Chaos-enhanced moth-flame optimization algorithm for global optimization

LI Hongwei[1], LIU Jianyong[1], CHEN Liang[1,2,*], BAI Jingbo[1], SUN Yangyang[3], and LU Kai[1]

1. College of Field Engineering, Army Engineering University of the PLA, Nanjing 210001, China;
2. Automobile Non-Commissioned Officer Academy, Army Military Transportation University, Bengbu 233011, China;
3. College of National Defense Engineering, Army Engineering University of the PLA, Bengbu 233011, China

Abstract: Moth-flame optimization (MFO) is a novel metaheuristic algorithm inspired by the characteristics of a moth's navigation method in nature called transverse orientation. Like other metaheuristic algorithms, it is easy to fall into local optimum and leads to slow convergence speed. The chaotic map is one of the best methods to improve exploration and exploitation of the metaheuristic algorithms. In the present study, we propose a chaos-enhanced MFO (CMFO) by incorporating chaos maps into the MFO algorithm to enhance its performance. The chaotic map is utilized to initialize the moths' population, handle the boundary overstepping, and tune the distance parameter. The CMFO is benchmarked on three groups of benchmark functions to find out the most efficient one. The performance of the CMFO is also verified by using two real engineering problems. The statistical results clearly demonstrate that the appropriate chaotic map (singer map) embedded in the appropriate component of MFO can significantly improve the performance of MFO.

Keywords: moth-flame optimization (MFO), chaotic map, metaheuristic, global optimization.

1. Introduction

There exists a class of nonlinear and non-convex optimization problems which are highly complex and difficult to solve in our daily life and work fields. These optimization problems include engineering design [1], feature selection [2], image segmentations [3], flow shop scheduling [4], parameter estimation [5], multi-objective optimization [6], etc. So far, these problems are difficult to solve by traditional approaches. Inspired by the great laws of nature, a series of metaheuristic algorithms have been proposed. Metaheuristic algorithms are very suitable for searching the global optimum or relative optimum by extracting useful information from a group of individuals. In the 1990s, several metaheuristic algorithms had been proposed such as genetic algorithm (GA) [7], particle swarm optimization (PSO) [8], ant colony optimization (ACO) [9], and differential evolution (DE) [10]. In the 21st century, some new metaheuristic algorithms were put forward such as artificial bee colony (ABC) [11], cuckoo search (CS) [12], harmony search (HS) [13], bat algorithm (BA) [14], chicken swarm optimization (CSO) [15], grey wolf optimizer (GWO) [16], firefly algorithm (FA) [17], whale optimization algorithm (WOA) [18], and ant lion optimizer (ALO) [19].

Recently, Mirjalili [20] proposed a novel metaheuristic optimization algorithm called moth-flame optimization (MFO). It is inspired by the movement of moths with respect to the source of light. The MFO has been quite successful in a wide range of applications such as engineering design [20 – 22], image segmentation [23,24], feature selection [25,26], power flow problem [27], optical network unit (ONU) placement [28], parameter extraction problem [29,30], and multi-objective optimization [22,31,32].

Chaos belongs to the characteristic of the nonlinear system, which is a deterministic system, while chaotic motion has random behaviors without random sequences. Chaos maps possess some uncertainty, ergodicity and stochasticity. Hence, chaos maps have been utilized to improve both exploration and exploitation by many metaheuristic algorithms. Some recent studies have employed the chaos theory in metaheuristic algorithms: Yang [33] applied chaotic maps to the mutation probability to boost the exploitation of GA and clearly outperform the standard GA. Guo [34] improved the performance of DE with the chaos mutation factor. Gandomi [35] validated that the tuning learning parameter by chaotic maps can improve the performance of PSO. Talatahari [36] utilized chaotic maps to tune the attractive movement of the fireflies in FA and confirmed no-

ticeable improvements of the new algorithms. Saremi [37] employed the chaotic maps for manipulating three operators of biogeography-based optimization (BBO) and the results demonstrated chaotic maps can significantly boost the performance. Huang [38,39] incorporated the chaotic theory into the cuckoo search (CS) algorithm and demonstrated that the improved algorithm is superior to the standard CS. The above researches show that the chaos theory has potential possibilities to improve the performance for metaheuristic algorithms. Therefore, it is easy to think of using chaos maps to improve the performance of MFO.

Currently, there are few works for improving the performance of MFO using the chaos theory. Wu [40] integrated the crisscross mechanism and the chaotic operator with MFO and tested it on eight benchmark functions. Simulation experiments showed that the improved algorithm can enhance the performance of MFO. However, it did not analyze the effect of chaotic maps for MFO separately and it only integrated chaotic maps with one of the components of MFO. This paper integrates ten chaotic maps into different components of MFO to extensively investigate the effectiveness of chaotic maps for improving exploration and exploitation of MFO. Firstly, we compare the original MFO which uses the set on the boundary mechanism and the MFO algorithm which uses the randomness of the bound handling mechanism. It is easy to get the conclusion that the MFO algorithm significantly outperforms the original MFO. Then we use chaotic maps to replace the pseudorandom sequences embodied in three components of MFO. To evaluate the proposed chaos-enhanced MFO (CMFO) algorithms and find the best CMFO, they are benchmarked on three groups of benchmark functions which consist of unimodal functions, multi-modal functions and composite functions. The results reveal noticeable improvements of the CMFO due to the application of Singer maps embedded in distance parameters, while the chaotic maps embedded in population initialization and boundary handing do not improve the performance of MFO. The results on engineering problems also show that the proposed CMFO outperforms other algorithms.

The organization of the paper is as follows: Section 2 describes the original MFO. The chaotic maps and the proposed CMFO are presented in Section 3. In addition, the CMFO is also evaluated through 18 benchmark functions and two benchmark engineering design cases in Section 4 and Section 5. Section 6 provides a summary of our work.

2. The original MFO algorithm

The original MFO algorithm is inspired by a unique way of flight of moths at night called transverse orientation mechanism. It helps them to fly in a straight line by maintaining a fixed angle with the remote moon [41]. However, the moths are often deceived by artificial light. When they try to maintain a similar angle with an artificial light which is closer to the moths compared to the moon, the transverse orientation mechanism leads them to fly along a spiral path and brings them to the artificial light. Mirjalili [20] simulated the natural phenomena of the spiral convergence of moths towards artificial light. In MFO, there are two important populations of solutions (moths and flames), which is different from some other metaheuristic algorithms. The moths represent the corresponding positions of the current solution randomly generated in the search domain. The flames represent the corresponding position of the better solutions obtained by the moths. Matrix M is used to represent the moths:

$$M = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1d} \\ m_{21} & m_{22} & \cdots & m_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n1} & m_{n2} & \cdots & m_{nd} \end{bmatrix} \tag{1}$$

where n is the number of moths and d is the number of dimensions. For all the moths, we assume an array for storing the corresponding fitness values as follows:

$$OM = \begin{bmatrix} OM_1 & OM_2 & \cdots & OM_n \end{bmatrix}^{\mathrm{T}}. \tag{2}$$

The flames also can be represented by a matrix F:

$$F = \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1d} \\ f_{21} & f_{22} & \cdots & f_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ f_{n1} & f_{n2} & \cdots & f_{nd} \end{bmatrix}. \tag{3}$$

We also assume that there exists an array for storing the corresponding fitness values:

$$OF = \begin{bmatrix} OF_1 & OF_2 & \cdots & OF_n \end{bmatrix}^{\mathrm{T}}. \tag{4}$$

The moths and flames are both solutions, while they adopt different updating methods in the evolutionary process. Each moth is assigned to one flame and its position is updated by a spiral around the assigned flame as follows:

$$M_i = S(M_i, F_j) \tag{5}$$

where S denotes a spiral function. Basically, any spiral functions satisfying certain conditions can be used [20]. In the original MFO, a logarithmic spiral is selected. The position of each moth is updated in terms of the corresponding flame as follows:

$$S(M_i, F_j) = D_i \cdot \mathrm{e}^{br} \cdot \cos(2\pi r) + F_j \tag{6}$$

$$D_i = |M_i - F_j| \tag{7}$$

$$r = a \cdot \mathrm{rand} + 1 \tag{8}$$

$$a = -1 + t \left(-\frac{1}{T} \right) \qquad (9)$$

where D_i denotes the distance between the ith moth and the jth flame, b is a constant for defining the shape of the logarithmic spiral, r is the distance parameter which defines the distance between the next position of the moth and the flame, T is the maximum number of the iterations, t is the number of the current iteration, a is linearly decreased from -1 to -2 over the the evolutionary process.

3. CMFO

Chaos maps have the properties of ergodicity and regularity, with the possibility to balance the exploration and exploitation of MFO. Accordingly, we propose a series of CMFO algorithms. We choose ten different chaotic maps as shown in Table 1. More details refer to literature [35,37]. In CMFO, chaotic maps are imbedded in three components to take the place of pseudorandom sequences in the original MFO. These components are population initialization, boundary handing and distance parameter r. Different chaotic maps are used in each component and ten variants of CMFO are obtained. Every chaotic map is tested independently on each component. The pseudocode of the CMFO algorithm is illustrated in Algorithm 1. In this algorithm, cc is a chaotic sequence which is generated by a chaotic map.

Table 1   Chaos maps

Number	Name	Chaotic map
1	Chebyshev	$x_{k+1} = \cos(k \arccos x_k)$
2	Circle map	$x_{k+1} = x_k + b - (a/2\pi) \sin(2\pi x_k) \cdot \mathrm{mod}(1), \quad a = 0.5 \text{ and } b = 0.2$
3	Gaussian map	$x_{k+1} = \begin{cases} 0, & x_k = 0 \\ 1/\mathrm{mod}\,(x_k, 1), & \text{otherwise} \end{cases}$
4	Iterative map	$x_{k+1} = \sin(a\pi/x_k), \quad a = 0.7$
5	Logistic map	$x_{k+1} = ax_k(1 - x_k), \quad a = 4$
6	Piecewise map	$x_{k+1} = \begin{cases} \dfrac{x_k}{P}, & 0 \leqslant x_k < P \\[2mm] \dfrac{x_k - P}{0.5 - P}, & P \leqslant x_k < 0.5 \\[2mm] \dfrac{1 - P - x_k}{0.5 - P}, & 0.5 \leqslant x_k \leqslant 1 - P \\[2mm] \dfrac{1 - x_k}{P}, & 1 - P \leqslant x_k \leqslant 1 \end{cases}$
7	Sine map	$x_{k+1} = \dfrac{a}{4} \sin(\pi x_k), \quad a = 4$
8	Singer map	$x_{k+1} = \mu(7.86 x_k - 23.31 x_k^2 + 28.75 x_k^3 - 13.302\,875 x_k^4), \quad u = 1.07$
9	Sinusoidal map	$x_{k+1} = ax_k^2 \sin(\pi x_k), \quad a = 2.3$
10	Tent map	$x_{k+1} = \begin{cases} \dfrac{x_k}{0.7}, & x_k < 0.7 \\[2mm] \dfrac{10}{3}(1 - x_k), & x_k \geqslant 0.7 \end{cases}$

