Contents lists available at ScienceDirect

Information Sciences

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

Chaotic Krill Herd algorithm

Gai-Ge Wang [a], Lihong Guo [b,*], Amir H. Gandomi [c,1], Guo-Sheng Hao [a], Heqi Wang [b]

[a] School of Computer Science and Technology, Jiangsu Normal University, Xuzhou, Jiangsu 221116, China
[b] Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun 130033, China
[c] Department of Civil Engineering, The University of Akron, Akron, OH 44325, USA

A R T I C L E I N F O

Article history:
Received 20 August 2012
Received in revised form 26 September 2013
Accepted 10 February 2014
Available online 3 March 2014

Keywords:
Global optimization problem
Krill Herd
Chaotic maps
Multimodal function

A B S T R A C T

Recently, Gandomi and Alavi proposed a meta-heuristic optimization algorithm, called Krill Herd (KH). This paper introduces the chaos theory into the KH optimization process with the aim of accelerating its global convergence speed. Various chaotic maps are considered in the proposed chaotic KH (CKH) method to adjust the three main movements of the krill in the optimization process. Several test problems are utilized to evaluate the performance of CKH. The results show that the performance of CKH, with an appropriate chaotic map, is better than or comparable with the KH and other robust optimization approaches.

1. Introduction

The process of optimization is essentially the choice of a vector within a search space. The selected vector can maximize/minimize an objective function to provide the best solution. Generally, modern intelligent approaches are used to deal with these types of optimization problems. Such optimization approaches can be categorized into two groups in view of their natures: (1) deterministic, and (2) random intelligent approaches. The deterministic approaches using gradient have a strict step. They produce the identical solution if its initial starting values are the same with each other when solving the same problem. Contrary to the deterministic approaches, gradient-free stochastic algorithms are based on random walks. Therefore, the optimization process cannot be repeated under any conditions. However, in most cases, both of them are capable of finding the same final optimal solutions [44]. Recently, nature-inspired meta-heuristic algorithms show a powerful and efficient performance for dealing with high-dimension nonlinear optimization problems [12,53].

To some extent, all meta-heuristic approaches manage to make a trade-off between intensification (local search) and randomization (global search) [50]. These robust nature-inspired meta-heuristic approaches are utilized to tackle NP-hard problems such as task-resource assignment. They can fully exploit the useful information of the whole population to find optimal solutions. Up to now, significant research has been done on evolution theory. The process of evolution is idealized as a kind of gradient-free method, called genetic algorithms (GAs) [6,18,55]. Since then various nature-inspired meta-heuristic approaches have been proposed such as evolutionary strategy (ES) [1,2], particle swarm optimization (PSO) [27,28,40,56], ant colony optimization (ACO) [4], differential evolution (DE) [10,19,38], firefly algorithm (FA) [7], biogeography-based optimization (BBO) [29,37], cuckoo search (CS) [9,49], probability-based incremental learning (PBIL) [36], big bang–big

* Corresponding author. Tel.: +1 3689827126.
E-mail addresses: gaigewang@163.com, gaigewang@gmail.com (G.-G. Wang), guolh@ciomp.ac.cn (L. Guo), a.h.gandomi@gmail.com, ag72@uakron.edu (A.H. Gandomi), guoshenghaoxz@tom.com (G.-S. Hao), whq200808@gmail.com (H. Wang).
[1] URL: http://www.gozips.uakron.edu/~ag72.

http://dx.doi.org/10.1016/j.ins.2014.02.123
0020-0255/© 2014 Elsevier Inc. All rights reserved.

crunch algorithm [5,23,25,26], harmony search (HS) [15,17,47], charged system search (CSS) [24], animal migration optimization (AMO) [31], teaching–learning-based optimization (TLBO) [3,35], bat algorithm (BA) [11,52], and the KH method [8]. In fact, KH is a new kind of swarm intelligence optimization approach inspired by the herding behavior of krill [8]. In KH, the objective function for the krill movement is decided by the distances of each krill from density of the krill swarm and food. The position for each krill is made up of three parts: (i) movement induced by other individuals (ii) foraging motion, and (iii) physical diffusion. The crucial advantage of KH algorithm is its simplicity, making KH implement easily and lending itself to parallel computation [44].

In general, KH lends itself strongly to exploitation. However, it cannot always implement global search well. Thus, in some cases, KH fails to find global optimal solution. The search strategy used in basic KH is mainly based on random walks. Thus, it cannot always deal with the problem successfully [45]. Different strategies have been added to the basic KH method [43,45] with the aim of improving its performance [45].

With the development of the nonlinear dynamics, chaos concept has been widely considered in various applications [34]. In this context, one of the most famous applications is the introduction of chaos theory into the optimization methods [48]. Up to now, the chaos theory has been successfully combined with several meta-heuristic optimization methods [13]. Some major efforts in this area includes hybridizing chaotic sequences with memetic differential evolution algorithm [22], FA [13], gravitational search algorithm [20], imperialist competitive algorithm [41], charged system search [33,42], PSO [14,46], and GAs [21].

In the present study, chaotic KH-based (CKH) methods are introduced for the purpose of accelerating the convergence of KH. Various one-dimensional chaotic maps are employed in place of the parameters used in KH. The performance of the proposed approach is tested on fourteen benchmark problems. Experimental results indicate that CKH performance is superior to KH, ACO, BA, CS, DE, ES, GA, PBIL, and PSO. This is mainly because deterministic chaotic signals are used to replace linearly declined values.

The organization of this paper is as follows. Firstly, a brief overview of the basic KH algorithm and 12 chaotic maps are given in Section 2. The detailed presentation of the proposed CKH approach is provided in Section 3. Subsequently, the tuning of the inertia weights and selecting the optimal CKH are described in Section 4. In addition, the performance of the CKH approach is verified using fourteen benchmarks. Finally, a summary of the present work is represented in Section 5.

2. Overview of the KH and chaotic maps

2.1. KH method

KH [8] is a new type of meta-heuristic method for solving optimization problems. This method is inspired by the herding of krill swarms when searching for food in nature. For each krill, its position in search space is influenced by three components described below [45]:

i. movement induced by other krill;
ii. foraging action;
iii. random diffusion.

For simplicity, the above three motions in KH can be idealized to the following Lagrangian model [8] as shown in Eq. (1).

$$\frac{dX_i}{dt} = N_i + F_i + D_i \tag{1}$$

where N_i, F_i, and D_i are, respectively, corresponding to the above three motions for the ith krill [8]. The krill number is represented by i, and t is considered as generation.

2.1.1. Motion induced by other krill

The direction of the first motion, α_i, is approximately calculated according to the following three factors: target effect, local effect, and repulsive effect. For krill i, this movement can be modeled below:

$$N_i^{new} = N^{\max}\alpha_i + \omega_n N_i^{old} \tag{2}$$

where

$$\alpha_i = \alpha_i^{local} + \alpha_i^{target} \tag{3}$$

and $N^{\max}$ is the maximum speed, ω_n is the inertia weight in [0,1], N_i^{old} is the previous motion, α_i^{local} and α_i^{target} are the local effect and the target effect, respectively [45]. According to the literature [8], we set $N^{\max}$ to 0.01 $(\mathrm{m\,s^{-1}})$ in our study.

2.1.2. Foraging motion

The second motion is determined by the two main factors: the food location and the previous experience with respect to the food position. For the ith krill, the expression of this motion can be provided below [45]:

$$F_i = V_f \beta_i + \omega_f F_i^{old} \tag{4}$$

where

$$\beta_i = \beta_i^{food} + \beta_i^{best} \tag{5}$$

and V_f is the foraging speed, ω_f is the inertia weight between 0 and 1, F_i^{old} is the previous foraging motion, β_i^{food} is the food attraction and β_i^{best} is the effect of the best fitness. In our study, we set V_f to 0.02 [8].

2.1.3. Physical diffusion

The third motion is essentially a random process. The model of this motion can be expressed according to two factors: a maximum diffusion speed and a random vector. This model can be expressed below [8]:

$$D_i = D^{\max}\delta \tag{6}$$

where $D^{\max}$ is the diffusion speed, and δ is the random vector in $[-1,1]$ [45].

2.1.4. Main procedure of the KH method

The second and third motion involve two global and two local schemes. These schemes can work simultaneously which makes KH a robust and efficient method [45]. The position of krill i from t to $t+\Delta t$ can be formulated by Eq. (7) [45].

$$X_i(t+\Delta t) = X_i(t) + \Delta t \frac{dX_i}{dt} \tag{7}$$

Note that Δt is a key constant and should be fine-tuned in terms of the specific problem. The reason is that Δt can be regarded as a scale factor of the speed vector. Moreover, in KH, the inertia weights (ω_n, ω_f) are set to 0.9 at the start of the KH to highlight exploration. They are later linearly decreased to 0.1 in order to stimulate exploitation [8].

Furthermore, in order to enhance the performance of the KH, genetic reproduction mechanisms have been introduced into the KH algorithm [8].

In general, the KH method can be described as shown in Fig. 1, and its responding flowchart is illustrated in Fig. 2. Further information about the above movements and KH method can be found in [8].

2.2. Chaotic maps

There is a special kind of random-based optimization algorithms, namely chaotic optimization algorithm (COA) [13]. COA employs chaotic variables rather than random variables [13]. The chaos has the property of the non-repetition and ergodicity. Therefore, it can perform downright searches at higher speeds compared to the stochastic searches that mainly rely on probabilities [13]. In the present study, 1-D, non-invertible maps are used to produce chaotic sets. Twelve distinguished 1-D

```
Krill herd algorithm
Begin
    Step 1: Initialization. Initialize the generation counter G, the population P of NP krill
            randomly, $V_f$, $D^{max}$, and $N^{max}$.
    Step 2: Fitness calculation . Calculate fitness for each krill according to its initial position.
    Step 3: While G < MaxGeneration do
            Sort the population according to their fitness.
            for i=1:NP (all krill) do
                Perform the following motion calculation.
                    Motion induced by other individuals
                    Foraging motion
                    Physical diffusion
                Implement the genetic operators.
                Update the krill position in the search space.
                Calculate fitness for each krill according to its new position.
            end for i
            G = G+1.
    Step 4: end while
End.
```

Fig. 1. Krill Herd algorithm.

