Journal of Computational Science 5 (2014) 224–232

Contents lists available at ScienceDirect

Journal of Computational Science

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

Chaotic bat algorithm

Amir H. Gandomi [a], Xin-She Yang [b,*]

[a] Department of Civil Engineering, The University of Akron, Akron, OH 44325, USA
[b] School of Science and Technology, Middlesex University, Hendon, London NW4 4BT, UK

ARTICLE INFO

Article history:
Received 5 December 2012
Received in revised form 18 July 2013
Accepted 4 October 2013
Available online 14 October 2013

Keywords:
Bat algorithm
Chaos
Metaheuristic
Global optimization

ABSTRACT

Bat algorithm (BA) is a recent metaheuristic optimization algorithm proposed by Yang. In the present study, we have introduced chaos into BA so as to increase its global search mobility for robust global optimization. Detailed studies have been carried out on benchmark problems with different chaotic maps. Here, four different variants of chaotic BA are introduced and thirteen different chaotic maps are utilized for validating each of these four variants. The results show that some variants of chaotic BAs can clearly outperform the standard BA for these benchmarks.

1. Introduction

Many design optimization problems are often highly nonlinear, which can typically have multiple modal optima, and it is thus very challenging to solve such multimodal problems. To cope with this issue, global optimization algorithms are widely attempted, however, traditional algorithms may not produce good results, and latest trends are to use new metaheuristic algorithms [1]. Metaheuristic techniques are well-known global optimization methods that have been successfully applied in many real-world and complex optimization problems [2,3]. These techniques attempt to mimic natural phenomena or social behavior so as to generate better solutions for optimization problem by using iterations and stochasticity [4]. They also try to use both intensification and diversification to achieve better search performance. Intensification typically searches around the current best solutions and selects the best candidate designs, while the diversification process allows the optimizer to explore the search space more efficiently, mostly by randomization [1].

In recent years, several novel metaheuristic algorithms have been proposed for global search. Such algorithms can increase the computational efficiency, solve larger problems, and implement robust optimization codes [5]. For example, Xin-She Yang [6] recently developed a promising metaheuristic algorithm, called bat algorithm (BA). Preliminary studies suggest that the BA can have superior performance over genetic algorithms and particle swarm optimization [6], and it can solve real world and engineering optimization problems [7–10]. On the other hand, recent advances in theories and applications of nonlinear dynamics, especially chaos, have drawn more attention in many fields [10]. One of these fields is the applications of chaos in optimization algorithms to replace certain algorithm-dependent parameters [11].

Previously, chaotic sequences have been used to tune parameters in metaheuristic optimization algorithms such as genetic algorithms [12], particle swarm optimization [13], harmony search [14], ant and bee colony optimization [15,16], imperialist competitive algorithm [17], firefly algorithm [18], and simulated annealing [19]. Such a combination of chaos with metaheuristics has shown some promise once the right set of chaotic maps are used. It is still not clear why the use of chaos in an algorithm to replace certain parameters may change the performance, however, empirical studies indeed indicate that chaos can have high-level of mixing capability, and thus it can be expected that when a fixed parameter is replaced by a chaotic map, the solutions generated may have higher mobility and diversity. For this reason, it may be useful to carry out more studies by introducing chaos to other, especially newer, metaheuristic algorithms.

Therefore, one of the aims of this paper is to introduce chaos into the standard bat algorithm, and as a result, we propose a chaos-based bat algorithm (CBA). As different chaotic maps may lead to different behavior of the algorithm, we then have a set of chaos-based bat algorithms. In these algorithms, we use different chaotic systems to replace the parameters in BA. Thus different methods that use chaotic maps as potentially efficient alternatives to pseudorandom sequences have been proposed. In order to evaluate the

* Corresponding author. Tel.: +44 2084112351.
E-mail addresses: a.h.gandomi@gmail.com, ag72@uakron.edu (A.H. Gandomi), x.yang@mdx.ac.uk (X.-S. Yang).

1877-7503/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.jocs.2013.10.002

proposed algorithms, a set of unimodal and multimodal mathematical benchmarks are utilized. The simulation results reveal the improvements of the new algorithms, due to the application of deterministic chaotic signals instead of constant values.

The rest of the paper is organized as follows: Section 2 presents the descriptions of the standard BA and proposes four different chaotic BAs. The chaotic maps that generate chaotic sequences in the BA steps are described in Section 3. Section 4 describes how to implement the simulations, while in Section 5, we discuss the tuning of the BA parameters and finding the best chaotic BA. Finally, Section 6 presents the unique features of the chaotic BAs and outlines directions for further research.

2. Bat algorithm

The bat-inspired metaheuristic algorithm, namely the bat algorithm, was recently proposed by Xin-She Yang [6], based on the echolocation of microbats [20]. In the real world, echolocation can have only a few thousandths of a second (up to about 8–10 ms) with a varying frequency in the region of 25–150 kHz, corresponding to the wavelengths of 2–14 mm in the air.

Microbats typically use echolocation for searching for prey. During roaming, microbats emit short pulses; however, when a potential prey is nearby, their pulse emits rates increase and the frequency is tuned up. The increase of the frequency, namely frequency-tuning, together with the speedup of pulse emission will shorten the wavelength of echolocations and thus increase accuracy of the detection. Nature has use frequency-tuning for many years, and in oil industry, they also use frequency tuning to detect different layers of potential oil reserves by increasing the frequency and energy for thinner layers. Bat algorithm was developed to use the key idea of frequency tuning based on the echolocation of microbats. In the standard bat algorithm, the echolocation characteristics of microbats can be idealized as the following three rules:

i. All bats use echolocation to sense distance, and they also 'know' the difference between food/prey and background barriers in some magical way;
ii. Bats fly randomly with velocity v_i at position x_i with a fixed frequency $f_{\min}$, varying wavelength λ and loudness A_0 to search for prey. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission $r \in [0,1]$, depending on the proximity of their target;
iii. Although the loudness can vary in many ways, we assume that the loudness varies from a large (positive) A_0 to a minimum constant value $A_{\min}$.

The basic steps of BA can be summarized as the pseudo code shown in Fig. 1.