Algorithm 1 The pseudocode for CMFO
1   Initialize chaotic value cc
2   for $i = 1 : n$
3   for $j = 1 : d$
4           $cc = \mathrm{chaotic}(cc)$;
5   initialize $[m_{ij}]_0 = lb_{min,j} + (ub_{max,j} - lb_{min,j}) * cc$
6   end
7   end
8   Calculate the fitness values of M_0 and store it in an array OM_0
9   While the termination condition is not satisfied
10   if t (the iteration counter) $== 1$
11          $OF_1 = \mathrm{sort}(OM_0)$ (sort OM_0 in descending order)
12          $F_1 = \mathrm{sort}(M_0)$ (sort M_0 according to OM_0 in descending order)
13   else
14          $OF_t = \mathrm{sort}(OM_{t-1}, OM_t)$ (sort $[OM_{t-1}, OM_t]$ in descending order)
15   end
16   Update a using (9)
17   for $i = 1 : n$
18          $cc = \mathrm{chaotic}(cc)$
19          $r = a \cdot cc + 1$
20   Calculate $D(i)$ using (7) with respect to $M_t(i,:)$ and $F_t(i,:)$
21   Update $M_{t+1}(i,:)$ using (5) and (6) with respect to $M_t(i,:)$ and $F_t(i,:)$
22          $I = M_{t+1}(i,:) < lb; J = M_{t+1}(i,:) < ub$
23   for $j = 1 : d$
24   if $I(j) == 1$ or $J(j) == 1$
25                  $cc = \mathrm{chaotic}(cc)$
26                  $M_{t+1}(i,j) = lb_{min,j} + (ub_{max,j} - lb_{min,j}) * cc$
27   end
28   end
29   end
30 End while

3.1   Chaotic maps for population initialization

Like most traditional metaheuristic algorithms, the original MFO also adopts the method of generating initial solutions based on uniformly distributed random number sequences. The initial position of each moth is set according to the following formula:

$$m_{ij} = lb_{min,j} + (ub_{max,j} - lb_{min,j}) \cdot \mathrm{rand}(0, 1) \qquad (10)$$

where m_{ij} denotes the jth dimension of the ith solution (moth), $ub_{max,j}$ and $lb_{min,j}$ are the upper and lower boundary values of the dimension j. $\mathrm{rand}(0, 1)$ is a uniformly distributed random number sequence in the range of (0,1). However, this method may lead to the moths far away from the optimal solution and deteriorate the global search

performance. Therefore, we use chaotic maps instead of $\mathrm{rand}(0,1)$ to initialize the initial position of moths in lines from 2 to 7.

3.2　Chaotic maps for boundary handing

In the original MFO, when certain dimensions of moths fly out of the boundary, the set on the boundary mechanism [42] is used to reset these moths as boundary values. As a result, lots of moths gathered on the boundary. Obviously, this method is suitable for the problem whose optimal solution is on the boundary [38]. Obviously, the randomness of the bound handling mechanism is another choice when facing the boundary overstepping. For the sake of convenience, the original MFO using the set on the boundary mechanism denotes by the improved MFO (IMFO), the original MFO using the randomness of the bound handling mechanism denotes by MFO in the later sections. In the next section, we will see that MFO outperforms IMFO through experiments. In our study, CMFO utilizes chaotic maps to take place of the randomness of the bound handling mechanism which is in lines from 23 to 28.

3.3　Chaotic maps for distance parameter r

In IMFO, distance parameter r determines the distance between the next position of a moth and its corresponding flame. With r changing, the location of the moth is distributed around the flame, forming a super ellipse around the flame in all dimensions, which guarantees the exploration and exploitation of the search space. However, the IMFO uses a uniformly distributed random number in the range of $(0,1)$ to randomize distance parameter r, which may deteriorate the global search capability. In our experiments, distance parameter r is turned by chaos maps, which is processed in lines 18 to 19. This strategy may enhance the balance between exploration and exploration.

4. Experimental results and discussion

To evaluate the performance of the proposed CMFO algorithms, they are tested on 18 benchmark functions. These test functions are divided into three groups: unimodal, multi-modal, and composite functions [43]. Unimodal test functions in Table 2 have a single optimum, so they can test the exploitation of algorithms. Multi-modal test functions in Table 3 have more than one local optimum but only one global optimum. Good optimization algorithms should be able to avoid all the local optima and determine the global optimum.

Table 2　Unimodal benchmark functions

Formulation	Search range
$F_1(\boldsymbol{X}) = \sum\limits_{i=1}^{n} ix_i^4 + \mathrm{random}[0,1]$	$x_i \in [-1.28, 1.28]$
$F_2(\boldsymbol{X}) = \sum\limits_{i=1}^{n-1} (100(x_{i+1}-x_i^2)^2+(1-x_i)^2)$	$x_i \in [-30, 30]$
$F_3(\boldsymbol{X}) = \sum\limits_{i=1}^{n} \left(\sum\limits_{j-1}^{i} x_j\right)^2$	$x_i \in [-100, 100]$
$F_4(\boldsymbol{X}) = \sum\limits_{i=1}^{n} |x_i| + \prod\limits_{i=1}^{n} |x_i|$	$x_i \in [-10, 10]$
$F_5(\boldsymbol{X}) = \sum\limits_{i=1}^{n} x_i^2$	$x_i \in [-100, 100]$
$F_6(\boldsymbol{X}) = \sum\limits_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	$x_i \in [-100, 100]$

Table 3　Multimodal benchmark functions

Formulation	Search range
$F_7(\boldsymbol{X}) = -20\exp\left(-0.2\sqrt{\dfrac{1}{n}\sum\limits_{i=1}^{n} x_i^2}\right) - \exp\left(\dfrac{1}{n}\sum\limits_{i=1}^{n}\cos 2\pi x_i^2\right) + 20 + e$	$x_i \in [-32, 32]$
$F_8(\boldsymbol{X}) = \dfrac{1}{4\,000}\sum\limits_{i=1}^{n} x_i^2 - \prod\limits_{i=1}^{n}\dfrac{\cos x_i}{\sqrt{i}} + 1$	$x_i \in [-600, 600]$
$F_9(\boldsymbol{X}) = \dfrac{\pi}{n}\left\{10\sin^2(\pi y_1) + \sum\limits_{i=1}^{n}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})] + (y_n-1)^2\right\} + \sum\limits_{i=1}^{n} u(x_i,10,100,4), y_i = 1 + (x_i+1)/4$	$x_i \in [-50, 50]$
$F_{10}(\boldsymbol{X}) = 0.1\left\{\sin^3(3\pi x_1) + \sum\limits_{i=1}^{n-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})] + (x_n-1)^2[1+\sin^2(2\pi x_n)]\right\} + \sum\limits_{i=1}^{n} u(x_i,5,100,4)$	$x_i \in [-50, 50]$
$F_{11}(\boldsymbol{X}) = \sum\limits_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i) + 10)$	$x_i \in [-5.12, 5.12]$
$F_{12}(\boldsymbol{X}) = 418.982\,9n - \sum\limits_{i=1}^{n}(x_i\sin\sqrt{|x_i|})$	$x_i \in [-500, 500]$

Table 4   Composite benchmark functions

Formulation	Search range
F_{13}(CF1) : $f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10} =$ Sphere Function $[\sigma_1, \sigma_2, \ldots, \sigma_{10}] = [1, 1, \ldots, 1]$ $[\lambda_1, \lambda_2, \ldots, \lambda_{10}] = [5/100, 5/100, 5/100, 5/100, 5/100, 5/100, 5/100, 5/100, 5/100, 5/100]$	$[-5, 5]$
F_{14}(CF2) : $f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10} =$ Griewank Function $[\sigma_1, \sigma_2, \ldots, \sigma_{10}] = [1, 1, \ldots, 1]$ $[\lambda_1, \lambda_2, \ldots, \lambda_{10}] = [5/100, 5/100, 5/100, 5/100, 5/100, 5/100, 5/100, 5/100, 5/100, 5/100]$	$[-5, 5]$
F_{15}(CF3) : $f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10} =$ Griewank Function $[\sigma_1, \sigma_2, \ldots, \sigma_{10}] = [1, 1, \ldots, 1]$ $[\lambda_1, \lambda_2, \ldots, \lambda_{10}] = [1, 1, \ldots, 1]$	$[-5, 5]$
F_{16}(CF4) : $f_1, f_2 =$ Ackley Function, $f_3, f_4 =$ Rastrigin Function, $f_5, f_6 =$ Weierstrass Function, $f_7, f_8 =$ Griewank Function, $f_9, f_{10} =$ Sphere Function $[\sigma_1, \sigma_2, \ldots, \sigma_{10}] = [1, 1, \ldots, 1]$ $[\lambda_1, \lambda_2, \ldots, \lambda_{10}] = [5/32, 5/32, 1, 1, 5/0.5, 5/0.5, 5/100, 5/100, 5/100, 5/100]$	$[-5, 5]$
F_{17}(CF5) : $f_1, f_2 =$ Rastrigin Function, $f_3, f_4 =$ Weierstrass Function, $f_5, f_6 =$ Griewank Function, $f_7, f_8 =$ Ackley Function, $f_9, f_{10} =$ Sphere Function $[\sigma_1, \sigma_2, \ldots, \sigma_{10}] = [1, 1, \ldots, 1]$ $[\lambda_1, \lambda_2, \ldots, \lambda_{10}] = [1/5, 1/5, 5/0.5, 5/0.5, 5/100, 5/100, 5/32, 5/32, 5/100, 5/100]$	$[-5, 5]$
F_{18}(CF6) : $f_1, f_2 =$ Rastrigin Function, $f_3, f_4 =$ Weierstrass Function, $f_5, f_6 =$ Griewank Function, $f_7, f_8 =$ Ackley Function, $f_9, f_{10} =$ Sphere Function $[\sigma_1, \sigma_2, \ldots, \sigma_{10}] = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$ $[\lambda_1, \lambda_2, \ldots, \lambda_{10}] = [0.1 \times 1/5, 0.2 \times 1/5, 0.3 \times 5/0.5, 0.4 \times 5/0.5, 0.5 \times 5/100, 0.6 \times 5/100, 0.7 \times 5/32, 0.8 \times 5/32,$ $0.9 \times 5/100, 1]$	$[-5, 5]$

For this reason, multi-modal test functions can test the exploration and local optima avoidance of algorithms. Composite test functions in Table 4 have a lot of local optima and different shapes [19]. Therefore, the balance of the exploration and exploitation can be tested by the last group of test functions. Note that the minima of all the benchmark functions are equal to 0.

In our experiments, all the algorithms are run 100 times. The number of dimensions is set to 30. For all the algorithms, the maximum number of iterations and the population size are equal to 100 and 30 respectively. Parameter b is set as 1.

Tables 5 – 17 present the experimental results for test functions described above. The average (mean) and standard deviation (SD) of the best solution obtained by the corresponding algorithms are listed in tables. To determine whether the results achieved by different algorithms are statistically different, a nonparametric statistical test called Wilcoxon's rank sum test is introduced. It is carried out at the default 5% significance level. The p values calculated in the Wilcoxon's rank sum test are shown in tables. N/A indicates the corresponding algorithm could not be compared with itself in the Wilcoxon's rank sum test. $p < 0.05$ means there is definitively difference between the results of the compared algorithms. Note that the best results are highlighted in bold and $p \geqslant 0.05$ are underlined.

4.1 Comparison of two kinds of cross-border processing methods

Table 5 shows the results of IMFO and MFO benchmarked on test functions. From Table 5, we can note that MFO obtains better results compared to IMFO on almost all test functions except F_{12}. F_{12} is a multi-modal test function and has a global optimal solution at point $\boldsymbol{X}^* = (420.968\,7, 420.968\,7, \ldots, 420.968\,7)$. We can see that point $\boldsymbol{X}^*$ is quite close to the bound.

