Received September 24, 2018, accepted October 14, 2018, date of publication October 24, 2018,
date of current version November 30, 2018.

*Digital Object Identifier 10.1109/ACCESS.2018.2876996*

# Chaos Enhanced Bacterial Foraging Optimization for Global Optimization

**QIAN ZHANG[1], HUILING CHEN[1], JIE LUO[1], YUETING XU[1], CHENGWEN WU[1], AND CHENGYE LI[2]**

[1]Department of Computer Science, Wenzhou University, Wenzhou 325035, China
[2]Department of Pulmonary and Critical Care Medicine, The First Affiliated Hospital, Wenzhou Medical University, Wenzhou 325000, China

Corresponding authors: Huiling Chen (chenhuiling.jlu@gmail.com) and Chengye Li (lichengye41@126.com)

This work was supported in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LY17F020012 and Grant LY13F020022, in part by the Science and Technology Plan Project, Wenzhou, China, under Grant ZG2017019 and Grant Y20170043, and in part by the Medical and Health Technology Projects of Zhejiang Province under Grant 2019315504.

**ABSTRACT** The recently developed Bacterial Foraging Optimization algorithm (BFO) is a nature-inspired optimization algorithm based on the foraging behavior of *Escherichia coli.* Due to its simplicity and effectiveness, BFO has been applied widely in many engineering and scientific fields. However, when dealing with more complex optimization problems, especially high dimensional and multimodal problems, BFO performs poorly in convergence compared to other nature-inspired optimization techniques. In this paper, we therefore propose an improved BFO, termed ChaoticBFO, which combines two chaotic strategies to achieve a more suitable balance between exploitation and exploration. Specifically, a chaotic initialization strategy is incorporated into BFO for bacterial population initialization to achieve acceleration throughout early steps of the proposed algorithm. Then, a chaotic local search with a 'shrinking' strategy is introduced into the chemotaxis step to escape from local optimum. The performance of ChaoticBFO was validated on 23 numerical well-known benchmark functions by comparing with 10 other competitive metaheuristic algorithms. Moreover, it was applied to two real-world benchmarks from IEEE CEC 2011. The experimental results demonstrate that ChaoticBFO is superior to its counterparts in both convergence speed and solution quality in most of the cases. This paper is of great significance for promoting the research, improvement and application of the BFO algorithm.

**INDEX TERMS** Bacterial foraging optimization, function optimization, chaotic local search, chaos theory.

## I. INTRODUCTION

In recent years, multiple nature-inspired optimization algorithms have been proposed, including Genetic Algorithm (GA) [1], [2], Particle Swarm Optimization (PSO) [3], [4], and Ant Colony Optimization (ACO) [5], [6], Fruit Fly optimization (FOA) [7], Grasshopper Optimization Algorithm (GOA) [8]. Based on the competitive-cooperative mechanism of E*scherichia coli.(E. coli)* in the foraging process, Passino [9] proposed a novel swarm intelligence algorithm called Bacterial Foraging Optimization algorithm (BFO), which consists mainly of four behaviors: chemotaxis, swarming, reproduction and elimination-dispersal.

Owing to its effectiveness, BFO has been widely and successfully applied in many fields such as optimal control [10], harmonic estimation [11], transmission loss reduction [12] and training neural network [13]. Dasgupta *et al.* [14] presented an algorithm based on BFO for the automatic detection of circular shapes from complicated and noisy images without using the conventional Hough transform method. Majhi *et al.* [15] used BFO to develop an efficient forecasting model for prediction of various stock indices. Sakthivel *et al.* [16] proposed an adaptive BFO for the design optimization of an energy efficient induction motor. Bhushan *and* Singh [17] introduced a BFO-based high performance speed control system for a DC motor that can track the desired trajectory with less computational time. Sanyal *et al.* [18] proposed an adaptive BFO for fuzzy entropy optimization that is suitable for thresholding-based image segmentation. Sathya *et al.* [19] proposed a multilevel thresholding approach for histogram-based image segmentation using a modified BFO. In [20], a new approach was proposed for edge detection using a combination of BFO and a probabilistic derivative technique derived from Ant Colony Systems. In [21], a modified algorithm called adaptive

2169-3536 © 2018 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

crossover BFO with improved recognition rate was proposed. Tang *et al.* [22] proposed a multilevel thresholding approach based on a modified BFO to enhance the applicability and practicality of optimal thresholding techniques.

However, when dealing with complex, high dimensional and multimodal problems, the original BFO is still easily trapped in local optima. Therefore, many improved BFO algorithms have been proposed in recent years. In [23], Jain et al. proposed a novel algorithm named Genetically Bacterial Swarm Optimization (GBSO), which hybridizes the best features of the three basic algorithms, GA, BFO and PSO. In [24], Tan et al. proposed an adaptive comprehensive learning BFO. In [25], a novel method called Chemotaxis Differential Evolution Optimization Algorithm (CDEOA) was proposed, which augments BFO with conditional introduction of Differential Evolution (DE) and Random Search operators. Yang *et al.* [26] proposed a new swarm intelligence algorithm based on BFO for structural learning of Bayesian networks. To address issues on real-parameter single objective optimization problems, Yang *et al.* [27] proposed a new BFO using newly designed chemotaxis and conjugation strategies. In [28], an effective BFO was proposed using a gravitational search strategy incorporated into the chemotaxis step and a swarm diversity strategy integrated into the reproduction step.

Although the above schemes have improved the original BFO to some extent, for the optimization problems of high dimensional functions, they are only effective for some high dimensional functions and have such defects as low universality.

In this paper, an improved BFO, termed ChaoticBFO, is proposed, which introduces two strategies into the original BFO to improve its effectiveness. Specifically, a chaotic initialization strategy is incorporated into BFO for bacterial population initialization to achieve acceleration throughout early steps of the proposed algorithm. Then, a chaotic local search with a 'shrinking' strategy is introduced to the chemotaxis step, enabling ChaoticBFO to explore a huge search space in the early run phase to avoid premature convergence while exploiting a small region in the later run phase to refine the final solutions. To the best of our knowledge, there are few papers applying chaos theory to BFO. In [29], an adaptive chaotic BFO (ACBFO) was proposed for data clustering problem. In [30], a chaotic local search based BFO was proposed, which introduces the DE operator and the chaotic search operator into the chemotaxis step of the original BFO.

In order to further improve the local searching ability of individuals, both our proposed algorithm and the algorithm described above in [30] introduce the chaotic local search into the chemotaxis step and use the logistic chaotic mapping to generate the chaotic sequence. However, they differ greatly in how to use the chaotic local search in the chemotaxis step. Specifically, the above algorithm implements the chaotic search for the entire population on the position of the current bacterium, whereas our proposed algorithm applies a chaotic local search with a 'shrinking' strategy to the optimal position in the current bacterial population.

ChaoticBFO is proposed to achieve the trade-off between exploration and exploitation and offer significantly better results than the original BFO in terms of solution accuracy and convergence speed. The performance of ChaoticBFO was validated on 23 well-known numerical benchmark functions by comparing with 10 other typical metaheuristic algorithms. It was also applied to two real-world benchmarks from IEEE CEC 2011. The results show that ChaoticBFO is more effective than the original BFO and its competitors in both convergence speed and solution quality in most of the experiments. The main contributions of this study are as follows:

a) In order to achieve a more suitable balance between exploitation and exploration for BFO, we have incorporated two chaotic strategies (chaotic initialization strategy and chaotic local search with a 'shrinking' strategy) into BFO.

b) The proposed method was validated on 23 numerical well-known benchmark functions and the experimental results show that the proposed method outperforms its competitors in terms of convergence speed and solution quality in most of the cases. Moreover, it was also successfully applied to two real-world engineering problems from IEEE CEC 2011 benchmark set.

c) This study is of great significance to analyze, improve and expand the application of the BFO algorithm. Moreover, it is of great value to the study of swarm intelligence optimization algorithms.

The remainder of this paper is organized as follows. Section 2 explains the original BFO. The chaos theory is described in Section 3. In Section 4, ChaoticBFO is proposed. In Section 5, the comprehensive results are reported and analyzed. Section 6 provides a comparative study on two real-world benchmarks from IEEE CEC 2011. Finally, conclusions and future work are summarized in Section 7.

## II. BACTERIAL FORAGING OPTIMIZATION ALGORITHM (BFO)

BFO [9], proposed by Passino in 2002, is a novel swarm intelligence algorithm based on the competitive-cooperative mechanism of *E.coli* in the foraging process. Simulating the foraging behavior of *E.coli*, BFO consists mainly of four behaviors: chemotaxis, swarming, reproduction, and elimination-dispersal. Suppose we seek to find the optimal solution to a problem, a virtual bacterium is actually one trial solution (regarded as a search-agent) that moves on the functional surface (see Figure 1) to locate the global optimum. As shown in Figure 1, the red circle represents the best bacterium to deal with the case.

The four main behaviors of bacteria are as follows:

**1) Chemotaxis:** Chemotaxis operation is the core of the algorithm, which simulates *E.coli*'s foraging behavior of swimming and tumbling. In areas poorer in food, bacteria tumble more frequently, whereas bacteria swim in areas where food is more abundant. Figure 2 depicts how clockwise and counterclockwise movement of a bacterium takes

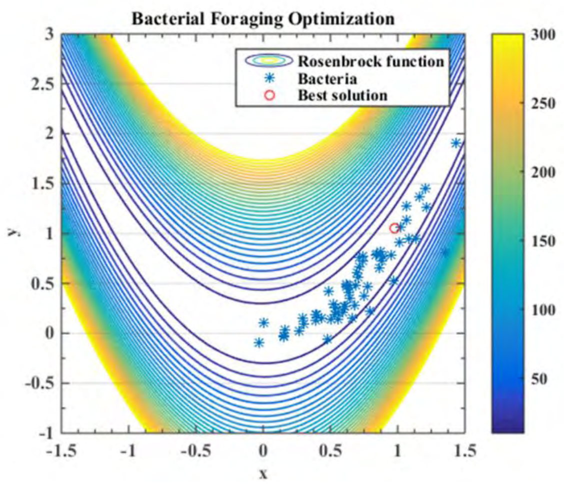


**FIGURE 1.** A bacterial swarm on a unimodal benchmark function surface.

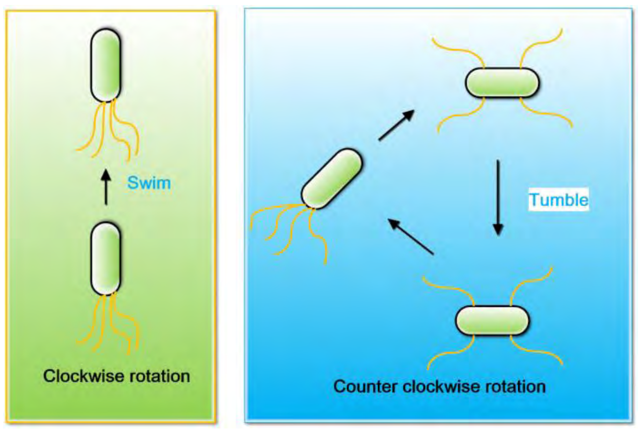


**FIGURE 2.** Swimming and tumbling of a bacterium.

place in a nutrient solution. The chemotaxis operation of the $i$th bacterium can be represented as:

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i) * dct_i$$
$$dct_i = \frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \tag{1}$$

Where the $\theta^i(j,k,l)$ represents the $i$th bacterium at the $j$th chemotaxis, $k$th reproductive, and $l$th elimination-dispersal step. $C(i)$ is the trend step length of bacterium $i$ in a random direction ($dct_i$). $\Delta$ is a random vector between $-1$ and 1.

**2) Swarming:** In the chemotaxis of bacteria in the foraging process, there are both attraction and repulsion among individual bacteria. Bacteria generate attraction information to allow individual bacteria to travel to the center of the population, bringing them together. However, at the same time, individual bacteria are kept at a distance based on their respective repulsion information.

**3) Reproduction:** According to the natural mechanism of survival of the fittest, after some time, bacteria with weak ability to seek food will eventually be eliminated, and bacteria with strong feeding ability will breed offspring to maintain the size of the population. By simulating this phenomenon, a reproduction operation is proposed. In a $S$-sized population, $S/2$ bacteria with poor fitness are eliminated and $S/2$ individuals with higher fitness self-replicate after the bacteria perform the chemotaxis operator.