For each bat (say i), we have to define its position $\boldsymbol{x}_i$ and velocity $\boldsymbol{v}_i$ in a d-dimensional search space, and they should be updated subsequently during the iterations. The new solutions $\boldsymbol{x}_i^t$ and velocities $\boldsymbol{v}_i^t$ at time step t can be calculated by

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta \tag{1}$$

$$\boldsymbol{v}_i^t = \boldsymbol{v}_i^{t-1} + (\boldsymbol{x}_i^{t-1} - \boldsymbol{x}^*)f_i \tag{2}$$

$$\boldsymbol{x}_i^t = \boldsymbol{x}_i^{t-1} + \boldsymbol{v}_i^t \tag{3}$$

where β in the range of [0,1] is a random vector drawn from a uniform distribution. Here $\boldsymbol{x}^*$ is the current global best location (solution) found so far, which is located after comparing all the solutions among all the n bats at the current iteration. As the product $\lambda_i f_i$ is the velocity increment, we can use either f_i (or λ_i) to adjust the velocity change while fixing the other factor λ_i (or f_i),

Bat Algorithm

Objective function $f(\mathrm{x})$, $\mathrm{x} = (x_1, \ldots, x_d)^T$
Initialize the bat population x_i $(i = 1,2, \ldots, n)$ *and* v_i
Define pulse frequency f_i *at* x_i
Initialize pulse rates r *and the loudness* A
while *(t <Max number of iterations)*
  Generate new solutions by adjusting frequency,
  and updating velocities and locations/solutions [equations (2) to (4)]
  if *(rand > r)*
    Select a solution among the best solutions
    Generate a local solution around the selected best solution
  end if
  Generate a new solution by flying randomly
  if *(rand < A & f*$(\mathrm{x}_i) < f(\mathrm{x}_*)$*)*
    Accept the new solutions
  end if
  Rank the bats and find the current best x_*
end while
Postprocess results and visualization

Fig. 1. Pseudo code of the bat algorithm (BA).

depending on the type of the problem of interest. In our implementation, we will use $f_{\min} = 0$ and $f_{\max} = 2$, though the actual range can vary, depending on the domain size of the problem of interest. Initially, each bat is randomly assigned a frequency that is drawn uniformly from $[f_{\min}, f_{\max}]$.

By looking at the bat algorithm more closely, we can see that BA can have global and local search abilities, depending on the parameters, and it can also automatically switch from global search to local search by tuning relevant parameters. This switch is controlled by α and γ to be introduce below. The local search is essentially a random walk around the current best solutions, and a new solution for each bat can be generated locally using:

$$x_{\text{new}} = x_{\text{old}} + \varepsilon A^t \tag{4}$$

where the random number ε is drawn from $[-1, 1]$, while $A^t = \langle A_i^t \rangle$ is the average loudness of all the bats at this time step. In fact, this is the main updating equation of simulated annealing. For this reason, simulated annealing can be thought as a very special case of the bat algorithm.

It is worth pointing out that, to a degree, BA can be considered as a balanced combination of local and global moves and this is manifested by controlling the loudness and pulse rate. For simplicity, we can also use $A_0 = 1$ and $A_{\min} = 0$, assuming $A_{\min} = 0$ means that a bat has just found the prey and temporarily stop emitting any sound. Now we have

$$A_i^{t+1} = \alpha A_i^t, \quad r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)] \tag{5}$$

where α and γ are constants. In fact, α is similar to the cooling factor of a cooling schedule in the simulated annealing. For any $0 < \alpha, \gamma < 1$, we have

$$A_i^t \to 0, \quad r_i^t \to r_i^0, \quad \text{as} \quad t \to \infty \tag{6}$$

In the simplest case, we can use $\alpha = \gamma$, and in the standard BA, we can use $\alpha = \gamma = 0.9$ to 0.975 in most cases, though have used $\alpha = \gamma = 0.9$ in our simulations. However, the main purpose of this paper is to use chaotic maps to tune these 4 parameters so as to see if a chaotic map can improve the efficiency of the bat algorithm. As we can see below, some chaotic maps can indeed enhance the efficient of BA.

3. Chaotic maps

In almost all metaheuristic algorithms with stochastic components, randomness is achieved by using some probability distributions, often uniform or Gaussian. In principle, it can be advantageous to replace such randomness by chaotic maps because chaos can have very similar properties of randomness with better statistical and dynamical properties. Such dynamical mixing is important to ensure that solutions generated by the algorithm can be diverse enough to potentially reach every mode in the multimodal objective landscape. The methods using chaotic maps to replace random variables are called chaotic optimization (CO). Due to the ergodicity and mixing properties of chaos, algorithms can potentially carry out iterative search steps at higher speeds than standard stochastic search with standard probability distributions [21]. To achieve such potential, we will use one-dimensional, non-invertible maps to generate a set of chaotic bat algorithm variants. In the rest of this section, we will first outline some of well-known, one-dimensional maps to be used for later simulations.

(a) Chebyshev map

The family of Chebyshev map can be defined as follows [22]:

$$x_{k+1} = \cos(k\cos^{-1}(x_k)) \tag{7}$$

(b) Circle map

The Circle map [23] is represented by the following equation:

$$x_{k+1} = x_k + b - \left(\frac{a}{2\pi}\right)\sin(2\pi x_k)\mathrm{mod}(1) \tag{8}$$

For $a = 0.5$ and $b = 0.2$, it generates chaotic sequence in (0, 1).

(c) Gauss/Mouse map

The following equations define Gaussian map [24]:

$$x_{k+1} = \begin{cases} 0 & x_k = 0 \\ \dfrac{1}{x_k \mathrm{mod}(1)} & \text{otherwise} \end{cases} \tag{9a}$$

$$\frac{1}{x_k \mathrm{mod}(1)} = \frac{1}{x_k} - \left\lfloor \frac{1}{x_k} \right\rfloor, \tag{9b}$$

which also generates chaotic sequences in (0, 1).

(d) Intermittency map

The intermittency map has two parts: linear and non-linear. It is formulated as [25]:

$$x_{k+1} = \begin{cases} \varepsilon + x_k + c x_k^n & 0 < x_k \le P \\ \dfrac{x_k - P}{1 - P} & P < x_k < 1 \end{cases}. \tag{10}$$

(e) Iterative map

The iterative chaotic map with infinite collapses [26] can be written as

$$x_{k+1} = \sin\left(\frac{a\pi}{x_k}\right) \tag{11}$$

where $a \in (0,1)$ is a suitable parameter.