Table 5   Results of MFO and IMFO

Func- tion	MFO			IMFO			Func- tion	MFO			IMFO		
	Mean	SD	p value	Mean	SD	p value		Mean	SD	p value	Mean	SD	p value
F_1	**1.44E+00**	**6.71E-01**	**N/A**	9.46E+00	6.03E+00	4.08E-33	F_{10}	**4.08E+06**	**3.52E+06**	**N/A**	4.80E+07	3.83E+07	2.99E-32
F_2	**2.19E+06**	**1.01E+06**	**N/A**	1.60E+07	1.22E+07	4.08E-33	F_{11}	**1.27E+02**	**2.15E+01**	**N/A**	2.23E+02	2.72E+01	7.99E-34
F_3	**1.56E+04**	**3.67E+03**	**N/A**	3.85E+04	9.44E+03	6.88E-34	F_{12}	5.44E+03	5.68E+02	8.34E-08	**4.95E+03**	**6.48E+02**	**N/A**
F_4	**1.91E+01**	**3.43E+00**	**N/A**	6.48E+01	1.70E+01	5.93E-34	F_{13}	**7.43E+01**	**4.99E+01**	**N/A**	9.77E+01	5.56E+01	2.38E-08
F_5	**3.41E+03**	**1.07E+03**	**N/A**	1.10E+04	4.92E+03	6.11E-34	F_{14}	**1.01E+02**	**5.04E+01**	**N/A**	1.18E+02	3.89E+01	8.69E-08
F_6	**3.35E+03**	**9.79E+02**	**N/A**	1.06E+04	4.32E+03	5.35E-32	F_{15}	**9.04E+02**	**1.97E+02**	**N/A**	1.33E+03	2.58E+02	4.62E-23
F_7	**1.15E+01**	**1.22E+00**	**N/A**	1.85E+01	1.66E+00	1.29E-33	F_{16}	**7.39E+02**	**5.51E+01**	**N/A**	7.95E+02	7.32E+01	1.11E-09
F_8	**3.19E+01**	**8.61E+00**	**N/A**	9.77E+01	3.37E+01	3.78E-34	F_{17}	**1.00E+02**	**2.36E+01**	**N/A**	1.29E+02	3.86E+01	9.53E-10
F_9	**7.71E+05**	**9.13E+05**	**N/A**	1.88E+07	3.19E+07	9.00E-34	F_{18}	**9.31E+02**	**9.37E+00**	**N/A**	9.78E+02	2.78E+01	4.33E-33

Therefore, when some dimensions of a solution cross the boundary, it will gather around point X^* using the methods of the set on the boundary mechanism. This also shows that the information of functions itself is very important and conductive for optimization algorithms. However, the positions of the optimal solutions are unknown, so the idea that they are randomly distributed in the solution space is appropriate. The results also verify that the performance of MFO with randomness of the bound handling mechanism is better than the performance of IMFO with the set on the boundary mechanism.

4.2 Comparison of MFO and CMFO

Tables 6–8 present the results of the CMFO algorithm with chaotic initialization population on three groups of test functions.

Table 6　Results of unimodal benchmark functions for chaotic initialization population

Algorithm	F_1			F_2			F_3		
	Mean	SD	p value	Mean	SD	p value	Mean	SD	p value
MFO	1.58E+00	7.84E-01	3.51E-01	2.37E+06	1.18E+06	1.40E-01	1.44E+04	3.37E+03	2.39E-01
CMFO1	1.43E+00	1.01E+00	4.73E-01	1.93E+06	9.89E+05	7.97E-01	1.57E+04	4.17E+03	4.99E-02
CMFO2	1.39E+00	5.36E-01	8.18E-01	2.06E+06	1.23E+06	6.55E-01	1.62E+04	4.52E+03	5.31E-02
CMFO3	5.98E+02	1.60E-02	6.80E-08	9.46E+08	6.41E+01	6.80E-08	5.92E+06	1.05E+04	6.80E-08
CMFO4	1.42E+00	4.23E-01	5.98E-01	2.36E+06	1.47E+06	1.72E-01	1.58E+04	4.35E+03	6.39E-02
CMFO5	**1.31E+00**	**4.11E-01**	**N/A**	2.32E+06	1.33E+06	2.18E-01	1.68E+04	3.26E+03	1.95E-03
CMFO6	1.56E+00	5.05E-01	5.65E-02	2.17E+06	1.37E+06	5.25E-01	1.49E+04	3.69E+03	1.26E-01
CMFO7	1.50E+00	6.09E-01	4.73E-01	2.72E+06	1.46E+06	4.11E-02	1.66E+04	3.65E+03	5.56E-03
CMFO8	1.63E+00	4.34E-01	2.56E-02	3.13E+06	1.34E+06	6.22E-04	1.71E+04	4.24E+03	4.70E-03
CMFO9	1.69E+00	7.26E-01	1.26E-01	2.79E+06	1.82E+06	1.08E-01	**1.33E+04**	**2.98E+03**	**N/A**
CMFO10	1.38E+00	5.93E-01	7.56E-01	**1.73E+06**	**6.83E+05**	**N/A**	1.70E+04	4.15E+03	6.04E-03

Algorithm	F_4			F_5			F_6		
	Mean	SD	p value	Mean	SD	p value	Mean	SD	p value
MFO	**1.67E+01**	**3.65E+00**	**N/A**	**3.11E+03**	**9.55E+02**	**N/A**	3.97E+03	1.29E+03	1.08E-01
CMFO1	1.82E+01	4.60E+00	2.98E-01	3.47E+03	7.80E+02	1.72E-01	3.82E+03	7.71E+02	8.10E-02
CMFO2	2.00E+01	3.49E+00	7.71E-03	3.30E+03	7.55E+02	3.51E-01	3.48E+03	9.04E+02	5.08E-01
CMFO3	2.27E+02	2.67E+02	6.80E-08	1.31E+05	1.08E+00	6.80E-08	1.27E+05	9.50E-01	6.80E-08
CMFO4	9.16E+01	3.25E+01	6.92E-07	3.74E+03	9.26E+02	3.60E-02	3.65E+03	1.20E+03	3.37E-01
CMFO5	1.90E+01	4.21E+00	1.08E-01	3.52E+03	7.22E+02	1.20E-01	3.43E+03	1.17E+03	8.18E-01
CMFO6	1.87E+01	4.33E+00	1.56E-01	3.67E+03	1.05E+03	8.59E-02	3.66E+03	9.74E+02	3.94E-01
CMFO7	1.88E+01	2.95E+00	3.15E-02	3.65E+03	1.10E+03	1.14E-01	3.36E+03	1.07E+03	8.60E-01
CMFO8	1.87E+01	2.88E+00	3.85E-02	3.88E+03	1.16E+03	4.68E-02	3.56E+03	8.50E+02	3.10E-01
CMFO9	1.91E+01	4.78E+00	6.79E-02	3.73E+03	1.12E+03	6.79E-02	**3.29E+03**	**9.79E+02**	**N/A**
CMFO10	1.84E+01	3.36E+00	9.09E-02	3.16E+03	8.82E+02	8.18E-01	3.53E+03	1.10E+03	5.25E-01

Table 7　Results of multimodal benchmark functions for chaotic initialization population

Algorithm	F_7			F_8			F_9		
	Mean	SD	p value	Mean	SD	p value	Mean	SD	p value
MFO	1.18E+01	1.01E+00	9.03E-01	2.92E+01	1.01E+01	5.98E-01	7.65E+05	1.02E+06	1.90E-01
CMFO1	1.18E+01	1.35E+00	9.46E-01	3.28E+01	1.04E+01	8.59E-02	**3.98E+05**	**3.94E+05**	**N/A**
CMFO2	1.19E+01	1.04E+00	4.09E-01	3.32E+01	1.04E+01	4.99E-02	9.36E+05	2.18E+06	1.64E-01
CMFO3	2.04E+01	3.83E-02	6.80E-08	1.16E+03	1.77E+02	6.80E-08	2.72E+09	2.85E-01	6.80E-08
CMFO4	1.18E+01	7.33E-01	8.82E-01	3.30E+01	9.07E+00	4.11E-02	8.87E+05	1.12E+06	3.10E-01
CMFO5	**1.17E+01**	**1.23E+00**	**N/A**	3.24E+01	5.27E+00	1.67E-02	5.01E+05	4.22E+05	3.51E-01
CMFO6	1.20E+01	1.10E+00	4.09E-01	3.04E+01	8.46E+00	2.85E-01	9.13E+05	8.67E+05	4.68E-02
CMFO7	1.21E+01	1.21E+00	2.18E-01	3.12E+01	7.58E+00	1.02E-01	7.22E+05	8.10E+05	3.65E-01
CMFO8	1.17E+01	1.24E+00	9.89E-01	3.25E+01	1.18E+01	1.40E-01	2.86E+06	3.47E+06	1.10E-05
CMFO9	1.18E+01	1.07E+00	9.68E-01	3.17E+01	8.96E+00	1.48E-01	1.16E+06	9.33E+05	6.56E-03
CMFO10	1.22E+01	1.12E+00	4.25E-01	**2.74E+01**	**7.79E+00**	**N/A**	6.53E+05	5.17E+05	1.08E-01

Algorithm	F_{10}			F_{11}			F_{12}		
	Mean	SD	p value	Mean	SD	p value	Mean	SD	p value
MFO	4.33E+06	3.27E+06	2.62E-01	**1.18E+02**	**1.76E+01**	**N/A**	5.51E+03	4.96E+02	5.63E-04
CMFO1	**2.78E+06**	**1.42E+06**	**N/A**	1.27E+02	2.01E+01	1.72E-01	5.34E+03	5.50E+02	7.71E-03
CMFO2	4.55E+06	3.96E+06	1.14E-01	1.24E+02	1.73E+01	3.23E-01	5.64E+03	6.12E+02	4.16E-04
CMFO3	4.47E+09	1.35E+00	6.80E-08	5.58E+02	3.21E+00	6.80E-08	1.06E+04	1.37E+02	6.80E-08
CMFO4	5.02E+06	5.89E+06	3.65E-01	1.28E+02	1.75E+01	1.14E-01	5.40E+03	4.82E+02	3.34E-03
CMFO5	5.86E+06	4.09E+06	3.97E-03	1.29E+02	1.53E+01	7.20E-02	5.46E+03	5.03E+02	1.95E-03
CMFO6	3.75E+06	4.20E+06	7.35E-01	1.33E+02	2.17E+01	4.11E-02	5.82E+03	5.74E+02	4.17E-05
CMFO7	4.99E+06	3.77E+06	1.44E-02	1.25E+02	2.18E+01	4.25E-01	5.15E+03	5.87E+02	1.40E-01
CMFO8	6.02E+06	4.52E+06	1.79E-04	1.30E+02	1.72E+01	4.99E-02	**4.88E+03**	**5.30E+02**	**N/A**
CMFO9	6.41E+06	8.58E+06	7.64E-02	1.34E+02	1.51E+01	6.56E-03	5.20E+03	4.20E+02	2.56E-02
CMFO10	3.78E+06	3.39E+06	5.25E-01	1.41E+02	2.12E+01	6.87E-04	5.73E+03	4.36E+02	2.60E-05

Table 8   Results of composite benchmark functions for chaotic initialization population

Algorithm	F_{13}			F_{14}			F_{15}		
	Mean	SD	p value	Mean	SD	p value	Mean	SD	p value
MFO	8.36E+01	4.63E+01	6.39E-02	9.27E+01	1.91E+01	8.39E-01	9.77E+02	2.61E+02	1.02E-01
CMFO1	7.02E+01	2.16E+01	4.73E-01	1.03E+02	4.97E+01	5.98E-01	9.47E+02	2.08E+02	9.09E-02
CMFO2	8.05E+01	4.70E+01	5.65E-02	1.21E+02	8.93E+01	4.57E-01	9.85E+02	1.59E+02	1.14E-02
CMFO3	1.48E+03	1.38E-01	6.80E-08	1.47E+03	4.18E+01	6.80E-08	2.08E+03	2.54E+00	6.80E-08
CMFO4	6.85E+01	2.55E+01	7.35E-01	9.13E+01	1.48E+01	8.39E-01	8.84E+02	1.57E+02	3.94E-01
CMFO5	6.63E+01	1.49E+01	3.51E-01	9.60E+01	1.63E+01	3.10E-01	8.36E+02	2.30E+02	9.89E-01
CMFO6	1.41E+02	4.36E+01	1.10E-05	1.74E+02	3.86E+01	1.20E-06	9.58E+02	2.02E+02	7.64E-02
CMFO7	7.86E+01	6.57E+01	3.79E-01	9.40E+01	1.66E+01	5.79E-01	**8.34E+02**	**1.85E+02**	**N/A**
CMFO8	6.98E+01	1.51E+01	2.29E-01	1.01E+02	5.14E+01	7.76E-01	9.79E+02	1.68E+02	1.23E-02
CMFO9	**6.27E+01**	**1.41E+01**	**N/A**	**9.12E+01**	**1.53E+01**	**N/A**	8.98E+02	2.03E+02	2.98E-01
CMFO10	1.04E+02	7.83E+01	1.79E-02	1.05E+02	6.65E+01	7.97E-01	9.14E+02	2.17E+02	3.51E-01