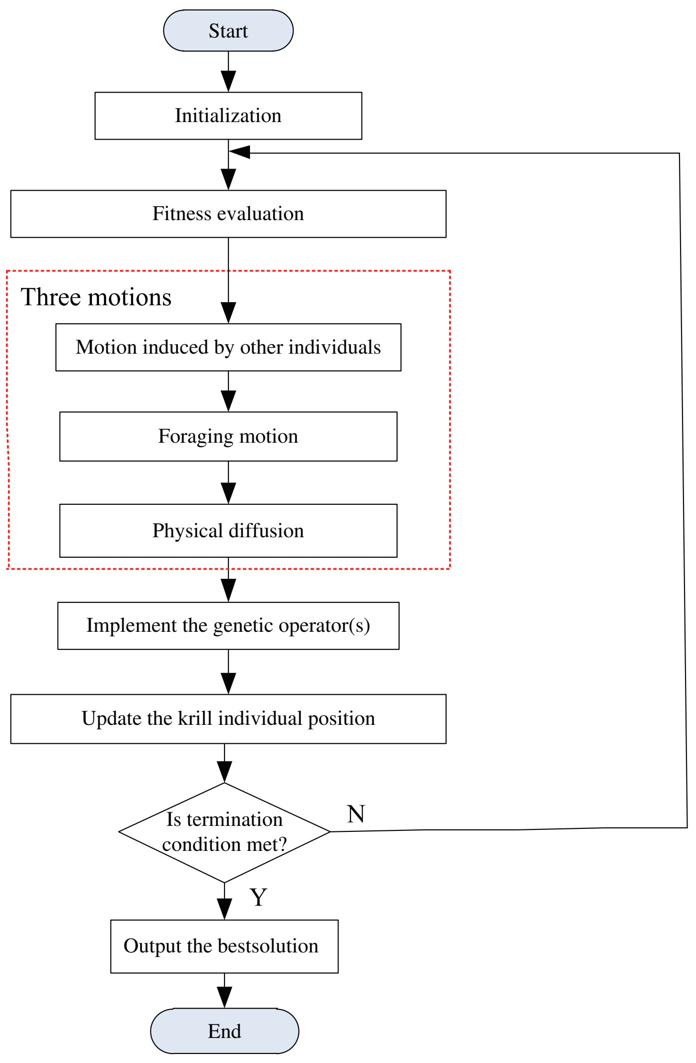
Start
Initialization
Fitness evaluation
Three motions
Motion induced by other individuals
Foraging motion
Physical diffusion
Implement the genetic operator(s)
Update the krill individual position
Is termination condition met?
N
Y
Output the bestsolution
End

Fig. 2. Flowchart of KH algorithm.

maps [13] are described in Table 1. In this table, k means the index of the chaotic sequence, and x_k represents the kth number in the chaotic sequence.

3. The proposed CKH approach

As it is presented in Eqs. (2) and (4), the main parameters of KH are the inertia weights (ω_n, ω_f) that represent the variations of the global optimal attraction. The values of the weights are vitally important in deciding the convergence speed and how to update the position of the krill in KH.

Though the basic KH is very powerful, the solutions are still deviating from each other at the end of the search. In KH, the inertia weights (ω_n, ω_f) are set to 0.9 at the early search stage to highlight exploration. Finally, they are linearly decreased to 0.1 to stimulate exploitation [8].

In the standard KH, there is no need to keep linearly decrements. In fact, a chaotic varying inertia weights (ω_n, ω_f) may be more favorable for the search, which may also make the method converge to the best solution with a fast speed as will be demonstrated in the following section. After normalization, the range of a chaotic map is always between 0 and 1. So, in this paper, we use chaotic maps to adjust inertia weights (ω_n, ω_f), and such chaos-improved KH can be named as the chaotic KH.

The basic framework of the CKH method and its responding flowchart are shown in Figs. 3 and 4, respectively. In Figs. 3 and 4, $1 \leqslant i \leqslant NP$ represents the krill number, and NP is the total number of the krill in the population, *i.e.* population size.

The tuning of inertia weights (ω_n, ω_f) using chaotic maps also improve the ability of KH to avoid local optima when solving a multimodal function. Comparing with other optimization methods, this could be a merit for this method.

Table 1
Twelve different chaotic maps.

No.	Name	Definition
M1	Chebyshev map	$x_{k+1} = \cos(k \cos^{-1}(x_k))$
M2	Circle map[a]	$x_{k+1} = x_k + b - (a/2\pi)\sin(2\pi k) \bmod (1)$
M3	Gaussian map	$x_{k+1} = \begin{cases} 0 & x_k = 0 \\ 1/x_k \bmod (1) & \text{otherwise} \end{cases}, \ 1/x_k \bmod (1) = \frac{1}{x_k} - \left[\frac{1}{x_k}\right]$
M4	Intermittency map	$x_{k+1} = \begin{cases} \varepsilon + x_k + c x_k^n & 0 < x_k \leqslant P \\ \frac{x_k - P}{1-P} & P < x_k < 1 \end{cases}$
M5	Iterative map	$x_{k+1} = \sin\left(\frac{a\pi}{x_k}\right), \ a \in (0,1)$
M6	Liebovitch map	$x_{k+1} = \begin{cases} \alpha x_k & 0 < x_k \leqslant P \\ \frac{P - x_k}{P_2 - P_1} & P_1 < x_k \leqslant P_2 \\ 1 - \beta(1 - x_k) & P_2 < x_k \leqslant 1 \end{cases}$
M7	Logistic map	$x_{k+1} = a x_k (1 - x_k)$
M8	Piecewise map	$x_{k+1} = \begin{cases} \frac{x_k}{P} & 0 \leqslant x_k < P \\ \frac{x_k - P}{0.5 - P} & P \leqslant x_k < \frac{1}{2} \\ \frac{1 - P - x_k}{0.5 - P} & \frac{1}{2} \leqslant x_k < 1 - P \\ \frac{1 - x_k}{P} & 1 - P \leqslant x_k < 1 \end{cases}$
M9	Sine map	$x_{k+1} = \frac{a}{4}\sin(\pi x_k), \ 0 < a \leqslant 4$
M10	Singer map	$x_{k+1} = \mu(7.86 x_k - 23.31 x_k^2 + 28.75 x_k^3 - 13.302875 x_k^4)$
M11	Sinusoidal map	$x_{k+1} = a x_k^2 \sin(\pi x_k)$
M12	Tent map	$x_{k+1} = \begin{cases} \frac{x_k}{0.7} & x_k < 0.7 \\ \frac{10}{3}(1 - x_k) & x_k \geqslant 0.7 \end{cases}$

[a] With $a = 0.5$ and $b = 0.2$, it geneguos rates chaotic sequence in (0,1).

CKH algorithm

```
Begin
    Step 1: Initialization. Initialize the generation counter G, the population P of NP krill
            randomly, $V_f$, $D^{max}$, and $N^{max}$ and KEEP; set initial value of the chaotic map $c_0$
            randomly, and inertia weights $\omega = (\omega_n, \omega_f) = c_0$.
    Step 2: Fitness calculation. Calculate fitness for each krill according to its initial position.
    Step 3: While G < MaxGeneration do
              Sort the population according to their fitness.
              Memorize the KEEP best krill.
              Update inertia weights using chaotic maps ($\omega = c_{t+1}$).
              for i=1: NP(all krill) do
                 Perform the following motion calculation.
                     Motion induced by the presence of other individuals
                     Foraging motion
                     Physical diffusion
                 Update the krill position in the search space.
                 Calculate fitness for each krill according to its new position.
              end for i
              Replace the KEEP worst krill with the KEEP best krill.
              G = G+1.
    Step 4: end while
End.
```

Fig. 3. CKH algorithm.

Moreover, the second significant improvement is the addition of elitism scheme to the CKH. As with other meta-heuristic methods, a kind of elitism is typically introduced in order to hold the optimal solutions. This puts a stop to the best solutions from being corrupted by motion calculation operators. Here, an elitism strategy is utilized to memorize the property of the best krill in CKH. Consequently, even if motion calculation operation ruins its corresponding krill, it has been held and can be recovered if required.