(f) Liebovitch map

This map, introduced by Liebovitch and Toth [22], can be written as

$$x_{k+1} = \begin{cases} \alpha x_k & 0 < x_k \le P_1 \\ \dfrac{P - x_k}{P_2 - P_1} & P_1 < x_k \le P_2 \\ 1 - \beta(1 - x_k) & P_2 < x_k \le 1 \end{cases} \tag{12}$$

where $\alpha < \beta$ that are defined as follows:

$$\alpha = \frac{P_2}{P_1}(1 - (P_2 - P_1)) \tag{13}$$

$$\beta = \frac{1}{P_2 - 1}((P_2 - 1) - P_1(P_2 - P_1)) \tag{14}$$

Here three equal limits are considered for the map.

(g) Logistic map

This classic logistic map appears in nonlinear dynamics of biological population evidencing chaotic behavior [27], and can be written as

$$x_{k+1} = a x_k (1 - x_k) \tag{15}$$

where x_k is the kth chaotic number, with k denoting the iteration number. Obviously, $x \in (0,1)$ under the conditions that the initial $x_0 \in (0,1)$ and that $x_0 \notin \{0.0, 0.25, 0.75, 0.5, 1.0\}$. In later experiments, $a = 4$ is used.

(h) Piecewise map

The so-called piecewise map is governed by the following equation [28]:

$$x_{k+1} = \begin{cases} \dfrac{x_k}{P} & 0 \le x_k < P \\ \dfrac{x_k - P}{0.5 - P} & P \le x_k < \dfrac{1}{2} \\ \dfrac{1 - P - x_k}{0.5 - P} & \dfrac{1}{2} \le x_k < 1 - P \\ \dfrac{1 - x_k}{P} & 1 - P \le x_k < 1 \end{cases} \tag{16}$$

where P is the control parameter between 0 and 0.5 and $x \in (0,1)$. Obviously, $P \neq 0$.

(i) Sawtooth map

The sawtooth map can be formulated by a mod function [29]:

$$x_{k+1} = 2x_k \mathrm{mode}(1) \tag{17}$$

(j) Sine map

The sine map is a unimodal map and can be given by [30]:

$$x_{k+1} = \frac{a}{4}\sin(\pi x_k) \tag{18}$$

where $0 < a \le 4$.

(k) Singer map

Singer's map is a one-dimensional system as given below [31]:

$$x_{k+1} = \mu(7.86x_k - 23.31x_k^2 + 28.75x_k^3 - 13.3x_k^4) \tag{19}$$

where μ is parameter between 0.9 and 1.08.

(l) Sinusoidal map

This iterator [27] can be defined as

$$x_{k+1} = a x_k^2 \sin(\pi x_k) \tag{20}$$

In a special case when $a = 2.3$ and $x_0 = 0.7$, it can be simplified as

$$x_{k+1} = \sin(\pi x_k) \tag{21}$$

(m) Tent map

This map is similar to the well-known logistic map. It displays some specific chaotic effects. This map cab be defined by the following equation [32]:

$$x_{k+1} = \begin{cases} \dfrac{x_k}{0.7} & x_k < 0.7 \\ \dfrac{10}{3}(1 - x_k) & x_k \ge 0.7 \end{cases} \tag{22}$$

Fig. 2 visualizes the chaotic value distributions of 50 iterations for all thirteen maps with random initial values. It should be noted that the chaotic maps do not produce values between 0 and 1 are normalized to have the same scale.