Algorithm	F_{16}			F_{17}			F_{18}		
	Mean	SD	p value	Mean	SD	p value	Mean	SD	p value
MFO	7.50E+02	5.17E+01	1.79E-02	1.02E+02	2.40E+01	7.15E-01	**9.32E+02**	**7.19E+00**	**N/A**
CMFO1	7.23E+02	5.36E+01	4.57E-01	1.01E+02	1.86E+01	4.57E-01	9.36E+02	9.46E+00	2.39E-01
CMFO2	7.30E+02	5.27E+01	2.08E-01	1.02E+02	1.63E+01	3.94E-01	9.34E+02	1.19E+01	7.15E-01
CMFO3	1.43E+03	3.87E+00	6.80E-08	1.52E+03	6.78E+01	6.80E-08	1.88E+03	1.42E-02	6.80E-08
CMFO4	7.80E+02	5.81E+01	2.00E-04	1.13E+02	3.46E+01	1.33E-01	9.34E+02	1.10E+01	8.18E-01
CMFO5	**7.08E+02**	**5.03E+01**	**N/A**	1.01E+02	2.87E+01	6.36E-01	9.37E+02	9.67E+00	8.59E-02
CMFO6	7.58E+02	4.35E+01	3.34E-03	1.95E+02	1.56E+01	6.80E-08	9.35E+02	1.09E+01	5.61E-01
CMFO7	7.44E+02	5.29E+01	1.33E-02	1.13E+02	2.49E+01	3.37E-02	9.35E+02	1.06E+01	4.73E-01
CMFO8	7.55E+02	5.07E+01	5.12E-03	1.04E+02	1.48E+01	1.08E-01	9.34E+02	1.07E+01	9.03E-01
CMFO9	7.66E+02	8.19E+01	1.44E-02	1.03E+02	1.60E+01	2.62E-01	9.34E+02	1.01E+01	5.61E-01
CMFO10	7.70E+02	3.67E+01	4.60E-04	**9.73E+01**	**1.72E+01**	**N/A**	9.36E+02	1.14E+01	2.29E-01

Table 9   Statistical results for chaotic initialization population

Group	Number	MFO	CMFO1	CMFO2	CMFO3	CMFO4	CMFO5	CMFO6	CMFO7	CMFO8	CMFO9	CMFO10
Group1	num_1	2	0	0	0	0	1	0	0	0	2	1
	num_2	4	5	5	0	4	4	6	3	2	4	4
Group2	num_1	1	2	0	0	0	1	0	0	1	0	1
	num_2	4	3	5	0	4	2	3	5	2	3	3
Group3	num_1	1	0	0	0	0	1	0	1	0	2	1
	num_2	4	6	5	0	5	5	2	3	4	3	3

CMFO1 to CMFO10 correspond to the algorithms initialized by ten different chaotic maps (Chebyshev, Circle, Gauss, Iterative, Logistic, Piecewise, Sine, Singer, Sinusoidal, and Tent). Table 9 shows a summary of Tables 6 – 8. In Table 9, num_1 represents the number of the test functions on which the algorithm performs better than the other algorithms. num_2 represents the number of the test functions on which the algorithm is equivalent to the algorithm which obtains the best solution.

It can be seen in Table 6 that CMFO3, CMFO7, and CMFO8 show worse results compared to the MFO algorithm. This shows that Gaussian, Sine, and Singer chaotic initialization populations on unimodal test functions are not able to improve the performance of MFO. The CMFO1, CMFO2, CMFO5, CMFO9 and CMFO10 show similar results compared to MFO on unimodal test functions. For multi-modal test functions showed in Table 7, the performance of the CMFO3, CMFO5, CMFO8, and CMFO9 are worse than MFO and the CMFO1, CMFO2

and CMFO7 obtain similar results compared to MFO. On the last group of the test functions, CMFO5 and CMFO9 obtain a little better results compared to MFO and show obviously better performance than the other variants of CMFO. In general, we can conclude that chaotic initialization population is not able to significantly improve the performance of MFO.

Tables 10 – 12 present the results of CMFO with chaotic boundary handing on three groups of test functions. CMFO11 to CMFO20 correspond to the algorithms with different chaotic maps for boundary handing. Table 13 shows a summary of Tables 10 – 12, num_1 and num_2 have the same meaning as in Table 9. Table 10 shows that CMFO11, CMFO13 – CMFO18 have worse results compared to the MFO algorithm and the other variants of CMFO cannot improve the performance of MFO on unimodal test functions. Table 11 shows that the CMFO12 and CMFO19 obtain better results than the MFO algorithm. This shows that the Circle and Sinusoidal chaotic

boundary handing are able to improve the performance of MFO. In contrast the other variants of CMFO algorithms show worse results compared to MFO on multi-modal test functions. For composite test functions, the performance of CMFO14, CMFO15 and CMFO17 – CMFO20 is equivalent to MFO, while the other variants of CMFO show worse results compared to the MFO algorithm. In general, CMFO19 can have similar performance with the MFO algorithm while the other variants of CMFO are inferior to the MFO algorithm on some test functions.

Table 10   Results of unimodal benchmark functions for chaotic boundary handing

Algorithm	F_1			F_2			F_3		
	Mean	SD	p value	Mean	SD	p value	Mean	SD	p value
MFO	1.58E+00	7.84E-01	1.14E-01	2.37E+06	1.18E+06	6.01E-02	**1.44E+04**	**3.37E+03**	**N/A**
CMFO11	3.97E+00	2.42E+00	1.38E-06	6.48E+06	4.57E+06	1.81E-05	2.66E+04	7.57E+03	2.69E-06
CMFO12	1.37E+00	6.84E-01	5.79E-01	2.04E+06	1.17E+06	5.61E-01	1.69E+04	3.55E+03	4.39E-02
CMFO13	8.55E+00	7.23E+00	1.80E-06	5.60E+06	2.84E+06	1.38E-06	2.58E+04	6.77E+03	1.20E-06
CMFO14	1.74E+00	7.01E-01	1.23E-02	2.16E+06	1.10E+06	3.23E-01	1.80E+04	4.66E+03	1.79E-02
CMFO15	2.46E+00	9.06E-01	1.10E-05	2.73E+06	1.21E+06	7.71E-03	2.00E+04	5.33E+03	3.05E-04
CMFO16	1.53E+00	5.77E-01	6.39E-02	2.03E+06	9.69E+05	5.08E-01	1.67E+04	5.05E+03	1.64E-01
CMFO17	1.66E+00	6.04E-01	1.44E-02	3.54E+06	2.37E+06	5.12E-03	1.97E+04	4.98E+03	9.21E-04
CMFO18	1.55E+00	5.08E-01	3.15E-02	2.88E+06	2.08E+06	5.65E-02	1.83E+04	5.45E+03	1.67E-02
CMFO19	**1.19E+00**	**4.42E-01**	**N/A**	2.42E+06	1.42E+06	8.59E-02	1.69E+04	4.60E+03	7.64E-02
CMFO20	1.37E+00	3.68E-01	1.26E-01	**1.73E+06**	**6.83E+05**	**N/A**	1.58E+04	5.17E+03	4.73E-01

Algorithm	F_4			F_5			F_6		
	Mean	SD	p value	Mean	SD	p value	Mean	SD	p value
MFO	**1.67E+01**	**3.65E+00**	**N/A**	**3.11E+03**	**9.55E+02**	**N/A**	3.97E+03	1.29E+03	2.34E-03
CMFO11	2.39E+01	4.39E+00	1.41E-05	6.74E+03	1.49E+03	2.22E-07	6.41E+03	1.34E+03	6.80E-08
CMFO12	1.97E+01	3.37E+00	1.06E-02	3.19E+03	1.02E+03	7.15E-01	**2.78E+03**	**5.88E+02**	**N/A**
CMFO13	4.77E+01	1.45E+01	6.80E-08	7.33E+03	2.50E+03	4.54E-07	7.47E+03	3.97E+03	2.96E-07
CMFO14	2.10E+01	4.94E+00	4.70E-03	3.68E+03	1.10E+03	1.02E-01	3.77E+03	1.16E+03	2.80E-03
CMFO15	2.17E+01	4.41E+00	8.36E-04	4.49E+03	1.34E+03	4.60E-04	4.21E+03	1.47E+03	4.16E-04
CMFO16	2.10E+01	4.47E+00	3.64E-03	3.91E+03	1.11E+03	2.75E-02	3.48E+03	8.14E+02	6.04E-03
CMFO17	2.23E+01	3.91E+00	1.04E-04	4.25E+03	1.00E+03	1.01E-03	4.49E+03	1.25E+03	1.60E-05
CMFO18	1.96E+01	3.90E+00	2.39E-02	3.78E+03	1.26E+03	1.02E-01	4.57E+03	1.31E+03	2.06E-06
CMFO19	1.90E+01	3.31E+00	4.99E-02	3.15E+03	9.35E+02	9.46E-01	3.45E+03	1.02E+03	1.55E-02
CMFO20	2.01E+01	3.49E+00	3.97E-03	3.79E+03	1.30E+03	6.79E-02	2.95E+03	7.72E+02	4.09E-01

Table 11   Results of multimodal benchmark functions for chaotic boundary handing

Algorithm	F_7			F_8			F_9		
	Mean	SD	p value	Mean	SD	p value	Mean	SD	p value
MFO	1.18E+01	1.01E+00	9.09E-02	2.92E+01	1.01E+01	7.15E-01	7.65E+05	1.02E+06	1.81E-01
CMFO11	1.43E+01	1.49E+00	3.94E-07	5.53E+01	1.52E+01	2.06E-06	4.83E+06	4.60E+06	1.20E-06
CMFO12	1.15E+01	9.23E-01	4.41E-01	**2.73E+01**	**7.23E+00**	**N/A**	**5.60E+05**	**9.72E+05**	**N/A**
CMFO13	1.56E+01	2.25E+00	2.56E-07	6.23E+01	2.85E+01	1.20E-06	3.58E+06	2.27E+06	2.06E-06
CMFO14	1.22E+01	9.62E-01	2.14E-03	3.57E+01	9.21E+00	6.56E-03	5.74E+05	7.89E+05	9.89E-01
CMFO15	1.26E+01	1.21E+00	4.60E-04	3.78E+01	1.17E+01	5.56E-03	1.25E+06	1.13E+06	4.32E-03
CMFO16	1.19E+01	9.71E-01	4.68E-02	3.34E+01	9.71E+00	3.15E-02	6.71E+05	5.97E+05	9.09E-02
CMFO17	1.25E+01	1.25E+00	7.58E-04	3.67E+01	7.97E+00	6.87E-04	1.57E+06	1.68E+06	2.34E-03
CMFO18	1.23E+01	1.26E+00	5.12E-03	3.17E+01	9.49E+00	1.08E-01	1.60E+06	1.71E+06	1.12E-03
CMFO19	1.19E+01	1.10E+00	6.39E-02	2.90E+01	8.21E+00	7.97E-01	5.94E+05	4.99E+05	2.50E-01
CMFO20	**1.12E+01**	**9.69E-01**	**N/A**	3.26E+01	8.36E+00	2.94E-02	5.64E+05	5.82E+05	6.17E-01

