur. J. Oper. Res.*, vol. 144, no. 3, pp. 581–597, 2003.
[36] Z. Hao, T. N. Zhang, J. H. Shen, and Y. Li, "Research on decision-makings of structure optimization based on improved Tent PSO," *Control Decis.*, vol. 23, no. 8, pp. 857–862, 2008.
[37] M. A. Mellal and E. J. Williams, "Parameter optimization of advanced machining processes using cuckoo optimization algorithm and hoopoe heuristic," *J. Intell. Manuf.*, vol. 27, no. 5, pp. 927–942, Oct. 2016.
[38] Y. Chen, J. Yu, Y. Mei, S. Zhang, X. Ai, and Z. Jia, "Trajectory optimization of multiple quad-rotor UAVs in collaborative assembling task," *Chin. J. Aeronaut.*, vol. 29, no. 1, pp. 184–201, Feb. 2016.
[39] S. Chen, J. Montgomery, and A. Bolufé-Röhler, "Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution," *Int. J. Speech Technol.*, vol. 42, no. 3, pp. 514–526, Apr. 2015.
[40] Q. Tang, Y. Shen, C. Hu, J. Zeng, and W. Gong, "Swarm intelligence: Based cooperation optimization of multi-modal functions," *Cognit. Comput.*, vol. 5, no. 1, pp. 48–55, Mar. 2013.
[41] J. Li, T. Chen, T. Zhang, and Y. X. Li, "A cuckoo optimization algorithm using elite opposition-based learning and chaotic disturbance," *J. Softw. Eng.*, vol. 10, no. 1, pp. 16–28, Jan. 2016.
[42] W. H. Lim and N. A. Mat Isa, "Teaching and peer-learning particle swarm optimization," *Appl. Soft Comput.*, vol. 18, pp. 39–58, May 2014.
[43] W.-N. Chen, J. Zhang, Y. Lin, N. Chen, Z.-H. Zhan, H. S.-H. Chung, Y. Li, and Y.-H. Shi, "Particle swarm optimization with an aging leader and challengers," *IEEE Trans. Evol. Comput.*, vol. 17, no. 2, pp. 241–258, Apr. 2013.

[44] L. Yang, Z. Weitao, and L. Shuntian, "Deep convolution blind separation of acoustic signals based on joint diagonalization," *J. Electron. Inf. Technol.*, vol. 41, no. 12, pp. 2951–2956, 2019.
[45] T. W. Lee, M. Girolami, and T. J. Sejnowski, "Independent compontment analysis using an extended informax algorithm for mixed sub-Gaussian and super-Gaussian sources," *Neural Comput.*, vol. 11, no. 1, pp. 417–441, 1999.
[46] G. Wen, C. Zhang, Z. Lin, Z. Shang, H. Wang, and Q. Zhang, "Independent component analysis based on genetic algorithms," in *Proc. 10th Int. Conf. Natural Comput. (ICNC)*, Aug. 2014, pp. 214–218.
[47] D. S. Watkins, *Fundamentals of Matrix Computations*, 2nd ed. New York, NY, USA: Wiley, 2002, pp. 192–194.

WEILIN LUO was born in 1995. He received the B.Eng. degree in communication engineering from Xidian University, Xi'an, China, in 2018. He is currently pursuing the master's degree. He has published more than four journal articles, among them, one article were retrieved by SCI/EI. He is involved in the National Natural Science Foundation Project. His research interests include swarm intelligence, blind source separation, and intelligent optimization.

HONGBIN JIN was born in 1976. He received the Ph.D. degree in military intelligence from the Air Force Radar Academy, Wuhan, China, in 2011. He was a Professor. He has published more than 40 journal articles as the major author, among them, 12 articles were retrieved by SCI/EI. His current research interests include target recognition, data fusion, and system performance evaluation.

HAO LI was born in 1981. He received the B.Eng. degree in radar engineering and the M.Eng. degree in information and communication engineering from the Air Force Radar Academy, Wuhan, China, and the Ph.D. degree in electronic science and technology from Air Force Engineering University, Xi'an, China, in 2017. He has published more than 20 journal articles as the major author, among them, 12 articles were retrieved by SCI/EI. He has hosted and participated in several national natural science foundations. His current research interests include swarm intelligence, UAV swarm, intelligence systems, and signal processing.

XI FANG was born in 1981. He received the Ph.D. degree from the Wuhan University of Technology, Wuhan, China, in 2011. He was an Associate Professor. He has published more than 12 journal articles as the major author, among them, five articles were retrieved by SCI/EI. His current research interests include swarm intelligence and mathematical modeling in engineering.

RONGHUA ZHOU was born in 1996. He received the B.Eng. degree in radar engineering from the Air Force Early Warning Academy, Wuhan, China. He is currently pursuing the master's degree. He has published more than two journal articles as the major author, among them, one article were retrieved by SCI/EI. He is involved in the National Natural Science Foundation Project. His research interests include swarm intelligence, passive positioning, and intelligent optimization.