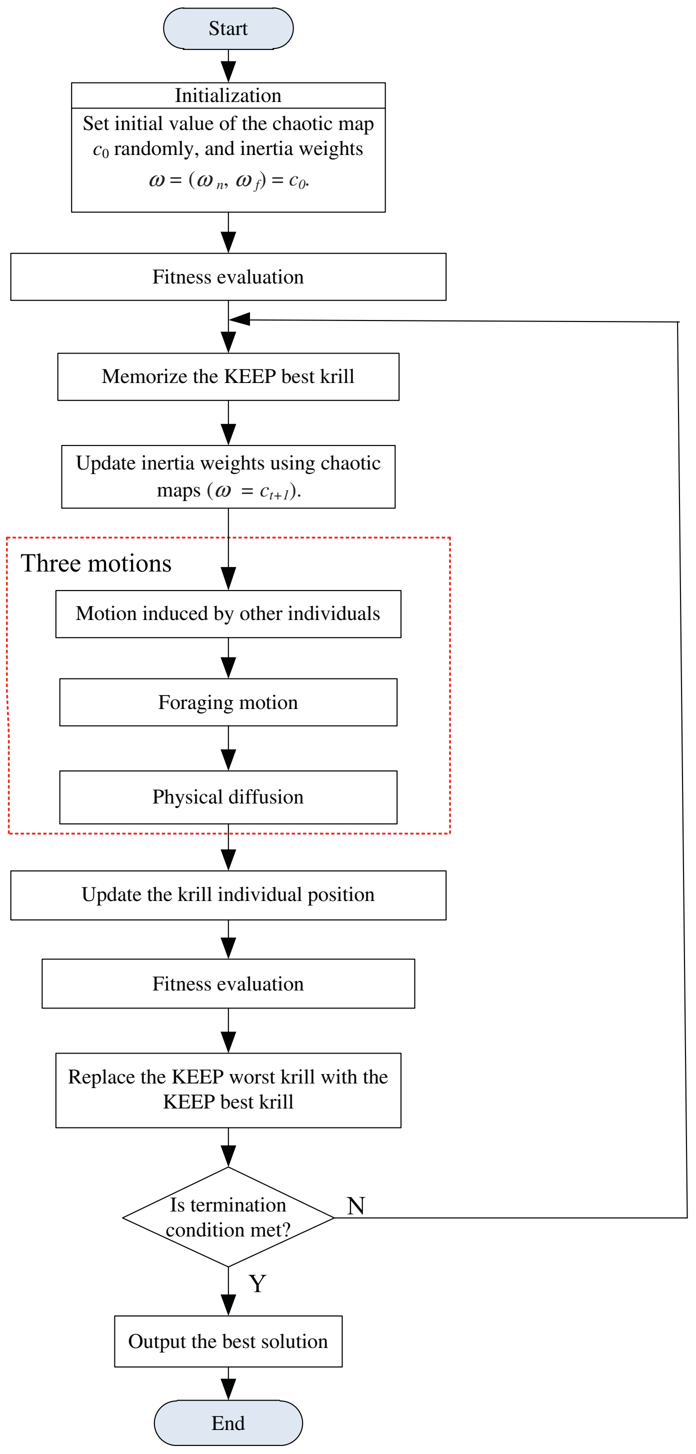
Start
Initialization
Set initial value of the chaotic map c_0 randomly, and inertia weights $\omega = (\omega_n, \omega_f) = c_0$.
Fitness evaluation
Memorize the KEEP best krill
Update inertia weights using chaotic maps ($\omega = c_{t+1}$).
Three motions
Motion induced by other individuals
Foraging motion
Physical diffusion
Update the krill individual position
Fitness evaluation
Replace the KEEP worst krill with the KEEP best krill
Is termination condition met?
N
Y
Output the best solution
End

Fig. 4. Flowchart of CKH algorithm.

4. Experimental results

In this section, various experiments on optimization benchmark problems are implemented to verify the performance of the proposed meta-heuristic CKH method.

In order to get an unbiased comparison of CPU times, all the experiments are performed using the same PC with the detailed settings as shown in Table 2.

Table 2
The detailed settings.

Name	Detailed settings
Hardware	
CPU	Pentium IV processor
Frequency	2.0 GHz
RAM	512 MB
Hard drive	160 GB
Software	
Operating system	Windows XP3
Language	MATLAB R2012a (7.14)

Table 3
Benchmark functions.

No.	Name	Definition
F01	Ackley	$f(\vec{x}) = 20 + e - 20 \cdot e^{-0.2 \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)}$
F02	Fletcher–Powell	$f(\vec{x}) = \sum_{i=1}^{n}(A_i - B_i)^2, \quad A_i = \sum_{j=1}^{n}(a_{ij}\sin\alpha_j + b_{ij}\cos\alpha_j) \quad B_i = \sum_{j=1}^{n}(a_{ij}\sin x_j + b_{ij}\cos x_j)$
F03	Griewank	$f(\vec{x}) = \sum_{i=1}^{n}\frac{x_i^2}{4000} - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
F04	Penalty #1	$f(\vec{x}) = \frac{\pi}{30}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2 \cdot [1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4), \ y_i = 1 + 0.25(x_i + 1)$
F05	Penalty #2	$f(\vec{x}) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2 \cdot [1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$
F06	Quartic *with noise*	$f(\vec{x}) = \sum_{i=1}^{n}(i \cdot x_i^4 + U(0,1))$
F07	Rastrigin	$f(\vec{x}) = 10 \cdot n + \sum_{i=1}^{n}(x_i^2 - 10 \cdot \cos(2\pi x_i))$
F08	Rosenbrock	$f(\vec{x}) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$
F09	Schwefel 2.26	$f(\vec{x}) = 418.9829 \times D - \sum_{i=1}^{D} x_i \sin(\lvert x_i\rvert^{1/2})$
F10	Schwefel 1.2	$f(\vec{x}) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$
F11	Schwefel 2.22	$f(\vec{x}) = \sum_{i=1}^{n}\lvert x_i\rvert + \prod_{i=1}^{n}\lvert x_i\rvert$
F12	Schwefel 2.21	$f(\vec{x}) = \max_i\{\lvert x_i\rvert, \ 1 \leqslant i \leqslant n\}$
F13	Sphere	$f(\vec{x}) = \sum_{i=1}^{n} x_i^2$
F14	Step	$f(\vec{x}) = 6 \cdot n + \sum_{i=1}^{n}\lfloor x_i\rfloor$

* In benchmark function F02, the matrix elements $\mathbf{a}_{n\times n}, \mathbf{b}_{n\times n} \in (-100, 100)$, $\boldsymbol{\alpha}_{n\times 1} \in (-\pi, \pi)$ are draw from uniform distribution.
* In benchmark functions F04 and F05, the definition of the function $u(x_i, a, k, m)$ is as follows:

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \leqslant x_i \leqslant a \\ k(-x_i - a)^m, & x_i < -a \end{cases}.$$

Fourteen different benchmark functions are used to evaluate our proposed meta-heuristic CKH method. The expressions and properties of these benchmarks can be presented in Tables 3 and 4, respectively [44]. More information with reference to all the benchmarks can be found in [54].

For all the algorithms used in this work, population size *NP*, an elitism parameter *Keep*, and maximum generation *Maxgen* are assigned to 50, 2 and 50, respectively. The optimal results obtained by each algorithm for each benchmark are explained in the following sections. It should be pointed out that different scales are used to normalize the results in the tables. Therefore, values cannot be comparative between them. The dimension of each function used in our work is 20 (*i.e.*, $d = 20$).

4.1. The performance of CKH with different chaotic maps

CKH is capable of significantly improving the solution quality by using the chaotic maps. Here, the work of adjusting the inertia weights $\omega = (\omega_n, \omega_f)$ are implemented. And the value of ω is substituted with various chaotic maps presented in Section 2.2. We normalized all the chaotic maps in [0.1,0.9] so as to achieve this.

100 implementations of the CKH algorithm are carried out on fourteen benchmarks to obtain typical statistical results. The results of the simulations are recorded in Tables 5 and 6. Table 5 represents the average values obtained by the CKH method, averaged over 100 simulations. Table 6 also illustrates the best values obtained by CKH method from 100 simulations. We must point out, the marks M1, M2,...,M12 in the Tables 5 and 6 are short for the twelve responding chaotic maps in the Section 2.2.

Table 4
Properties of benchmark functions, *lb* denotes lower bound, *ub* denotes upper bound, *opt* denotes optimum point.

No.	Function	*lb*	*ub*	*opt*	Continuity	Modality
F01	Ackley	−32.768	32.768	0	Continuous	Multimodal
F02	Fletcher–Powell	−π	π	0	Continuous	Multimodal
F03	Griewangk	−600	600	0	Continuous	Multimodal
F04	Penalty #1	−50	50	0	Continuous	Multimodal
F05	Penalty #2	−50	50	0	Continuous	Multimodal
F06	Quartic *with noise*	−1.28	1.28	1	Continuous	Multimodal
F07	Rastrigin	−5.12	5.12	0	Continuous	Multimodal
F08	Rosenbrock	−2.048	2.048	0	Continuous	Unimodal
F09	Schwefel 2.26	−512	512	0	Continuous	Multimodal
F10	Schwefel 1.2	−100	100	0	Continuous	Unimodal
F11	Schwefel 2.22	−10	10	0	Continuous	Unimodal
F12	Schwefel 2.21	−100	100	0	Continuous	Unimodal
F13	Sphere	−5.12	5.12	0	Continuous	Unimodal
F14	Step	−5.12	5.12	0	Discontinuous	Unimodal

Table 5
Mean normalized values with twelve different chaotic maps.

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12
F01	4.03	1.32	1.26	4.44	1.33	4.29	1.18	1.27	1.24	**1.00**	1.24	1.25
F02	**1.00**	1.22	1.25	2.92	1.19	2.82	1.22	1.20	1.24	1.13	1.24	1.32
F03	7.38	1.17	1.11	29.79	1.71	35.59	1.07	1.15	1.02	**1.00**	1.07	1.13
F04	37.50	1.48	1.01	6.7E3	6.58	7.6E3	1.24	1.67	1.01	**1.00**	1.38	1.14
F05	17.81	1.05	1.19	898.85	2.91	957.45	1.25	1.12	**1.00**	1.06	1.30	1.10
F06	4.62	1.15	1.12	132.64	1.95	131.40	1.20	1.21	1.03	**1.00**	1.21	1.13
F07	1.68	**1.00**	1.04	2.59	**1.00**	2.51	1.02	1.03	1.01	1.03	1.04	1.02
F08	1.99	1.24	1.17	15.49	1.45	15.03	1.16	1.19	1.09	**1.00**	1.20	1.10
F09	1.16	**1.00**	1.01	1.34	1.01	1.10	1.01	1.01	1.03	1.05	1.01	1.03
F10	1.05	1.06	1.08	2.73	1.11	2.61	1.05	1.05	1.03	**1.00**	1.04	1.02
F11	**1.00**	1.04	1.05	2.42	1.09	2.20	1.04	1.04	1.06	1.06	1.07	1.10
F12	4.17	1.06	1.09	6.65	1.29	6.57	1.11	1.05	**1.00**	1.04	1.06	1.02
F13	7.61	1.15	1.09	36.83	1.73	40.34	1.11	1.18	1.07	**1.00**	1.06	1.08
F14	8.69	1.29	1.17	39.13	1.82	42.42	1.08	1.11	1.03	**1.00**	1.16	1.12

Table 6
Best normalized values with twelve different chaotic maps.