Algorithm	F_{10}			F_{11}			F_{12}		
	Mean	SD	p value	Mean	SD	p value	Mean	SD	p value
MFO	4.33E+06	3.27E+06	4.90E-01	**1.18E+02**	**1.76E+01**	**N/A**	5.51E+03	4.96E+02	2.36E-06
CMFO11	1.35E+07	5.84E+06	3.50E-06	1.47E+02	1.58E+01	1.81E-05	5.13E+03	6.85E+02	1.44E-04
CMFO12	**3.30E+06**	**2.02E+06**	**N/A**	1.21E+02	2.20E+01	8.60E-01	5.82E+03	4.13E+02	6.92E-07
CMFO13	1.76E+07	9.50E+06	2.56E-07	1.91E+02	2.98E+01	2.22E-07	6.41E+03	4.99E+02	2.22E-07
CMFO14	5.80E+06	4.25E+06	6.79E-02	1.31E+02	2.04E+01	3.15E-02	5.22E+03	6.41E+02	1.79E-04
CMFO15	8.34E+06	6.46E+06	1.95E-03	1.41E+02	2.82E+01	3.97E-03	5.09E+03	6.07E+02	6.22E-04
CMFO16	5.25E+06	3.83E+06	1.02E-01	1.32E+02	2.06E+01	2.39E-02	5.34E+03	4.71E+02	1.81E-05
CMFO17	8.04E+06	7.99E+06	1.67E-02	1.40E+02	2.29E+01	2.34E-03	5.04E+03	2.82E+02	2.30E-05
CMFO18	6.67E+06	4.37E+06	6.56E-03	1.41E+02	2.25E+01	2.80E-03	4.69E+03	5.00E+02	1.67E-02
CMFO19	5.61E+06	5.55E+06	1.26E-01	1.28E+02	2.48E+01	1.81E-01	**4.31E+03**	**6.92E+02**	**N/A**
CMFO20	4.00E+06	2.49E+06	4.90E-01	1.31E+02	1.39E+01	2.23E-02	5.52E+03	5.86E+02	5.17E-06

Table 12　Results of composite benchmark functions for chaotic boundary handing

Algorithm	F_{13}			F_{14}			F_{15}		
	Mean	SD	p value	Mean	SD	p value	Mean	SD	p value
MFO	8.36E+01	4.63E+01	2.85E-01	9.27E+01	1.91E+01	1.48E-01	9.77E+02	2.61E+02	2.29E-01
CMFO11	1.69E+02	1.15E+02	6.92E-07	2.22E+02	1.28E+02	6.80E-08	1.52E+03	1.91E+02	1.43E-07
CMFO12	1.16E+02	8.27E+01	2.14E-03	1.13E+02	1.83E+01	1.05E-06	1.00E+03	2.14E+02	8.10E-02
CMFO13	2.69E+02	1.18E+02	6.80E-08	2.60E+02	1.03E+02	6.80E-08	1.63E+03	9.72E+01	6.80E-08
CMFO14	6.71E+01	1.15E+01	8.39E-01	9.35E+01	1.53E+01	4.68E-02	9.44E+02	2.57E+02	3.79E-01
CMFO15	6.92E+01	2.14E+01	8.39E-01	9.31E+01	1.78E+01	6.01E-02	9.92E+02	2.12E+02	1.14E-01
CMFO16	8.85E+01	4.91E+01	9.62E-02	1.25E+02	9.29E+01	1.44E-02	9.66E+02	2.56E+02	2.85E-01
CMFO17	6.84E+01	1.64E+01	7.56E-01	1.10E+02	4.94E+01	6.56E-03	9.81E+02	2.14E+02	1.20E-01
CMFO18	6.96E+01	1.61E+01	5.08E-01	**8.34E+01**	**1.08E+01**	**N/A**	8.89E+02	2.08E+02	9.68E-01
CMFO19	**6.56E+01**	**1.71E+01**	**N/A**	9.10E+01	1.42E+01	1.64E-01	9.47E+02	1.71E+02	2.98E-01
CMFO20	8.57E+01	6.92E+01	6.95E-01	9.29E+01	1.70E+01	3.15E-02	**8.80E+02**	**1.97E+02**	**N/A**

Algorithm	F_{16}			F_{17}			F_{18}		
	Mean	SD	p value	Mean	SD	p value	Mean	SD	p value
MFO	7.50E+02	5.17E+01	6.36E-01	1.02E+02	2.40E+01	5.98E-01	**9.32E+02**	**7.19E+00**	**N/A**
CMFO11	9.40E+02	1.25E+02	3.07E-06	2.29E+02	1.04E+02	6.80E-08	9.77E+02	2.24E+01	7.90E-08
CMFO12	7.59E+02	4.97E+01	1.90E-01	1.23E+02	3.66E+01	1.78E-03	9.33E+02	8.83E+00	9.89E-01
CMFO13	8.22E+02	7.85E+01	1.29E-04	3.04E+02	1.42E+02	6.80E-08	9.80E+02	2.47E+01	1.92E-07
CMFO14	7.35E+02	5.32E+01	8.82E-01	1.03E+02	2.24E+01	3.65E-01	9.34E+02	8.78E+00	8.60E-01
CMFO15	**7.34E+02**	**5.42E+01**	**N/A**	1.03E+02	2.14E+01	3.10E-01	9.35E+02	1.13E+01	6.17E-01
CMFO16	7.63E+02	7.11E+01	3.79E-01	1.21E+02	4.33E+01	4.39E-02	9.34E+02	1.01E+01	6.55E-01
CMFO17	7.40E+02	5.92E+01	9.89E-01	1.05E+02	3.39E+01	5.43E-01	9.34E+02	1.33E+01	6.75E-01
CMFO18	7.46E+02	5.69E+01	3.23E-01	**9.47E+01**	**1.23E+01**	**N/A**	9.39E+02	1.03E+01	2.94E-02
CMFO19	7.40E+02	4.10E+01	2.50E-01	9.63E+01	1.62E+01	8.60E-01	9.45E+02	8.33E+00	2.04E-05
CMFO20	7.42E+02	6.10E+01	7.15E-01	1.05E+02	2.59E+01	2.39E-01	9.36E+02	9.50E+00	1.20E-01

Table 13　Statistical results for chaotic boundary handing

Group	Number	MFO	CMFO11	CMFO12	CMFO13	CMFO14	CMFO15	CMFO16	CMFO17	CMFO18	CMFO19	CMFO20
Group1	num_1	3	0	1	0	0	0	0	0	0	1	1
	num_2	2	0	3	0	2	0	3	0	2	3	3
Group2	num_1	1	0	3	0	0	0	0	0	0	1	1
	num_2	4	0	2	0	2	0	2	0	1	5	2
Group3	num_1	1	0	0	0	0	1	0	0	2	1	1
	num_2	5	0	3	0	5	5	4	5	3	4	4

In other words, the chaotic boundary handing does not improve the performance of MFO.

From a theoretical point of view, chaotic initialization population and chaotic boundary handing can both improve the exploration of MFO. However only exploration cannot lead to better results for the metaheuristic algorithm, the exploitation and exploration must achieve a reasonable balance. From the results above, chaotic initialization population and chaotic boundary handing enhance the diversity of MFO. While they cannot boost the performance of MFO without the balance of exploitation and exploration. Such a conclusion can be drawn that chaotic maps replacing different random sequences may perform dramatically different for a given application difference. Therefore, it is baseless to specify some chaotic maps taking place of some random sequences without actual experiments.

The results of CMFO with chaotic maps for distance parameter r on three groups of test functions are presented in Tables 14 – 16. CMFO21 to CMFO30 correspond to the algorithms with different chaotic maps tuning distance parameter r. Table 17 shows a summary of Tables 14 – 16. num_1 and num_2 have the same meaning as in Table 9. From Table 14, we can see that CMFO28 shows much better results compared to the MFO algorithm on most unimodal test functions except F_3. Table 15 shows that CMFO28 yields the best results on most multi-modal functions except F_9. The p values prove that this superiority is statistically significant. On F_3 and F_9, the performance obtained by CMFO28 is equivalent to that of MFO. We also can see that CMFO28 achieves the best results on F_{14}, F_{17}, and F_{18} for composite test functions. The p values prove this superiority is statistically significant too. On the other composite functions, the performance of CMFO28 is also equivalent to that of CMFO28. In other words, the Singer chaotic tuning the distance parameter has success-

fully improved the performance of MFO. While the other variants of CMFO fail to outperform the MFO algorithm on almost all the test functions. Moreover, several representative convergence curves for CMFO28 and the MFO algorithm are shown in Fig. 1. As may be seen in this fig-ure, the CMFO28 algorithm imbedded with the Singer map for tuning distance parameter r has the fastest convergence rate. Considering Table 17 and Fig. 1, we can easily conclude that Singer maps improve the performance of MFO in terms of the balance of exploration and exploitation.

Table 14   Results of unimodal benchmark functions for tuning distance parameter r

Algorithm	F_1			F_2			F_3		
	Mean	SD	p value	Mean	SD	p value	Mean	SD	p value
MFO	1.58E+00	7.84E-01	3.60E-02	2.37E+06	1.18E+06	1.79E-04	**1.44E+04**	**3.37E+03**	**N/A**
CMFO21	2.04E+01	5.20E+00	6.80E-08	4.75E+07	7.79E+06	6.80E-08	4.24E+04	6.87E+03	6.80E-08
CMFO22	3.31E+00	1.24E+00	3.42E-07	8.32E+06	3.78E+06	6.80E-08	2.20E+04	5.15E+03	1.81E-05
CMFO23	3.76E+01	7.99E+00	6.80E-08	9.13E+07	1.19E+07	6.80E-08	5.39E+04	6.96E+03	6.80E-08
CMFO24	1.23E+00	5.55E-01	4.41E-01	2.07E+06	7.70E+05	7.41E-05	1.65E+04	3.75E+03	1.08E-01
CMFO25	2.49E+00	1.21E+00	7.58E-06	5.36E+06	2.30E+06	6.80E-08	1.55E+04	4.44E+03	4.09E-01
CMFO26	2.42E+00	1.01E+00	2.60E-05	5.27E+06	1.98E+06	7.90E-08	1.81E+04	3.07E+03	1.95E-03
CMFO27	3.20E+00	1.43E+00	7.95E-07	5.61E+06	2.45E+06	1.06E-07	2.03E+04	4.03E+03	6.61E-05
CMFO28	**1.11E+00**	**5.20E-01**	**N/A**	**1.11E+06**	**5.16E+05**	**N/A**	1.58E+04	5.77E+03	7.56E-01
CMFO29	1.82E+00	1.05E+00	4.70E-03	3.22E+06	1.89E+06	1.29E-04	2.00E+04	4.34E+03	8.29E-05
CMFO30	1.52E+00	3.86E-01	9.21E-04	2.20E+06	1.39E+06	2.34E-03	1.45E+04	3.28E+03	9.25E-01