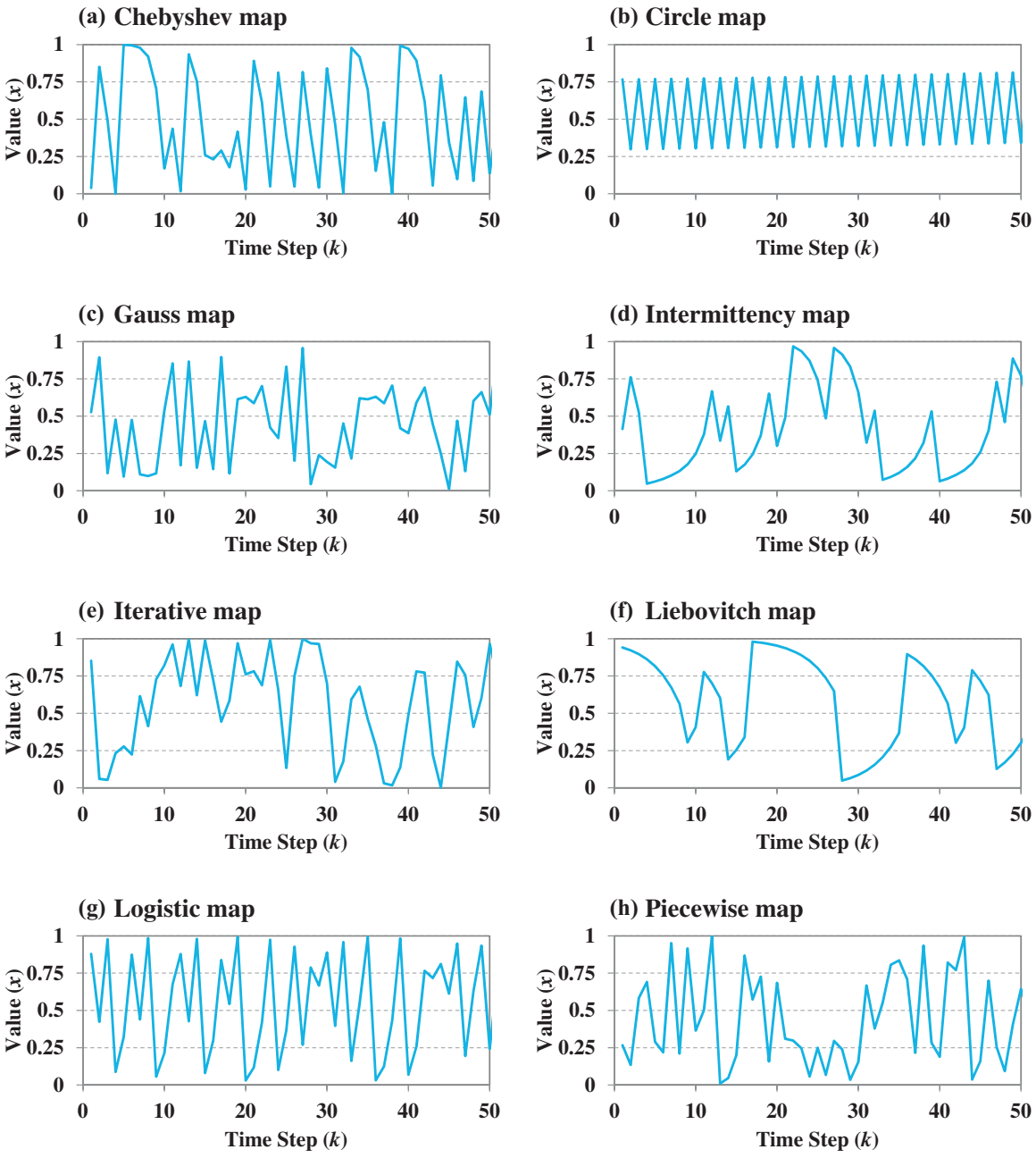

Fig. 2. Chaotic value distributions during 50 iteration.

4. Numerical simulations and results

4.1. Benchmark experiments

Different chaotic bat algorithms B have been benchmarked using six well-known numerical examples. The first three functions are unimodal, while the others are all multimodal functions. The detailed of benchmark functions are presented in Table 1.

The global minimum values of the above objectives are all $f(X^*) = 0$ and the global optima are located at the origin ($X^* = 0, 0, \ldots, 0$), except Rosenbrock and Penalized functions whose optima are located at $X^* = (1, 1, \ldots, 1)$. The search domain bounds of all the variables in the benchmark functions are -10 to 10 for all dimensions, and for the boundary constraint handling, we have used the evolutionary scheme [33].

4.2. Criterion for performance measures

There are many criteria in the literature for evaluating the performance of the algorithms, including the success rate, number of function evaluations, statistical variations and their combinations. Here, we will use the success rate which is defined as

$$S_r = 100 \times \frac{N_{\text{successful}}}{N_{\text{all}}} \tag{23}$$

where N_{all} is the number of all trials, and $N_{\text{successful}}$ is the number of trials which found the solution is successful. Here, we consider a run as a successful run when the found solution is very near to the global optimum. It should be noted that this distance varies with

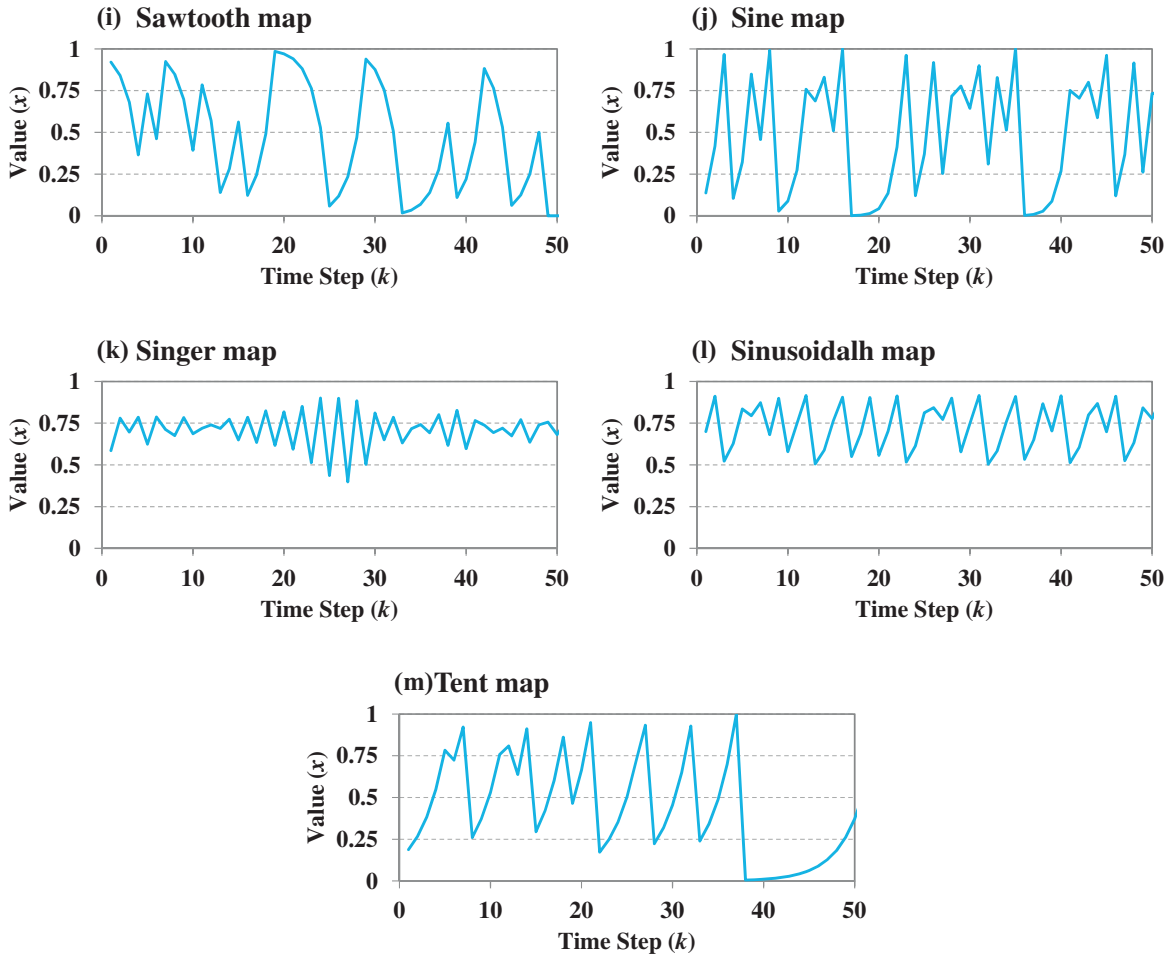

Fig. 2. (*Continued*)

Table 1
Benchmark problems.

