Applied Soft Computing Journal 86 (2020) 105884

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

An enhanced Bacterial Foraging Optimization and its application for training kernel extreme learning machine

Huiling Chen, Qian Zhang, Jie Luo, Yueting Xu, Xiaoqin Zhang *

College of Computer Science and Artificial Intelligence, Wenzhou University, Wenzhou, 325035, China

ARTICLE INFO

Article history:
Received 17 October 2018
Received in revised form 3 July 2019
Accepted 17 October 2019
Available online 31 October 2019

Keywords:
Bacterial Foraging Optimization
Chaotic local search
Gaussian mutation
Kernel extreme learning machine

ABSTRACT

The Bacterial Foraging Optimization (BFO) algorithm is a swarm intelligent algorithm widely used in various optimization problems. However, BFO suffers from multiple drawbacks, including slow convergence speed, inability to jump out of local optima and fixed step length. In this study, an enhanced BFO with chaotic chemotaxis step length, Gaussian mutation and chaotic local search (CCGBFO) is proposed for overcoming the existing weakness of original BFO. First, a chaotic chemotaxis step length operation is used to produce adaptive chemotaxis step length. Then, by combining the optimal position in the current bacteria with the Gaussian mutation operation to make full use of the information of the optimal position. Finally, a chaotic local search is introduced into the chemotaxis step to ensure that the algorithm can explore a large search space in the early stage. The performance of CCGBFO was evaluated on a comprehensive set of numerical benchmark functions including IEEE CEC2014 and CEC2017 problems. In addition, CCGBFO was also used to tune the key parameters of kernel extreme learning machine for dealing with the real-world problems. The experimental results show that the proposed CCGBFO significantly outperforms the original BFO in terms of both convergence speed and solution accuracy.

1. Introduction

Swarm intelligence refers to the characteristics that many simple individuals exhibit complex intelligence behaviors through cooperation. The study of swarm intelligence optimization algorithms, which simulate the search and optimization process into the evolution or foraging process of individuals, has received considerable attention from swarm intelligence researchers. Compared with the traditional gradient optimization algorithms [1], swarm intelligence optimization algorithms have more advantages. In recent years, several swarm intelligence optimization algorithms have emerged, including Genetic Algorithm (GA) [2,3], Particle Swarm Optimization (PSO) [4–6], and Ant Colony Optimization (ACO) [7–9], Moth-flame Optimizer (MFO) [10,11], Harris Hawk Optimizer (HHO) [12,13], Whale Optimization Algorithm (WOA) [14–17], Fruit fly optimization algorithm (FOA) [18,19]. The Bacterial Foraging Optimization (BFO) algorithm, proposed by Passino in 2002 [20], is a new swarm intelligent algorithm based on the foraging behavior of Escherichia coli (E. coli). BFO possesses a series of advantages including insensitivity to initial values and parameter selection, strong robustness, simplicity, ease of implementation, parallel processing and global search. Therefore, the BFO algorithm has been applied to a wide range of optimization problems such as color image enhancement [21], transmission loss reduction [22], active power filter synthesis [23], and machine learning.

The parameter tuning of chemotaxis step in BFO has been widely studied in the existing works. In [24], an improved BFO was proposed, which decreases the chemotaxis step linearly by decreasing its run-length unit linearly. Biswas et al. [25] used simple mathematical analysis of reproductive operations in BFO. In [26], in order to improve convergence speed and fine tune the search in multidimensional space, a linear variation and a nonlinear variation of the chemotaxis step were introduced. In [27], an adaptive BFO by adjusting run-length adaptively was used for an active noise control system. In [28], based on the chemotaxis operation, the influence of step length on the efficiency and accuracy of the algorithm was analyzed in detail and an adaptive adjustment algorithm for step length was proposed. In [29], Tan et al. proposed a variant of BFO with time-varying chemotaxis step length and comprehensive learning strategy. Yang et al. [30] proposed an improved BFO using new designed chemotaxis and conjugation strategies.

BFO has also been combined with other optimization methods [31–33]. In [34], Praveena et al. proposed a heuristic optimization method that combines BFO, PSO and Differential Evolution (DE). In [35], a new hybrid approach, consisting of PSO and BFO, was developed for the optimization of Power System

* Corresponding author.
E-mail address: zhangxiaoqinnan@gmail.com (X. Zhang).

https://doi.org/10.1016/j.asoc.2019.105884
1568-4946/

Stabilizers (PSSs) design in a multimachine power system. In [36], a novel algorithm with quantum behavior was proposed by combining quantum mechanics with BFO. Shao et al. [37] obtained a hybrid algorithm by mixing BFO with the Tabu Search (TS) algorithm, which is a potential method for pattern discovery. In [38], the authors introduced the use of DE and random search operators to enhance BFO. In [39], a gravitational search strategy was first integrated into the chemotaxis step, and then a swarm diversity strategy was incorporated into the reproduction process.

In this paper, an enhanced BFO called CCGBFO is proposed, and the main contributions are three-fold: First, a chaotic chemotaxis step length operation is incorporated into BFO to improve the original fixed step length. Therefore, the convergence speed and accuracy of the algorithm can be improved through adaptive variable chemotaxis step length. Then, by combining the optimal position in the current bacteria with the Gaussian mutation operation, the information of the optimal position in the current population is fully used, thereby increasing the diversity of the population and avoiding local optima in favor of global search. Finally, a chaotic local search with "shrinking" strategy operation is introduced into the chemotaxis step to ensure that the proposed algorithm can explore a large search space in the early stage, thereby avoiding falling into local optima, and that the search range can decrease in the later stage, thereby rendering the proposed algorithm converge to an optimal solution.

The work in [40] shares the most similarity to our work, but there are many differences between them: in [40], the DE and the chaotic local search operators are added to the chemotaxis step of the original BFO. However, although both their algorithm and our proposed method CCGBFO use chaotic local search in the chemotaxis step and logistic chaotic mapping to generate chaotic sequences, they differ greatly in the utilization of chaotic local search. Specifically, their algorithm applies chaotic local search for the entire population on the position of the current bacterium, while CCGBFO applies chaotic local search with a 'shrinking' strategy to the optimal position in the current bacterial population.

The remainder of this paper is organized as follows. Section 2 explains the original BFO. The chaos theory and Gaussian mutation are introduced in Section 3. In Section 4, CCGBFO is proposed. In Sections 5 and 6, the comprehensive results are presented. In Section 7, the application to KELM is described. In Section 8, conclusions and future work are summarized.

2. Bacterial foraging optimization (BFO)

BFO [20] is based on the foraging behavior of *E. coli*. In order to find a rich food source and avoid toxic areas, *E. coli* swims or tumbles by using its flagella, which can rotate in a clockwise or counterclockwise direction according to the environment.

According to natural selection (the survival of the fittest), bacteria with poor fitness will be eliminated, while bacteria with higher fitness will reproduce themselves. In addition, environmental changes can also lead to the extinction or proliferation of bacteria in the area. The two basic movements of bacteria: swimming and tumbling, are shown in Fig. 1.

By simulating the foraging behavior of *E. coli*, BFO is composed mainly of chemotaxis, swarming, reproduction, and elimination-dispersal operations.

(1) Chemotaxis: Chemotaxis is the search behavior of bacteria. When bacteria search for food, they produce two kinds of movement. One is swimming, which allows the bacterium to move in a certain direction; the other is tumbling, which changes the direction of movement. In the process of chemotaxis, bacteria will sense the level of nutrient content in the surrounding environment and thus guide the chemotaxis behavior. The chemotaxis operation of the ith bacterium at each step is shown as follow:

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

where $\theta^i(j,k,l)$ indicates the position of the ith bacterium after jth chemotaxis, kth reproduction, and lth elimination-dispersal operations, $C(i)$ is the moving step size of the ith bacterium, and Δ represents a unit vector in a random direction.

(2) Swarming: Bacteria interact with each other during the process of foraging. If the concentration of nutrients in a certain area is high, bacteria will send out signals and cause bacteria to accumulate at this area. Similarly, if the concentration of nutrients in a certain area is low, signals are also sent out in order that bacteria can be kept at a distance. This interaction can speed up their search for areas with high concentrations of nutrients.

(3) Reproduction: During the entire chemotaxis process, according to different fitness values, which areas are suitable for the survival of bacteria can be determined. Then 1/2 of the bacteria in the place where the nutrients are relatively poor are eliminated, and the other 1/2 bacteria self-replicate. Therefore, the number of bacteria is not changed in the process of reproduction. However, the convergence speed of the algorithm can be accelerated after the reproduction operation.

(4) Elimination-Dispersal: There may be a risk of local optimal solutions in the chemotaxis process. Therefore, the elimination-dispersal operation should occur in the process of bacterial foraging. Bacteria disappear at the current position with the probability of *Ped*, and appear in a random place. This process can help bacteria to forage in a new area and also help the algorithm to search for the global optimal solution.

3. Chaos theory and Gaussian mutation

3.1. Chaos theory

Over the past decade, the chaos theory has been widely applied to different fields of science such as chaos control [41], feature selection [42] and parameter optimization [43]. Chaotic sequences are characterized by their sensitive dependence on initial conditions, randomicity and ergodicity. In recent years, chaotic sequences have been introduced into various metaheuristic optimization algorithms.

Previous studies have shown that if chaos variables are used instead of random variables, there is a better chance to reach the promising searching area. The basic idea of chaos optimization is (1) to use a similar carrier method to introduce chaos states into the optimization variables, (2) to enlarge the traversal range of the chaos motions to the scope of the optimization variables to form the corresponding chaotic variables, and (3) to utilize the chaotic variables to search to make the search process more effective.

The improved method generates chaotic sequences using logistic mapping as shown in Eq. (2):

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

where μ is a control parameter, and it is set $\mu = 4$. Set$0 < ch_1 < 1$, and $ch_1 \neq 0.250.50.751$. It is not difficult to prove that when $\mu = 4$, the system is completely in chaos. S is the number of bacterial populations.

Fig. 2 shows the histogram distribution graph 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 1with uniform distribution) with 10^4 iterations. As can be seen from Fig. 2, the logistic map and

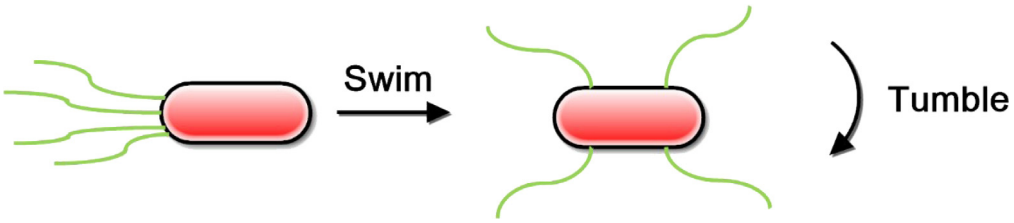

Fig. 1. Swimming and tumbling of a bacterium.

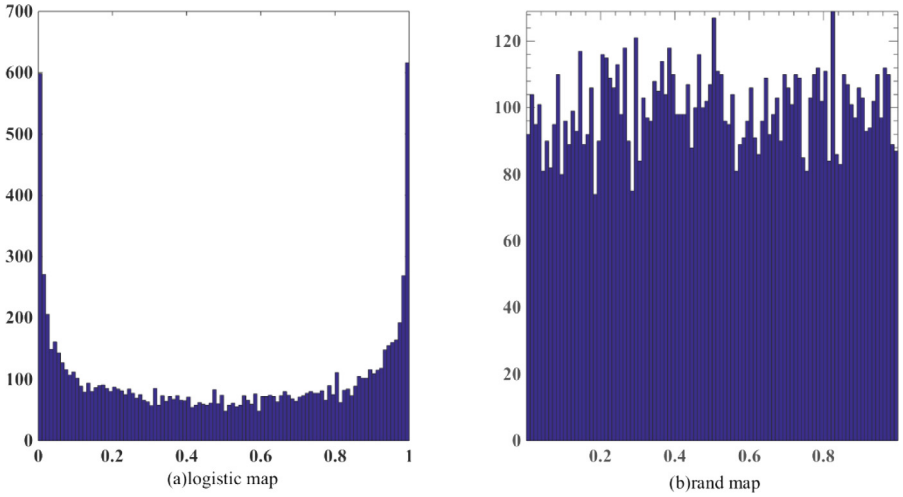

Fig. 2. Histogram distribution graph of (a) the logistic map and (b) the random map.

the random map are both located between the interval 0 and 1 but show completely different distributions. Specifically, between the interval 0 and 1, the logistic map has a greater probability of generating values near 0 and 1 than the random map. It is precisely because of this unique feature that the logistic map provides a different search range from the random map in local search, which helps the logistic map make local search faster than the random map.

Owing to its ergodicity and randomicity, chaotic search is effective in local optimization [11,15,16,44]. However, when the search space for exploring becomes larger, its performance decreases. In order to overcome this drawback, chaotic local search is introduced, which can make the search process avoid premature convergence and local optima stagnation. In [45], chaotic local search was incorporated into PSO to construct a chaotic PSO (CPSO). In [46], by using a chaotic local search with a 'shrinking' strategy, an effective memetic DE algorithm called DECLS was proposed. In [47], in order to improve solution accuracy and convergence speed, chaotic local search was integrated into the reduced Symbiotic Organisms Search.

3.2. Gaussian mutation

Gaussian mutation is known as adding a random vector obeying the Gaussian distribution to the state of the original individual. The Gaussian distribution (i.e. Normal distribution) plays a decisive role in mathematical statistics and its probability theory. If the random variable X obeys a Gaussian distribution with the mathematical expectation of μ and the variance of σ^2, it can be denoted as $N(\mu, \sigma^2)$. Its probability density function is shown in Eq. (3):

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{3}$$

The expected value μ determines its position, and the standard deviation σ determines the magnitude of the distribution. Many phenomena and random factors in the natural world can be described approximately with a Normal distribution.

4. Proposed method

Our proposed algorithm CCGBFO combines three operations. Specifically, a chaotic chemotaxis step length operation is incorporated into BFO to replace the original fixed chemotaxis step length, and then a Gaussian mutation operation and a chaotic local search with "shrinking" strategy operation are successively introduced into the chemotaxis step, and are allowed to operate only on the optimal position in the current bacterial population because applying these strategies on a general bacterial individual may slow down the search process and waste unnecessary iteration cycles. It should be noted that the 'shrinking' strategy used in this study was proposed in [46].

4.1. Chaotic chemotaxis step length operation

Step 1: Introduce the chaos theory, as shown in Eq. (2), generates chaotic sequences *ch* using the logistic map, and make the fixed step length C_i of bacteria i in the chemotaxis process map to the chaotic sequence to form a new chaotic chemotaxis step length C_i as shown in Eq. (4):

$$C_i = ch_i * C_i \quad i = 1, \ldots, S \tag{4}$$

Step 2: As shown in Eq. (5), sort the obtained chaotic chemotaxis step length in descending order to ensure that the bacterium can interact with the surrounding environment to choose the chemotaxis step length adaptively to prevent it from falling into the local optimal predicament.

$$C = \text{sort}\left(C, '\text{descend}'\right) \tag{5}$$

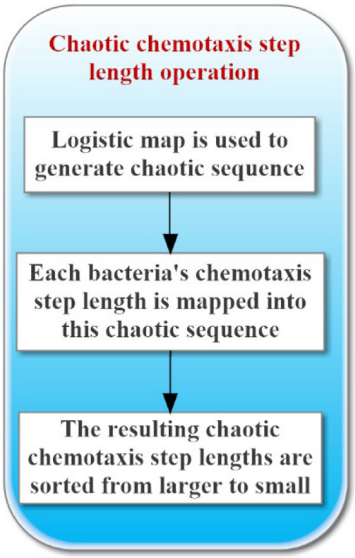

Fig. 3. Overall procedure of the chaotic chemotaxis step length operation.

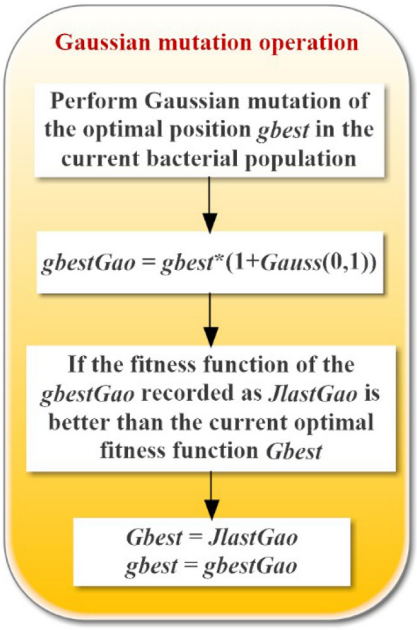

Fig. 4. Overall procedure of the Gaussian mutation operation.