Algorithm	F_4			F_5			F_6		
	Mean	SD	p value	Mean	SD	p value	Mean	SD	p value
MFO	1.67E+01	3.65E+00	6.22E-04	3.11E+03	9.55E+02	5.98E-01	3.97E+03	1.29E+03	7.58E-06
CMFO21	1.51E+02	2.01E+02	6.80E-08	2.59E+04	3.07E+03	6.80E-08	2.50E+04	2.70E+03	6.80E-08
CMFO22	3.37E+01	5.36E+00	6.80E-08	8.21E+03	1.99E+03	1.78E-03	8.34E+03	2.19E+03	6.80E-08
CMFO23	1.69E+03	3.14E+03	6.80E-08	3.59E+04	3.40E+03	6.80E-08	3.63E+04	3.25E+03	6.80E-08
CMFO24	2.06E+01	3.46E+00	6.01E-07	3.23E+03	7.25E+02	3.65E-01	3.92E+03	1.01E+03	1.80E-06
CMFO25	2.93E+01	4.77E+00	7.90E-08	5.35E+03	1.44E+03	3.10E-01	6.13E+03	1.77E+03	7.90E-08
CMFO26	2.61E+01	3.44E+00	7.90E-08	6.34E+03	2.22E+03	4.39E-02	5.99E+03	1.96E+03	2.56E-07
CMFO27	2.91E+01	5.11E+00	7.90E-08	6.26E+03	1.55E+03	5.43E-01	7.83E+03	2.01E+03	6.80E-08
CMFO28	**1.30E+01**	**2.51E+00**	**N/A**	**2.25E+03**	**6.79E+02**	**N/A**	**2.11E+03**	**5.79E+02**	**N/A**
CMFO29	1.79E+01	4.67E+00	9.28E-05	3.55E+03	1.35E+03	8.60E-01	3.36E+03	1.46E+03	2.34E-03
CMFO30	1.88E+01	3.56E+00	2.30E-05	3.04E+03	8.77E+02	2.39E-01	3.72E+03	9.56E+02	1.80E-06

Table 15   Results of multimodal benchmark functions for tuning distance parameter r

Algorithm	F_7			F_8			F_9		
	Mean	SD	p value	Mean	SD	p value	Mean	SD	p value
MFO	1.18E+01	1.01E+00	1.35E-03	2.92E+01	1.01E+01	9.05E-03	7.65E+05	1.02E+06	2.62E-01
CMFO21	1.87E+01	4.32E-01	6.80E-08	2.25E+02	2.30E+01	6.80E-08	5.73E+07	2.71E+07	6.80E-08
CMFO22	1.50E+01	6.98E-01	6.80E-08	8.25E+01	1.80E+01	6.80E-08	5.03E+06	4.02E+06	2.56E-07
CMFO23	1.94E+01	2.08E-01	6.80E-08	3.30E+02	2.77E+01	6.80E-08	1.42E+08	4.75E+07	6.80E-08
CMFO24	1.20E+01	1.05E+00	3.05E-04	3.50E+01	9.77E+00	4.68E-05	**4.36E+05**	**4.33E+05**	**N/A**
CMFO25	1.36E+01	9.40E-01	7.90E-08	5.13E+01	1.65E+01	1.23E-07	1.44E+06	1.07E+06	1.12E-03
CMFO26	1.40E+01	1.12E+00	7.90E-08	5.33E+01	1.35E+01	4.54E-07	1.20E+06	8.19E+05	2.14E-03
CMFO27	1.46E+01	7.20E-01	6.80E-08	6.02E+01	1.74E+01	6.80E-08	6.08E+06	6.06E+06	9.13E-07
CMFO28	**1.04E+01**	**1.21E+00**	**N/A**	**2.14E+01**	**5.73E+00**	**N/A**	4.82E+05	6.02E+05	6.75E-01
CMFO29	1.19E+01	1.41E+00	1.95E-03	3.05E+01	1.02E+01	1.95E-03	8.39E+05	8.56E+05	1.56E-01
CMFO30	1.15E+01	8.73E-01	1.78E-03	2.98E+01	6.77E+00	3.75E-04	8.89E+05	1.67E+06	1.21E-01

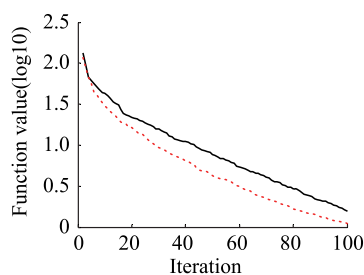| Algorithm	F_{10}			F_{11}			F_{12}		
	Mean	SD	p value	Mean	SD	p value	Mean	SD	p value
MFO	4.33E+06	3.27E+06	2.23E-02	1.18E+02	1.76E+01	2.39E-01	5.51E+03	4.96E+02	5.08E-01
CMFO21	1.64E+08	5.31E+07	6.80E-08	2.90E+02	1.85E+01	6.80E-08	7.88E+03	5.03E+02	6.80E-08
CMFO22	1.80E+07	8.20E+06	6.80E-08	1.82E+02	2.29E+01	1.06E-07	5.87E+03	5.56E+02	1.23E-02
CMFO23	3.24E+08	7.36E+07	6.80E-08	3.23E+02	2.18E+01	6.80E-08	7.97E+03	4.19E+02	6.80E-08
CMFO24	5.02E+06	5.54E+06	3.60E-02	1.22E+02	1.91E+01	8.10E-02	5.58E+03	4.61E+02	1.99E-01
CMFO25	1.19E+07	6.68E+06	1.80E-06	1.47E+02	2.01E+01	1.10E-05	5.53E+03	5.63E+02	4.09E-01
CMFO26	1.36E+07	7.04E+06	1.06E-07	1.64E+02	1.71E+01	3.94E-07	5.91E+03	5.54E+02	2.80E-03
CMFO27	1.37E+07	7.79E+06	1.23E-07	1.61E+02	2.66E+01	2.69E-06	5.58E+03	8.93E+02	4.09E-01
CMFO28	**2.29E+06**	**1.76E+06**	**N/A**	**1.11E+02**	**2.14E+01**	**N/A**	**5.39E+03**	**4.57E+02**	**N/A**
CMFO29	5.62E+06	4.18E+06	3.97E-03	1.55E+02	2.53E+01	7.58E-06	6.57E+03	5.69E+02	2.36E-06
CMFO30	4.49E+06	3.31E+06	1.23E-02	1.33E+02	2.11E+01	3.06E-03	5.71E+03	6.05E+02	1.08E-01

Table 16   Results of composite benchmark functions for tuning distance parameter r

Algorithm	F_{13}			F_{14}			F_{15}		
	Mean	SD	p value	Mean	SD	p value	Mean	SD	p value
MFO	8.36E+01	4.63E+01	2.85E-01	9.27E+01	1.91E+01	9.62E-02	9.77E+02	2.61E+02	3.34E-03
CMFO21	3.34E+02	5.06E+01	7.90E-08	3.57E+02	5.20E+01	9.17E-08	1.74E+03	2.17E+01	6.80E-08
CMFO22	1.33E+02	2.84E+01	4.54E-06	1.54E+02	3.72E+01	1.80E-06	1.40E+03	1.93E+02	1.92E-07
CMFO23	4.39E+02	7.87E+01	6.80E-08	4.81E+02	6.69E+01	6.80E-08	1.78E+03	2.00E+01	6.80E-08
CMFO24	9.81E+01	9.60E+01	9.46E-01	1.09E+02	4.52E+01	2.34E-03	1.04E+03	2.03E+02	2.47E-04
CMFO25	9.13E+01	2.03E+01	2.56E-03	1.17E+02	2.57E+01	6.61E-05	1.16E+03	2.13E+02	7.58E-06
CMFO26	1.27E+02	9.10E+01	3.06E-03	1.23E+02	2.04E+01	4.54E-06	1.27E+03	1.97E+02	6.92E-07
CMFO27	1.16E+02	6.47E+01	1.44E-04	1.35E+02	2.55E+01	1.58E-06	1.34E+03	1.66E+02	1.23E-07
CMFO28	8.53E+01	8.74E+01	5.25E-01	9.22E+01	4.96E+01	8.39E-01	**7.26E+02**	**2.28E+02**	**N/A**
CMFO29	1.30E+02	7.44E+01	2.30E-05	1.46E+02	3.10E+01	2.36E-06	1.20E+03	2.88E+02	2.30E-05
CMFO30	**7.56E+01**	**4.27E+01**	**N/A**	**9.08E+01**	**5.02E+01**	**N/A**	9.05E+02	2.43E+02	3.37E-02

Algorithm	F_{16}			F_{17}			F_{18}		
	Mean	SD	p value	Mean	SD	p value	Mean	SD	p value
MFO	7.50E+02	5.17E+01	9.46E-01	1.02E+02	2.40E+01	2.22E-04	9.32E+02	7.19E+00	3.34E-03
CMFO21	9.73E+02	6.97E+01	6.80E-08	3.85E+02	5.86E+01	6.80E-08	1.11E+03	3.37E+01	6.80E-08
CMFO22	7.69E+02	5.32E+01	2.23E-02	1.64E+02	3.09E+01	7.90E-08	9.69E+02	1.82E+01	1.66E-07
CMFO23	1.15E+03	6.60E+01	6.80E-08	5.07E+02	6.58E+01	6.80E-08	1.28E+03	3.51E+01	6.80E-08
CMFO24	**7.36E+02**	**5.68E+01**	**N/A**	1.15E+02	4.28E+01	4.17E-05	9.36E+02	1.49E+01	3.34E-03
CMFO25	7.52E+02	5.01E+01	9.09E-02	1.26E+02	1.74E+01	2.56E-07	9.58E+02	1.74E+01	2.22E-07
CMFO26	7.57E+02	6.14E+01	8.59E-02	1.39E+02	3.23E+01	1.23E-07	9.61E+02	1.46E+01	7.90E-08
CMFO27	7.70E+02	5.39E+01	3.37E-02	1.47E+02	3.67E+01	1.66E-07	9.61E+02	1.87E+01	1.23E-07
CMFO28	7.46E+02	4.38E+01	7.35E-01	**8.29E+01**	**1.17E+01**	**N/A**	**9.25E+02**	**5.67E+00**	**N/A**
CMFO29	7.95E+02	5.39E+01	8.36E-04	1.38E+02	3.46E+01	4.54E-06	9.51E+02	1.52E+01	2.56E-07
CMFO30	7.49E+02	5.34E+01	9.25E-01	9.76E+01	1.56E+01	1.95E-03	9.31E+02	9.25E+00	5.31E-02

Table 17   Statistical results of chaotic maps for tuning distance parameter r

Group	Number	MFO	CMFO21	CMFO22	CMFO23	CMFO24	CMFO25	CMFO26	CMFO27	CMFO28	CMFO29	CMFO30
Group1	num_1	1	0	0	0	0	0	0	0	5	0	0
	num_2	1	0	0	0	3	2	0	1	1	1	2
Group2	num_1	0	0	0	0	1	0	0	0	5	0	0
	num_3	3	0	0	0	2	1	0	1	1	1	2
Group3	num_1	0	0	0	0	1	0	0	0	3	0	2
	num_4	3	0	0	0	1	1	1	0	3	0	2