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12
F01	8.03	1.98	1.69	4.54	1.57	3.88	1.14	1.83	1.76	**1.00**	1.31	1.64
F02	1.44	1.50	1.98	3.59	1.57	2.60	1.93	1.10	1.41	1.41	2.19	**1.00**
F03	9.31	1.40	1.44	6.53	2.41	16.75	1.42	1.55	1.24	1.68	**1.00**	1.43
F04	3.12	2.23	1.91	4.0E4	2.34	2.1E5	3.81	4.17	2.07	**1.00**	3.09	3.28
F05	57.53	5.50	**1.00**	1.0E3	6.82	3.1E3	6.33	7.75	8.06	4.43	19.54	4.27
F06	5.20	1.19	2.34	48.35	3.67	75.48	**1.00**	1.19	1.80	1.16	2.58	2.06
F07	1.74	**1.00**	1.25	2.74	1.05	2.65	1.19	1.14	1.20	1.18	1.17	1.04
F08	2.00	1.14	1.26	4.84	1.86	7.92	1.26	1.16	1.07	**1.00**	1.13	1.01
F09	1.27	1.05	1.09	1.29	1.25	1.09	**1.00**	1.06	**1.00**	1.13	1.14	1.07
F10	1.20	1.21	1.31	1.58	1.12	2.95	1.35	1.13	1.24	1.07	1.11	**1.00**
F11	1.07	1.34	1.26	1.93	**1.00**	1.86	1.14	1.17	1.21	1.13	1.21	1.31
F12	10.33	1.58	1.41	10.63	2.12	6.48	1.64	1.86	1.35	1.51	**1.00**	1.39
F13	17.54	1.27	1.75	26.25	4.00	40.23	2.21	1.72	**1.00**	1.20	2.33	1.41
F14	18.66	1.87	1.69	50.27	3.62	41.49	1.42	1.48	1.15	**1.00**	1.93	1.60

From Tables 5 and 6, it can be seen that the basic KH combined with the M9 (Sine map) or M10 (Singer map) perform better than others. For M9 (Sine map) and M10 (Singer map), the M10 (Singer map) significantly outperforms the M9 (Sine map) when multiple runs are made. Accordingly, we choose it as the proper map to form the best chaotic KH (CKH) that will be studied in more detail in Section 4.2.

4.2. CKH vs. other optimization algorithms

The performance of CKH was compared on benchmark problems with nine other optimization algorithms [45]. The methods included in the comparative study are ACO [4], BA [51,52], CS [49], DE [30,38], ES [1,2], GA [18], KH [8], PBIL [36], and PSO

[16,27]. In DE, the DE/rand/1/bin scheme is used. It is worth mentioning that Gandomi and Alavi [8] have demonstrated that the KH II shows the best performance compared to the existing algorithms. Thus, in this work, KH II is considered as the basic KH method. The parameter settings in all experiments for each algorithm are shown in Table 7 [44].

Tables 8 and 9 record the average and the best results from 100 runs, respectively. Table 8 shows that, on average, CKH performs better than other methods on eleven of the fourteen benchmarks (F01–F08, and F12–F14) when searching for function minimum. CS, DE and GA are the second most effective, performing best on one of the fourteen benchmarks (F10, F11 and F09, respectively). From Table 9, it can be seen that, CKH performs better than other methods on eight of the fourteen benchmarks (F01, F03, F06–F08, and F12–F14). GA and ACO are the second and third most effective, respectively, performing the best on the benchmarks F02, F09, F10 and F04–F05 when multiple runs are made. CS and DE are the fourth most effec-

Table 7
Parameters settings.

Algorithms	Parameters
ACO	Initial pheromone value τ_0 = 1E−6, pheromone update constant Q = 20, exploration constant q_0 = 1, global pheromone decay rate ρ_g = 0.9, local pheromone decay rate ρ_l = 0.5, pheromone sensitivity s = 1, and visibility sensitivity β = 5
BA	Loudness A = 0.95, pulse rate r = 0.5, and scaling factor ε = 0.1
CKH	The foraging speed V_f = 0.02, the maximum diffusion speed $D^{\max}$ = 0.005, the maximum induced speed $N^{\max}$ = 0.01
CS	A discovery rate p_a = 0.25; for DE, a weighting factor F = 0.95 and a crossover constant CR = 0.4
DE	A weighting factor F = 0.95 and a crossover constant CR = 0.4
ES	The number of offspring λ = 10 produced in each generation, and standard deviation σ = 1 for changing solutions
GA	Roulette wheel selection, single point crossover with a crossover probability of 1, and a mutation probability of 0.01
KH	The foraging speed V_f = 0.02, the maximum diffusion speed $D^{\max}$ = 0.005, the maximum induced speed $N^{\max}$ = 0.01
PBIL	A learning rate of 0.05, 1 good population member and 0 bad population members to use to update the probability vector each generation, an elitism parameter of 1, and a 0 probability vector mutation rate
PSO	An inertial constant = 0.3, a cognitive constant = 1, and a social constant for swarm interaction = 1

Table 8
Mean normalized values.

	ACO	BA	CKH	CS	DE	ES	GA	KH	PBIL	PSO
F01	3.51	4.39	**1.00**	3.90	2.74	4.28	3.74	1.20	4.46	3.70
F02	2.65	3.84	**1.00**	1.87	1.04	2.68	1.08	1.06	2.36	2.27
F03	2.25	37.64	**1.00**	14.06	3.74	17.94	6.91	6.10	39.12	14.28
F04	3.8E3	7.6E3	**1.00**	343.53	24.30	2.9E3	45.91	435.21	6.0E3	545.34
F05	1.1E3	1.1E3	**1.00**	116.70	13.70	515.59	24.94	101.82	1.1E3	131.40
F06	8.68	126.33	**1.00**	20.39	3.13	106.70	6.57	24.48	121.83	24.12
F07	1.78	2.72	**1.00**	2.11	1.57	2.46	1.67	2.02	2.50	1.85
F08	14.74	15.34	**1.00**	4.08	2.17	19.11	3.62	5.02	15.56	4.28
F09	1.21	4.06	2.21	3.08	2.32	2.85	**1.00**	4.80	3.60	3.52
F10	1.57	3.75	1.02	**1.00**	2.14	2.42	1.63	5.14	2.40	1.64
F11	2.36	3.88	1.34	2.28	**1.00**	3.62	1.78	8.4E10	2.95	2.23
F12	4.85	8.51	**1.00**	5.72	6.35	7.66	6.66	1.73	8.16	6.55
F13	21.70	46.81	**1.00**	16.18	4.14	46.71	14.12	7.04	46.44	16.90
F14	3.54	42.51	**1.00**	15.68	3.90	26.29	8.22	6.43	44.57	15.78
Time	3.18	1.00	5.06	2.06	2.00	2.12	2.32	4.94	1.14	2.54

Table 9
Best normalized values.

	ACO	BA	CKH	CS	DE	ES	GA	KH	PBIL	PSO
F01	4.35	6.32	**1.00**	5.14	3.95	6.21	4.43	1.10	7.18	5.66
F02	10.52	12.96	2.86	6.07	3.42	9.50	**1.00**	18.56	6.78	7.94
F03	2.52	39.53	**1.00**	14.13	4.03	22.21	4.06	9.93	71.07	24.54
F04	**1.00**	7.0E7	163.92	3.1E5	1.2E4	4.8E7	155.30	1.6E7	2.9E8	3.6E6
F05	**1.00**	1.5E8	8.6E3	1.0E7	1.5E6	8.4E7	1.2E5	3.3E7	3.1E8	1.0E7
F06	6.78	111.99	**1.00**	14.21	3.02	191.23	2.23	51.11	169.98	27.53
F07	2.01	2.79	**1.00**	2.41	1.91	2.84	1.40	2.46	2.93	2.09
F08	13.56	9.19	**1.00**	2.79	2.12	16.53	1.65	4.47	16.31	1.97
F09	2.68	12.56	7.07	9.37	8.10	9.58	**1.00**	19.10	14.09	10.85
F10	1.15	2.98	1.01	**1.00**	2.60	2.63	**1.00**	2.29	2.80	1.72
F11	2.18	2.97	1.23	1.99	**1.00**	2.17	1.26	2.5E5	3.32	1.81
F12	7.94	14.43	**1.00**	9.41	12.12	17.02	5.38	2.33	18.18	10.41
F13	52.12	109.09	**1.00**	39.42	11.07	175.14	14.74	19.58	180.26	57.42
F14	2.83	48.10	**1.00**	18.54	5.39	43.68	3.58	8.93	86.33	23.91

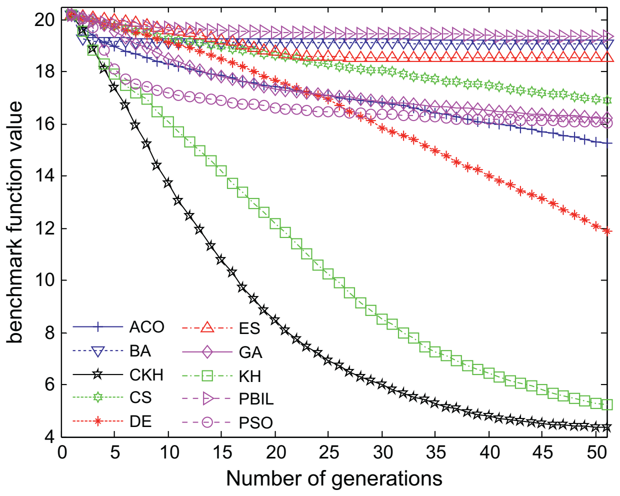

Fig. 5. Performance comparison on the F01 Ackley function.

tive. They have the best performance on the benchmarks F10 and F11, respectively. It can be seen that replacing inertia weights ω with chaotic maps can greatly enhance the performance of the KH.