The flowchart of the chaotic chemotaxis step length operation is shown in Fig. 3.

4.2. Gaussian mutation operation

Step 1: Apply Gaussian mutation to the optimal position *gbest* in the current bacterial population to generate the mutated position *gbestGao*. The expression is shown in Eq. (6):

$$gbestGao = gbest * (1 + \text{Gauss}(0, 1)) \tag{6}$$

where Gauss (0,1) is the standard Normal distribution.

Step 2: If the fitness function of the mutated position *gbestGao* is better than *Gbest* (*Gbest* represents the optimal fitness function in the current bacterial population), replace *Gbest* with the fitness of *gbestGao*, and *gbest* is updated with *gbestGao*.

Eq. (6) increases the Gaussian distribution random disturbance term $gbest * \text{Gauss}(0, 1)$on the basis of *gbest*, which makes full use of the information disturbance of the optimal position *gbest* in the current bacterial population. Therefore, Eq. (6) can not only enable bacteria to jump out of local optima and converge to global optima, but also improve convergence speed.

The flowchart of the Gaussian mutation operation is shown in Fig. 4.

4.3. Chaotic local search with "shrinking" strategy operation

Step 1: The chaotic variable ch_i is acquired through Eq. (2), and then ch_i is mapped to the search space [*lb*, *ub*] of current bacterial position to obtain a new vector CH_i, as shown in Eq. (7):

$$CH_i = lb + ch_i * (ub - lb) \quad i = 1, \ldots, S \tag{7}$$

Step 2: Combine the obtained $CH_i$with the optimal position *gbest* in the current bacterial population to form a new optimal individual position vector *sol* through chaotic local search, as shown in Eq. (8):

$$sol = (1 - setCan) * gbest + setCan * CH_i \quad i = 1, \ldots, S \tag{8}$$

where *setCan* is the shrinking factor, determined by Eq. (9):

$$setCan = 1 - \left| \frac{Intertime - 1}{Intertime} \right|^m \tag{9}$$

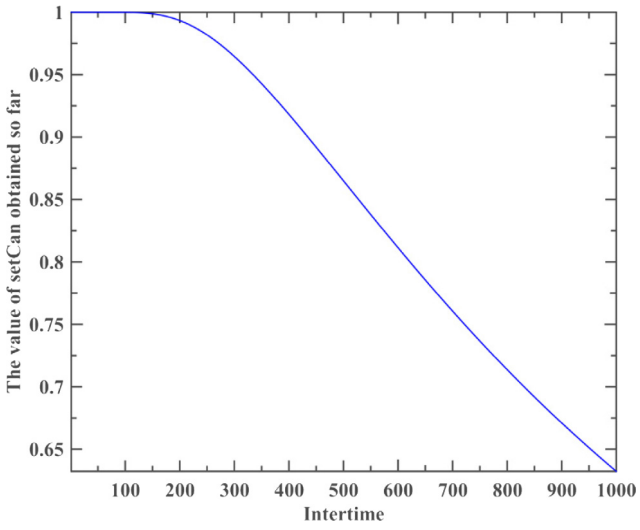

Fig. 5. Convergence curve of the value obtained from the *setCan* vector.

where *Intertime* represents the current iteration number of the algorithm and *m* masters the shrinking speed. The higher the *m* value, the slower the shrinking speed.

According to Eq. (9), we assume that $m = 1000$, the maximum number of iterations of the algorithm is 1000, and after 1000 iterations of the loop, the image of the value obtained from the *setCan* vector is shown in Fig. 5.

As can be clearly seen from Fig. 5, the value of the shrinking factor *setCan* gradually decreases as the number of iterations increases. Eq. (8) shows that the smaller the value of *setCan is*, the smaller the corresponding chaotic search scope becomes. Analysis of Fig. 5 shows that *setCan* is relatively large in the early stages of iterations, which helps to expand the range of search and increase the diversity of the population, while in the later stages of iterations, *setCan* becomes smaller, which helps to avoid falling into local optima and converge to global optima.

Step 3: If the fitness function value of the new optimal individual position vector *sol* is superior to *Gbest* (*Gbest* represents

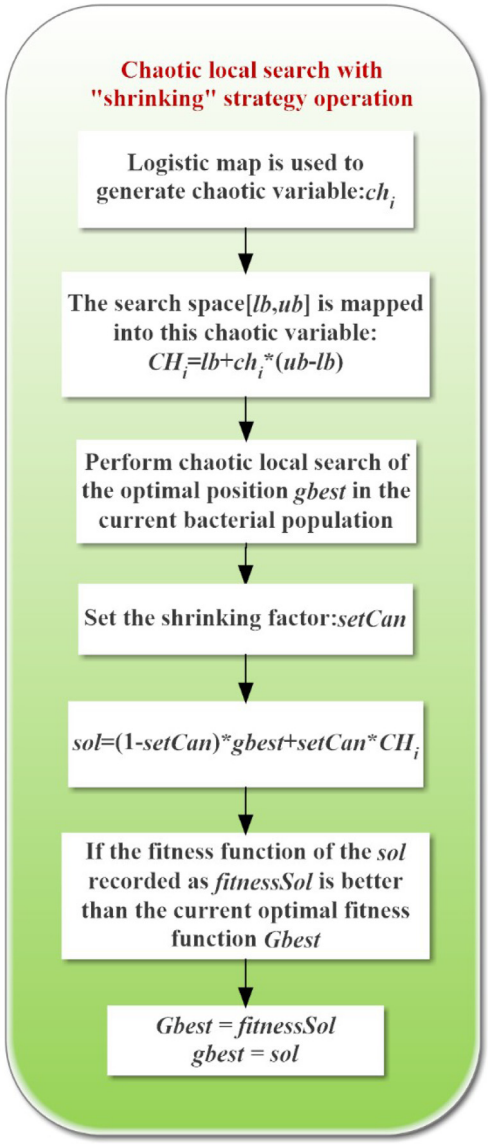

Fig. 6. Overall procedure of the chaotic local search with "shrinking" strategy operation.

the current optimal fitness function of the bacterial population), *Gbest* is replaced with the fitness of *sol* and *gbest* (*gbest* describes the optimal position in the current bacterial population) is re-recorded as *sol*. If the length of the chaotic sequence reaches *S*, chaotic local search ends; otherwise, skip to Step 1 to continue execution.

The flowchart of the chaotic local search with "shrinking" strategy operation is shown in Fig. 6.

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

The flowchart of the proposed CCGBFO is shown in Fig. 7.

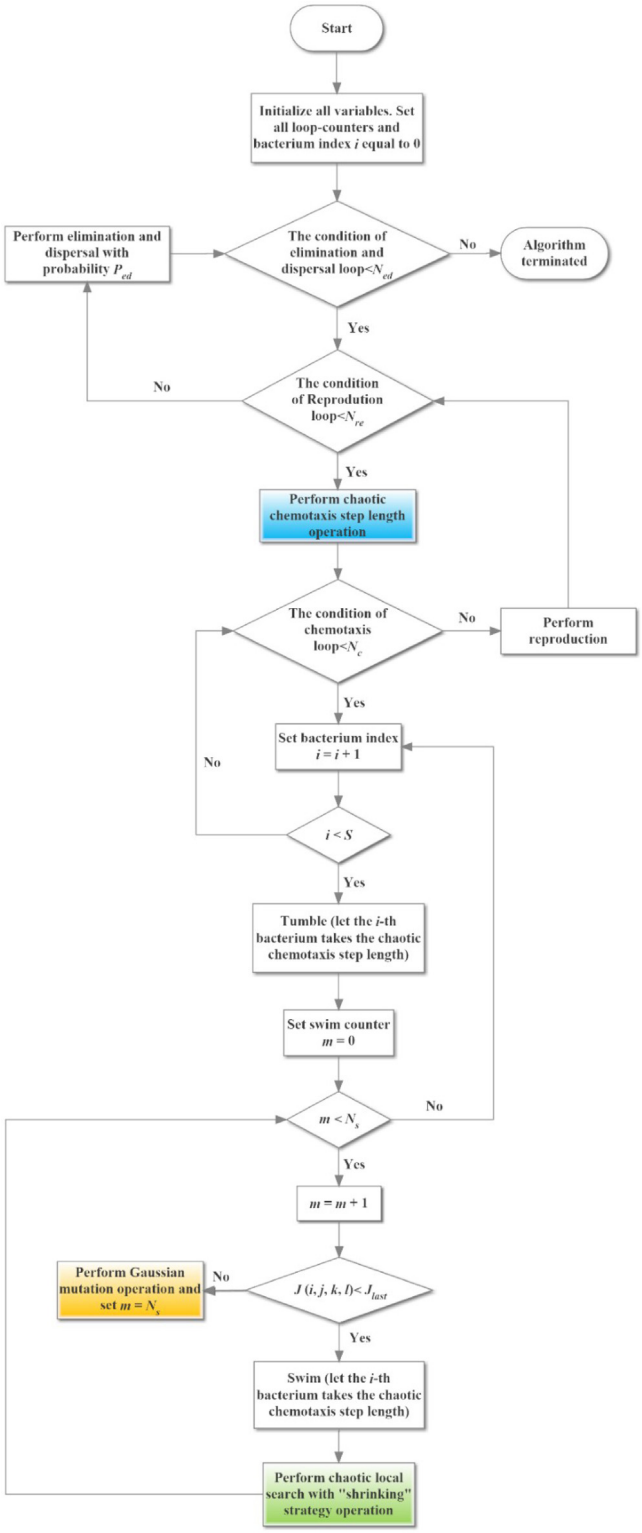

Fig. 7. Overall procedure of the proposed CCGBFO.

5. Comparative study on 23 numerical well-known benchmark functions

It is worth noting that all experiments involved in this paper were implemented in MATLAB R2014b software with Intel (R) Xeon (R) CPU E5-2660 v3 (2.60 GHz) and 16GB RAM configuration under Windows Server 2008 R2 operating system.

5.1. Benchmark function validation

The performance of the proposed algorithm CCGBFO was evaluated on 23 classical benchmark functions, which are, divided into three groups: unimodal, multimodal, or fixed-dimension multimodal, as listed in Table 1 where Dim indicates the dimension of the function, Range denotes the boundary of the

The main steps of CCGBFO

Begin

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, the current optimal value $Gbest$.

Main loop

For $ell = 1 : Ned$ /*Elimination and dispersal loop*/

For $K = 1 : Nre$ /*Reproduction loop*/

Perform chaotic chemotaxis step length operation described in 4.1

For $j = 1 : Nc$ /* chemotaxis loop*/

$Intertime = Intertime+1$; /* represent the number of iterations*/

For $i = 1 : S$

/*$fobj$ represents calculating the fitness of the i-th bacterium at the j-th chemotaxis, Kth reproductive, and ell-th elimination-dispersal steps.*/

$J(i,j,K,ell) = fobj(P(:,i,j,K,ell))$;

/* $Jlast$ stores the current optimal fitness.*/

$Jlast = J(i,j,K,ell)$;

/* $gbest$ represents the optimal position in the current bacterial population. */

$gbest(1,:) = P(:,i,j,K,ell)$;

Tumble according to Eq. (1)

/*Swim (for bacteria that seem to be headed in the right direction)*/

$m = 0$; /* Initialize counter for swim length*/

While $m < Ns$

$m = m+1$;

If $J(i,j+1,K,ell) < Jlast$

/* $Jlast$ stores the current optimal fitness.*/

$Jlast=J(i,j+1,K,ell)$;

Tumble according to Eq.(1)

If $Jlast < Gbest$

$Gbest = Jlast$;

$gbest(1,:) = P(:,i,j+1,K,ell)$;

End

Else

Perform Gaussian mutation operation described in 4.2

$m = Ns$;

End

Perform chaotic local search with "shrinking" strategy operation described in 4.3

End

End /*Go to next bacterium*/

/* The GGB vector saves the optimal value $Gbest$ for the current iteration.*/

$GGB(Intertime) = Gbest$;

End /*Go to the next chemotaxis*/

/*Reproduction*/

$Jhealth = sum(J(:,:,K,ell), 2)$; /* Set the health of each of the S bacteria*/

$[Jhealth, sortind] = sort(Jhealth)$; /*Sorts the nutrient concentration in order of ascending*/

/* Rearrange the bacterial population*/

$P(:,:,1,K+1,ell) = P(:,sortind,Nc+1,K,ell)$;

/*Keeps the chemotaxis parameters C with each bacterium at the next generation*/

$C(:,K+1) = C(sortind,K)$;

/*Split the bacteria (reproduction)*/

For $i = 1 : Sr$

/*The least fit do not reproduce, the most fit ones split into two identical copies*/

$P(:,i+Sr,1,K+1,ell) = P(:,i,1,K+1,ell)$;

$C(i+Sr,K+1) = C(i,K+1)$;

End

End /*Go to next reproduction*/

/*Elimination-Dispersal*/

For $m = 1 : S$

If $Ped > rand$ /*randomly generates a new individual anywhere in the solution space.*/

Reinitialize bacteria *m*

End

End

End /*Go to next Elimination-Dispersal*/

End

Table 1
Classical benchmark functions.

Group	Function	Dim	Range	f_{min}
Unimodal	$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]$	0
	$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[-10, 10]$	0
	$f_3(x) = \sum_{i=1}^{n} (\sum_{j-1}^{i} x_j)^2$	30	$[-100, 100]$	0
	$f_4(x) = \max_i \{\lvert x_i \rvert, 1 \le i \le n\}$	30	$[-100, 100]$	0
	$f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	$[-30, 30]$	0
	$f_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	30	$[-100, 100]$	0
	$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	30	$[-1.28, 1.28]$	0
Multimodal	$f_8(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30	$[-500,500]$	-418.9829×5
	$f_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	$[-5.12,5.12]$	0
	$f_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	30	$[-32,32]$	0
	$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$[-600,600]$	0
	$f_{12}(x) = \frac{\pi}{n}\left\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2\left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m)\begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	30	$[-50,50]$	0
	$f_{13}(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2\left[1 + \sin^2(3\pi x_i + 1)\right] + (x_n - 1)^2\left[1 + \sin^2(2\pi x_n)\right]\right\}$ $+ \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	$[-50,50]$	0
Fixed-dimension multimodal	$f_{14}(x) = \left(\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$	2	$[-65,65]$	1

(*continued on next page*)

Table 1 (*continued*).

Group	Function	Dim	Range	f_{min}
	$f_{15}(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1\left(b_i^2+b_ix_2\right)}{b_i^2+b_ix_3+x_4}\right]^2$	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
	$f_{18}(x)=\left[1+(x_1+x_2+1)^2\left(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2\right)\right]\times\left[30+(2x_1-3x_2)^2\times\left(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2\right)\right]$	2	[−2,2]	3
	$f_{19}(x)=-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{3}a_{ij}\left(x_j-p_{ij}\right)^2\right)$	3	[1,3]	−3.86
	$f_{20}(x)=-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{6}a_{ij}\left(x_j-p_{ij}\right)^2\right)$	6	[0,1]	−3.32
	$f_{21}(x)=-\sum_{i=1}^{5}\left[(X-a_i)(X-a_i)^T+c_i\right]^{-1}$	4	[0,10]	−10.1532
	$f_{22}(x)=-\sum_{i=1}^{7}\left[(X-a_i)(X-a_i)^T+c_i\right]^{-1}$	4	[0,10]	−10.4028
	$f_{23}(x)=-\sum_{i=1}^{10}\left[(X-a_i)(X-a_i)^T+c_i\right]^{-1}$	4	[0,10]	−10.5363

Table 2
Parameter settings for the algorithms used.