**4) Elimination-Dispersal:** In the foraging process, unexpected situations can not be ruled out such as the death of bacteria or their migration to new areas. To simulate this phenomenon, an elimination-dispersal operation is proposed. This operation occurs with a certain probability *Ped*. When a bacterial individual satisfies the probability *Ped*, it dies and randomly generates a new individual anywhere in the solution space. This new individual may be different from the original bacterium; however, it can jump out of the local optimal solution, promoting the search for the global optimal solution.

## III. CHAOS THEORY

Over the last decade, much progress has been made in the chaos theory. It has been used widely in different fields of science such as chaos control [31], feature selection [32], and parameter optimization [33]. Chaotic sequences have three basic dynamic properties: sensitive dependence on initial conditions, randomicity, and ergodicity. Chaotic sequences have been applied to various metaheuristic optimization algorithms in recent years. In [34], a novel GA with chaotic mutation was proposed by replacing the Gaussian mutation operator in real-coded GA with a chaotic mapping. Mingjun and Huanwen [35] introduced chaotic initialization and chaotic sequences into Simulated Annealing (SA) instead of Gaussian distribution. In [36], Alatas *et al.* proposed new PSO methods that use chaotic maps for parameter adaptation. In order to improve the overall searching performance of basic algorithms, other metaheuristic optimization algorithms also use the chaos theory, including Moth-Flame Optimization (MFO) [37], Firefly Algorithm (FA) [38], Artificial Bee Colony (ABC) [39], Biogeography-Based Optimization (BBO) [40], Krill Herd (KH) [41], Water Cycle Algorithm(WCA) [42], and Grey Wolf Optimizer (GWO) [43].

If chaos variables are used in the search, more advantage is gained over random search. The basic idea of chaos optimization is 1) to introduce chaos state into optimization variables by using a similar carrier method, 2) to magnify the traversal range of chaotic motions to the range of optimization variables, and 3) to use the chaos variables to search to make the search more effective.

The proposed method generates a chaotic sequence using logistic mapping as shown in equation (2):

$$ch_{i+1} = \mu ch_{i+1} * (1 - ch_i) \quad i = 1, \ldots, S-1 \tag{2}$$

Where $\mu$ is the control parameter, take $\mu = 4$. Set $0 < ch_1 < 1$, and $ch_1 \neq 0.25, 0.5, 0.75, 1$. Not difficult to prove that when $\mu = 4$, the system is completely in chaos. $S$ is the number of individuals.

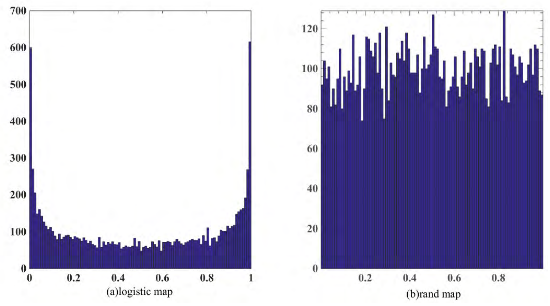


**FIGURE 3. Histogram distribution graphs of (a) the logistic map and (b) the random map.**

Figure 3 shows the histogram distribution graphs of (a) the logistic map (we set $\mu = 4$ and the initial value $ch_1 = 0.4501$) and (b) the random map (as the iteration increases, the data generate a random value between 0 and 1 with uniform distribution) with $10^4$ iterations. As shown in Figure 3, the two maps are located in the interval (0, 1). From these figures, it can be seen that the logistic map and the random map give different distributions. Specifically, the logistic map has much higher possibility to generate values near 0 and 1, whereas the probability that the random map generates different random values between 0 and 1 is almost the same. This means the logistic map provides a different search range from the random map in local search and explains why the logistic map makes local search faster than the random map.

Chaotic search usually works well in local optimization for its ergodicity and randomicity [44], [45]. However, its performance decreases when it explores a large search space. To overcome this shortcoming, chaotic local search was introduced. Due to the randomicity of chaotic local search, the search process can avoid premature convergence and local optima stagnation. In [46], chaotic local search was incorporated into PSO to construct a chaotic PSO (CPSO), where the parallel population-based evolutionary searching ability of PSO and chaotic searching behavior are reasonably combined. Jia *et al.* [47] proposed an effective memetic DE algorithm called DECLS, which utilizes a chaotic local search with a ‘shrinking’ strategy. In [48], a chaotic local search was integrated into the reduced Symbiotic Organisms Search (SOS) to form chaotic SOS (CSOS) for improving solution accuracy and convergence mobility.

## IV. PROPOSED METHOD

Our proposed algorithm ChaoticBFO combines two chaotic strategies. First, a chaotic initialization strategy is incorporated into BFO for bacterial population initialization. Then, a chaotic local search with a ‘shrinking’ strategy is introduced into the chemotaxis step. This proposed ‘shrinking’ strategy is a modified version of the method described in [47].

### A. CHAOTIC INITIALIZATION

*Step 1:* Through the chaos mechanism, the chaotic sequence is generated by using the logistic map generated by the Eq. (2), and the position $P$ of the initial bacterial population is mapped into the chaotic sequence to generate the position $PCh$ of the corresponding chaotic initial bacterial population. As shown in Eq. (3):

$$PCh = ch_i * P \quad i = 1, \ldots, S \tag{3}$$

*Step 2:* From the initial position $P$ of the bacterial population and its corresponding position $PCh$ of the chaotic initial bacterial population, $S$ superior individuals are selected as the initial solutions of bacterial populations. Loop execution $(S - 1)$ times.

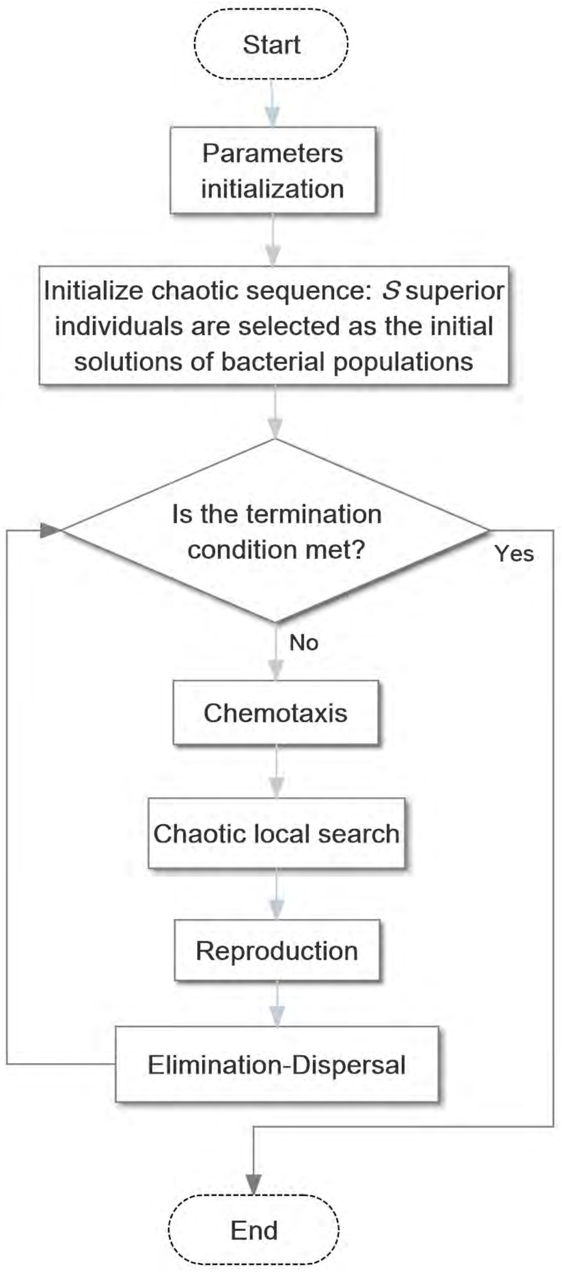

Start
Parameters initialization
Initialize chaotic sequence: S superior individuals are selected as the initial solutions of bacterial populations
Is the termination condition met?
Yes
No
Chemotaxis
Chaotic local search
Reproduction
Elimination-Dispersal
End

**FIGURE 4. Overall procedure of ChaoticBFO.**

### *B. CHAOTIC LOCAL SEARCH WITH A 'SHRINKING' STRATEGY*

*Step 1:* Before the chemotaxis operation of the $i$th bacterium, place the $i$th bacterium at the position of $j$th chemotaxis, $k$th reproductive, and $l$th elimination-dispersal as the optimal position*gbest* in the current bacterial population.

*Step 2:* The chaotic variable $ch_i$ generated in Eq. (2) is mapped into the chaotic vector $CH_i$ in the domain of definition [$lb$, $ub$], as shown in Eq. (4):

$$CH_i = lb + ch_i * (ub - lb) \quad i = 1, \ldots, S \tag{4}$$

Where $lb$ and $ub$ represent the lower and upper bounds of the initial solution, respectively.

Step 3: The chaotic vector $CH_i$ is linearly combined with the optimal position *gbest* to generate the candidate bacterial position *sol*, as shown in Eq. (5):

$$sol = (1 - setCan) * gbest + setCan * CH_i \quad i = 1, \ldots, S \tag{5}$$

Where *setCan* is the contraction factor, which is determined by Eq. (6):

$$setCan = exp\left(-Intertime \big/ Max\_iteration\right) \tag{6}$$

Where *Max_iteration* represents the maximum number of iterations of the algorithm and *Intertime* represents the current iteration number of the algorithm.

From Eq. (6), it can be seen that the contraction factor *setCan* decreases as the number of iterations increases. As shown in Eq. (5), the smaller the value of *setCan* is, the smaller the range of chaos search is. In the early iteration, *setCan* is larger, which helps to expand the search range and increase the diversity of the population. At the later stage of iteration, *setCan* is smaller, which helps to converge to the global optimal solution.

*Step 4:* If the candidate bacterial position *sol* is better than *Gbest* (*Gbest* represents the current optimal fitness function of the bacterial population), the fitness of *sol* is recorded as *Gbest* and *gbest* (*gbest* described in Step 1 represents the optimal position in the current bacterial population) is updated as *sol*. If the chaotic sequence length reaches $S$, local search ends; otherwise skip to Step 2 to continue execution.

The main steps of ChaoticBFO are described in detail as follows:

The flowchart of ChaoticBFO is shown in Figure 4.

**The Main Steps of ChaoticBFO**

```
Begin
Step 1: Parameter Initialization. Initialize the number of dimensions in the search space p, the swarm size of the population S, the number of chemotaxis steps Nc, the swimming length Ns, the number of reproduction steps Nre, the number of elimination-dispersal events Ned, the elimination-dispersal probability Ped, the size of the step C(i) taken in the random direction specified by the tumble.
Step 2: Population Initialization. As described in above Chaotic strategy, position P of the initial bacterial population is mapped into the chaotic sequence generated by the Eq. (2) to calculate the position PCh of the corresponding chaotic initial bacterial population according to Eq. (3). From the initial position P of the bacterial population and its corresponding position PCh of the chaotic initial bacterial population, S superior individuals are selected as the initial solutions of bacterial populations. Loop execution (S-1) times.
Step 3: For ell=1:Ned /*Elimination and dispersal loop*/
  For K=1:Nre /*Reproduction loop*/
    For j=1:Nc /* chemotaxis loop*/
        Intertime=Intertime+1;
        For i=1:s
          J(i,j,K,ell)=fobj(P(:,i,j,K,ell));
          Jlast=J(i,j,K,ell);
          gbest(1,:)= P(:,i,j,K,ell);
          Perform chaotic local search described in 4.2
          Tumble according to Eq.(1)
          /*Swim*/
          m=0;
          While m<Ns
            m=m+1;
            If J(i,j+1,K,ell)<Jlast
                Jlast=J(i,j+1,K,ell);
                Tumble according to Eq.(1)
            Else
      m=Ns ;
            End
          End
        End /*Go to next bacterium*/
      End /*Go to next chemotaxis*/
      /*Reproduction*/
      Jhealth=sum(J(:,:,K,ell),2);
      [Jhealth, sortind]=sort(Jhealth);
      /* Rearrange the bacterial population*/
      P(:,:,1,K+1,ell)=P(:,sortind,Nc+1,K,ell);
      /*Split the bacterium (reproduction)*/
    For i=1:Sr
          P(:,i+Sr,1,K+1,ell)=P(:,i,1,K+1,ell);
        End
      End /*Go to next reproduction*/
      /*Elimination-Dispersal*/
      For m=1:s
        If Ped>rand
          Reinitialize bacterium m
        End
      End
    End /* Go to next Elimination-Dispersal*/
End
```

## V. EXPERIMENTAL RESULTS AND DISCUSSION

### *A. BENCHMARK FUNCTION VALIDATION*

In order to test the performance of ChaoticBFO, 23 classical benchmark functions were used in this experiment. These benchmark functions are unimodal, multimodal, or fixed-dimension multimodal, as listed in Tables 1–3 where Dim indicates the dimension of the function, Range the boundary of the function's search space, and $f_{\min}$ the optimum value.