(a) F_1

(c) F_5

(b) F_2

(d) F_6

(e) F_7

(f) F_8

(g) F_9

(h) F_{10}

(i) F_{11}

(j) F_{15}

(k) F_{17}
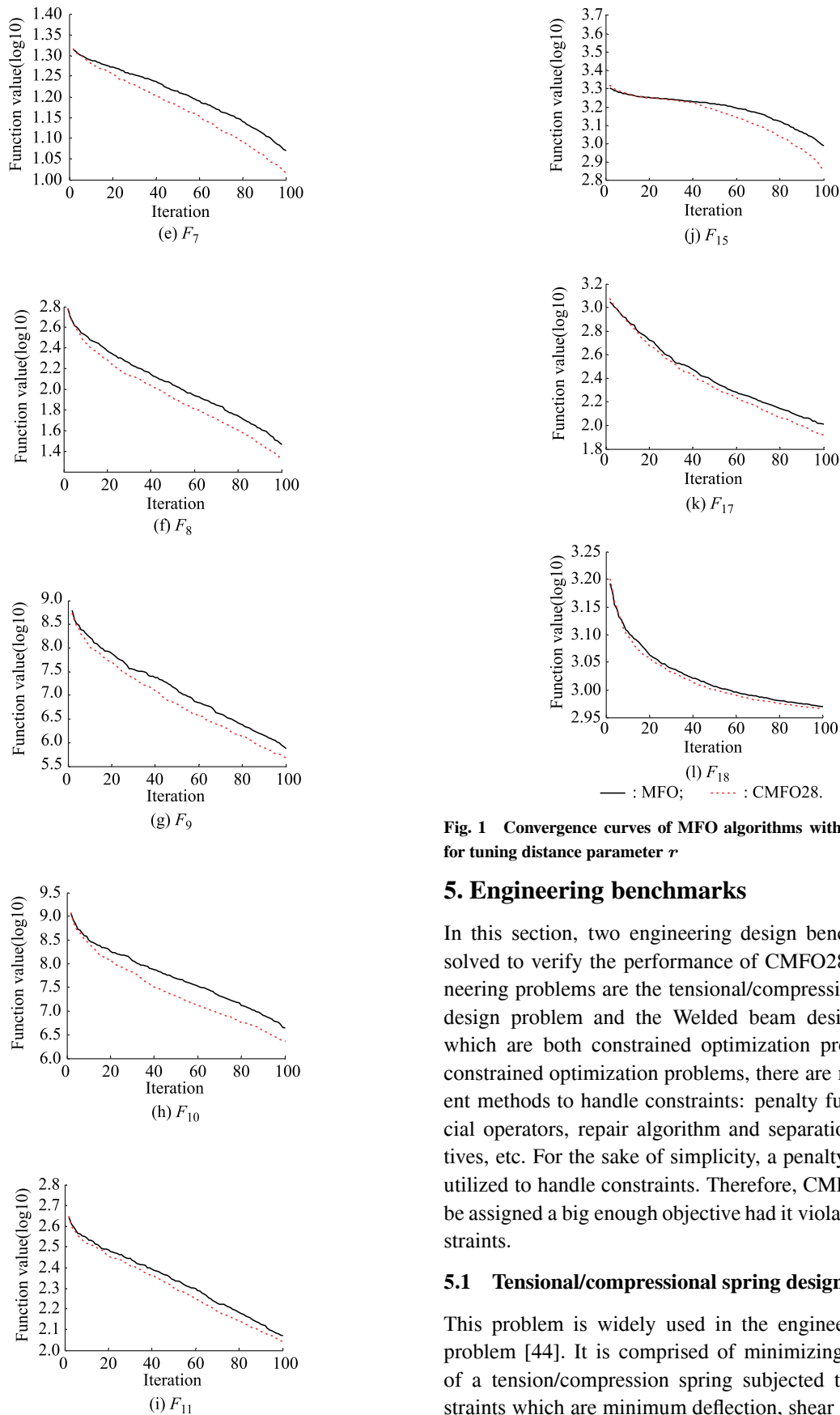

(l) F_{18}

— : MFO; ⋯⋯ : CMFO28.
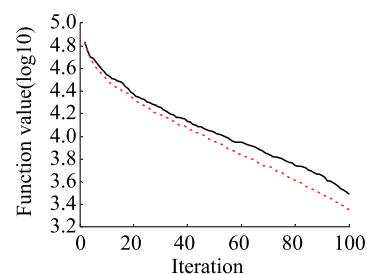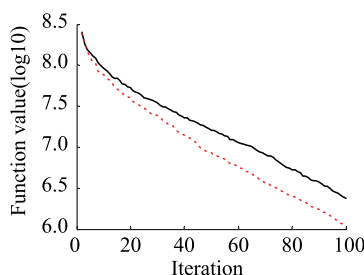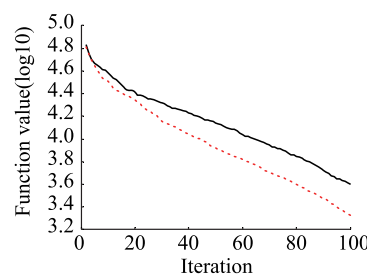
Fig. 1 Convergence curves of MFO algorithms with chaotic maps for tuning distance parameter r

5. Engineering benchmarks

In this section, two engineering design benchmarks are solved to verify the performance of CMFO28. The engineering problems are the tensional/compressional springs design problem and the Welded beam design problem which are both constrained optimization problems. For constrained optimization problems, there are many different methods to handle constraints: penalty function, special operators, repair algorithm and separation of objectives, etc. For the sake of simplicity, a penalty function is utilized to handle constraints. Therefore, CMFO28 would be assigned a big enough objective had it violated any constraints.

5.1 Tensional/compressional spring design problem

This problem is widely used in the engineering design problem [44]. It is comprised of minimizing the weight of a tension/compression spring subjected to four constraints which are minimum deflection, shear stress, surge

frequency, and limits on the outside diameter and on design variables. The design variables are the wire diameter $d(x_1)$, the mean coil diameter $D(x_2)$, and the number of active coils $N(x_3)$ as shown in Fig. 2. The mathematical formulation of this problem is shown below:

$$\min f(\boldsymbol{x}) = (x_3 + 2)x_2 x_1^2 \tag{11}$$

Subject to

$$g_1(\boldsymbol{x}) = 1 - \frac{x_2^3 x_3}{71\,785 x_1^4} \leqslant 0 \tag{12}$$

$$g_2(\boldsymbol{x}) = \frac{4x_2^2 - x_1 x_2}{12\,566(x_2 x_1^3 - x_1^4)} + \frac{1}{5\,108 x_1^2} - 1 \leqslant 0 \tag{13}$$

$$g_3(\boldsymbol{x}) = 1 - \frac{140.45 x_1}{x_2^2 x_3} \leqslant 0 \tag{14}$$

$$g_4(\boldsymbol{x}) = \frac{x_1 + x_2}{1.5} - 1 \leqslant 0 \tag{15}$$

where $0.05 \leqslant x_1 \leqslant 2.0, 0.25 \leqslant x_2 \leqslant 1.3, 2.0 \leqslant x_3 \leqslant 15.0$.
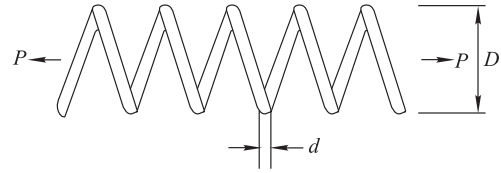
Fig. 2   Tension/compression spring design problem

This problem has been studied by several researches as a benchmark engineering problem. The comparisons of results obtained by different approaches are shown in Table 18. The results show that CMFO28 can find the best feasible solution compared to other approaches. Although, the modified oracle penalty function-based composite differential evolution (MoCoDE) [45] obtained the same results, the maximum number of function evaluation of MoCoDE (24 000) is far more than the maximum number of function evaluation of CMFO28 (30 000).

Table 18   Comparison of results for tensional/compressional spring design problem

Design variable	GA [46]	Coello & Montes [47]	CPSO [44]	CDE [48]	MoCoDE [45]	MFO28
$d(x_1)$	0.051 480	0.051 989	0.051 728	0.051 609	0.051 718	0.051 705
$D(x_2)$	0.351 661	0.363 965	0.357 644	0.354 714	0.357 418	0.357 103
$N(x_3)$	11.632 201	10.890 522	11.244 543	11.410 831	11.248 015	11.266 387
$f(\boldsymbol{x})$	0.012 705	0.012 681	0.0126 747	0.0126 702	0.012 665	0.012 665

5.2   Welded beam design problem

The Welded beam design problem is a practical design problem for constrained design optimization [44]. The objective is to minimize the fabricating cost of the Welded beam subjected to seven constraints. The problem has four design variables: $h(x_1)$, $l(x_2)$, $t(x_3)$ and $b(x_4)$. Fig. 3 shows its parameters. This problem can be formulated as follows:

$$\min f(\boldsymbol{x}) = 1.104\,71 x_1^2 x_2 + 0.048\,11 x_3 x_4 (14.0 + x_2) \tag{16}$$

Subject to

$$g_1(\boldsymbol{x}) = \tau(\boldsymbol{x}) - 13\,000 \leqslant 0 \tag{17}$$

$$g_2(\boldsymbol{x}) = \sigma(\boldsymbol{x}) - 30\,000 \leqslant 0 \tag{18}$$

$$g_3(\boldsymbol{x}) = x_1 - x_4 \leqslant 0 \tag{19}$$

$$g_4(\boldsymbol{x}) = 0.104\,71 x_1^2 +$$
$$0.048\,11 x_3 x_4 \cdot (14.0 + x_2) - 5.0 \leqslant 0 \tag{20}$$

$$g_5(\boldsymbol{x}) = 0.125 - x_1 \leqslant 0 \tag{21}$$

$$g_6(\boldsymbol{x}) = \delta(\boldsymbol{x}) - 0.25 \leqslant 0 \tag{22}$$

$$g_7(\boldsymbol{x}) = 6\,000 - P_c(\boldsymbol{x}) \leqslant 0 \tag{23}$$

where $\tau(\boldsymbol{x}) = \sqrt{(\tau')^2 + \dfrac{2\tau'\tau'' x_2}{2R} + (\tau'')^2}$, $\tau' = \dfrac{6\,000}{\sqrt{2}x_1 x_2}$, $\tau'' = \dfrac{MR}{J}$, $M = 6\,000\left(14 + \dfrac{x_2}{2}\right)$, $R = \dfrac{1}{2}\sqrt{x_2^2 + (x_1 + x_3)^2}$, $J = 2\sqrt{2}x_1 x_2 \left(\dfrac{x_2^2}{12} + \dfrac{(x_1 + x_3)^2}{4}\right)$, $\sigma(\boldsymbol{x}) = \dfrac{504\,000}{x_3^2 x_4}$, $\delta(\boldsymbol{x}) = \dfrac{2.195\,2}{x_3^2 x_4}$, $P_c(\boldsymbol{x}) = 64\,746.022(1 - 0.028\,234\,6 x_3)x_3 x_4^3$, $0.1 \leqslant x_1, x_4 \leqslant 2.0, 0.1 \leqslant x_2, x_3 \leqslant 10$.
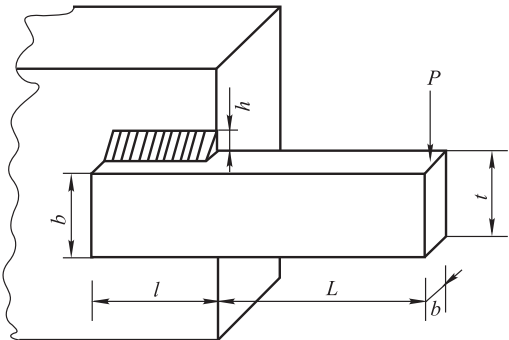

Fig. 3   Welded beam design problem

The best solutions obtained by different approaches are listed in Table 19. From Table 19, the results obtained by the CMFO28 are better than other approaches. Although, the results of MoCoDE are very close to the CMFO28 results, the maximum number of function evaluation of MoCoDE is 240 000 which is eight times the maximum number of function evaluation of CMFO28.

Table 19　Comparison of results for Welded beam design problem

Design variable	GA [46]	Coello & Montes [47]	CPSO [44]	CDE [48]	MoCoDE [45]	MFO28
$h(x_1)$	0.208 800	0.205 986	0.202 369	0.203 137	0.205 730	0.205 730
$l(x_2)$	3.420 500	3.471 328	3.544 214	3.542 998	3.470 489	3.470 489
$t(x_3)$	8.997 500	9.020 224	9.048 210	9.033 498	9.036 624	0.205 730
$b(x_4)$	0.210 000	0.206 480	0.205 723	0.206 179	0.205 730	0.205 7302
$f(\boldsymbol{x})$	1.748 309	1.728 226	1.728 024	1.733 462	1.724 852	1.724 852

6. Conclusions

This paper investigates the effectiveness of chaotic maps in improving the performance of the MFO algorithm. Ten different chaotic maps have been embedded into three components of initialization population, boundary handing and the distance parameter. To compare the variants of CMFO, 18 benchmark functions divided into unimodal, multimodal and composite problems are employed. The results reveal that chaotic maps used in initialization population and boundary handing components cannot improve the performance of MFO. It shows that exploration can increase the diversity of population but cannot directly improve the performance of MFO. On the contrary, the results of the comparative study suggest that Singer maps embedded into the distance parameter can significantly improve the performance of MFO which reconfirms that the balance of exploration and exploitation is vital to the performance of metaheuristic algorithms. The proposed CMFO(CMFO28) is evaluated on two complex engineering problems and the results clearly show that the CMFO can solve the real-world problems efficiently.

