DOI 10.1007/s13042-017-0739-8

ORIGINAL ARTICLE

Bat algorithm with triangle-flipping strategy for numerical optimization

Xingjuan Cai[1] · Hui Wang[2] · Zhihua Cui[1] · Jianghui Cai[1] · Yu Xue[3] · Lei Wang[4]

Received: 15 June 2017 / Accepted: 27 October 2017 / Published online: 13 November 2017

Abstract Bat algorithm (BA) is a novel population-based evolutionary algorithm inspired by echolocation behavior. Due to its simple concept, BA has been widely applied to various engineering applications. As an optimization approach, the global search characteristics determine the optimization performance and convergence speed. In BA, the global search capability is dominated by the velocity updating. How to update the velocity of bats may seriously affect the performance of BA. In this paper, we propose a triangle-flipping strategy to update the velocity of bats. Three different triangle-flipping strategies with five different designs are introduced. The optimization performance is verified by CEC2013 benchmarks in those designs against the standard BA. Simulation results show that the hybrid triangle-flipping strategy is superior to other algorithms.

Keywords Bat algorithm · Random triangle-flipping strategy · Directing triangle-flipping strategy · Hybrid triangle-flipping strategy

1 Introduction

Bio-inspired computation [1] is a collection for stochastic optimization algorithms inspired by biological phenomenon, such as particle swarm optimization [2, 3], brain storm optimisation [4], ant colony optimization [5, 6], fruit fly optimization algorithm (FOA) [7], elephant herding optimization [8], artificial physics optimization [9], social emotional optimisation algorithm [10], cuckoo search [11–14], firefly algorithm (FA) [15–18], and artificial bee colony (ABC) [19–21].

Bat algorithm (BA) [22] is a novel population-based stochastic bio-inspired computation inspired by bat behaviors. In the standard BA, bats fly in the search domain to seek food and avoid the emergent dangers from other bats, especially for the largest bat. Since the development of BA, different BA variants have been proposed to improve the performance. Li and Zhou [23] proposed a complex-valued bat algorithm where each bat is coded in complex number. The real and imaginary parts are updated as the same as the standard version. Due to this two-dimensional characteristic, the diversity and performance are both improved significantly. Saha et al. [24] introduced opposition-based learning (OBL) strategy into bat algorithm. The opposite position of each bat is investigated, then, the next population

✉ Zhihua Cui
zhihua.cui@hotmail.com

Xingjuan Cai
xingjuancai@163.com

Hui Wang
huiwang@whu.edu.cn

Jianghui Cai
jianghui@tyust.edu.cn

Yu Xue
xueyu_123@nuaa.edu.cn

Lei Wang
wanglei_tj@126.com

[1] School of Computer Science and Technology, Taiyuan University of Science and Technology, Taiyuan 030024, Shanxi, China

[2] Jiangxi Province Key Laboratory of Water Information Cooperative Sensing and Intelligent Processing, Nanchang Institute of Technology, Nanchang 330099, China

[3] School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 210044, China

[4] Department of Control Science and Engineering, Tongji University, Shanghai 201804, China

is selected from the current population and their opposite positions under the evaluation process of objective function. Chaos is a character of nonlinear systems and is generated randomly by simple deterministic systems. Currently, chaotic sequences are always applied to evolutionary algorithms to increase the global search capability. Gandomi and Yang [25] implemented several general chaotic sequences in BA, such as frequency, velocity disturbance, loudness and pulse emission rate. Similarly, Jordehi [26] also used chaos to improve the performance of BA.

To improve the local search capability, four differential evolutionary strategies [27, 28]: DE/rand/1/bin, DE/randToBest/1/bin, DE/best/2/bin and DE/best/1/bin were employed to replace the original local search pattern in the standard BA [29]. To enhance the local search region, the historical best position is replaced by one position obtained with optimal forage strategy [30].