**TABLE 1. Unimodal benchmark functions.**

| Function | Dim | Range | $f_{\min}$ |
|---|---|---|---|
| $f_1(x)=\sum_{i=1}^{n} x_i^2$ | 50 | [-100, 100] | 0 |
| $f_2(x)=\sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$ | 50 | [-10, 10] | 0 |
| $f_3(x)=\sum_{i=1}^{n}(\sum_{j-1}^{i} x_j)^2$ | 50 | [-100, 100] | 0 |
| $f_4(x)=\max_i\{\lvert x_i \rvert, 1\le i\le n\}$ | 50 | [-100, 100] | 0 |
| $f_5(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$ | 50 | [-30, 30] | 0 |
| $f_6(x)=\sum_{i=1}^{n}([x_i+0.5])^2$ | 50 | [-100, 100] | 0 |
| $f_7(x)=\sum_{i=1}^{n} i x_i^4 + random[0,1)$ | 50 | [-1.28, 1.28] | 0 |

**TABLE 2. Multimodal benchmark functions.**

| Function | Dim | Range | $f_{\min}$ |
|---|---|---|---|
| $f_8(x)=\sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$ | 50 | [-500,500] | $-418.9829\times 5$ |
| $f_9(x)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10]$ | 50 | [-5.12,5.12] | 0 |
| $f_{10}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$ | 50 | [-32,32] | 0 |
| $f_{11}(x)=\frac{1}{4000}\sum_{i=1}^{n} x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$ | 50 | [-600,600] | 0 |
| $f_{12}(x)=\frac{\pi}{n}\{10\sin(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2$ $+\sum_{i=1}^{n} u(x_i,10,100,4)$ $y_i=1+\frac{x_i+1}{4}$ $u(x_i,a,k,m)\begin{cases} k(x_i-a)^m & x_i>a \\ 0 & -a<x_i<a \\ k(-x_i-a)^m & x_i<-a \end{cases}$ | 50 | [-50,50] | 0 |
| $f_{13}(x)=0.1\begin{Bmatrix} \sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2[1+\sin^2(3\pi x_i+1)] \\ +(x_n-1)^2[1+\sin^2(2\pi x_n)] \end{Bmatrix}$ $+\sum_{i=1}^{n} u(x_i,5,100,4)$ | 50 | [-50,50] | 0 |

## B. SCALABILITY TEST

Through scalability testing, we can further explore the impact of changes in dimension on the quality of solutions and the efficacy of optimizers. For this purpose, 4 dimensions of F1-F23 functions were measured here: 10, 30, 100, and 200.

**TABLE 3. Fixed-dimension multimodal benchmark functions.**

| Function | Dim | Range | $f_{\min}$ |
|---|---|---|---|
| $f_{14}(x)=\left(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right)^{-1}$ | 2 | [-65,65] | 1 |
| $f_{15}(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_i x_2)}{b_i^2+b_i x_3+x_4}\right]^2$ | 4 | [-5, 5] | 0.00030 |
| $f_{16}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$ | 2 | [-5,5] | -1.0316 |
| $f_{17}(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos x_1+10$ | 2 | [-5,5] | 0.398 |
| $f_{18}(x)=\left[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)\right]$ $\times\left[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)\right]$ | 2 | [-2,2] | 3 |
| $f_{19}(x)=-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2\right)$ | 3 | [1,3] | -3.86 |
| $f_{20}(x)=-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2\right)$ | 6 | [0,1] | -3.32 |
| $f_{21}(x)=-\sum_{i=1}^{5}\left[(X-a_i)(X-a_i)^{\mathrm{T}}+c_i\right]^{-1}$ | 4 | [0,10] | -10.1532 |
| $f_{22}(x)=-\sum_{i=1}^{7}\left[(X-a_i)(X-a_i)^{\mathrm{T}}+c_i\right]^{-1}$ | 4 | [0,10] | -10.4028 |
| $f_{23}(x)=-\sum_{i=1}^{10}\left[(X-a_i)(X-a_i)^{\mathrm{T}}+c_i\right]^{-1}$ | 4 | [0,10] | -10.5363 |

All conditions were the same, and the population size and maximum number of iterations were set to 50 and 500, respectively. Table 4 presents the average error (Avg) and standard deviation (Stdv) values of the two algorithms in dealing with F1-F23 functions in 4 dimensions. In Table 4, each average error (Avg) which is smaller than another is marked in bold face, indicating the algorithm has better performance.

As can be seen from Table 4, ChaoticBFO performs better than BFO on most functions in all dimensions and still returns the results with obvious advantage at higher dimensions. Since it is more difficult to deal with F1-F23 functions in higher dimensions, the average error (Avg) also increases as the dimension increases. In the four different dimensions, ChaoticBFO obtains more accurate results than BFO on all functions from F1 to F15.

## C. COMPARATIVE STUDY

To verify its effectiveness, ChaoticBFO was compared with four well-known algorithms: PSO[3], Bat Algorithm (BA)

**TABLE 4. Scalability results.**

| | Dim | 10 | | 30 | | 100 | | 200 | |
|---|---|---|---|---|---|---|---|---|---|
| Id. | Metric | ChaoticBFO | BFO | ChaoticBFO | BFO | ChaoticBFO | BFO | ChaoticBFO | BFO |
| F1 | Avg | **1.25E-04** | 2.93E-03 | **1.35E-04** | 6.99E-03 | **4.27E-04** | 1.79E-02 | **6.47E-04** | 3.11E-02 |
| | Stdv | 1.30E-04 | 1.76E-03 | 1.21E-04 | 2.55E-03 | 4.86E-04 | 5.20E-03 | 8.97E-04 | 7.25E-03 |
| F2 | Avg | **5.84E-03** | 1.31E-01 | **1.73E-02** | 3.30E-01 | **3.41E-02** | 9.35E-01 | **5.73E-02** | 1.74E+00 |
| | Stdv | 4.14E-03 | 2.03E-02 | 1.23E-02 | 6.74E-02 | 2.20E-02 | 1.87E-01 | 3.11E-02 | 2.43E-01 |
| F3 | Avg | **1.28E-12** | 3.43E-12 | **3.23E-13** | 9.59E-13 | **7.32E-14** | 4.99E-13 | **4.82E-14** | 5.29E-13 |
| | Stdv | 2.36E-12 | 5.46E-12 | 6.44E-13 | 1.75E-12 | 1.03E-13 | 1.38E-12 | 7.60E-14 | 1.24E-12 |
| F4 | Avg | **4.54E-03** | 3.04E-02 | **4.19E-03** | 2.94E-02 | **3.23E-03** | 2.29E-02 | **3.16E-03** | 1.92E-02 |
| | Stdv | 2.76E-03 | 8.48E-03 | 2.93E-03 | 4.15E-03 | 2.08E-03 | 3.43E-03 | 1.95E-03 | 2.17E-03 |
| F5 | Avg | **2.84E-01** | 6.55E+04 | **7.31E-01** | 6.55E+04 | **7.64E-01** | 6.55E+04 | **1.44E+00** | 6.55E+04 |
| | Stdv | 2.28E-01 | 0 | 5.66E-01 | 0 | 7.16E-01 | 0 | 1.28E+00 | 0 |
| F6 | Avg | **3.07E-04** | 1.15E-02 | **7.50E-04** | 3.02E-01 | **1.18E-03** | 1.08E+01 | **1.90E-03** | 3.35E+01 |
| | Stdv | 4.87E-04 | 3.36E-03 | 7.19E-04 | 6.35E-02 | 1.08E-03 | 6.87E-01 | 1.70E-03 | 8.61E-01 |
| F7 | Avg | **2.42E-04** | 3.34E-03 | **3.79E-04** | 2.33E-03 | **3.66E-04** | 1.98E-03 | **3.52E-04** | 2.05E-03 |
| | Stdv | 1.82E-04 | 2.23E-03 | 2.42E-04 | 1.26E-03 | 2.50E-04 | 1.45E-03 | 2.97E-04 | 1.50E-03 |
| F8 | Avg | **-4.19E+03** | -1.62E+03 | **-1.26E+04** | -2.61E+03 | **-4.19E+04** | -4.82E+03 | **-8.38E+04** | -6.41E+03 |
| | Stdv | 1.76E-01 | 2.64E+02 | 5.28E-01 | 3.80E+02 | 1.64E+00 | 8.53E+02 | 2.75E+00 | 1.24E+03 |
| F9 | Avg | **-8.97E+01** | -8.95E+01 | -2.89E+02 | -2.89E+02 | **-9.88E+02** | -9.87E+02 | **-1.99E+03** | -1.98E+03 |
| | Stdv | 8.30E-02 | 3.08E-01 | 1.83E-01 | 4.89E-01 | 5.84E-01 | 8.76E-01 | 9.37E-01 | 1.78E+00 |
| F10 | Avg | **-2.13E+04** | -2.08E+04 | **-9.81E+12** | -9.10E+12 | **-2.16E+43** | -1.87E+43 | **-4.77E+86** | -4.07E+86 |
| | Stdv | 1.67E+02 | 5.44E+02 | 2.62E+11 | 5.97E+11 | 1.11E+42 | 2.07E+42 | 4.57E+85 | 7.16E+85 |
| F11 | Avg | **1.08E-04** | 1.39E-03 | **1.10E-04** | 3.17E-03 | **1.06E-04** | 8.95E-03 | **6.71E-05** | 1.26E-02 |
| | Stdv | 1.02E-04 | 6.31E-04 | 1.02E-04 | 1.23E-03 | 1.38E-04 | 2.99E-03 | 6.51E-05 | 3.55E-03 |
| F12 | Avg | **2.15E-11** | 1.49E-10 | **7.96E-12** | 7.60E-11 | **1.53E-12** | 1.30E-11 | **8.80E-13** | 2.71E-11 |
| | Stdv | 3.77E-11 | 2.32E-10 | 1.23E-11 | 9.51E-11 | 2.30E-12 | 1.97E-11 | 1.39E-12 | 3.46E-11 |
| F13 | Avg | **2.42E-06** | 6.33E-02 | **5.19E-05** | 9.92E-02 | **9.05E-05** | 9.92E-02 | **1.99E-04** | 9.92E-02 |
| | Stdv | 1.04E-05 | 2.58E-02 | 1.68E-04 | 5.77E-11 | 1.85E-04 | 1.37E-11 | 5.18E-04 | 7.54E-12 |
| F14 | Avg | **9.98E-01** | 1.89E+00 | **9.98E-01** | 1.89E+00 | **9.98E-01** | 2.45E+00 | **9.98E-01** | 2.22E+00 |
| | Stdv | 1.48E-10 | 1.23E+00 | 2.65E-10 | 1.15E+00 | 9.33E-11 | 1.81E+00 | 1.63E-10 | 1.75E+00 |
| F15 | Avg | **3.50E-04** | 4.83E-04 | **3.49E-04** | 5.05E-04 | **3.46E-04** | 5.23E-04 | **3.47E-04** | 4.78E-04 |
| | Stdv | 2.50E-05 | 9.62E-05 | 1.99E-05 | 1.20E-04 | 1.93E-05 | 1.38E-04 | 1.89E-05 | 7.37E-05 |
| F16 | Avg | -1.03E+00 | -1.03E+00 | -1.03E+00 | -1.03E+00 | -1.03E+00 | -1.03E+00 | -1.03E+00 | -1.03E+00 |
| | Stdv | 1.93E-06 | 3.58E-06 | 2.38E-06 | 3.14E-06 | 2.65E-06 | 4.32E-06 | 2.71E-06 | 2.67E-06 |
| F17 | Avg | 3.98E-01 | 3.98E-01 | 3.98E-01 | 3.98E-01 | 3.98E-01 | 3.98E-01 | 3.98E-01 | 3.98E-01 |
| | Stdv | 6.91E-07 | 1.23E-06 | 1.40E-06 | 1.12E-06 | 1.27E-06 | 1.85E-06 | 1.43E-06 | 1.49E-06 |
| F18 | Avg | 3.00E+00 | 3.00E+00 | 3.00E+00 | 3.00E+00 | 3.00E+00 | 3.00E+00 | 3.00E+00 | 3.00E+00 |
| | Stdv | 1.97E-04 | 3.10E-04 | 1.61E-04 | 1.33E-04 | 1.61E-04 | 1.89E-04 | 1.65E-04 | 1.90E-04 |
| F19 | Avg | -3.86E+00 | -3.86E+00 | -3.86E+00 | -3.86E+00 | -3.86E+00 | -3.86E+00 | -3.86E+00 | -3.86E+00 |
| | Stdv | 2.44E-04 | 5.58E-04 | 2.36E-04 | 4.47E-04 | 1.98E-04 | 4.44E-04 | 3.19E-04 | 3.31E-04 |
| F20 | Avg | -3.28E+00 | -3.28E+00 | -3.28E+00 | -3.29E+00 | -3.29E+00 | -3.29E+00 | -3.27E+00 | -3.28E+00 |
| | Stdv | 4.32E-02 | 2.23E-02 | 3.98E-02 | 1.31E-02 | 3.41E-02 | 1.45E-02 | 4.70E-02 | 1.54E-02 |
| F21 | Avg | **-1.02E+01** | -1.01E+01 | **-1.02E+01** | -1.01E+01 | **-1.02E+01** | -1.01E+01 | **-1.02E+01** | -1.01E+01 |
| | Stdv | 4.65E-04 | 1.00E-02 | 2.69E-04 | 9.82E-03 | 2.14E-04 | 1.16E-02 | 2.34E-04 | 1.26E-02 |
| F22 | Avg | -1.04E+01 | -1.04E+01 | -1.04E+01 | -1.04E+01 | **-1.04E+01** | -1.02E+01 | -1.04E+01 | -1.04E+01 |
| | Stdv | 2.30E-04 | 1.02E-02 | 3.18E-04 | 7.83E-03 | 1.59E-04 | 9.60E-01 | 2.08E-04 | 9.67E-03 |
| F23 | Avg | -1.05E+01 | -1.05E+01 | -1.05E+01 | -1.05E+01 | -1.05E+01 | -1.05E+01 | -1.05E+01 | -1.05E+01 |
| | Stdv | 2.56E-04 | 1.02E-02 | 3.14E-04 | 1.02E-02 | 2.65E-04 | 1.01E-02 | 2.76E-04 | 8.76E-03 |