ID	Name	Formula	Dimensions
F1	Sphere	$f(X) = \|\|X\|\| \quad \text{where} \quad \|\|X\|\| = \sqrt{\sum_{i=1}^{n} x_i^2}$	30
F2	Schwefel	$f(X) = \sum_{i=1}^{n} \|x_i\| + \prod_{i=1}^{n} \|x_i\|$	20
F3	Rosenbrock	$f(X) = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	10
F4	Ackley	$f(X) = -a \exp\left(-0.02\sqrt{n^{-1}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(n^{-1}\sum_{i=1}^{n} \cos(2\pi x_i)\right) + a + e, \; a = 20$	10
F5	Griewank	$f(X) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right)$	10
F6	Penalized	$f(X) = \frac{1}{10}\left(\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2(1 + \sin^2(3\pi x_{i+1})) + (x_n - 1)^2\right) + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$ $\text{where} \quad u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & a < x_i \\ 0 & -a \le x_i \le a \\ k(-x_i - a)^m & x_i < a \end{cases}$	10

Table 2
Success rate of CBA-I for benchmark functions with different chaotic maps.

Chaotic map name	F1	F2	F3	F4	F5	F6
Chebyshev map	57	37	34	96	65	34
Circle map	64	35	34	93	66	46
Gauss/mouse map	64	37	44	91	67	33
Intermittency map	60	35	44	92	61	32
Iterative map	60	41	44	99	73	33
Liebovitch map	65	42	39	96	65	40
Logistic map	58	32	34	95	70	39
Piecewise map	65	34	35	95	73	33
Sawtooth map	64	34	41	90	66	29
Sine map	69	43	38	94	67	34
Singer map	59	42	37	97	74	35
Sinusoidal map	65	37	36	98	67	26
Tent map	58	34	41	97	67	26

different search space. Thus, the criteria for a successful run can be defined as:

$$\sum_{d=1}^{D}(X_i^{gb} - X_i^*)^2 \le (UB - LB) \times 10^{-4} \tag{24}$$

where D is the dimension of the test function, X_i^{gb} is ith dimension of the obtained global best by the proposed algorithms; UB and LB are upper and lower bounds, respectively.

4.3. Initialization and parameter studies

We have also used 100 different runs for each parameter setting with completely random, different initial conditions. The final results are found to be almost independent of the initial configurations. In fact, we have also used statistical measures, such as the mean objective values and their standard deviations, to measure the performance of an algorithm, rather than relying simply on a few runs. This approach is reflected in many tables provided in the main text. Furthermore, in most cases in our implementation, we have carried out some extensive sensitivity studies of parameters such as the population size and attractiveness. From our simulations, we observed that the population size $n = 10$ to 40 would be sufficient for most problems.

5. Chaotic bat algorithms

In this section we have used chaotic maps in four different ways to tune the BA parameters and improve the performance, which will lead to a set of bat algorithm, or different variants of the chaotic bat algorithm. The flowchart of a schematic chaotic BA (CBA) is presented in Fig. 3. The following subsections describe how parameters can be tuned. We also present the simulation results, and then compare different chaotic BAs with each other as well as with the standard BA.

5.1. CBA-I

Parameter β of Eq. (1) is modified by chaotic maps (CM) during iterations and the frequency equation is modified as:

$$f_i = f_{\min} + (f_{\max} - f_{\min})CM_i \tag{25}$$

In the standard BA, β is a random number between 0 and 1 and in the CBA-I, it is a chaotic number between 0 and 1. The means of the best results over 100 runs of CBA-I for 13 different chaotic maps are presented in Fig. 4. The success rates of different maps are also presented in Table 2.

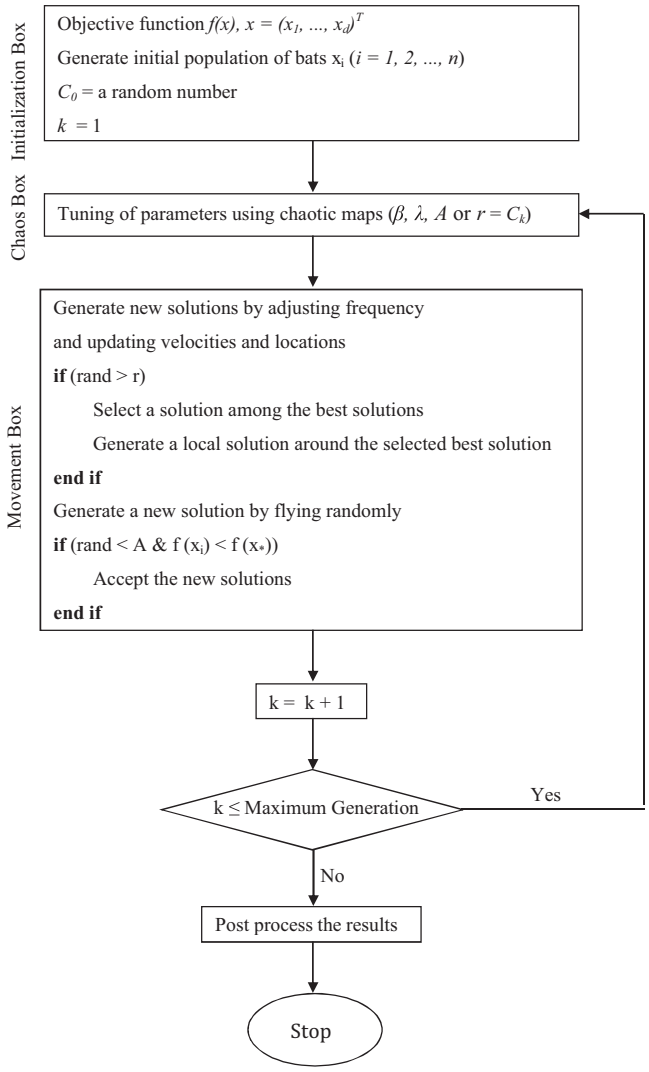

Fig. 3. Schematic flowchart of a chaotic bat algorithm (CBA).

5.2. CBA-II

Parameter λ_i of the velocity equation is modified by the selected chaotic maps and the Eq. (2) equation is modified by

$$\boldsymbol{v}_i^t = \boldsymbol{v}_i^{t-1} + (\boldsymbol{x}_i^t - \boldsymbol{x}^*)CM_i f_i \tag{26}$$

In the standard BA, λ is a random number between 0 and 1 via a normalized frequency, and in the CBA-II, it is a chaotic number between 0 and 1. The means of the best results over 100 runs of CBA-II for 13 different chaotic maps is presented in Fig. 5. The success rates of different maps are also presented in Table 3.