Different from the local search strategy, the global search capability is dominated by the velocity update equation, which is divided into two different parts: inertia and avoidance. The inertia is employed to describe the influence of the previous velocity, and the avoidance is used to collide with the largest bat. Generally, the velocity update equation plays an important role in directing the search for each bat. Many improvements aimed to change the current version. Xie et al. [31] used a random part associated with Lévy distribution to replace the avoidance. Khan et al. [32] added a new item associated with personal historical best position in the velocity update equation to improve the convergence speed. Furthermore, Bahmani-Firouzi and Azizipanah-Abarghooee designed four different velocity updating strategies to provide a balance between exploitation and exploration [33]. The swarm historical best position also plays an important role for many algorithms, for example, Xue et al. [34] incorporated this position to improve the performance of articial bee colony algorithm for global optimization. Similarly, Jaddi et al. [35] recognized the personal historical best position and the swarm historical best position as exploitation and exploration, respectively. Then, a different balance strategy was proposed to weight the two positions. Inspired by PSO [36, 37], the inertial weight was introduced into the velocity update equation [38, 39].The inertia influence can be controlled when the bats fall into local minima. With the control theory, the velocity update equation with inertia weight can be viewed as a special case of oscillation element, and one selection strategy is designed [40]. Furthermore, Yilmaz and Kucuksille [41] added a random item with two randomly selected bats to explore more search space.

Different from the inertia weight, some researchers suggest removing the inertia part so that the bats can fly with more selections. To increase the search space, the frequency is replaced by a random number generated with Lévy distribution [42]. Xie et al. [43] also incorporated the Lévy flight in the velocity update equation, but four randomly selected bats were used to guide the search pattern. Zhu et al. [44] replace the swarm historical best position with the mean best position to enhance the convergence speed.

Due to its simple concept, BA has been widely applied to many problems. Networks in real world more or less have overlapping community structure [45], however, Ma et al. [46] found that traditional community detection algorithms assumed that one vertex can only belong to one community. To solve this problem, Hassan et al. [47] designed a discrete bat algorithm to solve it, and experiments on real life networks show the ability of the proposed algorithm. Satellite images are a reliable source for investigating the temporal changes in crop cultivated areas, Senthilnath et al. [48] proposed a novel bat algorithm (BA)-based clustering approach for solving crop type classification problems using a multispectral satellite image. There are still many applications, such as visual tracking [49], distribution feeder reconfiguration [50], and redundancy allocation problem [51], due to the page limitation, please refer to the corresponding references.

Although there are many modifications for velocity update equations, however, there is still some room for it. In this paper, we provide several different velocity update equations to improve the performance.

The rest of this paper is organized as follows: Sect. 2 provides a brief description of the standard BA. Some analyses for the standard BA are illustrated in Sect. 3. Our proposed approach is presented in Sect. 4. Numerical experiments on the CEC2013 benchmark set are conducted in Sect. 5. Finally, some conclusions are given in Sect. 6.

2 Standard bat algorithm

In this paper, we only consider the following problem:

$$\min f(\vec{x}),\ [\vec{x} = (x_1, x_2, \dots, x_k, \dots, x_D) \in E], \tag{1}$$

where $E = [x_{\min}, x_{\max}]^D \subseteq R^D$ is the domain space, and $x_{\min} \leqslant x_k \leqslant x_{\max} (k = 1, 2, \dots, D)$.

Assume that there are N virtual bats, and the ith bat (i=1,2,…,N) can be represented as

$$< \vec{x}_i(t), \vec{v}_i(t), fr_i(t), A_i(t), r_i(t) >, \tag{2}$$

where $\vec{x}_i(t) = (x_{i1}(t), x_{i2}(t), ..., x_{ik}(t), ..., x_{iD}(t))$ and $\vec{v}_i(t) = (v_{i1}(t), v_{i2}(t), ..., v_{ik}(t), ..., v_{iD}(t))$ are the position and velocity of the ith bat in generation t, respectively, frequency $fr_i(t)$, loudness $A_i(t)$ and emission rate $r_i(t)$ are three required parameters.