[49], FA [50], and Flower Pollination Algorithm (FPA) [51]. In addition, to verify its efficiency, ChaoticBFO was also compared with the original BFO [9] and five other meta-heuristic algorithms: MFO [52], Dragonfly Algorithm (DA) [53], Sine Cosine Algorithm (SCA) [54], Grasshopper

**TABLE 5. Parameter settings for the algorithms used.**

| Method | Population size | Maximum generation | Other parameters |
|---|---|---|---|
| **BFO** | 50 | 500 | $\Delta \in$[-1, 1] |
| **BA** | 50 | 500 | *Q Frequency*$\in$[0 2]; *A Loudness*: 0.5; *r Pulse rate*: 0.5 |
| **DA** | 50 | 500 | $w\in$[0.9 0.2]; $s$ = 0.1; a = 0.1; $c$ = 0.7; $f$ = 1; $e$ = 1 |
| **FA** | 50 | 500 | $\beta_0$ =1; $\alpha$ $\in$[0 1]; $\gamma$ =1 |
| **FPA** | 50 | 500 | *switch probability p*=0.8; $\lambda$ =1.5 |
| **GOA** | 50 | 500 | *cMax*=1; *cMin*=0.00004 |
| **MFO** | 50 | 500 | $a\in$[-1 -2]; $b$=1 |
| **PSO** | 50 | 500 | *inertial weight*=1; $c_1$=2; $c_2$=2 |
| **SCA** | 50 | 500 | $a$=2 |
| **SSA** | 50 | 500 | $c_1\in$[0 1]; $c_2\in$[0 1] |

Optimization Algorithm (GOA) [55], Salp Swarm Algorithm (SSA) [56]. Parameter values for the above algorithms were set according to their original articles. Table 5 shows the parameter values for each algorithm. Moreover, to achieve unbiased results, 30 independent runs were performed on each benchmark function. In all the experiments, the population size and maximum number of iterations were set to 50 and 500, respectively.

The average results (Avg), standard deviation (Stdv), and overall rank of different algorithms in dealing with F1-F23 problems are presented in Table 6, Table 7, and Table 8. It should be noted that the ranking was based on the average results (Avg) of 30 independent runs. The convergence behavior of the algorithms in solving all the problems was also compared based on the logarithmic scale diagram in Figures 5-7. The results on the unimodal F1-F7 are listed in Table 6. It can be seen from Table 6 that ChaoticBFO outperforms all the other algorithms in dealing with all the cases (F1-F7). In addition, the ranking results also show that ChaoticBFO generates the best solution among all the algorithms.

According to the convergence curves shown in Figure 5, it can be seen that the convergence speed of ChaoticBFO is faster than that of the other algorithms in all cases. In dealing with F1, F2, F4, F6 and F7, ChaoticBFO converges very fast throughout early steps, whereas BA, DA, FA, FPA, GOA, SCA, SSA and PSO fail to improve the quality of solutions in solving F3 despite more explorative steps. For F2, although the other algorithms have the effect of convergence, ChaoticBFO has the fastest convergence speed. Moreover, although SSA converges fast in the later stages in tackling F6, ChaoticBFO obtains the optimal value.

**TABLE 6. Results on unimodal benchmark functions (F1-F7).**

| F | | ChaoticBFO | BFO | BA | DA | FA | FPA | GOA | MFO | PSO | SCA | SSA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Avg | 2.69E-04 | 1.14E-02 | 4.98E+01 | 3.52E+03 | 4.82E-02 | 5.50E+03 | 2.14E+02 | 3.85E+03 | 3.59E+02 | 5.42E+02 | 2.17E-03 |
| F1 | Stdv | 2.29E-04 | 3.99E-03 | 4.22E+00 | 1.74E+03 | 1.18E-02 | 9.55E+02 | 9.45E+01 | 5.22E+03 | 3.72E+01 | 7.40E+02 | 3.40E-03 |
| | Rank | 1 | 3 | 5 | 9 | 4 | 11 | 6 | 10 | 7 | 8 | 2 |
| | Avg | 2.47E-02 | 5.13E-01 | 3.95E+04 | 2.96E+01 | 9.96E-01 | 3.02E+02 | 2.19E+01 | 6.91E+01 | 3.56E+07 | 3.10E-01 | 5.01E+00 |
| F2 | Stdv | 1.34E-02 | 9.56E-02 | 2.10E+05 | 1.40E+01 | 1.61E-01 | 8.37E+02 | 1.91E+01 | 3.17E+01 | 1.17E+08 | 2.73E-01 | 1.97E+00 |
| | Rank | 1 | 3 | 10 | 7 | 4 | 9 | 6 | 8 | 11 | 2 | 5 |
| | Avg | 1.72E-13 | 1.45E-12 | 7.85E+02 | 3.84E+04 | 9.06E+03 | 1.04E+04 | 1.02E+04 | 6.06E+04 | 3.18E+03 | 4.35E+04 | 4.51E+03 |
| F3 | Stdv | 2.85E-13 | 2.81E-12 | 2.32E+02 | 2.39E+04 | 2.38E+03 | 2.41E+03 | 5.31E+03 | 2.22E+04 | 7.17E+02 | 1.06E+04 | 3.06E+03 |
| | Rank | 1 | 2 | 3 | 9 | 6 | 8 | 7 | 11 | 4 | 10 | 5 |
| | Avg | 4.94E-03 | 2.69E-02 | 1.20E+01 | 2.81E+01 | 1.24E+00 | 3.73E+01 | 1.64E+01 | 8.05E+01 | 7.44E+00 | 6.20E+01 | 1.49E+01 |
| F4 | Stdv | 2.70E-03 | 3.82E-03 | 4.24E+00 | 9.32E+00 | 9.84E-01 | 4.33E+00 | 3.19E+00 | 6.53E+00 | 6.85E-01 | 8.13E+00 | 2.80E+00 |
| | Rank | 1 | 2 | 5 | 8 | 3 | 9 | 7 | 11 | 4 | 10 | 6 |
| | Avg | 6.28E-01 | 6.55E+04 | 2.14E+04 | 1.17E+06 | 1.54E+02 | 1.49E+06 | 2.24E+04 | 6.00E+06 | 7.66E+05 | 2.12E+06 | 3.78E+02 |
| F5 | Stdv | 4.57E-01 | 0 | 6.19E+03 | 1.13E+06 | 1.99E+02 | 5.37E+05 | 1.69E+04 | 2.02E+07 | 1.49E+05 | 2.31E+06 | 4.96E+02 |
| | Rank | 1 | 6 | 4 | 8 | 2 | 9 | 5 | 11 | 7 | 10 | 3 |
| | Avg | 8.13E-04 | 1.90E+00 | 4.97E+01 | 3.07E+03 | 5.21E-02 | 5.35E+03 | 2.64E+02 | 5.59E+03 | 3.74E+02 | 6.51E+02 | 1.97E-03 |
| F6 | Stdv | 1.09E-03 | 3.14E-01 | 5.03E+00 | 2.46E+03 | 9.99E-03 | 8.83E+02 | 1.11E+02 | 8.46E+03 | 3.29E+01 | 5.93E+02 | 2.15E-03 |
| | Rank | 1 | 4 | 5 | 9 | 3 | 10 | 6 | 11 | 7 | 8 | 2 |
| | Avg | 4.05E-04 | 2.14E-03 | 2.00E+01 | 1.26E+00 | 2.73E-02 | 2.07E+00 | 7.76E-02 | 1.34E+01 | 3.61E+02 | 2.04E+00 | 2.85E-01 |
| F7 | Stdv | 2.97E-04 | 1.56E-03 | 8.80E+00 | 1.09E+00 | 1.10E-02 | 7.69E-01 | 2.38E-02 | 1.71E+01 | 5.12E+01 | 2.52E+00 | 5.83E-02 |
| | Rank | 1 | 2 | 10 | 6 | 3 | 8 | 4 | 9 | 11 | 7 | 5 |
| Sum of ranks | | 7 | 22 | 42 | 56 | 25 | 64 | 41 | 71 | 51 | 55 | 28 |
| Average rank | | 1 | 3.14286 | 6 | 7.83333 | 3.57143 | 9.14286 | 5.85714 | 10.14286 | 7.28571 | 7.85714 | 4 |
| Overall rank | | 1 | 2 | 6 | 8 | 3 | 10 | 5 | 11 | 7 | 9 | 4 |