5.3. CBA-III

The loudness (A) in BA is also replaced with the chaotic maps to improve the performance of the BA. It should be noted that although the loudness changes in a monotonic way between 0 and 1 in the conventional BA, the chaotic maps change during iterations differently. In the CBA-II, it is a chaotic number between 0 and 1. The means of the best results over 100 runs of CBA-III for 13 different chaotic maps are presented in Fig. 6. The success rates of the CBA-III for 13 different chaotic maps are presented in Table 4.

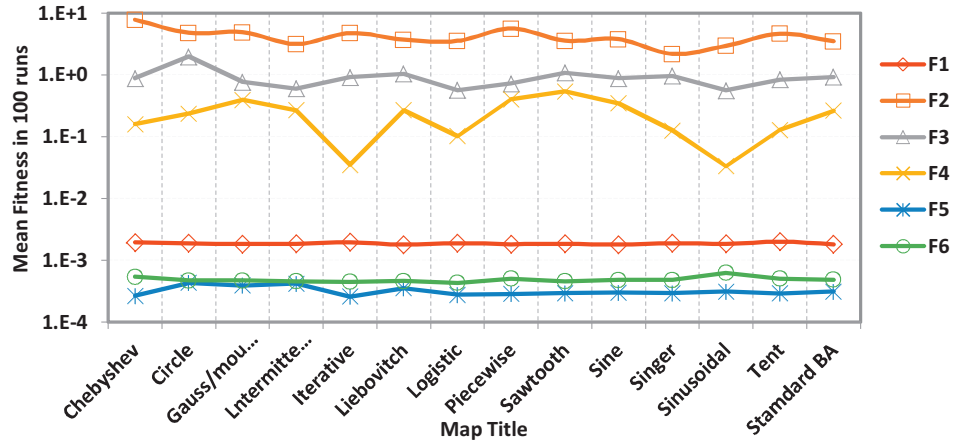

Fig. 4. Comparison of mean of best results for CBA-I.

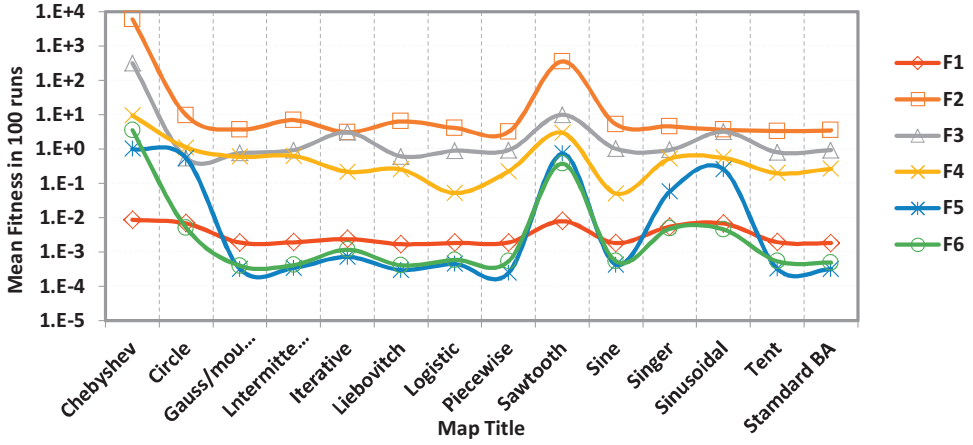

Fig. 5. Comparison of mean of best results for CBA-II.

5.4. CBA-IV

Chaotic maps can also be used to replace the pulse emission rate (r) to potentially improve the performance of the BA. In the standard BA, the pulse rate varies monotonically between 0 and 1, while, in the CBA-IV, it becomes a chaotic number between 0 and 1. The means of best results over 100 runs of CBA-IV for 13 different chaotic maps are presented in Fig. 7. The success rates of different maps are also presented in Table 5.

5.5. Overall analysis

To make overall comparisons of the different CBAs and standard BA, the success rates of different benchmark problems are

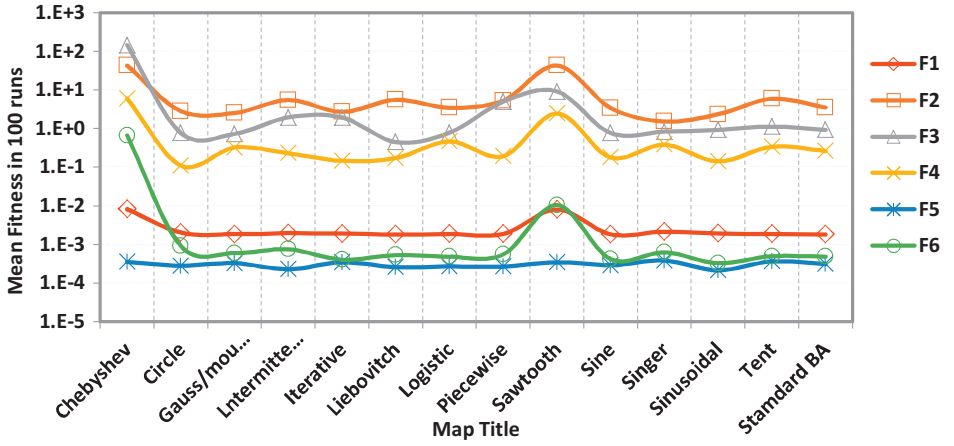

Fig. 6. Comparison of mean of best results for CBA-III.