In the next generation, the velocity is updated as follows:

$$v_{ik}(t+1) = v_{ik}(t) + (x_{ik}(t) - p_k(t)) \cdot fr_i(t), \tag{3}$$

where $\vec{p}(t) = (p_1(t), p_2(t), \dots, p_k(t), \dots, p_D(t))$ is the best position found so far by the entire swarm. Equation (3) can be viewed as a combination of the inertia part $v_{ik}(t)$ and the influence of $\vec{p}(t)$. The frequency $fr_i(t)$ is calculated as follows:

$$fr_i(t) = fr_{\min} + (fr_{\max} - fr_{\min}) \cdot rand_1, \tag{4}$$

where $fr_{\max}$ and $fr_{\min}$ are the maximum and minimum values of frequency, respectively, and $rand_1$ is a random number uniformly distributed within [0, 1].

To reflect the bat decision, the position is changed with some randomness. Let $rand_2$ be a random number uniformly distributed within [0, 1], if $rand_2 < r_i(t)$ is satisfied, the ith bat will execute the following global search pattern:

$$x'_{ik}(t+1) = x_{ik}(t) + v_{ik}(t+1). \tag{5}$$

Otherwise, the local search pattern is adopted as follows:

$$x'_{ik}(t+1) = p_k(t) + \varepsilon_{ik} \cdot \overline{A}(t), \tag{6}$$

where ε_{ik} is a random number generated by uniform distribution within [–1], $\overline{A}(t)$ is the average loudness of all bats, and

$$\overline{A}(t) = \frac{\sum_{i=1}^{n} A_i(t)}{n}. \tag{7}$$

After the $\vec{x}'_i(t+1) = (x'_{i1}(t+1), x'_{i2}(t+1), \ldots, x'_{ik}(t+1), \ldots, x'_{iD}(t+1))$ is obtained by Eqs. (5) and (6), the new $\vec{x}_i(t+1)$ is updated as follows:

$$\vec{x}_i(t+1) = \begin{cases} \vec{x}'_i(t+1), & \text{if } \ rand_3 < A_i(t) \ \wedge \ f(\vec{x}'_i(t+1)) < f(\vec{x}_i(t)) \\ \vec{x}_i(t), & \text{otherwise} \end{cases}, \tag{8}$$

where $rand_3$ is a random number generated by uniform distribution within [–1]. Similar with cuckoo search [52, 53], Eq. (8) implies that the position is updated only when the following two conditions are required: (1) a better position is obtained; and (2) the probability $A_i(t)$ is satisfied. If the position of the ith bat is updated, the corresponding loudness and emission rate $r_i(t+1)$ are replaced as follows:

$$A_i(t+1) = \alpha A_i(t), \tag{9}$$

$$r_i(t+1) = r(0) \cdot (1 - e^{-\gamma t}), \tag{10}$$

where $\alpha > 0$ and $\gamma > 0$ are two predefined parameters, and $A(0)$ and $r(0)$ are two initial values for loudness and emission rate, respectively.

The pseudo code of the standard bat algorithm is listed as follows:

Algorithm1: Standard Bat Algorithm

```
Begin
    For each bat, initialise the position, velocity and parameters;
    While (stop criterion is met)
        Randomly generate the frequency for each bat with Eq. (4);
        Update the velocity for each bat with Eq.(3);
        If $rand_2 < r_i(t)$
            Update the temp position for corresponding bat with Eq.(5);
        Else
            Update the temp position for corresponding bat with Eq.(6);
        End
        Evaluate its quality/fitness;
        Re-update the position for corresponding bat with Eq.(8);
        If the position is updated
            Update the loudness and emission rate with Eqs.(9) and (10), respectively;
        End
        Rank the bats and save the best position;
    End
    Output the best position;
End
```

3 Analysis for the standard bat algorithm

We only consider the global search pattern (Eqs. 3, (5), and will ignore the local search pattern (Eq. 6). From Eq. (8), the new position is updated when two conditions are satisfied ($rand_3 < A_i(t)$ and $f(\vec{x}'_i(t+1)) < f(\vec{x}_i(t))$). It means that some bats in the swarm are not updated in some generations. However, if the position is not updated, the velocity should be zero because of $\vec{v}_i(t) = \vec{x}_i(t) - \vec{x}_i(t-1) = 0$. Under this circumstance, the corresponding velocity and position update equations should be changed as follows:

$$v_{ik}(t+1) = (x_{ik}(t) - p_k(t)) \cdot f_i(t), \tag{11}$$

$$x_{ik}(t+1) = x_{ik}(t) + v_{ik}(t+1). \tag{12}$$

Based on Eqs. (11) and(12), we can get

$$x_{ik}(t+1) = x_{ik}(t) + (x_{ik}(t) - p_k(t)) \cdot f_i(t). \tag{13}$$

To clearly illustrate the characteristics of Eq. (13), Fig. 1 presents the possible search manners. As seen, the search range of Eq. (13) is small. This is not beneficial for global