**TABLE 7. Results on multimodal benchmark functions (F8-F13).**

| F | | ChaoticBFO | BFO | BA | DA | FA | FPA | GOA | MFO | PSO | SCA | SSA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Avg | -2.09E+04 | -3.34E+03 | -1.17E+04 | -7.37E+03 | -9.23E+03 | -1.12E+04 | -1.21E+04 | -1.32E+04 | -1.13E+04 | -5.03E+03 | -1.21E+04 |
| F8 | Stdv | 8.02E-01 | 6.32E+02 | 9.71E+02 | 9.32E+02 | 9.81E+02 | 2.37E+02 | 1.13E+03 | 1.63E+03 | 1.23E+03 | 3.67E+02 | 1.28E+03 |
| | Rank | 1 | 10 | 4 | 8 | 7 | 6 | 3 | 2 | 5 | 9 | 3 |
| | Avg | -4.89E+02 | -4.88E+02 | 4.93E+02 | 2.57E+02 | 4.92E+01 | 3.13E+02 | 1.18E+02 | 3.15E+02 | 6.76E+02 | 1.13E+02 | 6.31E+01 |
| F9 | Stdv | 3.49E-01 | 7.75E-01 | 3.39E+01 | 8.82E+01 | 9.27E+00 | 3.69E+01 | 4.18E+01 | 5.16E+01 | 3.26E+01 | 6.29E+01 | 2.04E+01 |
| | Rank | 1 | 2 | 10 | 7 | 3 | 8 | 6 | 9 | 11 | 5 | 4 |
| | Avg | -4.55E+21 | -4.23E+21 | 5.39E+00 | 1.01E+01 | 1.35E-01 | 1.56E+01 | 7.22E+00 | 1.95E+01 | 1.01E+01 | 1.59E+01 | 3.43E+00 |
| F10 | Stdv | 1.52E+20 | 3.45E+20 | 2.04E-01 | 2.44E+00 | 2.66E-02 | 1.28E+00 | 1.16E+00 | 7.70E-01 | 2.46E-01 | 7.02E+00 | 5.28E-01 |
| | Rank | 1 | 2 | 5 | 7 | 3 | 8 | 6 | 10 | 7 | 9 | 4 |
| | Avg | 1.08E-04 | 5.09E-03 | 8.70E-01 | 3.51E+01 | 1.43E-02 | 4.86E+01 | 2.73E+00 | 6.32E+01 | 1.10E+00 | 7.25E+00 | 8.24E-02 |
| F11 | Stdv | 1.44E-04 | 2.17E-03 | 3.90E-02 | 2.59E+01 | 3.21E-03 | 1.08E+01 | 8.89E-01 | 7.79E+01 | 8.00E-03 | 1.01E+01 | 3.65E-02 |
| | Rank | 1 | 2 | 5 | 9 | 3 | 10 | 7 | 11 | 6 | 8 | 4 |
| | Avg | 1.96E-12 | 4.34E-11 | 1.55E+01 | 2.24E+04 | 8.76E-04 | 2.08E+04 | 1.21E+01 | 1.72E+07 | 8.85E+00 | 1.18E+07 | 8.52E+00 |
| F12 | Stdv | 3.45E-12 | 5.63E-11 | 5.34E+00 | 5.89E+04 | 8.28E-04 | 2.93E+04 | 4.51E+00 | 6.49E+07 | 1.39E+00 | 1.32E+07 | 2.78E+00 |
| | Rank | 1 | 2 | 7 | 9 | 3 | 8 | 6 | 11 | 5 | 10 | 4 |
| | Avg | 4.12E-05 | 9.92E-02 | 8.39E+00 | 1.08E+06 | 8.46E-03 | 9.60E+05 | 1.42E+02 | 1.41E+07 | 1.25E+03 | 1.67E+07 | 5.76E+01 |
| F13 | Stdv | 1.17E-04 | 1.24E-11 | 1.41E+00 | 2.20E+06 | 2.27E-03 | 7.26E+05 | 1.61E+02 | 7.48E+07 | 9.59E+02 | 2.37E+07 | 1.77E+01 |
| | Rank | 1 | 3 | 4 | 9 | 2 | 8 | 6 | 10 | 7 | 11 | 5 |
| Sum of ranks | | 6 | 21 | 35 | 49 | 21 | 48 | 34 | 53 | 41 | 52 | 24 |
| Average rank | | 1 | 3.5 | 5.833333 | 8.166667 | 3.5 | 8 | 5.666667 | 8.833333 | 6.833333 | 8.666667 | 4 |
| Overall rank | | 1 | 2 | 5 | 8 | 2 | 7 | 4 | 10 | 6 | 9 | 3 |

The statistical results on multimodal F7-F13 are listed in Table 7. From Table 7, it can be seen that ChaoticBFO generates very competitive solutions compared to other algorithms. According to the overall rank, it can be seen that ChaoticBFO ranks first. The corresponding convergence plots shown in Figure 6 also demonstrates the relative superiority of ChaoticBFO in dealing with all test problems. It outperforms all its competitors in tackling F11 and F13 only

**TABLE 8.** **Results on fixed-dimension multimodal benchmark functions (F14-F23).**

| F | | ChaoticBFO | BFO | BA | DA | FA | FPA | GOA | MFO | PSO | SCA | SSA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| F14 | Avg | 9.98E-01 | 2.03E+00 | 3.97E+00 | 1.82E+00 | 1.88E+00 | 9.98E-01 | 9.98E-01 | 1.16E+00 | 1.76E+00 | 1.59E+00 | 1.03E+00 |
| | Stdv | 1.29E-10 | 1.26E+00 | 4.16E+00 | 1.04E+00 | 1.12E+00 | 4.02E-04 | 5.15E-16 | 7.38E-01 | 1.39E+00 | 9.25E-01 | 1.81E-01 |
| | Rank | 1 | 10 | 11 | 8 | 9 | 2 | 3 | 5 | 7 | 6 | 4 |
| F15 | Avg | 3.47E-04 | 5.13E-04 | 1.07E-02 | 4.64E-03 | 1.36E-03 | 7.16E-04 | 9.96E-03 | 1.45E-03 | 1.25E-03 | 9.51E-04 | 2.22E-03 |
| | Stdv | 2.77E-05 | 1.50E-04 | 9.87E-03 | 5.09E-03 | 2.11E-03 | 1.31E-04 | 1.61E-02 | 3.58E-03 | 4.71E-04 | 3.53E-04 | 4.94E-03 |
| | Rank | 1 | 2 | 11 | 9 | 6 | 3 | 10 | 7 | 5 | 4 | 8 |
| F16 | Avg | -1.03E+00 | -1.03E+00 | -1.03E+00 | -1.03E+00 | -1.03E+00 | -1.03E+00 | -1.03E+00 | -1.03E+00 | -1.03E+00 | -1.03E+00 | -1.03E+00 |
| | Stdv | 2.53E-06 | 4.85E-06 | 9.92E-04 | 1.75E-14 | 2.71E-09 | 1.42E-08 | 6.08E-13 | 6.78E-16 | 2.10E-03 | 4.36E-05 | 1.73E-14 |
| | Rank | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| F17 | Avg | 3.98E-01 | 3.98E-01 | 3.98E-01 | 3.98E-01 | 3.98E-01 | 3.98E-01 | 3.98E-01 | 3.98E-01 | 3.99E-01 | 3.99E-01 | 3.98E-01 |
| | Stdv | 1.18E-06 | 1.38E-06 | 3.82E-04 | 3.24E-16 | 1.23E-09 | 1.05E-09 | 2.10E-13 | 0.00E+00 | 1.03E-03 | 4.95E-04 | 8.51E-15 |
| | Rank | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 |
| F18 | Avg | 3.00E+00 | 3.00E+00 | 3.04E+00 | 3.00E+00 | 3.00E+00 | 3.00E+00 | 5.70E+00 | 3.00E+00 | 3.07E+00 | 3.00E+00 | 3.00E+00 |
| | Stdv | 1.62E-04 | 1.68E-04 | 4.31E-02 | 3.31E-12 | 3.77E-08 | 1.22E-06 | 1.48E+01 | 1.22E-15 | 1.07E-01 | 3.18E-05 | 2.20E-13 |
| | Rank | 1 | 1 | 2 | 1 | 1 | 1 | 4 | 1 | 3 | 1 | 1 |
| F19 | Avg | -3.86E+00 | -3.86E+00 | -3.84E+00 | -3.86E+00 | -3.86E+00 | -3.86E+00 | -3.81E+00 | -3.86E+00 | -3.85E+00 | -3.85E+00 | -3.86E+00 |
| | Stdv | 2.27E-04 | 3.61E-04 | 1.35E-02 | 1.01E-02 | 9.60E-10 | 1.23E-06 | 1.96E-01 | 2.71E-15 | 1.48E-02 | 2.91E-03 | 1.78E-13 |
| | Rank | 1 | 1 | 3 | 1 | 1 | 1 | 4 | 1 | 2 | 2 | 1 |
| F20 | Avg | -3.29E+00 | -3.28E+00 | -2.89E+00 | -3.20E+00 | -3.28E+00 | -3.32E+00 | -3.26E+00 | -3.25E+00 | -2.88E+00 | -3.01E+00 | -3.21E+00 |
| | Stdv | 3.40E-02 | 1.78E-02 | 1.27E-01 | 1.27E-01 | 6.02E-02 | 3.65E-03 | 6.29E-02 | 6.52E-02 | 1.86E-01 | 1.35E-01 | 3.88E-02 |
| | Rank | 2 | 3 | 9 | 7 | 3 | 1 | 4 | 5 | 10 | 8 | 6 |
| F21 | Avg | -1.02E+01 | -1.01E+01 | -4.70E+00 | -9.98E+00 | -8.50E+00 | -1.01E+01 | -5.88E+00 | -7.07E+00 | -4.01E+00 | -2.57E+00 | -8.06E+00 |
| | Stdv | 2.49E-04 | 1.01E-02 | 2.66E+00 | 9.22E-01 | 3.08E+00 | 4.82E-02 | 3.23E+00 | 3.45E+00 | 1.45E+00 | 1.84E+00 | 3.09E+00 |
| | Rank | 1 | 2 | 8 | 3 | 4 | 2 | 7 | 6 | 9 | 10 | 5 |
| F22 | Avg | -1.04E+01 | -1.04E+01 | -6.29E+00 | -1.02E+01 | -1.04E+01 | -1.03E+01 | -6.56E+00 | -9.06E+00 | -4.71E+00 | -4.05E+00 | -8.79E+00 |
| | Stdv | 2.64E-04 | 8.97E-03 | 2.91E+00 | 9.63E-01 | 1.20E-06 | 7.24E-02 | 3.51E+00 | 2.76E+00 | 1.89E+00 | 1.67E+00 | 2.77E+00 |
| | Rank | 1 | 1 | 7 | 3 | 1 | 2 | 6 | 4 | 8 | 9 | 5 |
| F23 | Avg | -1.05E+01 | -1.03E+01 | -5.26E+00 | -1.04E+01 | -1.05E+01 | -1.04E+01 | -7.09E+00 | -8.98E+00 | -5.21E+00 | -4.65E+00 | -9.22E+00 |
| | Stdv | 2.51E-04 | 9.77E-01 | 2.94E+00 | 9.79E-01 | 1.01E-06 | 8.54E-02 | 3.61E+00 | 2.88E+00 | 1.60E+00 | 1.45E+00 | 2.74E+00 |
| | Rank | 1 | 3 | 7 | 2 | 1 | 2 | 6 | 5 | 8 | 9 | 4 |
| Sum of ranks | | 11 | 25 | 60 | 36 | 28 | 16 | 46 | 36 | 55 | 52 | 36 |
| Average rank | | 1.1 | 2.5 | 6 | 3.6 | 2.8 | 1.6 | 4.6 | 3.6 | 5.5 | 5.2 | 3.6 |
| Overall rank | | 1 | 3 | 9 | 5 | 4 | 2 | 6 | 5 | 8 | 7 | 5 |

through 10 iterations. For F8, F9, and F10, it can be seen that ChaoticBFO converges fast during few searching steps compared with other methods. For F10, FA has a better result until 500 iterations, whereas ChaoticBFO outperforms FA and it found the good solution during the initial steps.

The results on F14 to F23 are listed in Table 8. The results in Table 8 show that ChaoticBFO generates the optimal solutions for problems F14, F15, and F21 in all 30 runs. The algorithms have the same exploration capabilities in tackling problems F16, F17, F18, and F19. In particular, the FPA algorithm outperforms other optimizers in tackling the problem F20. Moreover, in dealing with F22, ChaoticBFO, BFO and FA algorithms are more competitive than other algorithms. However, based on overall rank, ChaoticBFO is the best technique.

As shown in Figure 7, it can be seen that ChaoticBFO converges very fast and outperforms all other methods in dealing with F15, with BFO being the second-best. For F14, it can be seen that ChaoticBFO converges faster during few searching steps and generates better results than GOA and SSA. In tackling F17, F18 and F19, DA converges very fast throughout early steps, whereas ChaoticBFO is the second-best. For F21, F22 and F23, there is a close rivalry between DA and ChaoticBFO.