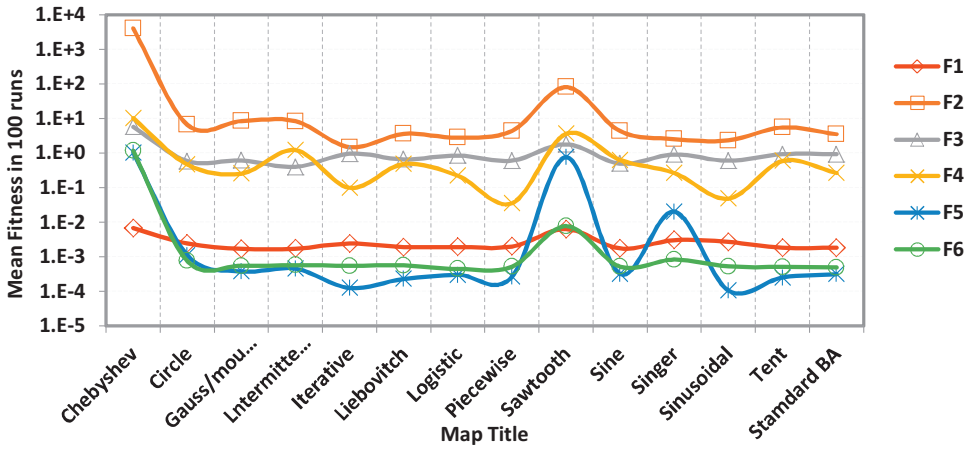

Fig. 7. Comparison of mean of best results for CBA-IV.

Table 3
Success rates of CBA-II for benchmark functions with different chaotic maps.

Chaotic map name	F1	F2	F3	F4	F5	F6
Chebyshev map	0	0	0	0	0	0
Circle map	0	1	3	1	0	0
Gauss/mouse map	63	36	47	92	68	39
Intermittency map	57	30	31	90	70	50
Iterative map	31	26	5	8	39	3
Liebovitch map	75	21	26	92	66	41
Logistic map	64	25	10	81	61	19
Piecewise map	62	26	35	92	76	34
Sawtooth map	0	0	0	0	2	0
Sine map	66	22	11	87	60	29
Singer map	0	10	0	0	9	0
Sinusoidal map	0	1	2	0	0	0
Tent map	56	36	34	92	73	33

Table 4
Success rates of CBA-III for benchmark functions with different chaotic maps.

Chaotic map name	F1	F2	F3	F4	F5	F6
Chebyshev map	0	0	0	0	63	0
Circle map	51	25	22	57	74	28
Gauss/mouse map	70	32	30	89	71	25
Intermittency map	57	27	18	78	78	17
Iterative map	59	36	22	98	72	51
Liebovitch map	68	23	17	92	70	29
Logistic map	66	28	30	92	70	29
Piecewise map	65	25	26	89	74	25
Sawtooth map	0	0	0	0	69	0
Sine map	65	30	27	88	72	29
Singer map	63	35	29	90	67	44
Sinusoidal map	57	35	34	98	74	57
Tent map	64	26	19	91	69	33

Table 5
Success rates of CBA-IV for benchmark functions with different chaotic maps.

Chaotic map name	F1	F2	F3	F4	F5	F6
Chebyshev map	0	0	0	0	0	0
Circle map	43	32	42	61	61	19
Gauss/mouse map	78	40	24	89	65	36
Intermittency map	74	20	1	67	59	24
Iterative map	28	40	54	99	86	31
Liebovitch map	70	31	14	90	70	32
Logistic map	64	32	30	95	72	33
Piecewise map	61	35	26	96	76	36
Sawtooth map	0	2	0	0	0	0
Sine map	73	33	22	90	65	30
Singer map	16	38	68	89	84	37
Sinusoidal map	26	39	77	99	93	42
Tent map	61	29	15	90	74	37

Table 6
Comparison of the success rates of all benchmark functions for CBAs and BA for using standard value and different chaotic maps.

Algorithm	CBA-I	CBA-II	CBA-III	CBA-IV
Replaced parameter	β	λ	A	r
Chebyshev map	323	0	63	0
Circle map	338	5	257	258
Gauss/mouse map	336	**345**	317	332
Intermittency map	324	328	275	245
Iterative map	**350**[a]	112	338	338
Liebovitch map	347	321	299	307
Logistic map	328	260	315	326
Piecewise map	335	325	304	330
Sawtooth map	324	2	69	2
Sine map	345	275	311	313
Singer map	344	19	328	332
Sinusoidal map	329	3	**355**	**376**
Tent map	323	324	302	306
Standard BA = 304				

[a] Bold sets are the best success rates for the parameters.

summarized and presented in Table 6. As it can be seen from this table, chaotic maps can improve the performance. For the CBA-I and CBA-II, the most suitable maps are the iterative map and Gauss/mouse map, respectively. From Table 6, Sinusoidal map is the most suitable for CBA-III and CBA-IV to replace with loudness (A) and pulse rate (r), respectively; however, it seems that it is not suitable to substitute with λ at all. Replacing pulse rate (r) in CBA-IV is more effective than others and we can say the best algorithm among these four is the CBA-IV with the Sinusoidal map. It is worth mentioning that although some chaotic maps improve the performance, some of them are not suitable at all. The improvement and degradation of each chaotic map can be attributed to the probability distribution (or generally statistic properties) of the map, as well as their time correlations.

6. Discussions and conclusions

The use of chaos has been one of the techniques to tune some of the parameters in metaheuristic algorithms, and this has become an active research topic in the recent optimization literature. In the present work, we have introduced chaos to the standard BA, and have developed a set of chaotic BA variants. In fact, four different CBAs have been proposed and thirteen different chaotic maps have been used to validate these algorithms. By comparing different chaotic BAs, the algorithm which uses the Sinusoidal map as its pulse rate (r), that is CBA-IV, is the best CBA. The results reveal that

the improvement of the new algorithms, due to the application of deterministic chaotic signals in place of constant and/or random values. Statistical results and the success rates of the CBAs suggest that the tuned algorithms can clearly improve the reliability of the global optimality, and they also enhance the quality of the results.

An interesting and yet important question is that how some chaotic maps can improve the performance of an algorithm, while others do not. It lacks a good theoretical framework for such analysis. Obviously, any progress in such analysis will no doubt provide insight into the working mechanism of chaotic metaheuristic algorithms and the combination of chaos and metaheuristics.