Method	Population size	Maximum generation	Other parameters
BFO	40	1000	$\Delta\in[-11]$
BA	40	1000	*Q Frequency*∈[0 2]; *A Loudness*: 0.5; *r Pulse rate*: 0.5
DA	40	1000	$w\in[0.9\ 0.2]$; $s=0.1$; a $=0.1$; $c=0.7$; $f=1$; $e=1$
FA	40	1000	$\beta_0=1$; $\alpha\in[0\ 1]$; $\gamma=1$
FPA	40	1000	*switch probability* $p=0.8$; $\lambda=1.5$
GOA	40	1000	$cMax=1$; $cMin=0.00004$
GWO	40	1000	$a\in[2\ 0]$
MFO	40	1000	$a\in[-1-2]$; $b=1$
ABC	40	1000	*limit* $=300$
PSO	40	1000	*inertial weight* $=1$; $c_1=2$; $c_2=2$
DE	40	1000	*Scaling Factor* $=[0.2\ 0.8]$; *Crossover Probability* $=0.2$

function's search space, and f_{min} represents the theoretical optimum.

5.2. Comparison of CCGBFO with well-known optimization algorithms

In order to evaluate its performance, CCGBFO was compared with 7 well-known meta-heuristic optimization algorithms including PSO [4], DE [48], Bat Algorithm (BA) [49], FA [50], ABC [51], Flower Pollination Algorithm (FPA) [52] and Grey Wolf Optimizer (GWO) [53]. Furthermore, CCGBFO was also compared with the original BFO [20] and three recently developed meta-heuristics: MFO [54], Dragonfly Algorithm (DA) [55], and Grasshopper Optimization Algorithm (GOA) [56]. The parameters of the algorithms mentioned above were set according to the parameter values of their original papers. Table 2 describes the specified values of parameters used in each algorithm. Moreover, 30 independent runs were performed on each benchmark function to ensure fair and unbiased results. In addition, the number of population size and the maximum number of iterations in all the experiments were set to 40 and 1000, respectively.

The average results (Avg), standard deviation (Stdv), and overall ranks of different algorithms in dealing with the F1–F23 functions are presented in Table 3. It should be noted that the ranking is based on the average results (Avg) of 30 independent runs. As can be seen from Table 3, on 6 of the 7 unimodal functions (excepting F6), CCGBFO achieves better results than the original BFO and other algorithms. On the six multimodal functions with the exception of F13, it can be seen that CCGBFO obtains very competitive solutions compared to other algorithms. On F13, although DE and ABC are more competitive than CCGBFO, CCGBFO outperforms the other nine algorithms. Regarding the ten fixed-dimension multimodal functions, CCGBFO obtains the exact optimal solutions on F15; and in dealing with the other nine functions (F14, F16, F17, F18, F19, F20, F21, F22 and F23), although CCGBFO and some algorithms can obtain the optimal value at the same time, the optimization performance of CCGBFO is still improved compared to the original BFO. For instance, on F14, CCGBFO, DA, FPA, GOA, ABC and DE can almost simultaneously achieve the optimal convergence effect; however, CCGBFO outperforms the original BFO. It can also be seen from Table 3 that, based on the overall rank, CCGBFO is the best algorithm, followed successively by DE, ABC, GWO, BFO, FA, FPA, GOA, BA, MFO, DA and PSO.

Fig. 8 shows the convergence curves of CCGBFO and other algorithms on the 12 selected benchmark problems (F1, F2, F3, F4, F8, F9, F11, F12, F14, F15, F22 and F23). In dealing with F1, F2, F3, F4 and F11, the convergence curves are very similar. After about 500 iterations, CCGBFO converges very quickly to the optimal location towards global values. The difference is that on F1, F2 and F4, GWO converges faster than all the other algorithms at the beginning of iteration. On F3, GWO also shows a very fast convergence speed throughout early steps; however, after about 500 iterations, both CCGBFO and BFO have a significant downward trend, but CCGBFO is faster because BFO falls into local optima in the later stage of the search. On F11, as the number

Table 3
Results obtained by CCGBFO and other well-known algorithms on the 23 benchmark functions(F1–F23).

F		CCGBFO	BFO	BA	DA	FA	FPA	GOA	GWO	MFO	ABC	PSO	DE
F1	Avg	0	7.99E−03	1.42E+01	7.78E+02	2.83E−03	1.48E+02	9.19E−01	2.63E−65	1.67E+03	2.59E−11	1.21E+02	4.03E−12
	Stdv	0	4.11E−03	1.39E+00	4.42E+02	8.65E−04	4.84E+01	7.54E−01	5.76E−65	3.79E+03	3.14E−11	1.38E+01	1.48E−12
	Rank	1	6	8	11	5	10	7	2	12	4	9	3
F2	Avg	0	3.38E−01	2.01E+01	1.19E+01	1.15E−01	2.07E+01	7.97E+00	2.62E−38	3.07E+01	1.03E−06	9.23E+01	4.31E−08
	Stdv	0	9.42E−02	1.58E+01	5.58E+00	5.83E−02	6.09E+00	1.88E+01	2.14E−38	1.95E+01	4.59E−07	1.04E+02	1.24E−08
	Rank	1	6	9	8	5	10	7	2	11	4	12	3
F3	Avg	0	3.10E−13	5.15E+01	8.35E+03	7.67E+02	3.74E+02	1.35E+03	1.03E−17	1.53E+04	1.25E+04	3.76E+02	2.59E+04
	Stdv	0	6.80E−13	1.36E+01	6.38E+03	3.09E+02	9.83E+01	8.81E+02	4.64E−17	1.12E+04	2.92E+03	9.00E+01	3.79E+03
	Rank	1	3	4	9	7	5	8	2	11	10	6	12
F4	Avg	0	2.94E−02	1.78E+00	1.73E+01	6.21E−02	1.52E+01	8.84E+00	4.82E−16	6.03E+01	2.32E+01	4.40E+00	2.03E+00
	Stdv	0	4.62E−03	3.04E−01	6.56E+00	1.53E−02	1.92E+00	3.24E+00	6.32E−16	1.12E+01	4.13E+00	2.67E−01	3.61E−01
	Rank	1	3	5	10	4	9	8	2	12	11	7	6
F5	Avg	3.50E−04	6.55E+04	3.39E+03	8.59E+04	9.69E+01	7.20E+03	4.25E+02	2.64E+01	1.24E+04	2.96E+00	1.35E+05	5.88E+01
	Stdv	4.03E−04	0	1.21E+03	9.88E+04	1.33E+02	6.27E+03	3.92E+02	7.74E−01	3.10E+04	3.62E+00	3.49E+04	3.20E+01
	Rank	1	10	7	11	5	8	6	3	9	2	12	4
F6	Avg	7.94E−06	1.54E−01	1.44E+01	5.97E+02	2.86E−03	1.47E+02	1.19E+00	4.66E−01	3.33E+03	3.50E−11	1.24E+02	4.78E−12
	Stdv	1.30E−05	2.34E−02	1.87E+00	4.02E+02	1.06E−03	5.19E+01	1.13E+00	3.22E−01	6.58E+03	4.20E−11	1.59E+01	2.64E−12
	Rank	3	5	8	11	4	10	7	6	12	2	9	1
F7	Avg	6.33E−05	3.81E−03	1.16E+01	2.36E−01	4.32E−03	1.44E−01	1.23E−02	7.56E−04	2.06E+00	1.34E−01	1.02E+02	2.49E−02
	Stdv	6.02E−05	2.45E−03	7.51E+00	1.26E−01	1.55E−03	3.91E−02	4.04E−03	4.02E−04	5.63E+00	3.01E−02	2.71E+01	4.63E−03
	Rank	1	3	11	9	4	8	5	2	10	7	12	6
F8	Avg	−1.29E+04	−2.86E+03	−7.12E+03	−5.80E+03	−6.92E+03	−8.13E+03	−7.44E+03	−6.06E+03	−8.94E+03	−1.22E+04	−6.80E+03	−1.25E+04
	Stdv	9.18E+02	4.95E+02	5.96E+02	6.62E+02	6.21E+02	2.08E+02	8.17E+02	6.36E+02	8.38E+02	1.43E+02	6.27E+02	2.60E+02
	Rank	1	12	7	11	8	5	6	10	4	3	9	2
F9	Avg	−2.90E+02	−2.89E+02	2.61E+02	1.40E+02	3.39E+01	1.14E+02	8.98E+01	9.02E−01	1.49E+02	5.09E−01	3.66E+02	5.90E+01
	Stdv	9.31E−04	5.26E−01	2.33E+01	3.18E+01	1.35E+01	1.93E+01	3.46E+01	2.83E+00	3.61E+01	5.27E−01	2.12E+01	5.66E+00
	Rank	1	2	11	9	5	8	7	4	10	3	12	6
F10	Avg	−1.07E+13	−9.04E+12	4.82E+00	7.92E+00	1.41E−02	9.87E+00	3.48E+00	1.52E−14	1.44E+01	2.60E−05	8.23E+00	5.25E−07
	Stdv	3.93E+09	7.90E+11	2.75E+00	2.00E+00	2.11E−03	1.21E+00	9.09E−01	2.18E−15	8.09E+00	1.37E−05	4.39E−01	1.53E−07
	Rank	1	2	8	9	6	11	7	3	12	5	10	4
F11	Avg	0	4.13E−03	5.59E−01	9.42E+00	3.64E−03	2.33E+00	5.33E−01	3.41E−03	1.20E+01	1.39E−03	1.02E+00	5.95E−11
	Stdv	0	1.45E−03	6.40E−02	5.03E+00	8.71E−04	4.29E−01	2.10E−01	6.81E−03	3.12E+01	3.58E−03	1.89E−02	6.22E−11
	Rank	1	6	8	11	5	10	7	4	12	3	9	2
F12	Avg	1.41E−14	2.26E−11	1.18E+01	5.36E+01	2.26E−05	4.29E+00	5.30E+00	3.47E−02	5.34E−01	2.77E−12	4.40E+00	4.48E−13
	Stdv	4.35E−14	3.38E−11	5.05E+00	1.59E+02	7.75E−06	8.21E−01	2.47E+00	2.10E−02	8.80E−01	7.91E−12	6.82E−01	3.20E−13
	Rank	1	4	11	12	5	8	10	6	7	3	9	2
F13	Avg	6.01E−07	9.92E−02	2.12E+00	2.28E+03	3.18E−04	2.01E+01	1.21E+01	3.41E−01	2.93E−01	5.68E−11	2.12E+01	2.92E−12
	Stdv	1.34E−06	1.75E−11	2.70E−01	7.06E+03	1.35E−04	5.35E+00	1.47E+01	1.77E−01	9.28E−01	1.61E−10	3.05E+00	1.60E−12
	Rank	3	5	8	12	4	10	9	7	6	2	11	1
F14	Avg	9.98E−01	1.86E+00	4.60E+00	9.98E−01	1.32E+00	9.98E−01	9.98E−01	4.20E+00	1.49E+00	9.98E−01	2.48E+00	9.98E−01
	Stdv	3.62E−13	9.99E−01	3.58E+00	9.26E−11	4.46E−01	1.17E−08	3.94E−16	3.95E+00	1.26E+00	9.22E−17	2.10E+00	0
	Rank	1	4	7	1	2	1	1	6	3	1	5	1
F15	Avg	3.41E−04	5.14E−04	6.17E−03	2.90E−03	7.52E−04	4.70E−04	6.05E−03	2.98E−03	1.60E−03	7.40E−04	1.20E−03	7.08E−04
	Stdv	2.48E−05	1.32E−04	8.71E−03	4.93E−03	2.22E−04	1.10E−04	1.20E−02	6.93E−03	3.56E−03	1.86E−04	2.39E−04	1.97E−04
	Rank	1	3	12	9	6	2	11	10	8	5	7	4

(*continued on next page*)

Table 3 (*continued*).

F		CCGBFO	BFO	BA	DA	FA	FPA	GOA	GWO	MFO	ABC	PSO	DE
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	−1.03E+00
	Stdv	2.88E−13	1.55E−06	3.70E−04	1.30E−05	1.43E−05	2.40E−11	2.90E−14	3.92E−09	6.78E−16	6.32E−16	1.41E−03	6.78E−16
	Rank	1	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	3.98E−01	3.98E−01	3.98E−01	3.98E−01
	Stdv	7.31E−13	6.20E−07	2.02E−04	6.64E−08	6.65E−10	1.79E−15	9.91E−15	3.30E−07	0	0	6.34E−04	0
	Rank	1	1	1	1	1	1	1	1	1	1	1	1
F18	Avg	3.00E+00	3.00E+00	3.03E+00	3.00E+00	3.00E+00	3.00E+00	5.70E+00	3.00E+00	3.00E+00	3.00E+00	3.08E+00	3.00E+00
	Stdv	1.84E−10	8.55E−05	2.69E−02	1.34E−05	1.58E−08	1.75E−12	1.48E+01	4.59E−06	1.11E−15	1.12E−04	7.28E−02	2.03E−15
	Rank	1	1	2	1	1	1	4	1	1	1	3	1
F19	Avg	−3.86E+00	−3.86E+00	−3.84E+00	−3.86E+00	−3.86E+00	−3.86E+00	−3.84E+00	−3.86E+00	−3.86E+00	−3.86E+00	−3.85E+00	−3.86E+00
	Stdv	2.10E−07	2.65E−04	1.91E−02	6.53E−04	4.46E−10	8.72E−12	7.72E−02	2.89E−03	2.71E−15	2.26E−15	1.04E−02	2.71E−15
	Rank	1	1	3	1	1	1	3	1	1	1	2	1
F20	Avg	−3.32E+00	−3.29E+00	−2.92E+00	−3.23E+00	−3.27E+00	−3.32E+00	−3.27E+00	−3.25E+00	−3.23E+00	−3.32E+00	−2.95E+00	−3.32E+00
	Stdv	3.89E−08	1.55E−02	1.04E−01	7.04E−02	6.05E−02	5.87E−05	6.09E−02	8.25E−02	5.37E−02	1.55E−15	1.58E−01	5.18E−05
	Rank	1	2	7	5	3	1	3	4	5	1	6	1
F21	Avg	−1.02E+01	−1.01E+01	−5.40E+00	−7.96E+00	−8.97E+00	−1.02E+01	−5.64E+00	−9.82E+00	−7.62E+00	−1.02E+01	−4.97E+00	−9.95E+00
	Stdv	8.89E−09	6.29E−03	2.85E+00	2.55E+00	2.71E+00	8.42E−05	3.18E+00	1.28E+00	2.81E+00	5.34E−15	1.69E+00	9.40E−01
	Rank	1	2	9	6	5	1	8	4	7	1	10	3
F22	Avg	−1.04E+01	−1.02E+01	−5.72E+00	−8.72E+00	−1.04E+01	−1.04E+01	−6.18E+00	−1.04E+01	−6.85E+00	−1.04E+01	−4.90E+00	−1.04E+01
	Stdv	1.05E−05	9.61E−01	2.87E+00	2.61E+00	7.28E−07	2.11E−04	3.62E+00	2.57E−04	3.47E+00	2.70E−12	1.54E+00	1.44E−01
	Rank	1	2	6	3	1	1	5	1	4	1	7	1
F23	Avg	−1.05E+01	−1.05E+01	−5.27E+00	−7.94E+00	−1.05E+01	−1.05E+01	−6.46E+00	−1.03E+01	−8.41E+00	−1.02E+01	−4.71E+00	−1.05E+01
	Stdv	9.89E−06	9.17E−03	3.11E+00	3.07E+00	8.15E−07	6.57E−04	3.73E+00	1.48E+00	3.33E+00	1.37E+00	1.49E+00	1.61E−11
	Rank	1	1	7	5	1	1	6	2	4	3	8	1
Sum of ranks		27	85	160	166	89	123	137	84	163	75	177	67
Average rank		1.1739	3.6957	6.9565	7.2174	3.8696	5.3478	5.9565	3.6522	7.0870	3.2609	7.6957	2.9130
Overall rank		1	5	9	11	6	7	8	4	10	3	12	2

Table 4
The calculated *p*-values from the benchmark functions (F1–F23) for CCGBFO versus other optimizers.