For future studies, it would be interesting to focus on the extension of the CMFO to handle mixed-type problems and discrete optimization problems. And, other chaotic maps applied to the MFO algorithm would be an important direction. In addition, the MFO algorithm combing with other optimization algorithms will also be potentially fruitful.

References

[1] GANDOMI A H, YANG X S, ALAVI A H. Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Engineering with Computers, 2013, 29(1): 17 – 35.

[2] MAFARJA M M, MIRJALILI S. Hybrid whale optimization algorithm with simulated annealing for feature selection. Neurocomputing, 2017, 260: 302 – 312.

[3] BHANDARI A K, SINGH V K, KUMAR A, et al. Cuckoo search algorithm and wind driven optimization based study of satellite image segmentation for multilevel thresholding using Kapur's entropy. Expert Systems with Applications, 2014, 41(7): 3538 – 3560.

[4] WANG H, WANG W J, SUN H, et al. A new cuckoo search algorithm with hybrid strategies for flow shop scheduling problems. Soft Computing, 2017, 21(15): 4297 – 4307.

[5] ZHANG X M. Parameter estimation of shallow wave equation via cuckoo search. Neural Computing & Applications, 2017, 28(12): 4047 – 4059.

[6] SAAD A, KHAN S A, MAHMOOD A. A multi-objective evolutionary artificial bee colony algorithm for optimizing network topology design. Swarm and Evolutionary Computation, 2018, 38: 187 – 201.

[7] HOLLAND J H. Adaptation in natural and artificial systems. Massachusetts: MIT Press, 1992.

[8] EBERHART R, KENNEDY J. A new optimizer using particle swarm theory. Proc. of the 6th International Symposium on Micro Machine and Human Science, 1995: 39 – 43.

[9] DORIGO M, MANIEZZO V, COLORNI A. Ant system: optimization by a colony of cooperating agents. IEEE Trans. on Systems Man & Cybernetics – Part B, 1996, 26(1): 29 – 41.

[10] STORN R, PRICE K. Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 1997, 11(4): 341 – 359.

[11] KARABOGA D, BASTURK B. An artificial bee colony (ABC) algorithm for numeric function optimization. Proc. of the IEEE Swarm Intelligence Symposium, 2006: 181 – 184.

[12] YANG X S, DEB S. Cuckoo search via levy flights. Mathematics, 2010: 210 – 214.

[13] ZONG W G, KIM J H, LOGANATHAN G V. A new heuristic optimization algorithm: harmony search. Simulation Transactions of the Society for Modeling & Simulation International, 2001, 76(2): 60 – 68.

[14] YANG X S. A new metaheuristic bat-inspired algorithm. Computer Knowledge & Technology, 2010, 284(12): 65 – 74.

[15] MENG X, LIU Y, GAO X, et al. A new bio-inspired algorithm: chicken swarm optimization. Proc. of the International Conference in Swarm Intelligence, 2014: 86 – 94.

[16] MIRJALILI S, MIRJALILI S M, LEWIS A. Grey wolf optimizer. Advances in Engineering Software, 2014, 69(3): 46 – 61.

[17] YANG X S. Nature-inspired metaheuristic algorithms. Frome: Luniver Press, 2008.

[18] MIRJALILI S, LEWIS A. The whale optimization algorithm. Advances in Engineering Software, 2016, 95: 51 – 67.

[19] MIRJALILI S. The ant lion optimizer. Advances in Engineering Software, 2015, 83(C): 80 – 98.

[20] MIRJALILI S. Moth-flame optimization algorithm: a novel

nature-inspired heuristic paradigm. Knowledge-Based Systems, 2015, 89: 228 – 249.

[21] JANGIR N, TRIVEDI I N, PANDYA M H, et al. Moth-flame optimization algorithm for solving real challenging constrained engineering optimization problems. Proc. of the IEEE Students' Conference on Electrical, Electronics and Computer Science, 2016: 1 – 5.

[22] SAVSANI V, TAWHID M A. Non-dominated sorting moth flame optimization (NS-MFO) for multi-objective problems. Engineering Applications of Artificial Intelligence, 2017, 63: 20 – 32.

[23] MUANGKOTE N, SUNAT K, CHIEWCHANWATTANA S, et al. Multilevel thresholding for satellite image segmentation with moth-flame based optimization. Proc. of the 13th International Joint Conference on Computer Science and Software Engineering, 2016: 460 – 465.

[24] ABD EL AZIZ M, EWEESC A A, HASSANIEN A E. Whale optimization algorithm and moth-flame optimization for multilevel thresholding image segmentation. Expert Systems with Applications, 2017, 83: 242 – 256.

[25] WANG M J, CHEN H L, YANG B, et al. Toward an optimal kernel extreme learning machine using a chaotic moth-flame optimization strategy with applications in medical diagnoses. Neurocomputing, 2017, 267: 69 – 84.

[26] HASSANIEN A E, GABER T, MOKHTAR U, et al. An improved moth flame optimization algorithm based on rough sets for tomato diseases detection. Computers and Electronics in Agriculture, 2017, 136: 86 – 96.

[27] BUCH H, TRIVEDI I N, JANGIR P. Moth flame optimization to solve optimal power flow with non-parametric statistical evaluation validation. Cogent Engineering, 2017, 4(1): 1286731.

[28] SINGH P, PRAKASH S. Optical network unit placement in fiber-wireless (FiWi) access network by moth-flame optimization algorithm. Optical Fiber Technology, 2017, 36: 403 – 411.

[29] ALLAM D, YOUSRI D A, ETEIBA M B. Parameters extraction of the three diode model for the multi-crystalline solar cell/module using moth-flame optimization algorithm. Energy Conversion and Management, 2016, 123: 535 – 548.

[30] YAMANY W, FAWZY M, THARWAT A, et al. Moth-flame optimization for training multi-layer perceptrons. Proc. of the 11th International Computer Engineering Conference, 2015: 267 – 272.

[31] VIKAS, NANDA S J. Multi-objective moth flame optimization. Proc. of the International Conference on Advances in Computing, Communications and Informatics, 2016: 2470 – 2476.

[32] DUBEY H M, PANDIT M, PANIGRAHI B K. An overview and comparative analysis of recent bio-inspired optimization techniques for wind integrated multi-objective power dispatch. Swarm and Evolutionary Computation, 2018, 38: 12 – 34.

[33] YANG L J, CHEN T L. Application of chaos in genetic algorithms. Communications in Theoretical Physics, 2002, 38(2): 168 – 172.

[34] GUO Z, CHENG B, YE M, et al. Self-adaptive chaos differential evolution. Proc. of the International Conference on Natural Computation, 2006: 972 – 975.

[35] GANDOMI A H, YUN G J, YANG X S, et al. Chaos-enhanced accelerated particle swarm optimization. Communications in Nonlinear Science and Numerical Simulation, 2013, 18(2): 327 – 340.

[36] GANDOMI A H, YANG X S, TALATAHARI S, et al. Firefly algorithm with chaos. Communications in Nonlinear Science and Numerical Simulation, 2013, 18(1): 89 – 98.

[37] SAREMI S, MIRJALILI S, LEWIS A. Biogeography-based optimisation with chaos. Neural Computing & Applications, 2014, 25(5): 1077 – 1097.

[38] HUANG L, DING S, YU S H, et al. Chaos-enhanced cuckoo search optimization algorithms for global optimization. Applied Mathematical Modelling, 2016, 40(5/6): 3860 – 3875.

[39] WANG G G, DEB S, GANDOMI A H, et al. Chaotic cuckoo search. Soft Computing, 2016, 20(9): 3349 – 3362.

[40] WU W M, LI Z X, LIN Z Y, et al. Moth-flame optimization algorithm based on chaotic crisscross operator. Computer Engineering & Applications, 2018, 54(3): 136 – 141. (in Chinese)

[41] GASTON K J, BENNIE J, DAVIES T W, et al. The ecological impacts of nighttime light pollution: a mechanistic appraisal. Biological Reviews, 2013, 88(4): 912 – 927.

[42] PADHYE N, DEB K, MITTAL P. Boundary handling approaches in particle swarm optimization. Advances in Intelligent Systems & Computing, 2013, 201: 287 – 298.

[43] LIANG J J, SUGANTHAN P N, DEB K. Novel composition test functions for numerical global optimization. Proc. of the Swarm Intelligence Symposium, 2005: 68 – 75.

[44] HE Q, WANG L. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Engineering Applications of Artificial Intelligence, 2007, 20(1): 89 – 99.

[45] SECTION I. Composite differential evolution with modified oracle penalty method for constrained optimization problems. Mathematical Problems in Engineering, 2014, 1: 1 – 15.

[46] COELLO C A C. Use of a self-adaptive penalty approach for engineering optimization problems. Computers in Industry, 2000, 41(2): 113 – 127.

[47] COELLO C A C, MONTES E M. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. Advanced Engineering Informatics, 2002, 16(3): 193 – 203.

[48] HUANG F Z, WANG L, HE Q. An effective co-evolutionary differential evolution for constrained optimization. Applied Mathematics and Computation, 2007, 186(1): 340 – 356.

Biographies

LI Hongwei was born in 1978. He received his M.S. degree from University of Science and Technology of the PLA in 2002. He is an associate professor in College of Field Engineering, Army Engineering University of the PLA. His current research interests are military operations research and intelligent unmanned technology.
E-mail: 727802081@qq.com

LIU Jianyong was born in 1961. He received his Ph.D. degree from University of Science and Technology of the PLA in 2004. He is a professor in College of Field Engineering, Army Engineering University of the PLA. His current research interests are military operations research and intelligent unmanned technology.
E-mail: jianyong1212@126.com

CHEN Liang was born in 1981. He received his B.S. degree and M.S. degree from University of Science and Technology of the PLA in 2004 and 2009, respectively. He is a Ph.D. candidate at College of Field Engineering, Army Engineering University of the PLA. He is a lecturer in Army Military Transportation University. His main research interests include operations research, swarm intelligence, multi-objective optimization and intelligent unmanned technology.
E-mail: chenbb0708@163.com

BAI Jingbo was born in 1982. He received his B.S. degree and M.S. degree from University of Science and Technology of the PLA in 2005 and 2009, respectively. He is a lecturer and a Ph.D. candidate in College of Field Engineering, Army Engineering University of the PLA. His current research interest includes mission planning of military problems.
E-mail: baijingbo1982@163.com

SUN Yangyang was born in 1983. He received his M.S. degree from University of Science and Technology of the PLA in 2009. He is a lecturer and a Ph.D. candidate in College of National Defense Engineering, Army Engineering University of the PLA. His current research interests include optical fiber sensing and system integration.
E-mail: bryant8011@163.com
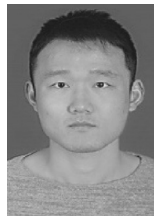
LU Kai was born in 1991. He received his B.S. degree and M.S. degree from Ordnance Engineering College in 2014 and 2017, respectively. He is a Ph.D. candidate in College of Field Engineering, Army Engineering University of the PLA. His research interest include bridge design theory and key technologies.
E-mail: xikaikaixi@outlook.com