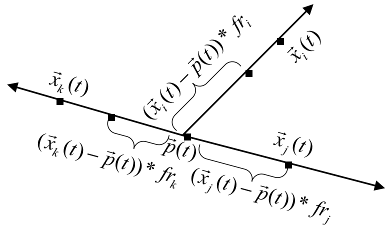

$\vec{x}_k(t)$
$(\vec{x}_i(t)-\vec{p}(t))*fr_i$
$\vec{x}_i(t)$
$\vec{p}(t)$
$\vec{x}_j(t)$
$(\vec{x}_k(t)-\vec{p}(t))*fr_k$
$(\vec{x}_j(t)-\vec{p}(t))*fr_j$

Fig. 1 Possible search manners of Eq. (13)

search. To tackle this issue, we propose a triangle-flipping strategy to extend the search range of BA.

4 Bat algorithm with triangle-flipping strategy

To avoid such problem, we can consider the following manner:

$$\vec{x}_i(t+1) = \vec{x}_i(t) + (\vec{x}_m(t) - \vec{x}_u(t)) \cdot f_i(t), \tag{14}$$

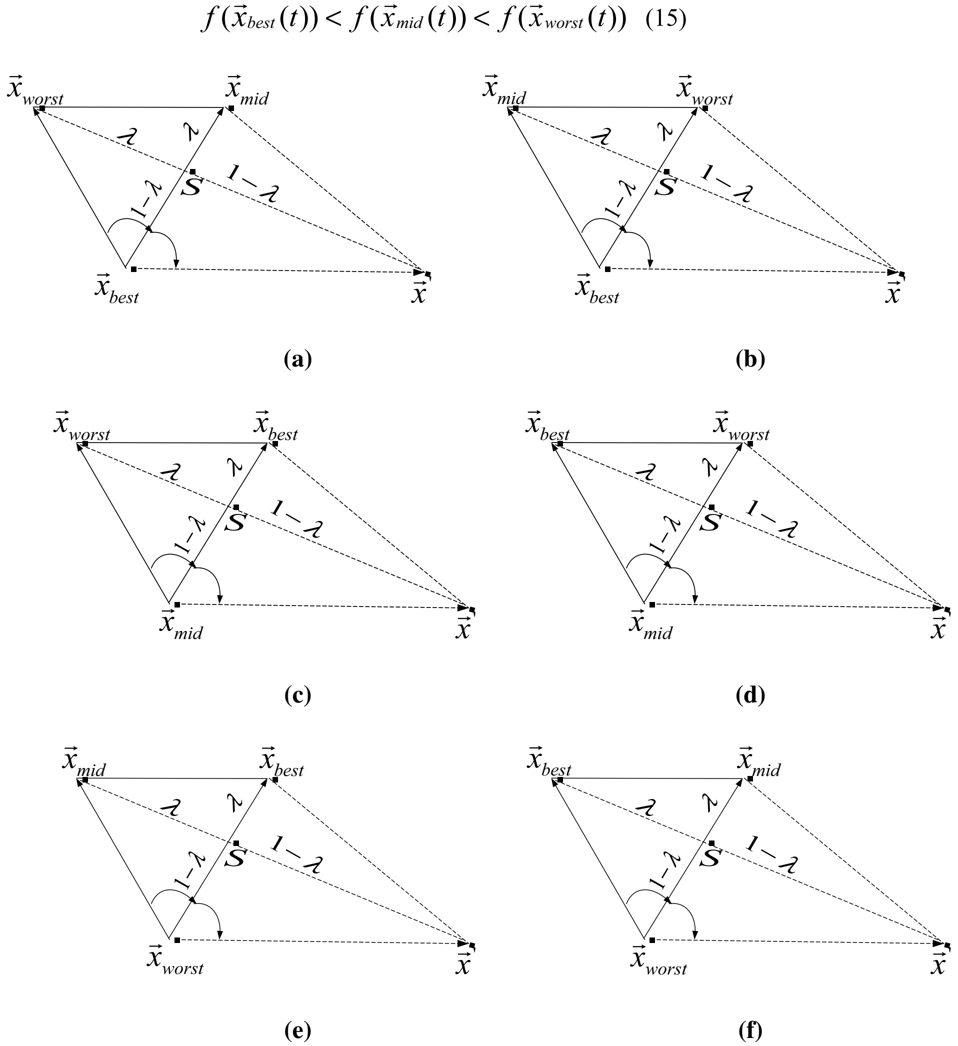
where $\vec{x}_m(t)$ and $\vec{x}_u(t)$ are two randomly selected positions. Due to $\vec{x}_i(t)$, $\vec{x}_m(t)$ and $\vec{x}_u(t)$ have different fitness values, there are six different manners presented in Fig. 2.