F	BFO	BA	DA	FA	FPA	GOA	GWO	MFO	ABC	PSO	DE
F1	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
F2	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
F3	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
F4	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
F5	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
F6	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	2.88E−06	4.73E−06	1.73E−06	1.73E−06	1.73E−06
F7	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
F8	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
F9	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
F10	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
F11	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.56E−02	1.73E−06	1.73E−06	1.73E−06	1.73E−06
F12	1.92E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
F13	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
F14	1.92E−06	1.73E−06	**3.97E−01**	6.34E−06	1.92E−06	3.01E−05	1.73E−06	**5.24E−01**	5.46E−06	1.73E−06	5.46E−06
F15	1.92E−06	1.73E−06	1.73E−06	1.73E−06	7.69E−06	1.92E−06	1.48E−02	1.73E−06	1.73E−06	1.73E−06	1.73E−06
F16	1.73E−06	1.73E−06	**9.14E−01**	1.73E−06	4.73E−06	1.65E−03	1.73E−06	2.55E−06	2.55E−06	1.73E−06	2.55E−06
F17	1.73E−06	1.73E−06	**1.63E−01**	1.73E−06	4.45E−06	1.25E−04	1.73E−06	3.78E−06	3.78E−06	1.73E−06	3.78E−06
F18	1.73E−06	1.73E−06	**4.17E−01**	1.73E−06	**7.86E−02**	1.11E−03	1.73E−06	1.73E−06	2.96E−03	1.73E−06	1.73E−06
F19	1.73E−06	1.73E−06	5.22E−06	3.18E−06	1.73E−06	2.60E−05	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
F20	1.73E−06	1.73E−06	1.73E−06	4.28E−02	1.73E−06	4.72E−02	1.73E−06	5.31E−05	1.73E−06	1.73E−06	3.11E−05
F21	1.73E−06	1.73E−06	2.88E−06	1.73E−06	1.73E−06	1.15E−04	1.73E−06	4.72E−02	1.73E−06	1.73E−06	9.63E−04
F22	1.73E−06	1.73E−06	1.92E−06	3.11E−05	3.52E−06	1.48E−03	1.73E−06	8.73E−03	1.73E−06	1.73E−06	2.58E−03
F23	1.73E−06	1.73E−06	7.69E−06	1.40E−02	1.73E−06	3.38E−03	1.73E−06	**9.75E−01**	**9.37E−02**	1.73E−06	1.73E−06

Table 5
Parameter settings for the improved algorithms used.

Method	Population size	Maximum generation	Other parameters
ALCPSO	40	1000	*inertia weight* $w = 0.4$; *acceleration coefficients* $c_1 = c_2 = 2$; *lifespan* $= 60$; $T = 2$; *pro* $= 1/n$ (n indicates dimension of the problem)
BLPSO	40	1000	*inertia weight* $w\in[0.9\ 0.2]$; *acceleration coefficients* $c = 1.496$; *Maximum immigration and rates* $I = 1$; *Maximum emigration rates* $E = 1$
CLPSO	40	1000	*inertia weight* $w\in[0.9\ 0.2]$; *acceleration coefficients* $c = 1.496$; *refreshing gap* $m = 5$;
TVPSO	40	1000	*inertia weight* $w\in[0.9\ 0.4]$; *acceleration coefficients* $c_1\in[2.5\ 0.5]$, $c_2\in[0.5\ 2.5]$
PSOBFO	40	1000	$c_1 = 1.2$, $c_2 = 0.5$; $\Delta\in[-1,\ 1]$

of iterations increases, the optimal value of the DE algorithm gradually decreases compared to other algorithms, but CCGBFO has the fastest convergence speed. On F9, CCGBFO converges to the optimal value at the beginning of explorative steps. In dealing with F8, F12 and F15, the convergence performance of CCGBFO is second to none. In addition, although CCGBFO has the same convergence effect as some algorithms on F14, F22 and F23, the speed of CCGBFO is the fastest before 50 iterations at the initial stage.

In this paper, the Wilcoxon rank-sum test [57] at 5% significance level is used to judge whether the improvements achieved by CCGBFO are meaningful over other competitors. The p-values of CCGBFO versus other optimizers on the 23 benchmark functions are presented in Table 4. The p-values greater than 0.05 are marked in boldface, indicating that the differences are not significant. As can be seen from Table 4, there are only eight p-values that are greater than 0.05, demonstrating that CCGBFO can significantly outperform the other competitors in most cases. Notably, CCGBFO is statistically better than the original BFO on most of the 23 benchmark functions.

5.3. Comparison of CCGBFO with state-of-art optimization algorithms

In this subsection, in order to further verify its superiority, CCGBFO was compared with PSOBFO (an improved BFO) [58] and multiple variants of the PSO algorithm including PSO with an Aging Leader and Challengers (ALCPSO) [59], Biogeography-based Learning PSO (BLPSO) [60], Comprehensive Learning PSO (CLPSO) [61] and Time-Varying PSO (TVPSO) [62], on the 23 classical benchmark functions. Parameter values for each algorithm were set according to their original papers. The specific parameter settings are shown in Table 5. For fair comparison, 30 independent runs were performed on each function. The number of population size was set to 40 and the maximum number of iterations was set to 1000 in all experiments.

Table 6 presents the average results (Avg), standard deviation (Stdv), and overall ranks of the six algorithms on the F1–F23 benchmark functions. Fig. 9 shows the convergence curves of different algorithms on some benchmark functions.

As can been from Table 6, in dealing with F1, F2, F3, F4, F9, F10 and F11 functions, CCGBFO and PSOBFO obtain better results than the other improved algorithms, while CCGBFO and CLPSO outperform the other competitors on F20 and F21. In tackling with F5 and F6, PSOBFO and ALCPSO have relatively good performance. On F16, F17, F18 and F19, all the algorithms achieve almost the same optimization effect according to the rank. Notably, on F7, F8, F12, F13 and F15, CCGBFO outperforms all other optimizers. To sum up, according to ranking, CCGBFO is the best algorithm for solving the 23 benchmark functions.

Convergence curves were plotted on six of the 23 benchmark functions, as shown in Fig. 9. As can been seen from Fig. 9,

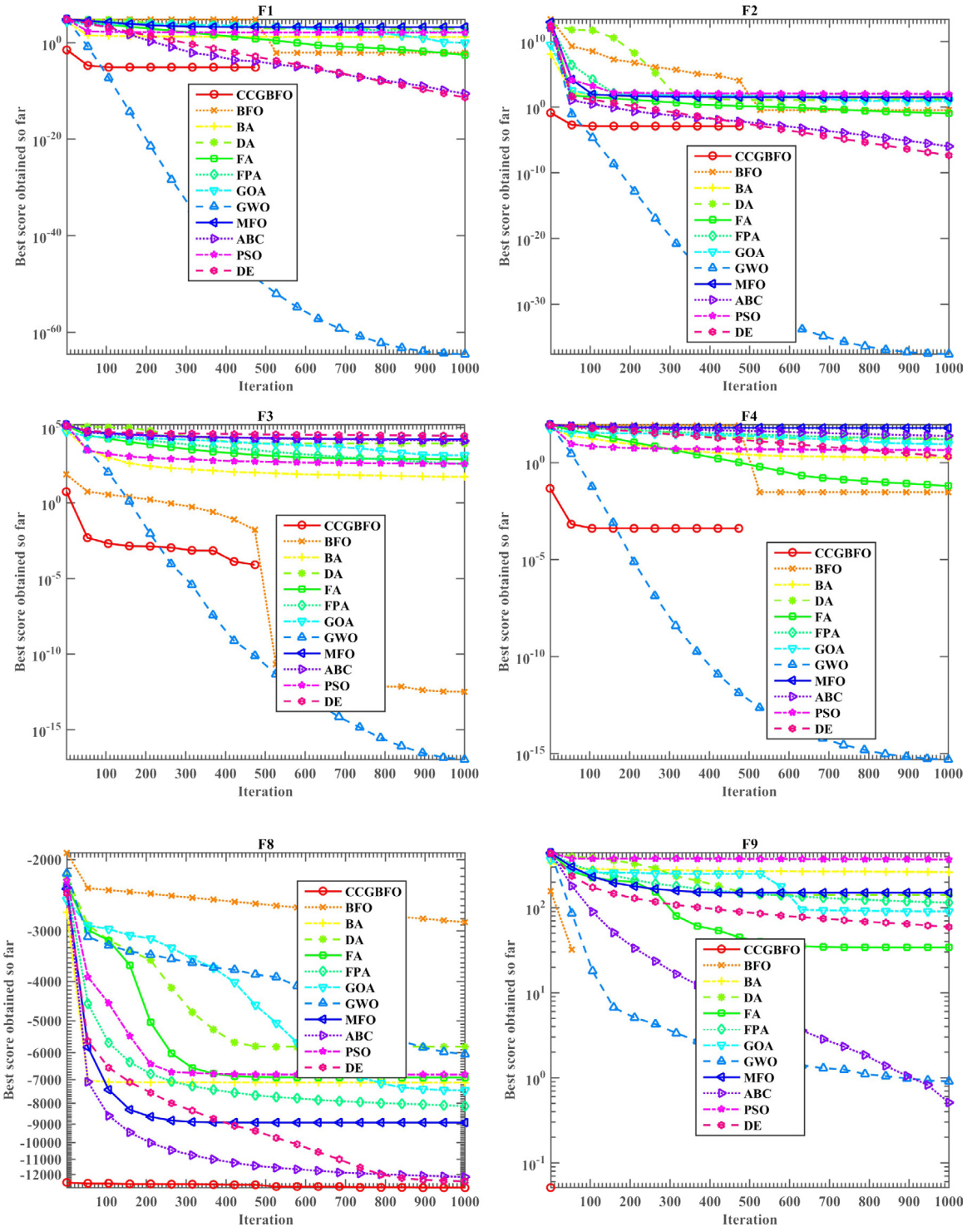

Fig. 8. Convergence curve of the CCGBFO and other algorithms.

CCGBFO has relative superiority in tackling F7, F8, F12, F13 and F15. Specifically, CCGBFO leads the other optimizers in the early stage of iteration on F7, F8, F13 and F15. Notably, on F8, CCGBFO is ahead of the other algorithms when it starts exploring, and on F13, CCGBFO reaches the optimal value only through approximately 100 iterations. On F12, CCGBFO continues to explore after 700 iterations, while other algorithms begin to slow down and fall into local optima. On F23, although the other algorithms eventually get close to the optimal value, CCGBFO outperforms them in the initialization step.

The p-values for each algorithm relative to CCGBFO on F1–F23 benchmark functions are presented in Table 7. The p-values that are not less than 0.05 are marked in bold face. From Table 7, it can be seen that CCGBFO has significant advantage over CLPSO on all the 23 benchmark functions. In addition, CCGBFO also significantly outperforms the other remaining algorithms on most of the functions.

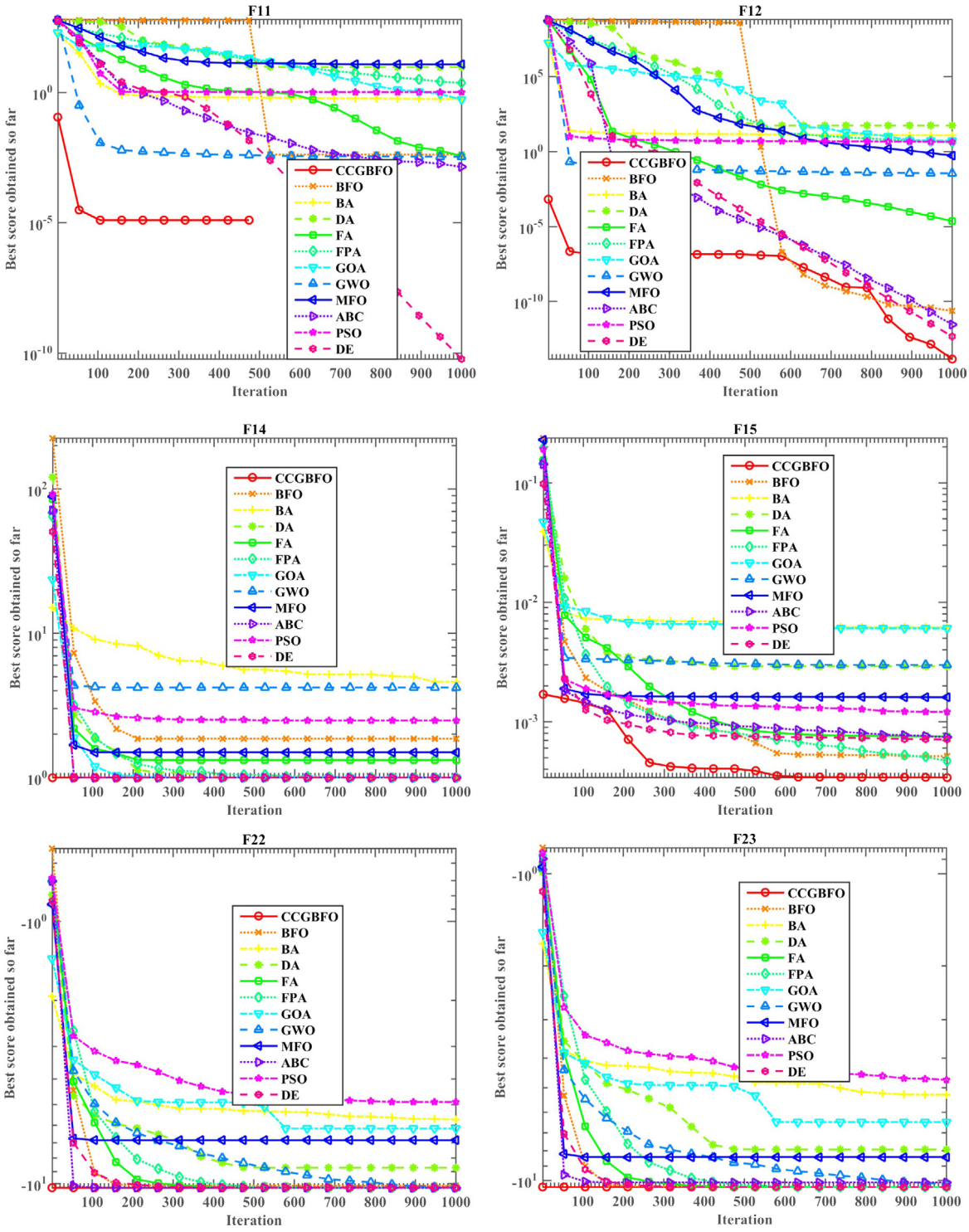

Fig. 8. (*continued*).

6. Comparative study on the IEEE CEC2014 and CEC2017 test functions

In this section, the test functions in CEC2014 and CEC2017 were used to further investigate the performance of CCGBFO. These well-known functions are divided into unimodal, multimodal, hybrid and composition categories. More information on the CEC2014 and CEC2017 test functions can be found in [63] and [64], respectively. For all the test functions, the search range is from −100 to 100, and the test dimension is 30. Tables 8 and 9 list brief information and optimum values for the CEC2014 and CEC2017 test functions, respectively.

CCGBFO was not only compared with the original BFO, but also with an improved BFO named PSOBFO. The parameters for the algorithms were set according to Tables 2 and 5. For fair comparison, in all experiments, the same population size and the maximum number of iterations were to 40 and 1000, respectively.

Table 10 presents the test results and the overall ranks of the proposed CCGBFO and other algorithms on the 30 test functions of CEC2014 and CEC2017. Figs. 10 and 11 show the convergence

Table 6
Results obtained by CCGBFO and other improved algorithms on the 23 benchmark functions (F1–F23).