Moreover, the calculated time demands of the ten optimization algorithms were similar. The average running times for these algorithms are collected and presented in Table 8. BA was the most efficient algorithm, while CKH was the tenth fastest of the ten methods.

Furthermore, optimization process of each algorithm is given in Figs. 5–18. The values shown in these figures are the average function optimum achieved from 100 runs. Here, all the values are not normalized, and they are true function values. In addition, it should be noted that the global optima of the benchmarks (F04, F05, and F11) are illustrated in the form of the semi-logarithmic convergence plots. Also, KH II is short for KH in Figs. 5–18.

Fig. 5 shows the values obtained by the ten methods on the F01 function, which is a multimodal function with a narrow global minimum basin ($F01_{min} = 0$) and many minor local optima. From Fig. 5, CKH performance is superior to the nine methods in optimization process, while KH II performs the second best in this benchmark. Moreover, all the methods have the almost same initial values, while CKH overtakes the other 9 methods. All methods clearly outperform the PBIL algorithm.

Fig. 6 shows the optimization values for F02 function. For this case, the figure clearly shows that CKH, DE and GA, KH II performances differ from each other. From Table 8 and Fig. 6, CKH has the best performance for this multimodal function. ACO, BA, CS, ES, PBIL, and PSO fail to find the best solution in this benchmark function.

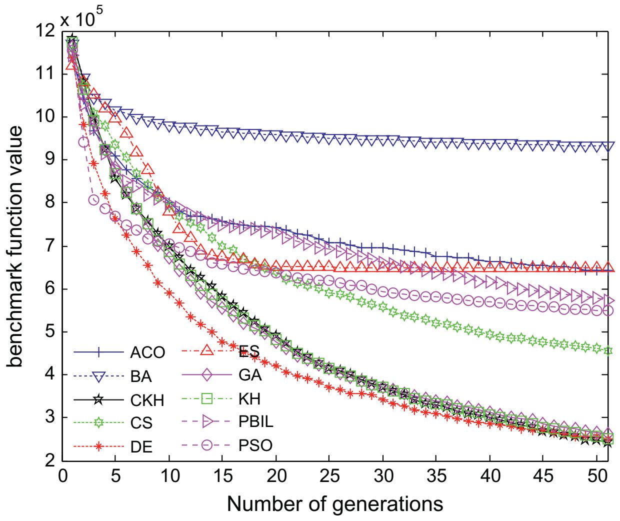

Fig. 6. Performance comparison on the F02 Fletcher–Powell function.

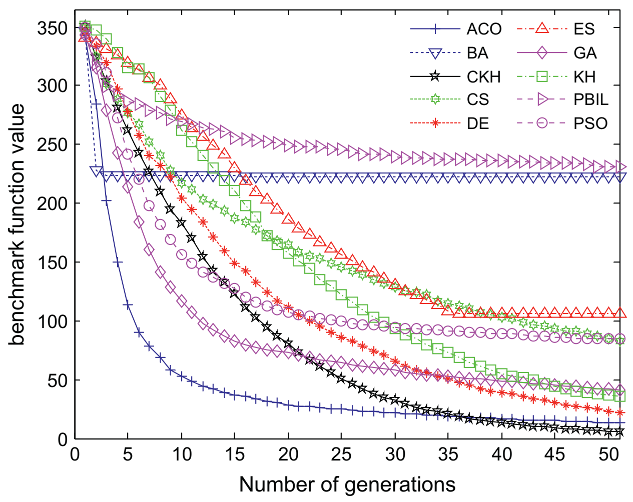

Fig. 7. Performance comparison on the F03 Griewank function.

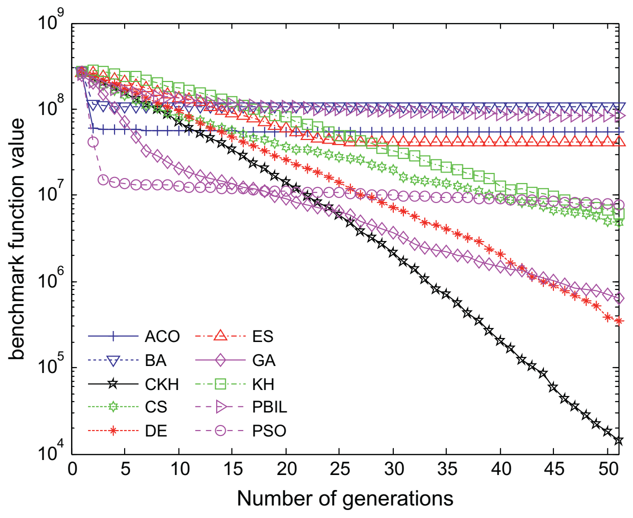

Fig. 8. Performance comparison on the F04 Penalty #1 function.

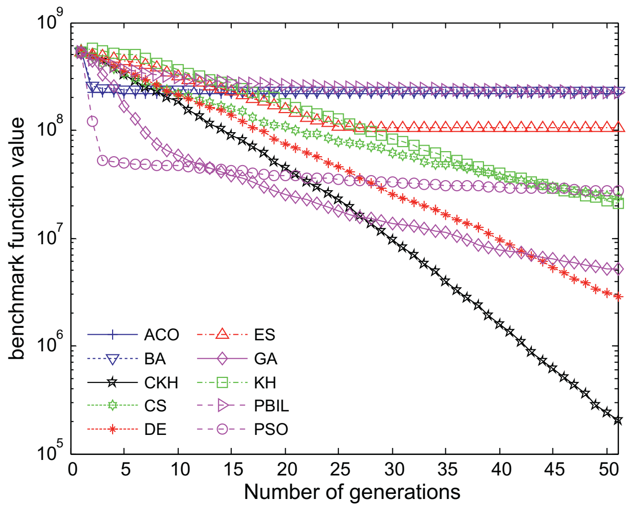

Fig. 9. Performance comparison on the F05 Penalty #2 function.

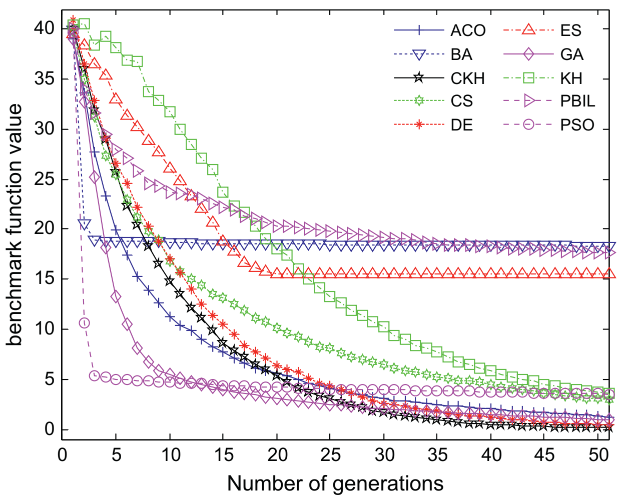

Fig. 10. Performance comparison on the F06 Quartic (*with noise*) function.

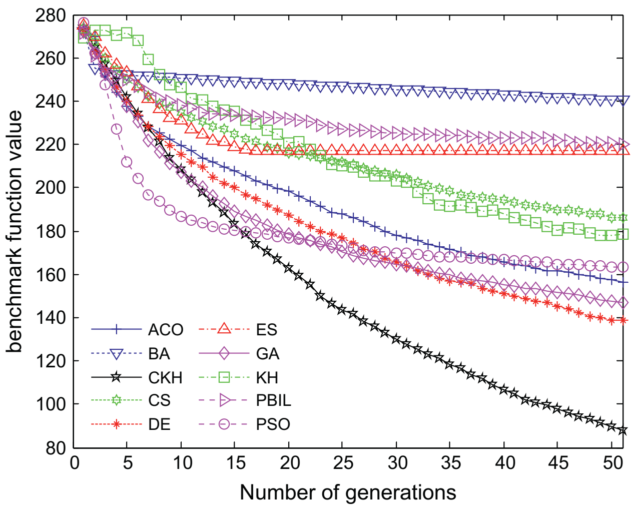

Fig. 11. Performance comparison on the F07 Rastrigin function.

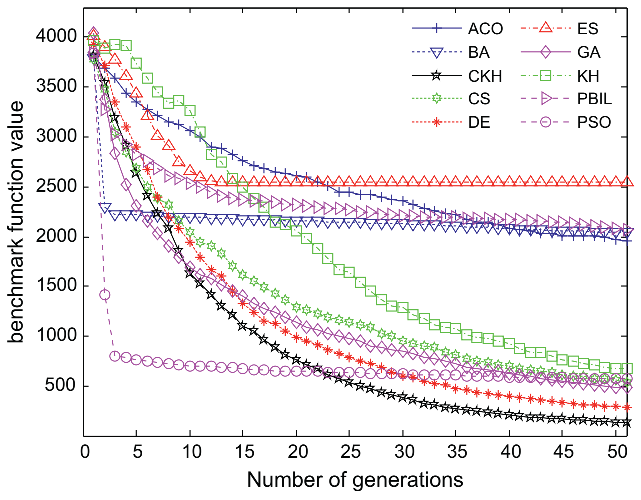

Fig. 12. Performance comparison on the F08 Rosenbrock function.