As can be seen from Figs.6-7, in dealing with the F9, F10 and F16 problems, the convergence curve of ChaoticBFO is incomplete. It is because ChaoticBFO has converged to the global optimal value at the initial stage. Therefore, the corresponding convergence curve is represented as a red dot on the F9 and F10 problems. Similarly, for the F9 problem, the convergence curve of BFO suddenly disappears at about 150 iterations because it has obtained the optimal solution. Especially when solving F16 (Fig.7-c), all convergence curves are missing except that there are three different color points representing FA, FPA and MFO from top to bottom. It is because all the algorithms used have been able to find the global optimal value at a very fast speed at the beginning of iteration, with these three methods converging relatively slowly compared with other algorithms.

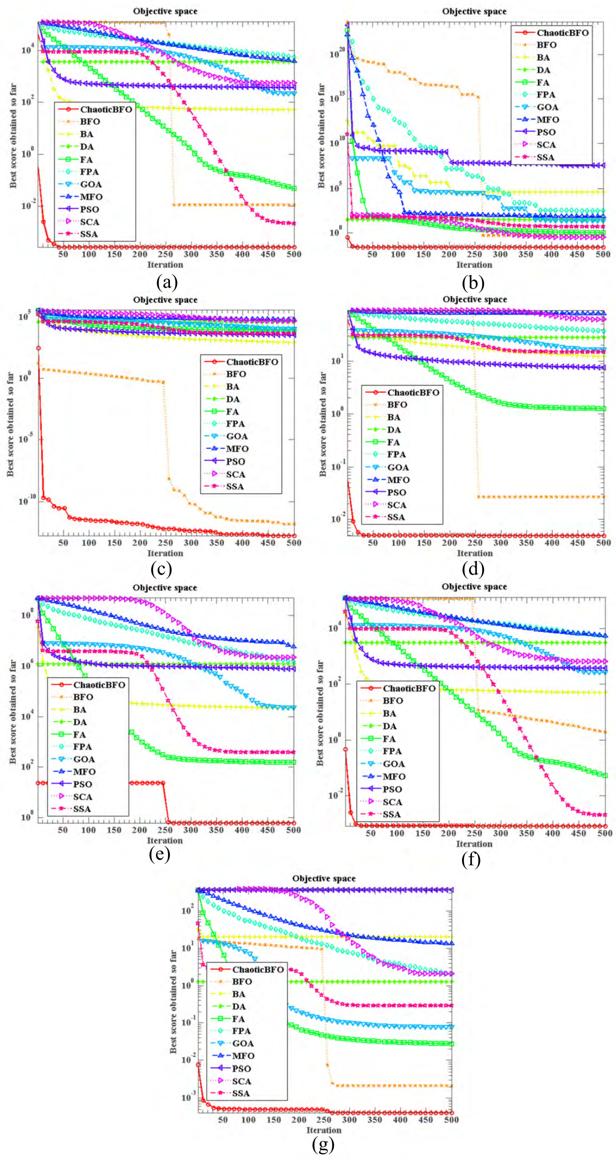


**FIGURE 5.** **Convergence curves on unimodal functions (F1-F7).**

As can be seen from the results, ChaoticBFO can achieve acceleration throughout early steps, because it utilizes chaos initialization strategy in the initialization step. Moreover, the adoption of chaotic local search with a 'shrinking' strategy enables ChaoticBFO to escape from local optimum with much better chance.

The Wilcoxon rank-sum test [57] at 5% significance is also utilized to judge whether the improvements achieved by ChaoticBFO are meaningful over the other optimizers. When p-values are less than 0.05, it can be determined that ChaoticBFO is significantly superior to the other approaches. If not, the improvements are not statistically significant.

The p-values are presented in Tables 9-11. In each table, p-values greater than 0.05 are shown in bold face, indicating that the differences are not significant. It can be seen in Table 9 that all p-values are less than 0.05, demonstrating

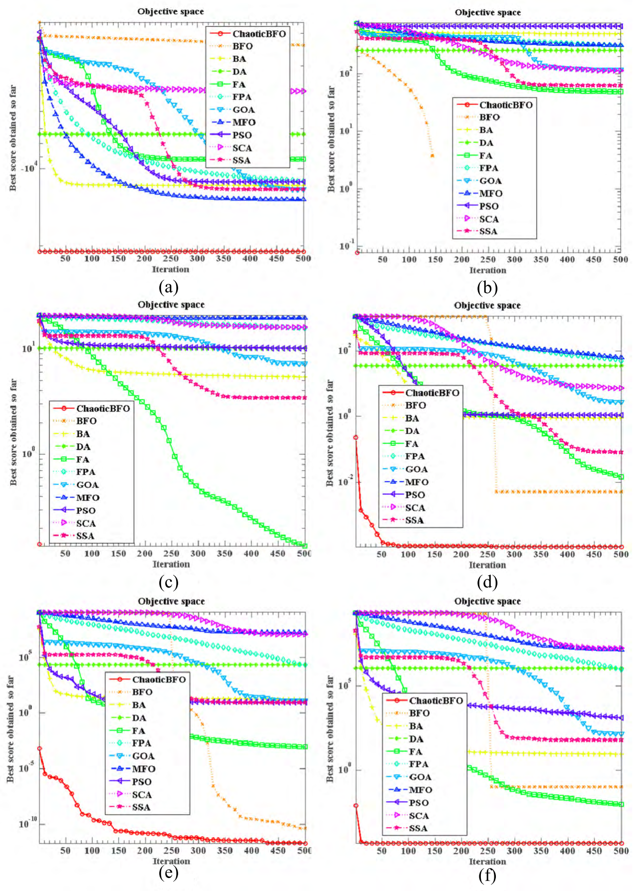


**FIGURE 6.** **Convergence curves on multimodal functions (F8-F13).**

that the performance of ChaoticBFO is statistically improved compared to other optimizers on functions F1-F7. The p-values of ChaoticBFO and other algorithms for F8-F13 are provided in Table 10. With all p-values less than 0.05, the results show that ChaoticBFO significantly outperforms the other competitors. The advantage of ChaoticBFO in dealing with F8-F13 can verify that it has a reasonable exploration capacity.

From Table 11, it can be noted that ChaoticBFO statistically prevails in over half of the test functions compared to other methods. Based on p-values for all 23 functions, it can be concluded that ChaoticBFO is significantly better than the original BFO on all functions except for F16, F17, F18, F19, F22 and F23. This is because the former takes advantage of the chaos initialization strategy and chaotic local search with a 'shrinking' strategy to achieve a more suitable balance between exploration and exploitation.

## *D. WALL-CLOCK TIME COST*

The wall-clock time consumed by ChaoticBFO and BFO on F1-F23 is shown in Table 12. It should be noted that wall-clock time cost is shown in seconds. It can be seen from Table 12 that ChaoticBFO consumes more wall-clock time than BFO in tackling all the test functions. The main reason for the greater time cost of ChaoticBFO is that two

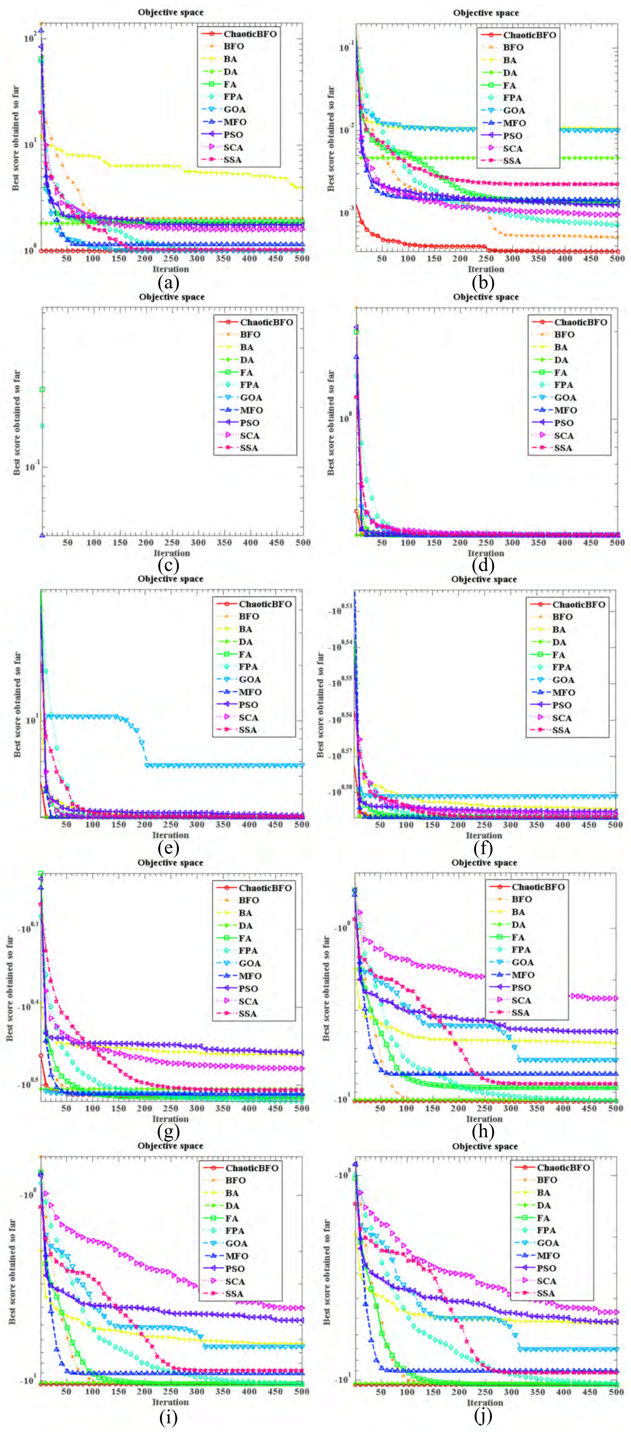


FIGURE 7. Convergence curves on fixed-dimension multimodal functions (F14-F23).

chaotic strategies (chaotic initialization strategy and chaotic local search with a 'shrinking' strategy) are introduced in the original BFO to improve performance and achieve a better balance between exploitation and exploration. Although the wall-clock time cost of ChaoticBFO is greater than that of BFO, it can be found from the above experimental results that ChaoticBFO is superior to BFO in most cases. Therefore, it is of great value to introduce the two chaotic strategies into BFO.

TABLE 9. The calculated p-values from the functions (F1-F7) for ChaoticBFO versus other optimizers.

| Problem | BFO | BA | DA | FA | FPA |
|---|---|---|---|---|---|
| F1 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 |
| F2 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 |
| F3 | 1.40E-02 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 |
| F4 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 |
| F5 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 |
| F6 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 |
| F7 | 1.00E-05 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 |
| | GOA | MFO | PSO | SCA | SSA |
| F1 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.80E-05 |
| F2 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 5.00E-06 | 2.00E-06 |
| F3 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 |
| F4 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 |
| F5 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 |
| F6 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 3.16E-03 |
| F7 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 |

TABLE 10. The calculated p-values from the functions (F8-F13) for ChaoticBFO versus other optimizers.

| Problem | BFO | BA | DA | FA | FPA |
|---|---|---|---|---|---|
| F8 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 |
| F9 | 5.08E-03 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 |
| F10 | 6.90E-05 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 |
| F11 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 |
| F12 | 4.40E-05 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 |
| F13 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 |
| | GOA | MFO | PSO | SCA | SSA |
| F8 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 |
| F9 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 |
| F10 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 |
| F11 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 |
| F12 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 |
| F13 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 |

## VI. COMPARATIVE STUDY ON TWO REAL-WORLD BENCHMARKS FROM IEEE CEC 2011

In this section, ChaoticBFO was applied to two real-world engineering problems from IEEE CEC 2011 benchmark set [58]. Detailed information on these problems is shown in Table 13. One of the problems is Parameter Estimation for Frequency-Modulated (FM) Sound Waves, and the other is Static Economic Load Dispatch (ELD) Problem Instance 1.