F		CCGBFO	ALCPSO	BLPSO	CLPSO	TVPSO	PSOBFO
F1	Avg	0	2.94E−17	1.47E+03	3.50E+02	7.12E−10	0
	Stdv	0	7.80E−17	2.79E+02	7.13E+01	1.52E−09	0
	Rank	1	2	5	4	3	1
F2	Avg	0	1.37E−04	1.42E+01	9.01E+00	7.55E−03	0
	Stdv	0	7.49E−04	1.43E+00	1.18E+00	1.70E−02	0
	Rank	1	2	5	4	3	1
F3	Avg	0	1.97E+02	1.16E+04	1.59E+04	2.73E+00	0
	Stdv	0	1.22E+02	1.62E+03	2.38E+03	3.59E+00	0
	Rank	1	3	4	5	2	1
F4	Avg	0	4.38E+00	2.51E+01	4.04E+01	3.26E−01	0
	Stdv	0	1.62E+00	2.19E+00	3.79E+00	1.34E−01	0
	Rank	1	3	4	5	2	1
F5	Avg	2.98E−04	5.27E+01	2.95E+05	5.39E+04	5.05E+01	0
	Stdv	3.08E−04	5.45E+01	1.17E+05	1.70E+04	5.00E+01	0
	Rank	2	4	6	5	3	1
F6	Avg	1.17E−05	8.43E−17	1.39E+03	3.69E+02	5.50E−09	1.51E−01
	Stdv	2.45E−05	2.55E−16	3.15E+02	8.10E+01	1.59E−08	2.84E−02
	Rank	3	1	6	5	2	4
F7	Avg	5.33E−05	7.89E−02	4.14E−01	1.78E−01	3.40E−02	4.61E−03
	Stdv	4.12E−05	4.40E−02	9.56E−02	5.71E−02	1.90E−02	2.57E−03
	Rank	1	4	6	5	3	2
F8	Avg	−1.27E+04	−1.02E+04	−4.79E+03	−8.70E+03	−6.95E+03	−2.81E+03
	Stdv	5.92E+02	5.79E+02	3.47E+02	3.66E+02	8.63E+02	4.56E+02
	Rank	1	2	5	3	4	6
F9	Avg	−2.90E+02	6.48E+01	1.96E+02	1.22E+02	4.93E+01	−2.90E+02
	Stdv	4.89E−03	2.10E+01	1.56E+01	1.53E+01	1.04E+01	0
	Rank	1	3	5	4	2	1
F10	Avg	−1.07E+13	3.62E−01	9.09E+00	7.08E+00	5.21E−01	−1.07E+13
	Stdv	1.07E+10	6.85E−01	5.83E−01	6.06E−01	7.34E−01	3.97E−03
	Rank	1	2	5	4	3	1
F11	Avg	0	1.32E−02	1.35E+01	4.08E+00	2.18E−02	0
	Stdv	0	1.58E−02	2.44E+00	7.97E−01	2.41E−02	0
	Rank	1	2	5	4	3	1
F12	Avg	1.21E−14	2.04E−01	5.03E+02	1.43E+01	1.44E−09	1.44E−09
	Stdv	3.25E−14	3.87E−01	1.01E+03	3.11E+00	6.77E−09	2.65E−09
	Rank	1	3	5	4	2	2
F13	Avg	9.59E−07	4.33E−03	1.12E+05	1.90E+03	1.83E−03	9.92E−02
	Stdv	1.77E−06	6.66E−03	8.57E+04	2.12E+03	4.16E−03	2.55E−09
	Rank	1	3	6	5	2	4
F14	Avg	9.98E−01	9.98E−01	9.98E−01	9.98E−01	1.06E+00	1.73E+00
	Stdv	5.75E−13	5.83E−17	0	0	2.52E−01	1.42E+00
	Rank	1	1	1	1	2	3
F15	Avg	3.55E−04	5.15E−04	5.91E−04	5.82E−04	4.26E−04	7.66E−04
	Stdv	2.89E−05	3.85E−04	5.86E−05	9.68E−05	2.54E−04	2.39E−04
	Rank	1	3	5	4	2	6
F16	Avg	−1.03E+00	−1.03E+00	−1.03E+00	−1.03E+00	−1.03E+00	−1.03E+00
	Stdv	8.91E−13	6.58E−16	6.78E−16	5.90E−16	6.58E−16	4.65E−05
	Rank	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
	Stdv	4.35E−13	0	0	0	0	1.24E−05
	Rank	1	1	1	1	1	1
F18	Avg	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00
	Stdv	2.63E−11	1.41E−15	1.28E−15	1.30E−15	9.69E−16	2.10E−03
	Rank	1	1	1	1	1	1
F19	Avg	−3.86E+00	−3.86E+00	−3.86E+00	−3.86E+00	−3.86E+00	−3.86E+00
	Stdv	6.03E−07	2.63E−15	2.71E−15	2.71E−15	2.71E−15	1.66E−03
	Rank	1	1	1	1	1	1
F20	Avg	−3.32E+00	−3.26E+00	−3.30E+00	−3.32E+00	−3.29E+00	−3.26E+00
	Stdv	6.03E−08	6.05E−02	4.65E−02	4.02E−08	5.35E−02	1.81E−02
	Rank	1	4	2	1	3	4
F21	Avg	−1.02E+01	−7.71E+00	−9.22E+00	−1.02E+01	−7.82E+00	−1.01E+01
	Stdv	7.73E−09	2.92E+00	2.44E+00	7.56E−03	3.21E+00	2.82E−02
	Rank	1	5	3	1	4	2

(*continued on next page*)

curves of different algorithms in dealing with some of the 30 test functions of CEC2014 and CEC2017, respectively. The Wilcoxon rank-sum test at 5% significance level was also employed in this experiment and the p-values are presented in Table 11.

Table 6 (*continued*).

F		CCGBFO	ALCPSO	BLPSO	CLPSO	TVPSO	PSOBFO
F22	Avg	−1.04E+01	−8.84E+00	−1.04E+01	−1.04E+01	−8.86E+00	−1.03E+01
	Stdv	2.75E−08	2.62E+00	1.36E−15	6.09E−02	2.91E+00	3.26E−02
	Rank	1	4	1	1	3	2
F23	Avg	−1.05E+01	−9.83E+00	−1.05E+01	−1.05E+01	−1.01E+01	−1.03E+01
	Stdv	7.42E−06	2.18E+00	1.81E−15	5.07E−03	1.54E+00	9.72E−01
	Rank	1	4	1	1	3	2
Sum of ranks		26	59	84	70	55	49
Average rank		1.1304	2.5652	3.6522	3.0435	2.3913	2.1304
Overall rank		1	4	6	5	3	2

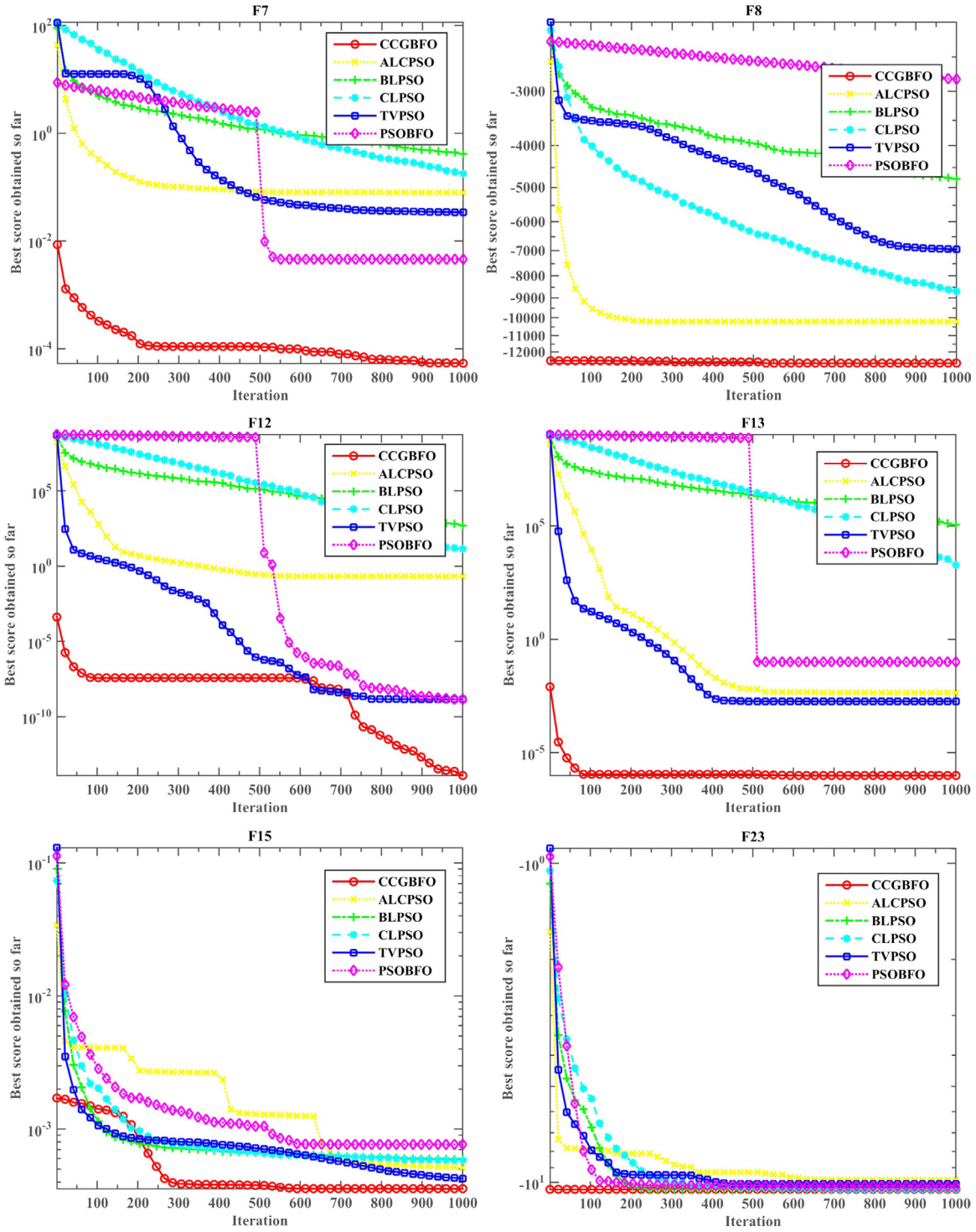

Fig. 9. Convergence curves of CCGBFO and other improved algorithms.

Table 7
The calculated *p*-values from the 23 benchmark functions (F1–F23) for CCGBFO versus other improved optimizers.

F	ALCPSO	BLPSO	CLPSO	TVPSO	PSOBFO
F1	1.73E−06	1.73E−06	1.73E−06	1.73E−06	**1.00E+00**
F2	1.73E−06	1.73E−06	1.73E−06	1.73E−06	**1.00E+00**
F3	1.73E−06	1.73E−06	1.73E−06	1.73E−06	**1.00E+00**
F4	1.73E−06	1.73E−06	1.73E−06	1.73E−06	**1.00E+00**
F5	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
F6	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
F7	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
F8	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
F9	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
F10	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
F11	1.95E−04	1.73E−06	1.73E−06	1.73E−06	**1.00E+00**
F12	1.92E−06	1.73E−06	1.73E−06	5.22E−06	1.73E−06
F13	**6.44E−01**	1.73E−06	1.73E−06	**8.59E−02**	1.73E−06
F14	5.25E−06	5.25E−06	5.25E−06	1.06E−03	1.73E−06
F15	**4.53E−01**	1.73E−06	1.92E−06	**1.59E−01**	2.13E−06
F16	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
F17	2.55E−06	2.55E−06	2.55E−06	2.55E−06	1.73E−06
F18	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
F19	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
F20	2.07E−02	**1.65E−01**	3.50E−02	**6.73E−01**	1.73E−06
F21	**1.02E−01**	1.48E−02	2.35E−06	**3.82E−01**	1.73E−06
F22	**6.44E−01**	1.73E−06	1.73E−06	**3.71E−01**	1.73E−06
F23	2.77E−03	1.73E−06	1.49E−05	3.59E−04	1.73E−06

6.1. Comparative results on the CEC2014 test functions

As can be seen from Table 10, according to the overall rank on the 30 test functions, CCGBFO ranks first, followed successively by PSOBFO and BFO. Specifically, CCGBFO obtains the best results on the three unimodal test functions. Notably, PSOBFO is outperformed by the original BFO in dealing with CE01 and CE03.

On the multimodal test functions (CE04 to CE16), it can be seen that CCGBFO obtains the best results on CE04, CE05, CE06, CE07, CE08, CE09, CE13, CE14 and CE15. However, in tackling CE10, CE11, and CE12, CCGBFO is outperformed by PSOBFO and BFO. Interestingly, the three algorithms are equally effective on CE16.

On the hybrid test functions (CE17 to CE22), CCGBFO takes the first place. PSOBFO outperforms BFO on CE17, CE18 and CE19 but is outperformed by BFO on CE20, CE21 and CE22.

On the remaining eight composition functions (CE23 to CE30), both CCGBFO and PSOBFO outperform BFO on CE23, CE27, CE28, CE29 and CE30; It can also be seen that the three algorithms are equally effective on CE24 and CE25. In addition, CCGBFO is the best algorithm for tackling CE26.

As can been from the convergence curves shown in Fig. 10, CCGBFO shows the best convergence behavior and outperforms PSOBFO and BFO at the early stage of iteration. Especially in tackling CE03, CE15 and CE23, CCGBFO converges almost to the optimal value at the beginning of exploration. In addition, although PSOBFO converges rapidly to the optimal values after 500 iterations on CE23 and CE30, CCGBFO obtains the optimal solutions in the early stage.

As can be from Table 11, there are only about 1/6 of the p-values are greater than 0.05. Therefore, it can be concluded that CCGBFO is statistically superior to the original BFO and PSOBFO in most of the CEC2014 test functions.

6.2. Comparative results on the CEC2017 test functions

As shown in Table 10, based on the overall rank on the CE01–CE30 test functions of CEC2017, CCGBFO ranks first, followed successively by PSOBFO and BFO. Specifically, on the three unimodal test functions, CCGBFO is obviously superior to BFO and PSOBFO. Notably, PSOBFO is outperformed by the original BFO on CE03. On the multimodal test functions (CE04–CE10), it can be seen that CCGBFO is the best algorithm in dealing with CE04, CE05 and CE08. However, BFO outperforms both CCGBFO and PSOBFO on CE06. In addition, PSOBFO is the best optimizer for tackling CE07, CE09 and CE10. On the ten hybrid test functions (CE11–CE20), with the exception of CE20, CCGBFO obtains

Table 8
Summary of the CEC2014 Test Functions.

ID	Name of function	Class	Optimum
CE01	Rotated High Conditioned Elliptic Function	Unimodal	100
CE02	Rotated Bent Cigar Function	Unimodal	200
CE03	Rotated Discus Function	Unimodal	300
CE04	Shifted and Rotated Rosenbrock's Function	Multimodal	400
CE05	Shifted and Rotated Ackley's Function	Multimodal	500
CE06	Shifted and Rotated Weierstrass Function	Multimodal	600
CE07	Shifted and Rotated Griewank's Function	Multimodal	700
CE08	Shifted Rastrigin's Function	Multimodal	800
CE09	Shifted and Rotated Rastrigin's Function	Multimodal	900
CE10	Shifted Schwefel's Function	Multimodal	1000
CE11	Shifted and Rotated Schwefel's Function	Multimodal	1100
CE12	Shifted and Rotated Katsuura Function	Multimodal	1200
CE13	Shifted and Rotated HappyCat Function [6]	Multimodal	1300
CE14	Shifted and Rotated HGBat Function [6]	Multimodal	1400
CE15	Shifted and Rotated Expanded Griewank's plus Rosenbrock's Function	Multimodal	1500
CE16	Shifted and Rotated Expanded Scaffer's F6 Function	Multimodal	1600
CE17	Hybrid Function 1 (N = 3)	Hybrid	1700
CE18	Hybrid Function 2 (N = 3)	Hybrid	1800
CE19	Hybrid Function 3 (N = 4)	Hybrid	1900
CE20	Hybrid Function 4 (N = 4)	Hybrid	2000
CE21	Hybrid Function 5 (N = 5)	Hybrid	2100
CE22	Hybrid Function 6 (N = 5)	Hybrid	2200
CE23	Composition Function 1 (N = 5)	Composition	2300
CE24	Composition Function 2 (N = 3)	Composition	2400
CE25	Composition Function 3 (N = 3)	Composition	2500
CE26	Composition Function 3 (N = 3)	Composition	2600
CE27	Composition Function 5 (N = 5)	Composition	2700
CE28	Composition Function 6 (N = 5)	Composition	2800
CE29	Composition Function 7 (N = 3)	Composition	2900
CE30	Composition Function 8 (N = 3)	Composition	3000

Table 9
Summary of the CEC2017 test functions.