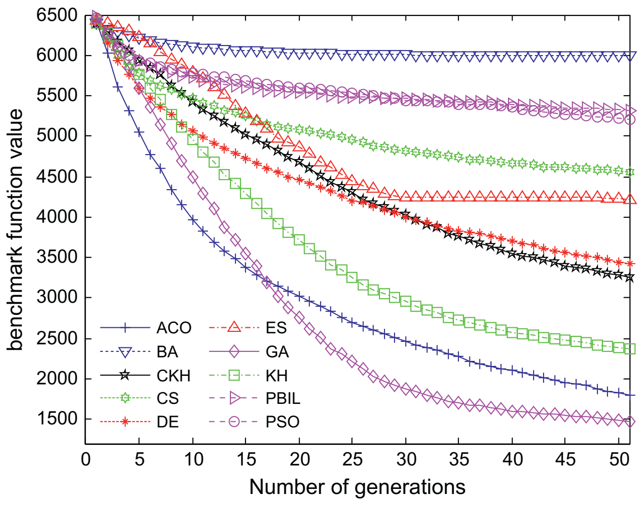

Fig. 13. Performance comparison on the F09 Schwefel 2.26 function.

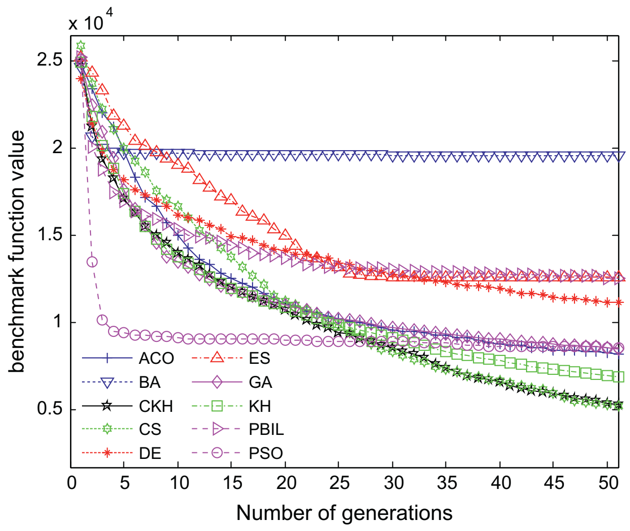

Fig. 14. Performance comparison on the F10 Schwefel 1.2 function.

Fig. 7 shows the optimization values for F03 function. F03 has a strange property, as it is much easier to solve for higher dimensions than lower dimensions [32]. From Fig. 7, it can be seen that CKH has the best performance for this benchmark. For other algorithms, ACO, BA, GA, and PSO have the fastest convergence rate initially towards the global optimum. However, they are outperformed by CKH after some generations. Further, ACO and DE work very well because they rank 2 and 3, respectively, among ten algorithms.

Fig. 8 illustrates the values for F04 Penalty #1 function. From Fig. 8, apparently, CKH overtakes all other approaches. For this case, PSO and GA show a faster convergence rate initially, however they seem to be trapped into sub-optimal values as the procedure proceeds, especially PSO. Although slower, DE performs the second best in the optimization process.

Fig. 9 illustrates the values achieved on F05 function. For this benchmark, very similar to F04 in Fig. 8, CKH significantly overtakes all other approaches in the optimization process.

Fig. 10 illustrates the values achieved for the ten methods when using F06. With reference to this benchmark, very similar to F02 Fletcher–Powell function as shown in Fig. 6, the figure reveals that ACO, CKH, DE and GA perform slightly different from each other. However, Table 8 and Fig. 10 show that CKH provides better results for this problem. BA, CS, ES, KH II, PBIL, and PSO do not succeed in this benchmark function. At last, DE and GA converge to the value that is very close to CKH's.

Fig. 11 illustrates the functions values for the F07 Rastrigin function that is a difficult multimodal function with a unique global minimum of $F07_{min} = 0$ and several local optima. When attempting to solve F07, methods may be converged into a local value. Therefore, a method that can keep a larger diversity is well capable of producing better values. At first glance

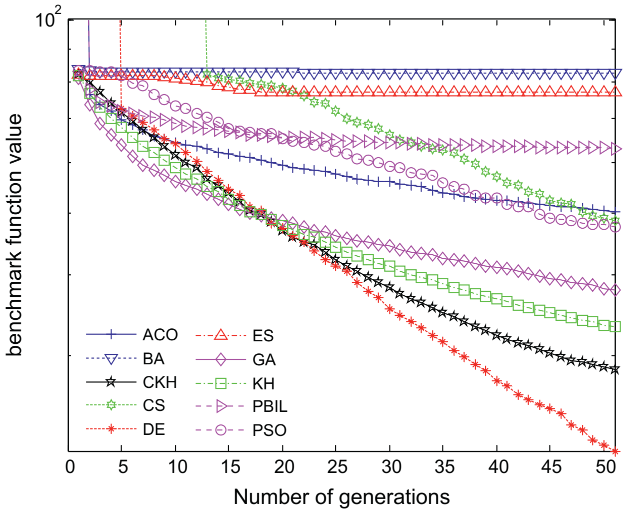

Fig. 15. Performance comparison on the F11 Schwefel 2.22 function.

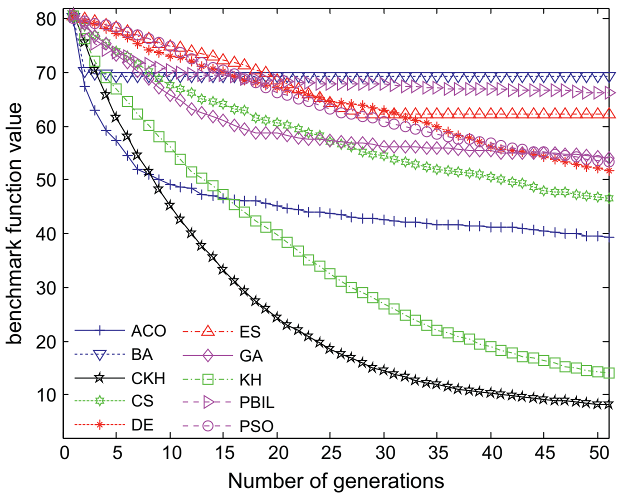

Fig. 16. Performance comparison on the F12 Schwefel 2.21 function.

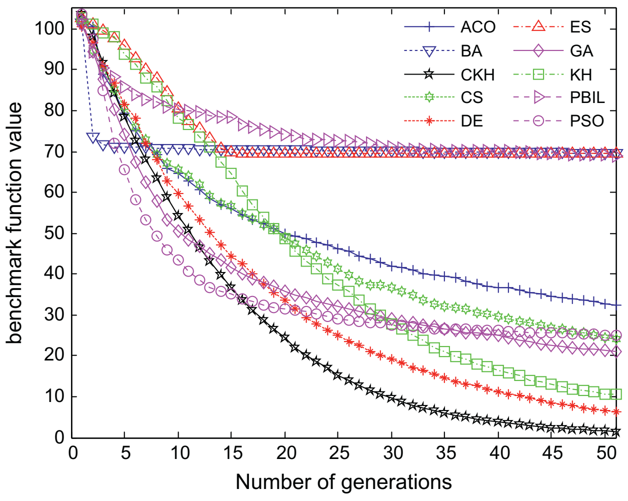

Fig. 17. Performance comparison on the F13 Sphere function.

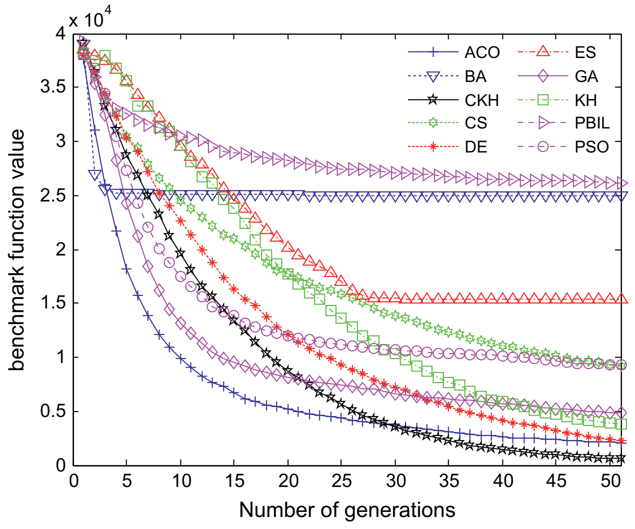

Fig. 18. Performance comparison on the F14 Step function.

it is clear that CKH greatly overtakes all other approaches. Moreover, PSO has the fastest convergence rate initially towards global value. However, it is outperformed by CKH after 16 generations. Further, DE and GA compares well compared with other algorithms.

Fig. 12 shows the functions values for F08 function, which can also be regarded as a multimodal function. For this case, analogous to the F07 in Fig. 11, clearly, CKH greatly overtakes all other approaches. In addition, PSO has the fastest convergence rate initially towards the global solution. However, it is outperformed by CKH after 22 generations. Further, the performance of DE and GA is very good for this case.

Fig. 13 reveals the equivalent values for the F09 Schwefel 2.26 function. From Fig. 13, GA performs better than other nine methods involving CKH in the optimization process, while ACO performs the second best in this multimodal benchmark. Moreover, KH II is superior to CKH.