**TABLE 11. The calculated p-values from the functions (F14-F23) for ChaoticBFO versus other optimizers.**

| Problem | BFO | BA | DA | FA | FPA |
|---|---|---|---|---|---|
| F14 | 2.44E-04 | 3.70E-05 | 8.24E-04 | 2.00E-06 | **1.02E-01** |
| F15 | 4.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 |
| F16 | **1.00E+00** | **1.00E+00** | **1.00E+00** | **1.00E+00** | **1.00E+00** |
| F17 | **1.00E+00** | 1.43E-02 | **1.00E+00** | **1.00E+00** | **1.00E+00** |
| F18 | **1.00E+00** | 1.20E-05 | **1.00E+00** | **1.00E+00** | **1.00E+00** |
| F19 | **1.00E+00** | 4.00E-06 | **6.56E-02** | **1.00E+00** | **1.00E+00** |
| F20 | 3.60E-02 | 2.00E-06 | 2.47E-03 | **9.75E-01** | 3.00E-06 |
| F21 | 4.32E-08 | 2.00E-06 | **1.80E-01** | 1.60E-02 | 2.60E-07 |
| F22 | **1.00E+00** | 2.00E-06 | **3.17E-01** | **1.00E+00** | 2.17E-04 |
| F23 | **3.17E-01** | 2.00E-06 | **3.17E-01** | **1.00E+00** | 6.10E-05 |
| | GOA | MFO | PSO | SCA | SSA |
| F14 | **1.00E+00** | **1.80E-01** | 7.99E-04 | 1.32E-03 | **3.17E-01** |
| F15 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 | 2.00E-06 |
| F16 | **1.00E+00** | **1.00E+00** | **3.17E-01** | **1.00E+00** | **1.00E+00** |
| F17 | **1.00E+00** | **1.00E+00** | 5.77E-04 | 7.89E-04 | **1.00E+00** |
| F18 | **3.17E-01** | **1.00E+00** | 8.00E-06 | **1.00E+00** | **1.00E+00** |
| F19 | **1.57E-01** | **1.00E+00** | 1.70E-05 | 3.41E-07 | **1.00E+00** |
| F20 | **1.66E-01** | 9.03E-03 | 2.00E-06 | 2.00E-06 | 7.00E-06 |
| F21 | 7.70E-05 | 7.39E-04 | 2.00E-06 | 2.00E-06 | 4.64E-03 |
| F22 | 2.71E-04 | 2.69E-02 | 2.00E-06 | 2.00E-06 | 1.12E-02 |
| F23 | 6.32E-04 | 1.41E-02 | 2.00E-06 | 2.00E-06 | 2.69E-02 |

These problems can be used to evaluate the performance of stochastic optimization algorithms. For more details about these problems, refer to [58].

In order to verify the efficiency of ChaoticBFO on CEC2011 R01 and R02 test problems, ChaoticBFO was compared against the original BFO [9], four well-known algorithms: PSO [3], BA [49], FA [50], and FPA [51] as well as four recently developed metaheuristic algorithms: MFO [52], SCA [54], GOA [55], and SSA [56]. The parameter values for the above algorithms were set according to their original articles. Table 5 shows the best recommended parameter values for each algorithm. Moreover, in order to achieve unbiased results, all experiments were conducted under the same conditions and followed the recommendations provided in CEC 2011 IEEE congress [58]. In all the experiments, the population size and maximum number of iterations were set to 50 and 1000, respectively, and the results were recorded based on 30 independent runs for each approach.

Tables 14 and 15 show the comparison of ChaoticBFO versus other optimizers in dealing with the R01 problem and the R02 problem, respectively. The rank of each method based on the average results of 30 independent runs and the p-values of the Wilcoxon rank-sum test [57] at 5% significance are also provided in Tables 14 and 15. In each table, each

**TABLE 12. Wall-clock time costs on test functions (F1-F23).**

| Problem | ChaoticBFO | BFO |
|---|---|---|
| F1 | 7.17 | 2.03 |
| F2 | 7.95 | 1.84 |
| F3 | 215.44 | 2.48 |
| F4 | 6.32 | 1.94 |
| F5 | 10.63 | 1.16 |
| F6 | 6.75 | 2.29 |
| F7 | 18.92 | 2.22 |
| F8 | 9.94 | 2.61 |
| F9 | 12.00 | 1.92 |
| F10 | 12.25 | 2.09 |
| F11 | 13.47 | 2.57 |
| F12 | 44.07 | 4.94 |
| F13 | 43.80 | 4.92 |
| F14 | 81.30 | 6.14 |
| F15 | 12.16 | 1.98 |
| F16 | 8.73 | 1.54 |
| F17 | 7.95 | 1.50 |
| F18 | 8.51 | 1.48 |
| F19 | 18.04 | 2.17 |
| F20 | 18.15 | 2.13 |
| F21 | 20.78 | 2.39 |
| F22 | 24.26 | 2.67 |
| F23 | 29.53 | 2.95 |

**TABLE 13. Description of real-world optimization problems.**

| ID | Name of the problem | Class | No. of Dimensions |
|---|---|---|---|
| R01 | Parameter Estimation for FM Sound Waves | Bound constrained | 6 |
| R02 | The ELD Instance 1 | Inequality constraints | 6 |

p-value greater than 0.05 is shown in bold face, indicating the differences are not significant.

From Table 14, it can be seen that ChaoticBFO is the best optimizer in dealing with the R01 problem. According to the average results, it can reduce the optimum cost of BFO by up to 23.50%. Regarding the ranking results, ChaoticBFO is followed successively by the FA, PSO, MFO, GOA, FPA, SSA, SCA, BA and BFO methods. Moreover, the results show that ChaoticBFO significantly outperforms the BFO, BA, SCA algorithms according to the p-values.

The results in Table 15 show that ChaoticBFO also ranks first in dealing with the R02 problem, whereas BFO, PSO and SCA perform poorly. The results demonstrate that ChaoticBFO, which utilizes chaotic initialization and

**TABLE 14.** Comparison of ChaoticBFO versus other optimizers in dealing with R01 problem.

| | Avg | Stdv | Rank | p-value |
|---|---|---|---|---|
| ChaoticBFO | 17.7865 | 2.0168 | 1 | NA |
| BFO | 23.2491 | 1.9390 | 10 | 0.000002 |
| BA | 22.2108 | 3.2132 | 9 | 0.000008 |
| FA | 17.9155 | 5.5999 | 2 | **0.718888** |
| FPA | 18.9230 | 2.3435 | 6 | **0.028486** |
| GOA | 18.7058 | 4.3338 | 5 | **0.338856** |
| MFO | 18.1120 | 5.7084 | 4 | **0.829013** |
| PSO | 18.0921 | 4.2074 | 3 | **0.797098** |
| SCA | 21.1886 | 6.2048 | 8 | 0.006424 |
| SSA | 19.7802 | 3.3975 | 7 | **0.016566** |

**TABLE 15.** Comparison of ChaoticBFO versus other optimizers in dealing with R02 problem.

| | Avg | Stdv | Rank | p-value |
|---|---|---|---|---|
| ChaoticBFO | 15455.6187 | 2.8837 | 1 | NA |
| BFO | 15790.5953 | 236.9747 | 8 | 0.000002 |
| BA | 15531.0001 | 44.8371 | 7 | 0.000002 |
| FA | 15492.0490 | 13.7950 | 4 | 0.000002 |
| FPA | 15456.5864 | 5.5733 | 2 | **0.416534** |
| GOA | 15496.2378 | 21.5129 | 5 | 0.000002 |
| MFO | 15486.3413 | 21.4709 | 3 | 0.000002 |
| PSO | 16656.1331 | 3827.3664 | 9 | 0.000005 |
| SCA | 138178.8847 | 68209.0150 | 10 | 0.000002 |
| SSA | 15508.1260 | 35.9843 | 6 | 0.000002 |

chaotic local search with a 'shrinking' strategy, can effectively improve the performance of the algorithm in dealing with R01 and R02 test problems. In addition, the combination of the two chaotic strategies can achieve a better balance between exploration and exploitation.

## VII. CONCLUSIONS AND FUTURE WORK

For BFO to solve complex optimization problems efficiently and effectively, we propose an improved BFO called ChaoticBFO in this paper. First, a chaotic initialization strategy is applied to the bacterial population initialization to increase the population of initial solutions for the selection of the best initial population, making the algorithm converge faster in the initial stage. Then, in the chemotaxis step, the chaotic local search with a 'shrinking' strategy is used to avoid the algorithm getting into the local optimal solution, thus obtaining the global optimal solution during the search process. To sum up, the combination of the two chaotic strategies can achieve a reasonable balance between exploration and exploitation.

In future research, there are many aspects worth expanding. For example, in order to achieve better performance, the original BFO can hybridize with other state-of-the-art metaheuristic algorithms. Moreover, it is necessary to apply ChaoticBFO to more practical problems in real-world scenarios. Furthermore, extending ChaoticBFO to the binary version and multi-objective [59], [60] version is also an interesting direction. In addition, in view of the high time-complexity of the proposed algorithm in dealing with relatively complex problems, it would be helpful to adopt parallel processing technologies based on GPU and multi-threading to further improve the computational capacity and speed of the algorithm.

## ACKNOWLEDGEMENTS

*(Huiling Chen and Chengye Li contributed equally to this work.)*

## REFERENCES

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Cambridge, MA, USA: MIT Press, 1992, pp. 126–137.

[2] K. S. Tang *et al.*, "Genetic algorithms and their applications," *IEEE Signal Process. Mag.*, vol. 13, no. 6, pp. 22–37, 1996.

[3] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw.*, Dec. 1995, pp. 1942–1948.

[4] W. Deng, H. Zhao, X. Yang, J. Xiong, M. Sun, and B. Li, "Study on an improved adaptive PSO algorithm for solving multi-objective gate assignment," *Appl. Soft Comput.*, vol. 59, pp. 288–302, Oct. 2017.

[5] K. Socha and M. Dorigo, "Ant colony optimization for continuous domains," *Eur. J. Oper. Res.*, vol. 185, no. 3, pp. 1155–1173, 2008.

[6] C. Twomey, T. Stützle, M. Dorigo, M. Manfrin, and M. Birattari, "An analysis of communication policies for homogeneous multi-colony ACO algorithms," *Inf. Sci.*, vol. 180, no. 12, pp. 2390–2404, 2010.

[7] L. Shen *et al.*, "Evolving support vector machines using fruit fly optimization for medical data classification," *Knowl.-Based Syst.*, vol. 96, pp. 61–75, Mar. 2016.

[8] J. Luo, H. Chen, Y. Xu, H. Huang, and X. Zhao, "An improved grasshopper optimization algorithm with application to financial stress prediction," *Appl. Math. Model.*, vol. 64, pp. 654–668, Dec. 2018.

[9] 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.

[10] D. H. Kim and J. H. Cho, "Adaptive tuning of PID controller for multivariable system using bacterial foraging based optimization," in *Advances in Web Intelligence—AWIC* (Lecture Notes in Computer Science), vol. 3528, P. S. Szczepaniak, J. Kacprzyk, and A. Niewiadomski, Eds. Berlin, Germany: Springer, 2005.

[11] S. Mishra, "A hybrid least square-fuzzy bacterial foraging strategy for harmonic estimation," *IEEE Trans. Evol. Comput.*, vol. 9, no. 1, pp. 61–73, Feb. 2005.

[12] M. Tripathy, S. Mishra, L. L. Lai, and Q. P. Zhang, "Transmission loss reduction based on FACTS and bacteria foraging algorithm," in *Parallel Problem Solving from Nature—PPSN IX* (Lecture Notes in Computer Science), vol. 4193, T. P. Runarsson, H. G. Beyer, E. Burke, J. J. Merelo-Guervós, L. D. Whitley, and X. Yao, Eds. Berlin, Germany: Springer, 2006.

[13] H. K. Dong and C. H. Cho, "Bacteria foraging based neural network fuzzy learning," in *Proc. Indian Int. Conf. Artif. Intell.*, Pune, India, Dec. 2005, pp. 2030–2036.

[14] S. Dasgupta, S. Das, A. Biswas, and A. Abraham, "Automatic circle detection on digital images with an adaptive bacterial foraging algorithm," *Soft Comput.*, vol. 14, no. 11, pp. 1151–1164, 2010.