ID	Name of function	Class	Optimum
CE01	Shifted and Rotated Bent Cigar Function	Unimodal	100
CE02	Shifted and Rotated Sum of Different Power Function	Unimodal	200
CE03	Shifted and Rotated Zakharov Function	Unimodal	300
CE04	Shifted and Rotated Rosenbrocks Function	Multimodal	400
CE05	Shifted and Rotated Rastrigins Function	Multimodal	500
CE06	Shifted and Rotated Expanded Scaffers F6 Function	Multimodal	600
CE07	Shifted and Rotated Lunacek Bi_Rastrigin Function	Multimodal	700
CE08	Shifted and Rotated Non-Continuous Rastrigins Function	Multimodal	800
CE09	Shifted and Rotated Levy Function	Multimodal	900
CE10	Shifted and Rotated Schwefels Function	Multimodal	1000
CE11	Hybrid Function 1 (N = 3)	Hybrid	1100
CE12	Hybrid Function 2 (N = 3)	Hybrid	1200
CE13	Hybrid Function 3 (N = 3)	Hybrid	1300
CE14	Hybrid Function 4 (N = 4)	Hybrid	1400
CE15	Hybrid Function 5 (N = 4)	Hybrid	1500
CE16	Hybrid Function 6 (N = 4)	Hybrid	1600
CE17	Hybrid Function 6 (N = 5)	Hybrid	1700
CE18	Hybrid Function 6 (N = 5)	Hybrid	1800
CE19	Hybrid Function 6 (N = 5)	Hybrid	1900
CE20	Hybrid Function 6 (N = 6)	Hybrid	2000
CE21	Composition Function 1 (N = 3)	Composition	2100
CE22	Composition Function 2 (N = 3)	Composition	2200
CE23	Composition Function 3 (N = 4)	Composition	2300
CE24	Composition Function 4 (N = 4)	Composition	2400
CE25	Composition Function 5 (N = 5)	Composition	2500
CE26	Composition Function 6 (N = 5)	Composition	2600
CE27	Composition Function 7 (N = 6)	Composition	2700
CE28	Composition Function 8 (N = 6)	Composition	2800
CE29	Composition Function 9 (N = 3)	Composition	2900
CE30	Composition Function 10 (N = 3)	Composition	3000

Table 10
Results for CEC2014 and CEC2017 problems (CE01–CE30).

CE14		CCGBFO	PSOBFO	BFO	CE17	CCGBFO	PSOBFO	BFO
	Avg	1.507E+09	2.248E+09	2.007E+09		6.244E+10	7.682E+10	7.728E+10
CE01	Stdv	2.422E+08	2.809E+08	4.879E+08	CE01	5.453E+09	1.847E+09	2.928E+08
	Rank	1	3	2		1	2	3
	Avg	7.820E+10	9.407E+10	9.456E+10		6.371E+50	3.140E+55	7.620E+55
CE02	Stdv	5.608E+09	2.964E+09	3.588E+08	CE02	2.095E+51	1.241E+56	3.142E+56
	Rank	1	2	3		1	2	3
	Avg	8.167E+04	2.225E+05	2.085E+05		8.408E+04	2.144E+05	2.064E+05
CE03	Stdv	5.096E+03	5.826E+04	5.079E+04	CE03	6.060E+03	6.262E+04	4.900E+04
	Rank	1	3	2		1	3	2
	Avg	1.660E+04	2.224E+04	2.230E+04		1.834E+04	2.960E+04	2.942E+04
CE04	Stdv	1.931E+03	3.538E+02	9.853E+01	CE04	2.685E+03	2.489E+03	2.518E+03
	Rank	1	2	3		1	3	2
	Avg	5.200E+02	5.203E+02	5.203E+02		9.367E+02	9.583E+02	9.714E+02
CE05	Stdv	2.709E−03	3.216E−02	3.318E−02	CE05	2.973E+01	4.143E+01	3.162E+01
	Rank	1	2	2		1	2	3
	Avg	6.439E+02	6.469E+02	6.471E+02		6.870E+02	6.803E+02	6.802E+02
CE06	Stdv	1.279E+00	2.468E+00	1.594E+00	CE06	7.613E+00	7.851E+00	8.466E+00
	Rank	1	2	3		3	2	1
	Avg	1.450E+03	1.666E+03	1.637E+03		1.417E+03	1.361E+03	1.362E+03
CE07	Stdv	5.551E+01	7.441E+01	1.250E+02	CE07	2.584E+01	7.102E+00	9.213E+00
	Rank	1	3	2		3	1	2
	Avg	1.137E+03	1.156E+03	1.153E+03		1.151E+03	1.176E+03	1.179E+03
CE08	Stdv	1.805E+01	3.758E+01	4.179E+01	CE08	1.630E+01	2.019E+01	1.888E+01
	Rank	1	3	2		1	2	3
	Avg	1.211E+03	1.230E+03	1.232E+03		9.377E+03	9.012E+03	9.058E+03
CE09	Stdv	1.340E+01	6.076E+00	5.863E+00	CE09	1.462E+03	1.422E+03	1.196E+03
	Rank	1	2	3		3	1	2
	Avg	7.776E+03	6.462E+03	6.545E+03		7.695E+03	6.137E+03	6.341E+03
CE10	Stdv	5.371E+02	4.695E+02	4.007E+02	CE10	5.802E+02	6.025E+02	4.195E+02
	Rank	3	1	2		3	1	2
	Avg	7.918E+03	6.430E+03	6.271E+03		9.693E+03	2.865E+04	2.566E+04
CE11	Stdv	5.256E+02	4.199E+02	3.571E+02	CE11	2.120E+03	1.021E+04	1.032E+04
	Rank	3	2	1		1	3	2

(*continued on next page*)

Table 10 (*continued*).

CE14		CCGBFO	PSOBFO	BFO	CE17	CCGBFO	PSOBFO	BFO
CE12	Avg	1.202E+03	1.201E+03	1.201E+03	CE12	1.555E+10	2.052E+10	2.369E+10
	Stdv	4.573E−01	2.152E−01	2.025E−01		3.501E+09	5.333E+09	3.451E+09
	Rank	2	1	1		1	2	3
CE13	Avg	1.309E+03	1.310E+03	1.310E+03	CE13	1.504E+10	2.243E+10	1.915E+10
	Stdv	5.978E−01	5.096E−01	5.743E−01		4.641E+09	7.321E+09	7.727E+09
	Rank	1	2	2		1	3	2
CE14	Avg	1.652E+03	1.769E+03	1.765E+03	CE14	7.806E+06	1.768E+07	9.398E+06
	Stdv	2.032E+01	2.079E+01	2.886E+01		6.612E+06	3.478E+07	1.071E+07
	Rank	1	3	2		1	3	2
CE15	Avg	1.952E+05	5.993E+05	6.038E+05	CE15	1.351E+09	3.094E+09	2.947E+09
	Stdv	2.785E+04	1.238E+04	1.462E+04		5.767E+08	1.385E+09	1.278E+09
	Rank	1	2	3		1	3	2
CE16	Avg	1.613E+03	1.613E+03	1.613E+03	CE16	6.699E+03	6.761E+03	6.736E+03
	Stdv	1.804E−01	3.588E−01	2.737E−01		1.296E+03	2.024E+03	1.673E+03
	Rank	1	1	1		1	3	2
CE17	Avg	1.768E+08	2.054E+08	2.117E+08	CE17	6.446E+03	2.033E+04	1.218E+04
	Stdv	8.153E+07	1.461E+08	1.372E+08		4.536E+03	2.986E+04	1.218E+04
	Rank	1	2	3		1	3	2
CE18	Avg	5.098E+09	7.825E+09	8.526E+09	CE18	1.124E+08	1.907E+08	1.245E+08
	Stdv	1.782E+09	2.788E+09	3.372E+09		9.270E+07	2.280E+08	1.472E+08
	Rank	1	2	3		1	3	2
CE19	Avg	2.433E+03	2.595E+03	2.602E+03	CE19	1.189E+09	3.605E+09	3.815E+09
	Stdv	6.957E+01	5.551E+01	4.497E+01		5.865E+08	1.517E+09	1.047E+09
	Rank	1	2	3		1	2	3
CE20	Avg	5.759E+05	1.821E+06	1.721E+06	CE20	2.949E+03	2.846E+03	2.890E+03
	Stdv	4.614E+05	1.512E+06	2.263E+06		1.818E+02	1.840E+02	1.903E+02
	Rank	1	3	2		3	1	2
CE21	Avg	5.153E+07	1.042E+08	7.996E+07	CE21	2.783E+03	2.728E+03	2.741E+03
	Stdv	3.351E+07	8.435E+07	5.281E+07		4.853E+01	5.131E+01	6.606E+01
	Rank	1	3	2		3	1	2
CE22	Avg	1.471E+04	2.696E+04	2.261E+04	CE22	9.460E+03	8.132E+03	8.138E+03
	Stdv	1.301E+04	4.164E+04	4.046E+04		4.739E+02	5.187E+02	5.224E+02
	Rank	1	3	2		3	1	2
CE23	Avg	2.500E+03	2.500E+03	2.502E+03	CE23	3.639E+03	3.639E+03	3.661E+03
	Stdv	0	0	5.151E−01		1.570E+02	1.471E+02	1.536E+02
	Rank	1	1	2		1	1	2
CE24	Avg	2.600E+03	2.600E+03	2.600E+03	CE24	3.869E+03	4.147E+03	4.227E+03
	Stdv	0	0	4.770E−02		1.454E+02	2.583E+02	2.529E+02
	Rank	1	1	1		1	2	3
CE25	Avg	2.700E+03	2.700E+03	2.700E+03	CE25	5.859E+03	7.826E+03	7.891E+03
	Stdv	0	0	6.947E−03		5.048E+02	3.102E+02	5.733E+01
	Rank	1	1	1		1	2	3
CE26	Avg	2.724E+03	2.781E+03	2.788E+03	CE26	1.233E+04	1.373E+04	1.331E+04
	Stdv	1.266E+01	3.101E+01	2.431E+01		7.258E+02	1.456E+03	1.284E+03
	Rank	1	2	3		1	3	2
CE27	Avg	2.900E+03	2.900E+03	2.901E+03	CE27	3.404E+03	4.902E+03	4.943E+03
	Stdv	0	0	1.636E−01		1.983E+02	5.046E+02	5.605E+02
	Rank	1	1	2		1	2	3
CE28	Avg	3.000E+03	3.000E+03	3.001E+03	CE28	3.327E+03	9.337E+03	9.437E+03
	Stdv	0	0	2.963E−01		2.867E+01	4.185E+02	2.907E+01
	Rank	1	1	2		1	2	3
CE29	Avg	3.100E+03	3.100E+03	2.785E+06	CE29	1.090E+04	1.374E+04	1.479E+04
	Stdv	0	0	5.173E+05		4.053E+03	1.021E+04	1.202E+04
	Rank	1	1	2		1	2	3
CE30	Avg	3.200E+03	3.200E+03	1.910E+05	CE30	2.160E+09	2.928E+09	2.816E+09
	Stdv	0	0	3.743E+04		1.264E+09	1.537E+09	1.702E+09
	Rank	1	1	2		1	3	2
Sum of ranks		35	58	64		44	64	70
Average rank		1.1667	1.9333	2.1333		1.4667	2.1333	2.3333
Overall rank		1	2	3		1	2	3

the best results. Notably, in tackling CE11, CE13, CE14, CE15, CE16, CE17 and CE18, PSOBFO is outperformed by BFO. However, PSOBFO obtains the optimal solution on CE20. Finally, on the ten composition functions (CE21–CE30), with the exception of CE21 and CE22, CCGBFO outperforms the other two algorithms.

As can be seen from the convergence curves shown in Fig. 11, with the exception of CE03 and CE04, CCGBFO has obvious convergence acceleration and outperforms the other two algorithms during the first 100 iterations. On CE03, although the convergence curves of PSOBFO and BFO have a rapid downward trend before 300 and 450 iterations, respectively, they fall into local optima at the later stage of the iteration; however, CCGBFO reaches the

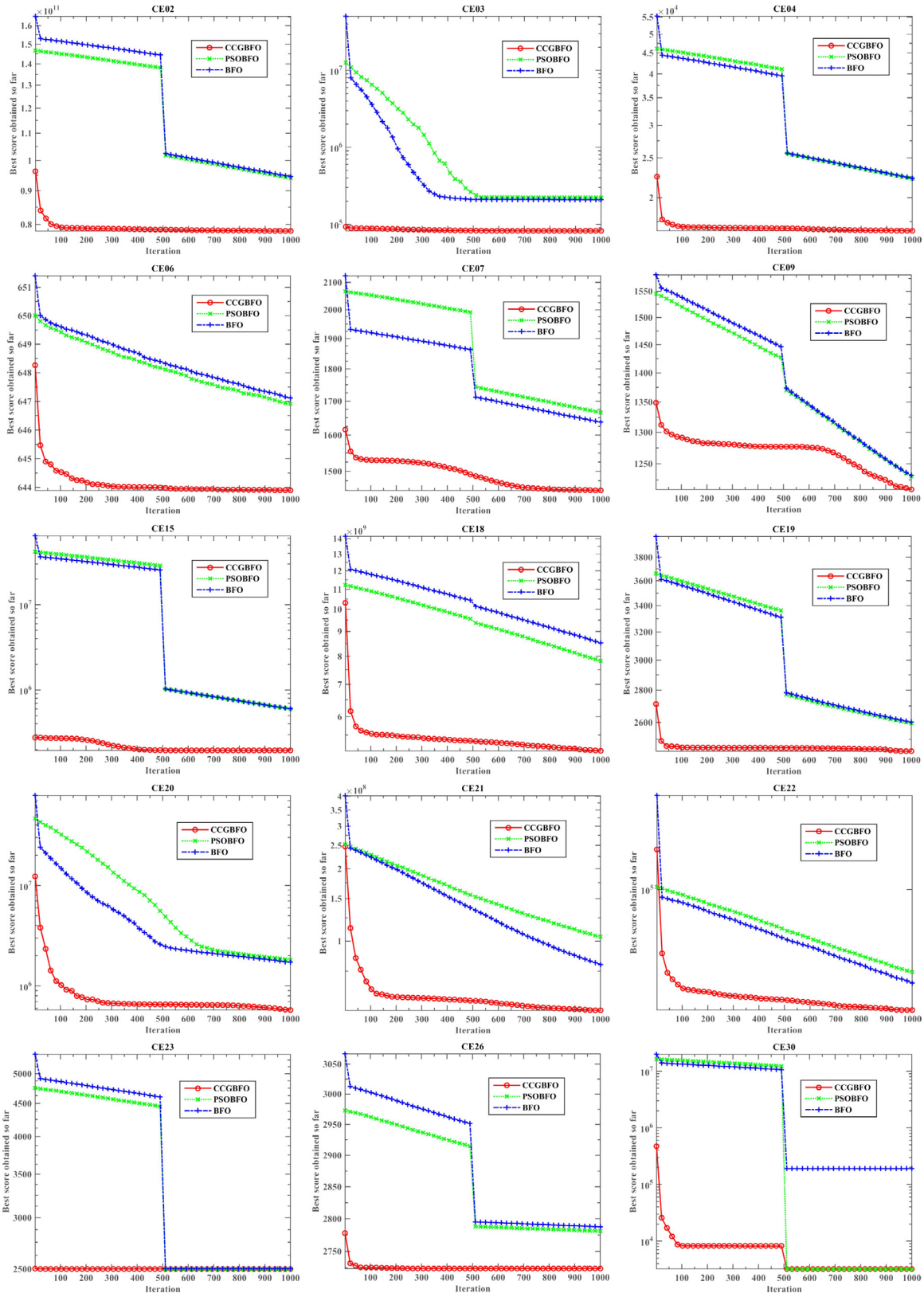

Fig. 10. Convergence curves for the 15 selected CEC2014 test functions.

optimal position almost at the beginning. On CE04, the convergence curves of PSOBFO and BFO have the same trend, whereas CCGBFO reaches a superior position. Based on the p-values shown in Table 11, it can be concluded that CCGBFO is statistically better than the other two algorithms in tackling most of the test functions of CEC2017.