References

[1] X.S. Yang, Nature-Inspired Metaheuristic Algorithms, Luniver Press, Bristol, UK, 2008.
[2] W. Annicchiarico, J. Periaux, M. Cerrolaza, G. Winter, Evolutionary Algorithms and Intelligent Tools in Engineering Optimization, WIT Press, Portland, OR, 2005.
[3] A.H. Gandomi, X.S. Yang, S. Talatahari, A.H. Alavi, Metaheuristic Applications in Structures and Infrastructures, Elsevier, Waltham, MA, 2013.
[4] E. Talbi, Metaheuristics: From Design to Implementation, John Wiley & Sons, Hoboken, New Jersey, USA, 2009.
[5] X.S. Yang, Engineering Optimization: An Introduction with Metaheuristic Applications, John Wiley & Sons, Hoboken, NJ, 2010.
[6] X.S. Yang, A new metaheuristic bat-inspired algorithm, in: J.R. Gonzalez, et al. (Eds.), Nature Inspired Cooperative Strategies for Optimization (NISCO 2010). Studies in Computational Intelligence, Springer Berlin, 284, Springer, Berlin, 2010, pp. 65–74.
[7] X.S. Yang, A.H. Gandomi, Bat algorithm: a novel approach for global engineering optimization, Engineering Computation 29 (5) (2012) 464–483.
[8] T.C. Bora, L.S. Coelho, L. Lebensztajn, Bat-inspired optimization approach for the brushless DC wheel motor problem, IEEE Transactions on Magnetics 48 (2) (2012) 947–950.
[9] A.H. Gandomi, X.S. Yang, S. Talatahari, A.H. Alavi, Bat algorithm for constrained optimization tasks, Neural Computing & Applications 22 (6) (2013) 1239–1255.
[10] L.T. Pecora, Carroll, synchronization in chaotic system, Physical Review Letters 64 (8) (1990) 821–824.
[11] D. Yang, G. Li, G. Cheng, On the efficiency of chaos optimization algorithms for global optimization, Chaos Solution & Fractal 34 (2007) 1366–1375.
[12] G. Gharooni-fard, F. Moein-darbari, H. Deldari, A. Morvaridi, Scheduling of scientific workflows using a chaos-genetic algorithm, Procedia Computer Science 1 (2010) 1445–1454.
[13] A.H. Gandomi, G.J. Yun, X.S. Yang, S. Talatahari, Chaos-enhanced accelerated particle swarm algorithm, Communications in Nonlinear Science and Numerical Simulation 18 (2) (2013) 327–340.
[14] B. Alatas, Chaotic harmony search algorithms, Applied Mathematics and Computation 216 (2010) 2687–2699.
[15] W. Gong, S. Wang, Chaos ant colony optimization and application, in: 4th International Conference on Internet Computing for Science and Engineering, 2009, pp. 301–303.
[16] B. Alatas, Chaotic bee colony algorithms for global numerical optimization, Expert Systems with Applications 37 (2010) 5682–5687.
[17] S. Talatahari, R. Sheikholeslami, B. Farahmand Azar, A.H. Gandomi, Imperialist competitive algorithm combined with chaos for global optimization, Communications in Nonlinear Science and Numerical Simulations 17 (3) (2012) 1312–1319.
[18] A.H. Gandomi, X.S. Yang, S. Talatahari, A.H. Alavi, Firefly algorithm with chaos, Communications in Nonlinear Science and Numerical Simulation 18 (1) (2013) 89–98.
[19] J. Mingjun, T. Huanwen, Application of chaos in simulated annealing, Chaos, Solitons & Fractals 21 (2004) 933–941.
[20] P. Richardson, Bats, Natural History Museum, London, 2008.
[21] L. Coelho, V.C. Mariani, Use of chaotic sequences in a biologically inspired algorithm for engineering design optimization, Expert Systems with Applications 34 (2008) 1905–1913.
[22] M.S. Tavazoei, M. Haeri, Comparison of different one-dimensional maps as chaotic search pattern in chaos optimization algorithms, Applied Mathematics and Computation 187 (2007) 1076–1085.
[23] R.C. Hilborn, Chaos and Nonlinear Dynamics: An Introduction for Scientists and Engineer, 2nd ed., Oxford Univ. Press, New York, 2004.
[24] D. He, C. He, L. Jiang, H. Zhu, G. Hu, Chaotic characteristic of a one-dimensional iterative map with infinite collapses, IEEE Transactions on Circuits and Systems 48 (7) (2001) 900–906.
[25] A. Erramilli, R.P. Singh, P. Pruthi, Modeling packet traffic with chaotic maps, Royal Institute of Technology, ISRN KTH/IT/R-94/18-SE, Stockholm-Kista, Sweden, August, 1994.
[26] R.M. May, Simple mathematical models with very complicated dynamics, Nature 261 (1976) 459–467.
[27] Y. Li, S. Deng, D. Xiao, A novel Hash algorithm construction based on chaotic neural network, Neural Computing and Applications 20 (2011) 133–141.
[28] A.G. Tomida, Matlab toolbox and GUI for analyzing one-dimensional chaotic maps, in: International Conference on Computational Sciences and Its Applications ICCSA, IEEE Press, 2008, pp. 321–330.
[29] A. Wolf, Quantifying chaos with Lyapunov exponents, in: A.V. Holden (Ed.), Chaos, Princeton University Press, Princeton, NJ, 1986.
[30] R.L. Devaney, An Introduction to Chaotic Dynamical Systems, Addison-Wesley, Redwood City, CA, 1987.
[31] H. Peitgen, H. Jurgens, D. Saupe, Chaos and Fractals, Springer-Verlag, Berlin, Germany, 1992.
[32] E. Ott, Chaos in Dynamical Systems, Cambridge University Press, Cambridge, UK, 2002.
[33] A.H. Gandomi, X.S. Yang, Evolutionary boundary constraint handling scheme, Neural Computing & Applications 21 (6) (2012) 1449–1462.

Amir H. Gandomi is a Researcher in Department of Civil Engineering at the University Akron, USA. He was selected as an elite in 2008 by Iranian National Institute of Elites. He used to be a lecturer in Tafresh University and serve as a researcher in National Elites Foundation. Amir H. Gandomi has published over 70 journal papers and several discussion papers, conference papers and book chapters. He has two patents and has published three books in Elsevier. His research interests are Metaheuristics modeling and optimization.

Xin-She Yang is a Reader at Middlesex University, UK and adjunct Professor at Reykjavik University, Iceland. He is also a Distinguished Visiting Professor at Xi'an Polytechnic University and Harbin Engineering University, China. He has published 15 books and more than 200 peer-reviewed papers. He is the IEEE CIS task force chair for Business Intelligence and Knowledge Management, the President of International Consortium for Optimization and Modelling in Science and Industry (iCOMSI), and the Editor-in-Chief of International Journal of Mathematical Modelling and Numerical Optimisation (IJMMNO).