Fig. 14 displays the values for F10 function. From Fig. 14, for this case, analogous to the F02 Fletcher–Powell function as shown in Fig. 6, it can be concluded that: (i) the performance of CKH and CS is similar. (ii) KH II is only inferior to CKH and CS, and performs the third best in this benchmark function. However, referring to Table 8 and Fig. 14, CS slightly outperforms CKH and the other approaches. PSO has the fastest convergence rate initially at finding the global minimum. However, it is overtaken by CKH and CS after 26 iterations.

Fig. 15 reveals the function values for F11 Schwefel 2.22 function. From Fig. 15, DE has the stable convergence rate towards the global solution and overtakes all other approaches in this unimodal benchmark function. CKH is only inferior to DE, and performs the second best in this unimodal function.

Fig. 16 reveals the function values for F12 Schwefel 2.21 function. At first glance, it is obvious that CKH has the fastest convergence rate towards the global solution in the whole optimization process. CKH reaches the optimal solution significantly earlier than other algorithms. KH II is only inferior to CKH, and performs the second best in this unimodal function.

Fig. 17 shows the function values for F13 Sphere function, which is also known as *de Jong*'s function, and has a single global value $F13_{min} = 0$, therefore, it is easy to solve. From Fig. 17, CKH has the fastest convergence rate towards the global solution and overtakes all other methods. Fig. 17 shows that GA performs better than DE at the beginning of the search, but DE eventually finds the value of CKH, while GA cannot. KH II fails to find the global value within the maximum number of iterations.

Fig. 18 displays the function values for F14 Step function. CKH shows the fastest convergence rate towards the global solution and overtakes all other methods. Referring to Fig. 18, ACO shows a little faster convergence than DE. Both ACO and DE are only inferior to CKH and reach a minimum that is very close to CKH's.

Considering the results shown in Figs. 5–18, it can be concluded that the CKH's performance is superior to or at least quite competitive with the other nine algorithms. Further, benchmarks F02, F04–F08, and F10 illustrate that PSO performs much better than the other methods initially, while later it is overtaken by other approaches.

4.3. Comparisons using t-test

Based on the final search results of 100 independent trials on every function, Table 10 presents the *t* values on every function of the two-tailed test with the 5% level of significance between the CKH and other optimization methods. In the table, the value of *t* with 198 degrees of freedom is significant at $\alpha = 0.05$ by a two-tailed test. **Boldface** indicates that CKH performs

Table 10
Comparisons between CKH and other methods at $\alpha = 0.05$ on a two-tailed t-tests.

	ACO	BA	CS	DE	ES	GA	KH	PBIL	PSO
F01	**65.34**	**121.01**	**84.83**	**62.15**	**118.07**	**70.03**	**12.03**	**131.81**	**96.11**
F02	**27.74**	**24.46**	**17.31**	0.41	**23.41**	0.97	**25.90**	**22.50**	**24.36**
F03	**18.19**	**34.53**	**34.66**	**32.93**	**45.49**	**17.60**	**34.59**	**74.05**	**63.74**
F04	**6.10**	**14.50**	**12.72**	**8.37**	**19.39**	**5.57**	**20.14**	**24.72**	**13.82**
F05	**7.07**	**15.99**	**13.08**	**16.05**	**25.77**	**6.59**	**24.55**	**39.43**	**15.60**
F06	**14.58**	**18.21**	**17.46**	**17.26**	**35.63**	**11.62**	**32.35**	**43.30**	**23.09**
F07	**31.77**	**52.85**	**48.61**	**28.53**	**62.35**	**23.33**	**39.06**	**70.16**	**40.54**
F08	**37.70**	**25.27**	**24.03**	**30.04**	**37.33**	**17.06**	**33.43**	**42.38**	**24.08**
F09	**26.39**	**36.54**	**18.13**	1.46	**12.61**	**28.36**	**39.12**	**29.90**	**21.10**
F10	**10.16**	**18.63**	**2.41**	**22.72**	**25.45**	**10.23**	**15.24**	**26.84**	**12.46**
F11	**22.33**	1.01	**15.17**	**11.29**	**29.63**	**7.60**	**1.00**	**43.07**	**9.08**
F12	**43.68**	**67.68**	**57.24**	**79.07**	**95.40**	**43.09**	**12.68**	**109.18**	**45.62**
F13	**34.57**	**36.20**	**41.53**	**35.45**	**67.82**	**25.75**	**37.26**	**66.18**	**59.59**
F14	**19.71**	**40.08**	**35.81**	**32.87**	**60.56**	**18.21**	**31.53**	**66.39**	**52.55**

Table 11
The number of fitness evaluations for different methods.

	ACO	BA	CKH	CS	DE	ES	GA	KH	PBIL	PSO
F01	50000	50000	19028	26820	**18380**	50000	50000	28478	50000	50000
F02	50000	50000	49995	50000	50000	50000	50000	**49981**	50000	50000
F03	50000	50000	49995	50000	50000	50000	50000	**49981**	50000	50000
F04	50000	50000	**15109**	23040	16600	50000	21410	30683	50000	50000
F05	50000	50000	**7797**	25660	18480	50000	42160	21713	50000	50000
F06	2160	50000	2282	2260	3140	50000	**1450**	5949	50000	27890
F07	50000	50000	49995	50000	50000	50000	50000	**49981**	50000	50000
F08	50000	50000	49995	50000	50000	50000	50000	**49981**	50000	50000
F09	50000	50000	49995	50000	50000	50000	50000	**49981**	50000	50000
F10	50000	50000	**43813**	50000	50000	50000	50000	49981	50000	50000
F11	50000	50000	22260	19780	**15100**	50000	50000	47836	50000	50000
F12	50000	50000	**10261**	50000	47280	50000	50000	17969	50000	50000
F13	50000	50000	**3939**	8120	8760	50000	18220	8425	50000	50000
F14	50000	50000	**8645**	30380	19980	50000	50000	21804	50000	50000

significantly better than the compared algorithms. For instance comparing the CKH and DE, the former significantly outperformed the latter on twelve functions (F01, F03–F08, and F10–F14), does as good as the latter on two functions (F02 and F09). This indicates that the CKH generally performs better than DE on the solution accuracy. Though the CKH performed slightly weaker on some functions, Table 10 also reveals that it outperforms the other nine methods for most functions.

4.4. The study of the number of fitness evaluations

In order to fully investigate the advantage of the CKH method, the number of fitness evaluations is further studied. We look at the number of fitness evaluations for every function for each method to the fixed function values. In the experiments, the fixed function values are set to $opt + 1$. opt indicates the optimum for every function that can be found in fifth column of Table 4. The maximum of fitness evaluations is set to 50,000. That is to say, if a method cannot converge to the values $opt + 1$ before the maximum number of fitness evaluations is reached (50,000), it will stop, too, and return only up to 50,000 fitness evaluations. All the results are recorded in Table 11. Table 11 shows that, for the six functions F04–F05, F10, F12–F14, CKH can find the satisfactory results with the least fitness evaluations. KH is somewhat inferior to the CKH method in terms of number of fitness evaluations, and it can converge to the fixed values with the least fitness evaluations for the five functions F02–F03 and F07–F09. Not only do DE and GA also perform well, but also they use the least evaluations when solving the test problems F01, F11 and F06.

5. Discussion

From the experiments conducted in Sections 4.1–4.4 on the benchmarks, it is demonstrated that CKH performs better than or comparable to the basic KH and other optimization approaches. Moreover, the implementation of CKH is simple and easy, and no effect is made to fine-tune the parameters. The work carried out in this paper demonstrates the CKH to be robust over all kinds of benchmark functions.

There are many ways to verify the performance of the meta-heuristic optimization algorithms. Benchmark evaluation is a simple way that is widely used. However, it is not a perfect approach. Firstly, the parameters used in these methods cannot be adjusted carefully. It is well-known that for these optimization methods, they always have some randomness. Therefore, different tuning parameters might make a big difference in their performance. Secondly, most practical application problems are not necessarily related with benchmark functions. Finally, benchmark evaluation might arrive at diverse conclusions if we change population size or maximum number of generations [44].

When a meta-heuristic method is used to solve a certain problem, the running time is a key factor. If a method wastes a long time searching for the best or second best solution, it must be impractical [45]. CKH does not seem to demand an impractical amount of computational effort. In comparison with other algorithms considered in this study, CKH was the tenth fastest. In CKH, calculating the chaotic sequence takes a relatively long time. How to make the CKH's converge faster still deserves further scrutiny.

6. Conclusions

The chaos theory and KH method are hybridized in order to design a novel improved meta-heuristic chaotic KH method (CKH) for solving optimization problems. Various chaotic maps are used to regulate the inertia weights, ω, of KH. The Singer map is selected as its ω through comparing various chaotic KH variants to form the best CKH. The simulations show that the usage of deterministic chaotic signals instead of linearly decreasing values is an important modification of the KH method. The experimental results show that the tuned KH significantly enhances the reliability of the global optimality and the quality of the solutions. In addition, the results reveal that CKH considerably enhances the KH's performances on most multimodal and unimodal problems. Moreover, CKH is simple and easy to implement.

Another advantage of CKH is that there are fewer parameters to adjust compared with other population-based optimization methods. The performance of a method that relies on parameters is mainly determined by those parameters. In such cases, it is hard to select the optimal parameters for an optimization method [14]. The present study proposes a new technique to deal with this difficult issue by using chaotic maps in place of these parameters. In CKH, the chaos theory is used to adjust the key parameter, ω. However, the introduction of chaos theory leads to another problem, which makes the approach take a long time to select the best chaotic map in place of the parameter, ω.