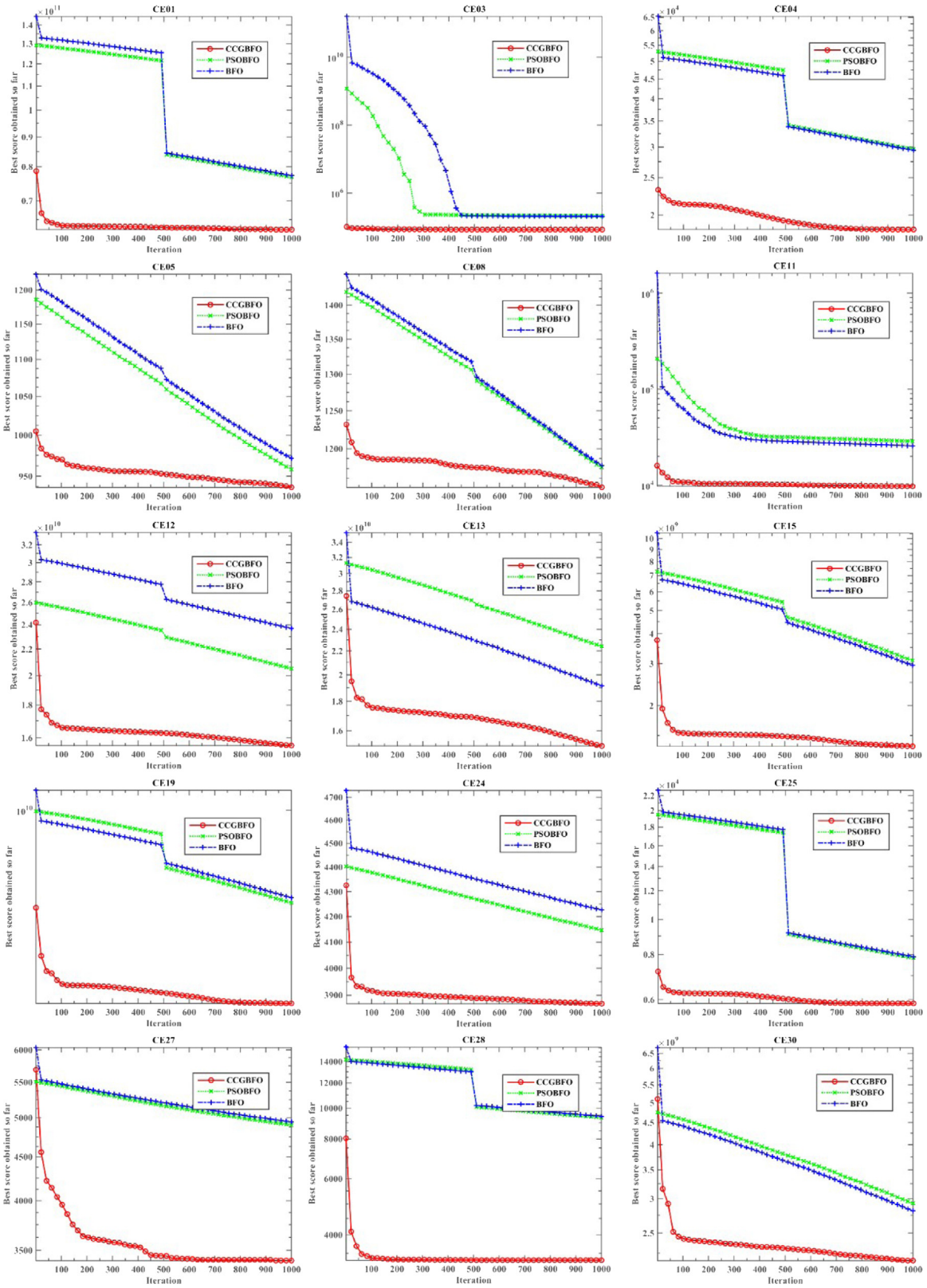

Fig. 11. Convergence curves for the 15 selected CEC2017 test functions.

7. Experimental results on real-world optimization of KELM

According to [65], the kernel method can also be applied to the Extreme Learning Machine(ELM) to generate KELM, which solves the problem of random initialization of ELM and has high classification accuracy, good generalization ability and high degree of robustness.

Table 11
The calculated *p*-values from the CEC2014 and CEC2017 test functions (CE01–CE30) for CCGBFO versus other optimizers.

CE14	PSOBFO	BFO	CE17	PSOBFO	BFO
CE01	2.88E−06	2.22E−04	CE01	1.73E−06	1.73E−06
CE02	1.73E−06	1.73E−06	CE02	5.31E−05	2.05E−04
CE03	1.73E−06	1.73E−06	CE03	1.73E−06	1.73E−06
CE04	1.73E−06	1.73E−06	CE04	1.73E−06	1.73E−06
CE05	1.73E−06	1.73E−06	CE05	1.66E−02	4.90E−04
CE06	5.31E−05	1.73E−06	CE06	9.84E−03	2.11E−03
CE07	2.60E−06	4.07E−05	CE07	1.73E−06	1.92E−06
CE08	2.56E−02	**5.71E−02**	CE08	1.60E−04	8.92E−05
CE09	3.52E−06	3.18E−06	CE09	**2.29E−01**	**2.37E−01**
CE10	3.88E−06	2.13E−06	CE10	1.92E−06	1.73E−06
CE11	1.92E−06	1.92E−06	CE11	1.92E−06	1.92E−06
CE12	5.22E−06	6.98E−06	CE12	5.29E−04	4.29E−06
CE13	2.35E−06	8.47E−06	CE13	1.25E−04	9.84E−03
CE14	1.73E−06	1.92E−06	CE14	**2.21E−01**	**5.58E−01**
CE15	1.73E−06	1.73E−06	CE15	3.72E−05	4.45E−05
CE16	2.16E−05	2.88E−06	CE16	**8.61E−01**	**6.14E−01**
CE17	**7.50E−01**	**3.82E−01**	CE17	7.73E−03	1.40E−02
CE18	6.16E−04	2.60E−05	CE18	**2.13E−01**	**5.72E−01**
CE19	2.60E−06	2.35E−06	CE19	6.34E−06	1.73E−06
CE20	1.11E−03	3.50E−02	CE20	3.68E−02	**3.18E−01**
CE21	3.85E−03	1.85E−02	CE21	1.48E−03	5.67E−03
CE22	**3.09E−01**	**8.13E−01**	CE22	2.13E−06	1.73E−06
CE23	**1.00E+00**	1.73E−06	CE23	**8.45E−01**	**2.89E−01**
CE24	**1.00E+00**	1.73E−06	CE24	3.72E−05	6.98E−06
CE25	**1.00E+00**	1.73E−06	CE25	1.73E−06	1.73E−06
CE26	2.35E−06	4.29E−06	CE26	1.06E−04	1.11E−03
CE27	**1.00E+00**	1.73E−06	CE27	1.73E−06	1.73E−06
CE28	**1.00E+00**	1.73E−06	CE28	1.73E−06	1.73E−06
CE29	**1.00E+00**	1.73E−06	CE29	**4.65E−01**	**1.47E−01**
CE30	**1.00E+00**	1.73E−06	CE30	4.72E−02	**1.31E−01**

The output of the ELM model is as shown in Eq. (10):

$$f(x_i) = h(x_i)H^T \left(\frac{I}{C} + HH^T\right)^{-1} T \tag{10}$$

where $h(x_i) = [g(a_1, x_i+b_i)\cdots g(a_L, x_i+b_L)]$ is the hidden layer output. Then the hidden layer output matrix H can be expressed as shown in Eq. (11):

$$H = \begin{bmatrix} h(x_1) \\ \cdots \\ h(x_N) \end{bmatrix}_{N\times L} \tag{11}$$

The hidden layer output $h(x_i)$ of each sample is considered as the nonlinear mapping of the sample x_i, so Eq. (12) is obtained:

$$\begin{aligned} HH^T &= \begin{bmatrix} h(x_1) \\ \cdots \\ h(x_N) \end{bmatrix}_{N\times L} g \begin{bmatrix} h(x_1) \\ \cdots \\ h(x_N) \end{bmatrix}^T_{N\times L} \\ &= \begin{bmatrix} h(x_1)\cdot h(x_1) \ldots h(x_1)\cdot h(x_N) \\ \cdots \\ h(x_N)\cdot h(x_1) \ldots h(x_N)\cdot h(x_N) \end{bmatrix}_{N\times N} \end{aligned} \tag{12}$$

Each item in Eq. (12) is the inner product form. According to the theory of kernel function, an implicit mapping can be constructed instead of the inner product of the mapping. That is, constructing a kernel function instead, as shown in Eq. (13):

$$HH^T(i,j) = K(x_i, x_j) \tag{13}$$

So there are Eqs. (14) and (15):

$$HH^T = \Omega_{ELM} = \begin{bmatrix} K(x_1,x_1) \ldots K(x_1,x_N) \\ \cdots \\ K(x_N,x_1) \ldots K(x_N,x_N) \end{bmatrix} \tag{14}$$

$$h(x)H^T = \begin{bmatrix} K(x,x_1) \\ \cdots \\ K(x,x_N) \end{bmatrix} \tag{15}$$

So the KELM model output is as shown in Eq. (16):

$$f(x) = h(x)H^T \left(\frac{I}{C} + HH^T\right)^{-1} T = \begin{bmatrix} K(x,x_1) \\ \cdots \\ K(x,x_N) \end{bmatrix} \left(\frac{I}{C} + \Omega_{ELM}\right)^{-1} \tag{16}$$

From the above, it can be seen that KELM can map the input samples into high dimensional space by introducing a kernel function, and replace the random hidden layer output matrix of ELM with the stable kernel matrix in order to obtain the only determined output weight. Finally, the stable predictive output is acquired.

The Gaussian kernel function is the most frequently used kernel function.

$$K(u,v) = exp(-\gamma \|u-v\|^2) \tag{17}$$

The setting of the parameters (C, γ) has a great impact on the performance of KELM.

In this section, CCGBFO was used to optimize the two key parameters of KELM to generate the resulting model called CCGBFO-KELM, which was applied to two different real-life scenarios, the diabetes disease diagnosis problem and the Australian credit problem.

Fig. 12 illustrates the framework of the proposed methodology. As shown in Fig. 12, the proposed CCGBFO-KELM consists mainly of two parts, namely parameter optimization and classification performance evaluation of SVM. In the process of parameter optimization, take nine tenths of the entire dataset as the training set combined with the CCGBFO strategy and the optimal parameter pairs are selected through 5-fold cross validation. After that, the optimization model is acquired by performing the training process with the parameter pairs obtained. In order to carry out the prediction task more accurately, 10-fold cross validation is utilized in this process.

In order to evaluate its effectiveness, CCGBFO-KELM was compared with four other classical machine learning algorithms, including KNN, CART, BP and SVM, as well as BFO-KELM, in terms of four different evaluation metrics, namely, classification accuracy (ACC), sensitivity, and specificity and Matthews correlation coefficient (MCC).

$$ACC = \frac{TP+TN}{TP+FP+FN+TN} \times 100\% \tag{18}$$

$$Sensitivity = \frac{TP}{TP+FN} \times 100\% \tag{19}$$

$$Specificity = \frac{TN}{FP+TN} \times 100\% \tag{20}$$

$$MCC = \frac{TP\times TN - FP\times FN}{\sqrt{(TP+FP)\times(TP+FN)\times(TN+FP)\times(TN+FN)}} \times 100\% \tag{21}$$

Where TP is the true positive samples, TN is the true negative samples, FP is the false positive samples, and FN is the false negative samples.

7.1. Diabetes disease diagnosis problem

The Pima Indians diabetes dataset consists of 768 cases (500 normal cases, 268 cases of diabetes) that were from Indian female patients. The focus of the data is to predict whether an Indian woman has diabetes. This dataset has 8 features, including

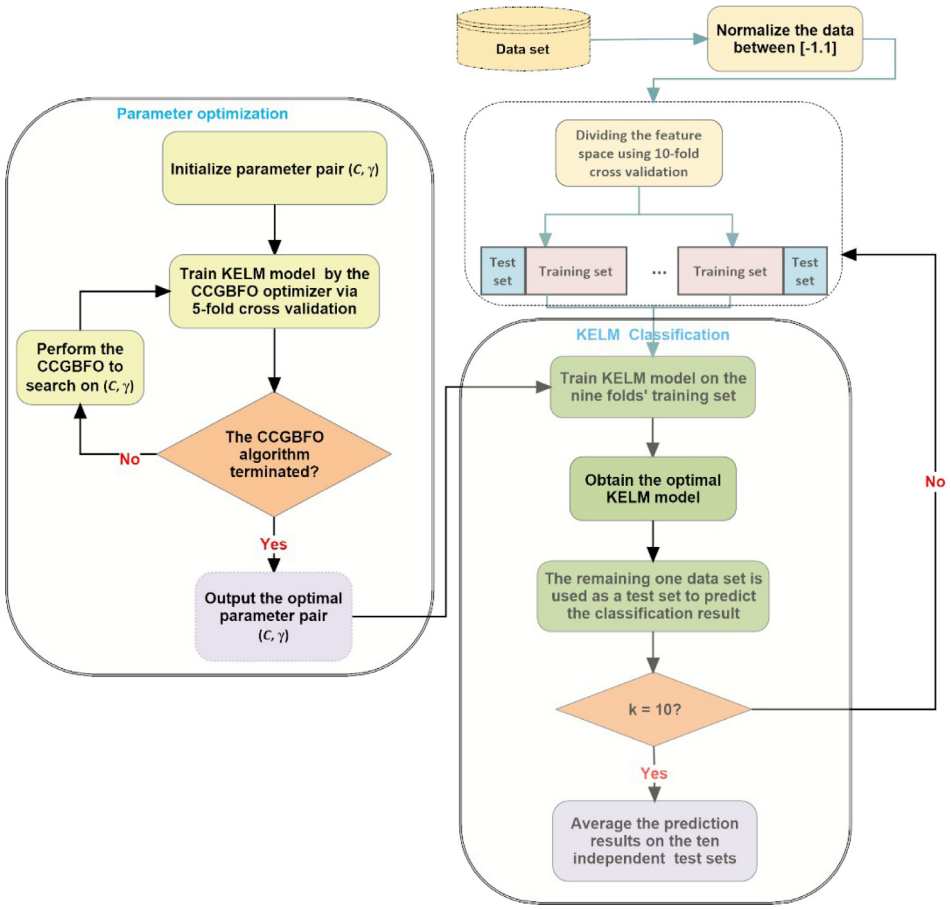

Fig. 12. Flowchart of the proposed CCGBFO-KELM model.

Table 12
The results obtained by CCGBFO-KELM on the Pima Indians diabetes dataset.

Fold	ACC	Sensitivity	Specificity	MCC
#1	0.7792	0.4783	0.9074	0.4351
#2	0.8052	0.5385	0.9412	0.5469
#3	0.8052	0.5667	0.9574	0.5928
#4	0.8442	0.7500	0.8980	0.6591
#5	0.7632	0.5185	0.8980	0.4603
#6	0.6883	0.5000	0.8085	0.3248
#7	0.7662	0.5714	0.8776	0.4781
#8	0.8182	0.5455	0.9273	0.5264
#9	0.8312	0.8214	0.8367	0.6456
#10	0.7368	0.4231	0.9000	0.3760
Avg.	0.7838	0.5713	0.8952	0.5045
Dev.	0.0472	0.1224	0.0453	0.1106

(1) number of times pregnant, (2) plasma glucose concentration a 2 hours in an oral glucose tolerance test, (3) diastolic blood pressure, (4) triceps skin fold thickness, (5) 2-Hour serum insulin, (6) body mass index, (7) diabetes pedigree function, and (8) age.

Table 12 shows the detailed results obtained by CCGBFO-KELM on this dataset via 10-fold cross validation. As illustrated in Table 12, CCGBFO-KELM achieves the average results of 78.38% ACC, 57.13% sensitivity, 89.52% specificity, and 0.5045 MCC.

Fig. 13 shows the comparison between the CCGBFO-KELM model and other test methods.

It can be clearly seen from Fig. 13 that CCGBFO-KELM is superior to BFO-KELM in terms of the four evaluation metrics. In addition, regarding the ACC metric, the CCGBFO-KELM model has the highest accuracy, whereas BP has the lowest accuracy. Among the six models tested, the variance of BFO-KELM is the smallest

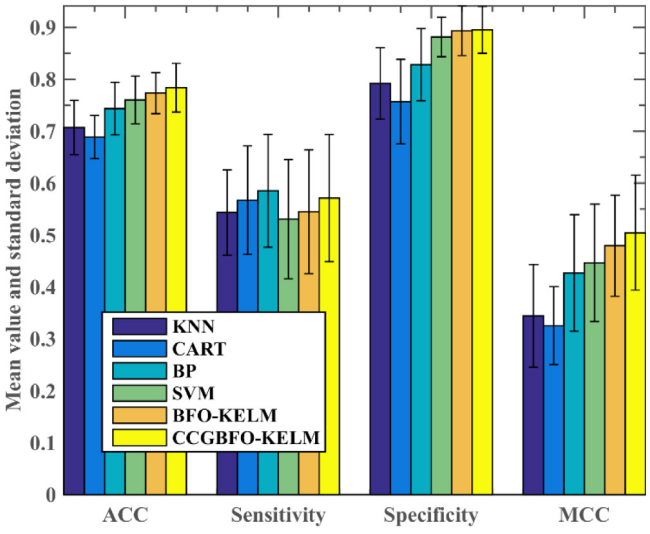

Fig. 13. The classification performance obtained by the six models in terms of ACC, sensitivity, specificity and MCC on the Pima Indians diabetes dataset.

and the variance of CCGBFO-KELM, CART, and SVM is almost the same, while the variance of KNN is the largest. It indicates that the proposed method is relatively robust and stable in diagnosing the Pima Indians diabetes. According to the sensitivity metric, BP obtains the best results with a larger variance and CGGBO-KELM ranks second, while SVM obtains the worst results. With regard to the specificity metric, CCGBFO-KELM obtains the best results,

Table 13
The results obtained by CCGBFO-KELM on the Australian credit dataset.