[15] R. Majhi, G. Panda, B. Majhi, and G. Sahoo, "Efficient prediction of stock market indices using adaptive bacterial foraging optimization (ABFO) and BFO based techniques," *Expert Syst. Appl.*, vol. 36, no. 6, pp. 10097–10104, 2009.

[16] V. P. Sakthivel, R. Bhuvaneswari, and S. Subramanian, "Design optimization of three-phase energy efficient induction motor using adaptive bacterial foraging algorithm," *COMPEL-Int. J. Comput. Math. Electr. Electron. Eng.*, vol. 29, no. 3, pp. 699–726, 2010.

[17] B. Bhushan and M. Singh, "Adaptive control of DC motor using bacterial foraging algorithm," *Appl. Soft Comput.*, vol. 11, no. 8, pp. 4913–4920, 2011.

[18] N. Sanyal, A. Chatterjee, and S. Munshi, "An adaptive bacterial foraging algorithm for fuzzy entropy based image segmentation," *Expert Syst. Appl.*, vol. 38, no. 12, pp. 15489–15498, 2011.

[19] P. D. Sathya and R. Kayalvizhi, "Modified bacterial foraging algorithm based multilevel thresholding for image segmentation," *Eng. Appl. Artif. Intell.*, vol. 24, no. 4, pp. 595–615, 2011.

[20] O. P. Verma, M. Hanmandlu, P. Kumar, S. Chhabra, and A. Jindal, "A novel bacterial foraging technique for edge detection," *Pattern Recognit. Lett.*, vol. 32, no. 8, pp. 1187–1196, 2011.

[21] R. Panda and M. K. Naik, "A novel adaptive crossover bacterial foraging optimization algorithm for linear discriminant analysis based face recognition," *Appl. Soft Comput.*, vol. 30, pp. 722–736, May 2015.

[22] K. Tang, X. Xiao, J. Wu, J. Yang, and L. Luo, "An improved multilevel thresholding approach based modified bacterial foraging optimization," *Appl. Intell.*, vol. 46, no. 1, pp. 214–226, 2017.

[23] T. Jain, M. J. Nigam, and S. Alavandar, "A hybrid genetically-bacterial foraging algorithm converged by particle swarm optimisation for global optimisation," *Int. J. Bio-Inspired Comput.*, vol. 2, no. 5, pp. 340–348, 2010.

[24] L. Tan, F. Lin, and H. Wang, "Adaptive comprehensive learning bacterial foraging optimization and its application on vehicle routing problem with time windows," *Neurocomputing*, vol. 151, pp. 1208–1215, Mar. 2015.

[25] Y. E. Yıldız and O. Altun, "Hybrid achievement oriented computational chemotaxis in bacterial foraging optimization: A comparative study on numerical benchmark," *Soft Comput.*, vol. 19, no. 12, pp. 3647–3663, 2015.

[26] C. Yang, J. Ji, J. Liu, J. Liu, and B. Yin, "Structural learning of Bayesian networks by bacterial foraging optimization," *Int. J. Approx. Reasoning*, vol. 69, pp. 147–167, Feb. 2016.

[27] C. Yang, J. Ji, J. Liu, and B. Yin, "Bacterial foraging optimization using novel chemotaxis and conjugation strategies," *Inf. Sci.*, vol. 363, pp. 72–95, Oct. 2016.

[28] W. Zhao and L. Wang, "An effective bacterial foraging optimizer for global optimization," *Inf. Sci.*, vol. 329, pp. 719–735, Feb. 2016.

[29] Y.-T. Zhang and C. Huang, "An improved bacterial foraging optimization based approach to data clustering," in *Proc. Int. Conf. Artif. Intell. Sci. Technol.*, 2017, pp. 91–99.

[30] F. Zhao, Y. Liu, Z. Shao, X. Jiang, C. Zhang, and J. Wang, "A chaotic local search based bacterial foraging algorithm and its application to a permutation flow-shop scheduling problem," *Int. J. Comput. Integr. Manuf.*, vol. 29, no. 9, pp. 962–981, 2016.

[31] L. Zhang and C. Zhang, "Hopf bifurcation analysis of some hyperchaotic systems with time-delay controllers," *Kybernetika*, vol. 44, no. 1, pp. 35–42, 2008.

[32] G. I. Sayed, A. E. Hassanien, and A. T. Azar, "Feature selection via a novel chaotic crow search algorithm," *Neural Comput. Appl.*, to be published, doi: 10.1007/s00521-017-2988-6.

[33] A. Tharwat and A. E. Hassanien, "Chaotic antlion algorithm for parameter optimization of support vector machine," *Appl. Intell.*, vol. 48, no. 3, pp. 670–686, 2018.

[34] L. J. Yang and T. L. Chen, "Application of chaos in genetic algorithms," *Commun. Theor. Phys.*, vol. 38, no. 2, pp. 168–172, 2002.

[35] J. Mingjun and T. Huanwen, "Application of chaos in simulated annealing," *Chaos, Solitons Fractals*, vol. 21, no. 4, pp. 933–941, 2004.

[36] B. Alatas, E. Akin, and A. B. Ozer, "Chaos embedded particle swarm optimization algorithms," *Chaos, Solitons Fractals*, vol. 40, no. 4, pp. 1715–1734, 2009.

[37] M. Wang *et al.*, "Toward an optimal kernel extreme learning machine using a chaotic moth-flame optimization strategy with applications in medical diagnoses," *Neurocomputing*, vol. 267, pp. 69–84, Dec. 2017.

[38] A. H. Gandomi, X.-S. Yang, S. Talatahari, and A. H. Alavi, "Firefly algorithm with chaos," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 18, no. 1, pp. 89–98, 2013.

[39] B. Wu and S. Fan, "Improved artificial bee colony algorithm with chaos," in *Computer Science for Environmental Engineering and EcoInformatics—CSEEE* (Communications in Computer and Information Science), vol. 158, Y. Yu, Z. Yu, and J. Zhao, Eds. Berlin, Germany: Springer, 2011.

[40] S. Saremi, S. Mirjalili, and A. Lewis, "Biogeography-based optimisation with chaos," *Neural Comput. Appl.*, vol. 25, no. 5, pp. 1077–1097, 2014.

[41] G.-G. Wang, L. H. Guo, A. H. Gandomi, G.-S. Hao, and H. Q. Wang, "Chaotic krill herd algorithm," *Inf. Sci.*, vol. 274, pp. 17–34, Aug. 2014.

[42] A. A. Heidari, R. Ali Abbaspour, and A. R. Jordehi, "An efficient chaotic water cycle algorithm for optimization tasks," *Neural Comput. Appl.*, vol. 28, no. 1, pp. 57–85, 2017.

[43] M. Kohli and S. Arora, "Chaotic grey wolf optimization algorithm for constrained optimization problems," *J. Comput. Des. Eng.*, vol. 5, no. 4, pp. 458–472, Oct. 2018.

[44] M. S. Tavazoei and M. Haeri, "Comparison of different one-dimensional maps as chaotic search pattern in chaos optimization algorithms," *Appl. Math. Comput.*, vol. 187, no. 2, pp. 1076–1085, 2007.

[45] L. Wang, D.-Z. Zheng, and Q. S. Lin, "Survey on chaotic optimization methods," *Comput. Technol. Autom.*, vol. 20, no. 1, pp. 1–5, 2001.

[46] B. Liu, L. Wang, Y.-H. Jin, F. Tang, and D. X. Huang, "Improved particle swarm optimization combined with chaos," *Chaos, Solitons Fractals*, vol. 25, no. 5, pp. 1261–1271, 2005.

[47] D. Jia, G. Zheng, and M. K. Khan, "An effective memetic differential evolution algorithm based on chaotic local search," *Inf. Sci.*, vol. 181, no. 15, pp. 3175–3187, 2011.

[48] S. Saha and V. Mukherjee, "A novel chaos-integrated symbiotic organisms search algorithm for global optimization," *Soft Comput.*, vol. 22, no. 11, pp. 3797–3816, Jun. 2018.

[49] X. S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies for Optimization—NICSO)* (Studies in Computational Intelligence), vol. 284, J. R. González, D. A. Pelta, C. Cruz, G. Terrazas, and N. Krasnogor, Eds. Berlin, Germany: Springer, 2010.

[50] X.-S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *Int. J. Bio-Inspired Comput.*, vol. 2, no. 2, pp. 78–84, 2010.

[51] X.-S. Yang, *Flower Pollination Algorithm for Global Optimization*. Berlin, Germany: Springer, 2012.

[52] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowl.-Based Syst.*, vol. 89, pp. 228–249, Nov. 2015.

[53] S. Mirjalili, "Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Comput. Appl.*, vol. 27, no. 4, pp. 1053–1073, 2016.

[54] S. Mirjalili, "SCA: A sine cosine algorithm for solving optimization problems," *Knowl.-Based Syst.*, vol. 96, pp. 120–133, Mar. 2016.

[55] S. Saremi, S. Mirjalili, and A. Lewis, "Grasshopper optimisation algorithm: Theory and application," *Adv. Eng. Softw.*, vol. 105, pp. 30–47, Mar. 2017.

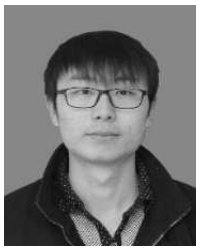
[56] S. Mirjalili, S. A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp swarm algorithm: A bio-inspired optimizer for engineering design problems," *Adv. Eng. Softw.*, vol. 114, pp. 163–191, Dec. 2017.

[57] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, Mar. 2011.

[58] S. Das and P. N. Suganthan, "Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems," Nanyang Technol. Univ., Singapore, Tech. Rep. CEC2011, 2010.

[59] J. Yi, D. Huang, S. Fu, H. He, and T. Li, "Multi-objective bacterial foraging optimization algorithm based on parallel cell entropy for aluminum electrolysis production process," *IEEE Trans. Ind. Electron.*, vol. 63, no. 4, pp. 2488–2500, Apr. 2016.

[60] J. Yi, J. Bai, W. Zhou, H. He, and L. Yao, "Operating parameters optimization for the aluminum electrolysis process using an improved quantum-behaved particle swarm algorithm," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3405–3415, Aug. 2018.



**QIAN ZHANG** received the bachelor's degree from the Department of Computer Science and Technology, Xinyang Normal University, China. He is currently a Graduate Student (majoring in computer software and theory) with Wenzhou University, China. His research interests are data mining, pattern recognition, evolutionary computation, as well as their applications such as medical diagnosis and bankruptcy prediction, among others.

**HUILING CHEN** received the Ph.D. degree from the Department of Computer Science and Technology, Jilin University, China. He is currently an Associate Professor with the Department of Computer Science and Technology, Wenzhou University, China. His current research interests center on machine learning and data mining, as well as their applications to medical diagnosis and bankruptcy prediction. He is currently a Reviewer of the IEEE Transactions on Systems, Man, and Cybernetics, Part b. He has published over 100 papers in international journals and conference proceedings, including *Pattern Recognition*, *Expert Systems with Applications*, *Knowledge-Based Systems*, *Soft Computing*, *Neurocomputing* and PAKDD, among others.

**JIE LUO** received the bachelor's degree from the Department of Mathematics and Computer Science, Yichun University, China. He is currently a Graduate Student (majoring in computer software and theory) with Wenzhou University, China. His research interests are data mining, pattern recognition, evolutionary computation, as well as their applications such as medical diagnosis and bankruptcy prediction, among others.

**YUETING XU** received the bachelor's degree from the Department of Computer Science and Technology, Ningbo Dahongying University, China. She is currently a Graduate Student (majoring in computer software and theory) with Wenzhou University, China. Her research interests are data mining, machine learning, and evolutionary computation and their applications to medical diagnosis.

**CHENGWEN WU** received the Ph.D. degree from the Department of Computer Science and Technology, Zhejiang University, China. He is currently a Lecturer with the Department of Computer Science and Technology, Wenzhou University, China. His current research interests center on machine learning and data mining. He has published over 20 papers in international journals and conference proceedings.

**CHENGYE LI** received the M.D. degree from the School of medicine, Shanghai Jiao Tong University, China. He is currently an Attending Physician with the Department of Pulmonary and Critical Care Medicine, Wenzhou Medical University, China. His research interests are data mining and machine learning, as well as their applications to medical diagnosis.