A limitation of this study is that only fourteen benchmarks are used to verify the performance of CKH. Thus, in order to evaluate the proposed method more in-depth, more comprehensive problems should be tested such as the high-dimensional ($d \geqslant 20$) CEC 2005 test suite [39]. Furthermore, CKH may be compared with other state-of-the-art optimization methods. In the future, the CKH method can be extended to solve constrained optimization problems such as the constrained optimization CEC 2005 test suite [39]. Herein, two mechanisms are used to improve the KH method. An interesting topic for future research would be investigating the real impact of elitism and the use of chaos in order to analyze their contributions to the algorithm performance. For more verification, the CKH method might be applied to solve practical engineering problems such as benchmark structural optimization problems. We will also perform mathematical analysis of the proposed method using dynamic system such as Markov chain to prove and explain its convergence.

References

[1] T. Back, Evolutionary Algorithms in Theory and Practice, Oxford University Press, 1996.
[2] H. Beyer, The Theory of Evolution Strategies, Springer, 2001.
[3] M. Črepinšek, S.-H. Liu, L. Mernik, A note on teaching–learning-based optimization algorithm, Inform. Sci. 212 (2012) 79–93.
[4] M. Dorigo, T. Ant Colony, Ant Colony Optimization, MIT Press, 2004.
[5] O.K. Erol, I. Eksin, A new optimization method: Big Bang–Big Crunch, Adv. Eng. Softw. 37 (2) (2006) 106–111.
[6] A.H. Gandomi, A.H. Alavi, Multi-stage genetic programming: a new strategy to nonlinear system modeling, Inform. Sci. 181 (23) (2011) 5227–5239.
[7] A.H. Gandomi, X.-S. Yang, A.H. Alavi, Mixed variable structural optimization using Firefly Algorithm, Comput. Struct. 89 (23–24) (2011) 2325–2336.
[8] A.H. Gandomi, A.H. Alavi, Krill Herd: a new bio-inspired optimization algorithm, Commun. Nonlinear Sci. Numer. Simul. 17 (12) (2012) 4831–4845.
[9] A.H. Gandomi, S. Talatahari, X.S. Yang, S. Deb, Design optimization of truss structures using cuckoo search algorithm, Struct. Des. Tall Spec. Build. 22 (17) (2013) 1330–1349.
[10] A.H. Gandomi, X.-S. Yang, S. Talatahari, S. Deb, Coupled eagle strategy and differential evolution for unconstrained and constrained global optimization, Comput. Math. Appl. 63 (1) (2012) 191–200.
[11] A.H. Gandomi, X.-S. Yang, A.H. Alavi, S. Talatahari, Bat algorithm for constrained optimization tasks, Neural Comput. Appl. 22 (6) (2013) 1239–1255.
[12] A.H. Gandomi, X.S. Yang, S. Talatahari, A.H. Alavi, Metaheuristic Applications in Structures and Infrastructures, Elsevier, 2013.
[13] A.H. Gandomi, X.S. Yang, S. Talatahari, A.H. Alavi, Firefly algorithm with chaos, Commun. Nonlinear Sci. Numer. Simul. 18 (1) (2013) 89–98.
[14] A.H. Gandomi, G.J. Yun, X.-S. Yang, S. Talatahari, Chaos-enhanced accelerated particle swarm optimization, Commun. Nonlinear Sci. Numer. Simul. 18 (2) (2013) 327–340.
[15] Z.W. Geem, J.H. Kim, G.V. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation 76 (2) (2001) 60–68.
[16] S. Gholizadeh, F. Fattahi, Design optimization of tall steel buildings by a modified particle swarm algorithm, Struct. Des. Tall Spec. Build. 23 (4) (2014) 285–301.
[17] S. Gholizadeh, A. Barzegar, Shape optimization of structures for frequency constraints by sequential harmony search algorithm, Eng. Optimiz. 45 (6) (2013) 627–646.
[18] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, 1998.
[19] N. Hachicha, B. Jarboui, P. Siarry, A fuzzy logic control using a differential evolution algorithm aimed at modelling the financial market dynamics, Inform. Sci. 181 (1) (2011) 79–91.
[20] X. Han, X. Chang, A chaotic digital secure communication based on a modified gravitational search algorithm filter, Inform. Sci. 208 (2012) 14–27.
[21] X. Han, X. Chang, An intelligent noise reduction method for chaotic signals based on genetic algorithms and lifting wavelet transforms, Inform. Sci. 218 (2013) 103–118.
[22] 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.
[23] A. Kaveh, S. Talatahari, Size optimization of space trusses using Big Bang–Big Crunch algorithm, Comput. Struct. 87 (17–18) (2009) 1129–1140.

[24] A. Kaveh, S. Talatahari, A novel heuristic optimization method: charged system search, Acta Mech. 213 (3–4) (2010) 267–289.
[25] A. Kaveh, S. Talatahari, Optimal design of Schwedler and ribbed domes via hybrid Big Bang–Big Crunch algorithm, J. Constr. Steel Res. 66 (3) (2010) 412–419.
[26] A. Kaveh, S. Talatahari, A discrete big bang–big crunch algorithm for optimal design of skeletal structures, Asian J. Civ. Eng. 11 (1) (2010) 103–122.
[27] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceeding of the IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.
[28] R.J. Kuo, Y.J. Syu, Z.-Y. Chen, F.C. Tien, Integration of particle swarm optimization and genetic algorithm for dynamic clustering, Inform. Sci. 195 (2012) 124–140.
[29] X. Li, J. Wang, J. Zhou, M. Yin, A perturb biogeography based optimization with mutation for global numerical optimization, Appl. Math. Comput. 218 (2) (2011) 598–609.
[30] X. Li, M. Yin, An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure, Adv. Eng. Softw. 55 (2013) 10–31.
[31] X. Li, J. Zhang, M. Yin, Animal migration optimization: an optimization algorithm inspired by animal migration behavior, Neural Comput. Appl. (2013) 1–11.
[32] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput. 10 (3) (2006) 281–295.
[33] B. Nouhi, S. Talatahari, H. Kheiri, Chaos embedded charged system search for practical optimization problems, Int. J. Optimiz. Civ. Eng. 3 (1) (2013) 23–36.
[34] L. Pecora, T. Carroll, Synchronization in chaotic system, Phys. Rev. Lett. 64 (8) (1990) 821–824.
[35] R.V. Rao, V.J. Savsani, D.P. Vakharia, Teaching–learning-based optimization: an optimization method for continuous non-linear large scale problems, Inform. Sci. 183 (1) (2012) 1–15.
[36] B. Shumeet, Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning, Carnegie Mellon University, 1994.
[37] D. Simon, Biogeography-based optimization, IEEE Trans. Evol. Comput. 12 (6) (2008) 702–713.
[38] 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.
[39] P. Suganthan, N. Hansen, J. Liang, K. Deb, Y. Chen, A. Auger, S. Tiwari, in: Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-parameter Optimization, Nanyang Technological University, 2005.
[40] S. Talatahari, M. Kheirollahi, C. Farahmandpour, A.H. Gandomi, A multi-stage particle swarm for optimum design of truss structures, Neural Comput. Appl. 23 (5) (2013) 1297–1309.
[41] S. Talatahari, B. Farahmand Azar, R. Sheikholeslami, A.H. Gandomi, Imperialist competitive algorithm combined with chaos for global optimization, Commun. Nonlinear Sci. Numer. Simul. 17 (3) (2012) 1312–1319.
[42] S. Talatahari, A. Kaveh, R. Sheikholeslami, Engineering design optimization using chaotic enhanced charged system search algorithms, Acta Mech. 223 (10) (2012) 2269–2285.
[43] G. Wang, L. Guo, H. Wang, H. Duan, L. Liu, J. Li, Incorporating mutation scheme into krill herd algorithm for global numerical optimization, Neural Comput. Appl. 24 (3–4) (2014) 853–871.
[44] G. Wang, L. Guo, A novel hybrid bat algorithm with harmony search for global numerical optimization, J. Appl. Math. 2013 (2013) 1–21.
[45] G. Wang, L. Guo, A.H. Gandomi, L. Cao, A.H. Alavi, H. Duan, J. Li, Lévy-flight krill herd algorithm, Math. Probl. Eng. 2013 (2013) 1–14.
[46] W. Xu, Z. Geng, Q. Zhu, X. Gu, A piecewise linear chaotic map and sequential quadratic programming based robust hybrid particle swarm optimization, Inform. Sci. 218 (2013) 85–102.
[47] P. Yadav, R. Kumar, S.K. Panda, C.S. Chang, An Intelligent Tuned Harmony Search algorithm for optimisation, Inform. Sci. 196 (2012) 47–72.
[48] D. Yang, G. Li, G. Cheng, On the efficiency of chaos optimization algorithms for global optimization, Chaos, Solitons Fractals 34 (4) (2007) 1366–1375.
[49] X.S. Yang, S. Deb, Cuckoo search via Lévy flights, in: Proceeding of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), IEEE Publications, 2009, pp. 210–214.
[50] X.S. Yang, Nature-inspired Metaheuristic Algorithms, second ed., Luniver Press, 2010.
[51] X.S. Yang, A new metaheuristic bat-inspired algorithm, in: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), 2010, pp. 65–74.
[52] X.S. Yang, A.H. Gandomi, Bat algorithm: a novel approach for global engineering optimization, Eng. Comput. 29 (5) (2012) 464–483.
[53] X.S. Yang, A.H. Gandomi, S. Talatahari, A.H. Alavi, Metaheuristics in Water, Geotechnical and Transport Engineering, Elsevier, 2013.
[54] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput. 3 (2) (1999) 82–102.
[55] Q. Yuan, F. Qian, W. Du, A hybrid genetic algorithm with the Baldwin effect, Inform. Sci. 180 (5) (2010) 640–652.
[56] Y. Zhang, D. Huang, M. Ji, F. Xie, Image segmentation using PSO and PCM with Mahalanobis distance, Expert Syst. Appl. 38 (7) (2011) 9036–9040.