Fold	ACC	Sensitivity	Specificity	MCC
#1	0.8841	0.8649	0.9063	0.7692
#2	0.8986	0.8889	0.9167	0.7861
#3	0.8551	0.8780	0.8214	0.6995
#4	0.8261	0.8529	0.8000	0.6535
#5	0.8406	0.9474	0.7097	0.6862
#6	0.8841	0.8333	0.9394	0.7740
#7	0.8841	0.8864	0.8800	0.7548
#8	0.8551	0.8182	0.9200	0.7133
#9	0.8406	0.7895	0.9032	0.6898
#10	0.8986	0.9615	0.8605	0.8008
Avg.	0.8667	0.8721	0.8657	0.7327
Dev.	0.0263	0.0536	0.0706	0.0503

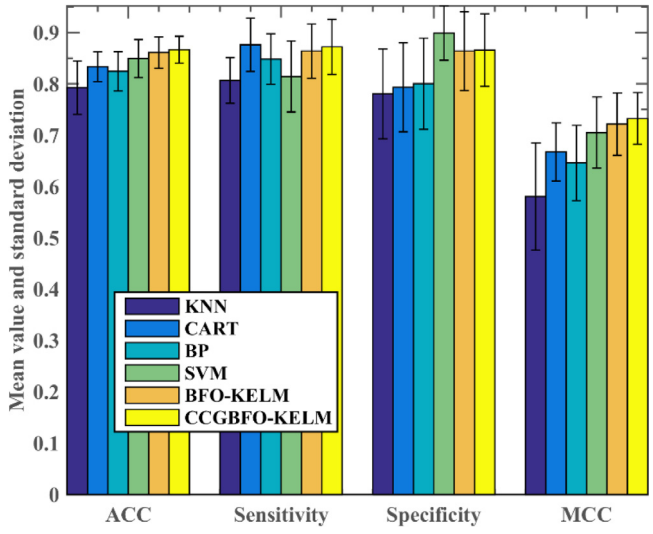

Fig. 14. The classification performance obtained by the six models in terms of ACC, sensitivity, specificity and MCC on the Australian credit dataset.

followed successively by BFO-KELM, SVM, BP, KNN and CART. In terms of the MCC metric, CCGBFO-KELM obtains the best results, followed successively by BFO-KELM, SVM, BP, KNN and CART.

7.2. Australian credit problem

The Australian credit dataset was obtained from the UCI repository, which consists of 307 reputable applicant instances and 383 instances with poor reputation. There are 6 numerical and 8 categorical attributes in each instance. To protect the confidentiality of the data, all attribute names and values have been replaced with meaningless symbols. Table 13 presents the detailed results obtained by CCGBFO-KELM on the Australian credit dataset.

From Table 13, it can be seen that CCGBFO-KELM achieves the average results of 86.67% ACC, 87.21% sensitivity, 86.57% specificity, and 0.7327 MCC.

Fig. 14 shows the comparison between the CCGBFO-KELM model and other test methods.

As can be seen from Fig. 14, CCGBFO-KELM is better than BFO-KELM in terms of all the 4 evaluation metrics. As for the ACC metrics, it can be seen that the CCGBFO-KELM has the highest classification accuracy with the smallest variance during the process of 10-fold cross validation, whereas KNN has the lowest accuracy with the largest variance. This proves the robustness and stableness of the proposed method on the Australian credit dataset. Regarding the sensitivity metric, although CART takes the first place, it is only slightly better than CCGBFO-KELM. Interestingly, KNN has the poorest sensitivity but has the smallest variance. With regard to the specificity metric, SVM obtains the best results, followed successively by CCGBFO-KELM, BFO-KELM, BP, CART and KNN. According to the MCC metric, CCGBFO-KELM ranks first with the smallest variance, followed successively by BFO-KELM, SVM, CART, BP and KNN.

8. Conclusions and future work

Given multiple drawbacks suffered by BFO, including slow convergence speed, low search accuracy, and inability to jump out of local optima, in this study, an improved BFO, called CCGBFO, is proposed, which combines the chaotic chemotaxis step length, Gaussian mutation and chaotic local search with a 'shrinking' strategy operators. CCGBFO was compared with other state-of-the-art optimization methods on a series of classical benchmark problems. The experimental results show that CCGBFO significantly outperforms other competitors including the original BFO, in terms of convergence speed and solution accuracy, indicating that combining these three strategies can achieve a better balance between exploration and exploitation. In addition, CCGBFO was used to solve the parameter selection problem of KELM and obtained very accurate prediction results on two real-world problems.

The future research directions mainly include the following aspects. First, in order to further enhance the credibility of research, we need to adopt the latest algorithm that has been improved as a comparison algorithm. Second, the parameter setting of BFO is a complex and important problem. In general, the selection of parameters is usually based on experience and lacks the support of scientific theory. Therefore, in future research, we need to analyze the parameter values from the theoretical point of view and establish a solid theoretical basis for parameter adjustment. In addition, the fusion of different algorithms is an important research direction for the improvement of intelligent optimization algorithms. There are many emerging intelligent algorithms, and the unique mechanisms of these algorithms can be integrated into BFO in order that the best optimization results can be obtained.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.asoc.2019.105884.

Acknowledgments

This research was partially supported by National Natural Science Foundation of China (61922064), Zhejiang Provincial Natural Science Foundation, China (LR17F030001), Science and Technology Plan Project of Wenzhou, China (ZG2017016, ZG2017019), the Graduate Scientific Research Foundation of Wenzhou University, China (3162018022, 3162018024), Zhejiang University Students Science and Technology Innovation Activity Plan, China (2019R429043).

References

[1] X. Zhang, et al., Robust low-rank tensor recovery with rectification and alignment, IEEE Trans. Pattern Anal. Mach. Intell. (2019) http://dx.doi.org/10.1109/TPAMI.2019.2929043:1-1.
[2] J.H. Holland, Adaptation in Natural and Artificial Systems, MIT Press, 1992, pp. 126–137.
[3] E. Andrassyova, M. Mach, Genetic algorithms and their applications, 1996.
[4] J. Kennedy, R. Eberhart, Particle swarm optimization, in: IEEE International Conference on Neural Networks - Conference Proceedings, 1995.

[5] W. Deng, et al., Study on an improved adaptive PSO algorithm for solving multi-objective gate assignment, Appl. Soft Comput. 59 (2017).
[6] X. Zhang, et al., Multiple object tracking via species-based particle swarm optimization, IEEE Trans. Circuits Syst. Video Technol. 20 (11) (2010) 1590–1602.
[7] K. Socha, M. Dorigo, Ant colony optimization for continuous domains, European J. Oper. Res. 185 (3) (2008) 1155–1173.
[8] C. Twomey, et al., An analysis of communication policies for homogeneous multi-colony ACO algorithms, Inf. Sci. Int. J. 180 (12) (2010) 2390–2404.
[9] W. Deng, J. Xu, H. Zhao, An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem, IEEE Access 7 (2019) 20281–20292.
[10] Y. Xu, et al., Enhanced Moth-flame optimizer with mutation strategy for global optimization, Inform. Sci. 492 (2019) 181–203.
[11] Y. Xu, et al., An efficient chaotic mutative moth-flame-inspired optimizer for global optimization tasks, Expert Syst. Appl. 129 (2019) 135–155.
[12] A.A. Heidari, et al., Harris hawks optimization: Algorithm and applications, Future Gener. Comput. Syst. 97 (2019) 849–872.
[13] H. Chen, et al., Parameters identification of photovoltaic cells and modules using diversification-enriched Harris hawks optimization with chaotic drifts, J. Cleaner Prod. (2019) 118778.
[14] A.A. Heidari, et al., An enhanced associative learning-based exploratory whale optimizer for global optimization, Neural Comput. Appl. (2019) http://dx.doi.org/10.1007/s00521-019-04015-0.
[15] H. Chen, et al., A balanced whale optimization algorithm for constrained engineering design problems, Appl. Math. Model. 71 (2019) 45–59.
[16] J. Luo, et al., Multi-strategy boosted mutative whale-inspired optimization approaches, Appl. Math. Model. 73 (2019) 109–123.
[17] H. Chen, et al., An efficient double adaptive random spare reinforced whale optimization algorithm, Expert Syst. Appl. (2019) 113018.
[18] X. Zhang, et al., Gaussian mutational chaotic fruit fly-built optimization and feature selection, Expert Syst. Appl. 141 (2020) 112976.
[19] H. Chen, et al., Efficient multi-population outpost fruit fly-driven optimizers: Framework and advances in support vector machines, Expert Syst. Appl. (2019) 112999.
[20] K.M. Passino, Biomimicry of bacterial foraging for distributed optimization and control, IEEE Control Syst. 22 (3) (2002) 52–67.
[21] M. Hanmandlu, et al., A novel optimal fuzzy system for color image enhancement using bacterial foraging, IEEE Trans. Instrum. Meas. 58 (8) (2009) 2867–2879.
[22] M. Tripathy, et al., Transmission loss reduction based on FACTS and bacteria foraging algorithmi, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, 2006, pp. 222–231.
[23] S. Mishra, C.N. Bhende, Bacterial foraging technique-based optimized active power filter for load compensation, IEEE Trans. Power Deliv. 22 (1) (2007) 457–465.
[24] B. Niu, et al., A novel bacterial foraging optimizer with linear decreasing chemotaxis step, in: Proceedings - 2010 2nd International Workshop on Intelligent Systems and Applications, ISA 2010, 2010.
[25] A. Biswas, et al., Analysis of the reproduction operator in an artificial bacterial foraging system, Appl. Math. Comput. 215 (9) (2010) 3343–3355.
[26] B. Niu, et al., Novel bacterial foraging optimization with time-varying chemotaxis step, Int. J. Artif. Intell. 7 (11A) (2011) 257–273.
[27] S. Gholami-Boroujeny, M. Eshghi, Active noise control using an adaptive bacterial foraging optimization algorithm, Signal Image Video Process. 8 (8) (2012) 1507–1516.
[28] J. Li, et al., Analysis and improvement of the bacterial foraging optimization algorithm, J. Comput. Sci. Eng. 8 (1) (2014) 1–10.
[29] L. Tan, F. Lin, H. Wang, Adaptive comprehensive learning bacterial foraging optimization and its application on vehicle routing problem with time windows, Neurocomputing 151 (P3) (2015) 1208–1215.
[30] C. Yang, et al., Bacterial foraging optimization using novel chemotaxis and conjugation strategies, Inform. Sci. 363 (2016) 72–95.
[31] Q. Zhang, et al., Chaos-induced and mutation-driven schemes boosting salp chains-inspired optimizers, IEEE Access 7 (2019) 31243–31261.
[32] Q. Zhang, et al., Chaos enhanced bacterial foraging optimization for global optimization, IEEE Access 6 (2018) 64905–64919.
[33] Z. Cai, et al., An intelligent Parkinsons' disease diagnostic system based on a chaotic bacterial foraging optimization enhanced fuzzy KNN approach, Comput. Math. Methods Med. 2018 (2018) 24.
[34] P. Praveena, K. Vaisakh, S. Rama Mohana Rao, A bacterial foraging and PSO-DE algorithm for solving dynamic economic dispatch problem with valve-point effects, in: Proceedings - 1st International Conference on Integrated Intelligent Computing, ICIIC 2010, 2010.
[35] S.M. Abd-Elazim, E.S. Ali, A hybrid particle swarm optimization and bacterial foraging for optimal power system stabilizers design, Int. J. Electr. Power Energy Syst. 46 (1) (2013) 334–341.
[36] G.Y. Zhang, Y.G. Wu, Y.X. Tan, Bacterial foraging optimization algorithm with quantum behavior, Dianzi Yu Xinxi Xuebao/J. Electron. Inf. Technol. 35 (3) (2013) 614–621.
[37] L. Shao, Y. Chen, Bacterial foraging optimization algorithm integrating tabu search for motif discovery, in: 2009 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2009, 2009.
[38] Y.E. Yıldız, O. Altun, Hybrid achievement oriented computational chemotaxis in bacterial foraging optimization: a comparative study on numerical benchmark, Soft Comput. 19 (12) (2015) 3647–3663.
[39] W. Zhao, L. Wang, An effective bacterial foraging optimizer for global optimization, Inform. Sci. 329 (2016) 719–735.
[40] F. Zhao, et al., A chaotic local search based bacterial foraging algorithm and its application to a permutation flow-shop scheduling problem, Int. J. Comput. Integr. Manuf. 29 (9) (2016) 962–981.
[41] L. Zhang, C. Zhang, Hopf bifurcation analysis of some hyperchaotic systems with time-delay controllers, Kybernetika 44 (1) (2008) 35–42.
[42] G.I. Sayed, A.E. Hassanien, A.T. Azar, Feature selection via a novel chaotic crow search algorithm, Neural Comput. Appl. (2017) 1–18.
[43] A. Tharwat, A.E. Hassanien, Chaotic antlion algorithm for parameter optimization of support vector machine, Appl. Intell. 48 (3) (2018) 670–686.
[44] X. Zhao, et al., Chaos enhanced grey wolf optimization wrapped ELM for diagnosis of paraquat-poisoned patients, Comput. Biol. Chem. 78 (2019) 481–490.
[45] B. Liu, et al., Improved particle swarm optimization combined with chaos, Chaos Solitons Fractals 25 (5) (2005) 1261–1271.
[46] D. Jia, G. Zheng, M. Khurram Khan, An effective memetic differential evolution algorithm based on chaotic local search, Inform. Sci. 181 (15) (2011) 3175–3187.
[47] S. Saha, V. Mukherjee, A novel chaos-integrated symbiotic organisms search algorithm for global optimization, Soft Comput. (2017) 1–20.
[48] R. Storn, K. Price, Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.
[49] X.S. Yang, A new metaheuristic bat-inspired algorithm, Stud. Comput. Intell. (2010) 65–74.
[50] X.S. Yang, Firefly algorithm, stochastic test functions and design optimization, Int. J. Bio-Inspired Comput. 2 (2) (2010) 78–84.
[51] D. Karaboga, An idea based on honey bee swarm for numerical optimization, 2005.
[52] X.-S. Yang, Flower Pollination Algorithm for Global Optimization, Springer, Berlin, Heidelberg, 2012.
[53] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61.
[54] S. Mirjalili, Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm, Knowl. Based Syst. 89 (2015) 228–249.
[55] S. Mirjalili, Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems, Neural Comput. Appl. 27 (4) (2016) 1053–1073.
[56] S. Saremi, S. Mirjalili, A. Lewis, Grasshopper optimisation algorithm: Theory and application, Adv. Eng. Softw. 105 (2017) 30–47.
[57] J. Derrac, et al., A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evol. Comput. 1 (1) (2011) 3–18.
[58] W. Korani, Tunning of PID controller using bacterial foraging orientec by particle swarm optimization.
[59] W.N. Chen, et al., Particle swarm optimization with an aging leader and challengers, IEEE Trans. Evol. Comput. 17 (2) (2013) 241–258.
[60] X. Chen, et al., Biogeography-based learning particle swarm optimization, Soft Comput. 21 (24) (2016) 1–23.
[61] J.J. Liang, et al., Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput. 10 (3) (2006) 281–295.
[62] H.-L. Chen, et al., A novel bankruptcy prediction model based on an adaptive fuzzy k-nearest neighbor method, Knowl.-Based Syst. 24 (8) (2011) 1348–1359.
[63] J.J. Liang, B.Y. Qu, P.N. Suganthan, Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization, 2013.
[64] A. LaTorre, J.M. Pena, A comparison of three large-scale global optimizers on the CEC 2017 single objective real parameter numerical optimization benchmark, in: 2017 IEEE Congress on Evolutionary Computation, CEC 2017 - Proceedings, 2017.
[65] G.B. Huang, et al., Extreme learning machine for regression and multiclass classification, IEEE Trans. Syst. Man Cybern. B 42 (2) (2012) 513–529.