To avoid the confusion, three positions are re-typed as:$\vec{x}_{best}$, $\vec{x}_{mid}$ and $\vec{x}_{worst}$, and it means

$$f(\vec{x}_{best}(t)) < f(\vec{x}_{mid}(t)) < f(\vec{x}_{worst}(t)). \tag{15}$$

Figure 2a can be regarded as a heuristic search pattern, the vector $\overrightarrow{x_{best}x_{worst}}$ is turned to $\overrightarrow{x_{best}x_{mid}}$, while the final

Fig. 2 Six different manners for Eq. (14)



(a) **(b)**

(c) **(d)**

(e) **(f)**

point from $\vec{x}_{worst}$ move to $\vec{x}_{mid}$, this process improves the performance significantly because $f(\vec{x}_{mid}(t)) < f(\vec{x}_{worst}(t))$. With the same method, we can turn the circle further to next point $\vec{x}'$, but how to compute this point? For convenience, we assume rational, in other words, the following equation is satisfied:

$$(1-\lambda)\vec{x}_{mid} + \lambda\vec{x}_{best} = (1-\lambda)\vec{x}_{worst} + \lambda\vec{x}'. \tag{16}$$

It means

$$\vec{x}' = \vec{x}_{best} + \frac{1-\lambda}{\lambda}(\vec{x}_{mid} - \vec{x}_{worst}). \tag{17}$$

Suppose $f_i(t) = \frac{1-\lambda}{\lambda}$, we have

$$\vec{x}' = \vec{x}_{best} + (\vec{x}_{mid} - \vec{x}_{worst}) \cdot f_i(t). \tag{18}$$

With the same methods, Fig. 2c, e are also illustrated as:

$$\vec{x}' = \vec{x}_{mid} + (\vec{x}_{best} - \vec{x}_{worst}) \cdot f_i(t), \tag{19}$$

$$\vec{x}' = \vec{x}_{worst} + (\vec{x}_{best} - \vec{x}_{mid}) \cdot f_i(t). \tag{20}$$

As a heuristic search pattern, Eqs. (18)–(20) can be viewed as a local search pattern, while the last three figures can be viewed as the global search patterns listed as follow:

$$\vec{x}' = \vec{x}_{best} + (\vec{x}_{worst} - \vec{x}_{mid}) \cdot f_i(t), \tag{21}$$

$$\vec{x}' = \vec{x}_{mid} + (\vec{x}_{worst} - \vec{x}_{best}) \cdot f_i(t), \tag{22}$$

$$\vec{x}' = \vec{x}_{worst} + (\vec{x}_{mid} - \vec{x}_{best}) \cdot f_i(t). \tag{23}$$

1. Directing triangle-flipping strategy

It is obvious that the Eqs. (18)–(20) are meta-heuristic strategies aiming to improve the local search capability. Therefore, we call all of them (please refer to Fig. 2a, c, e) are directing triangle-flipping strategy. With this manner, $\vec{x}_m(t)$ is better than $\vec{x}_u(t)$. However, this strategy may mislead a wrong search direction, because we always want to provide a local search.

2. Random triangle-flipping strategy

For Eqs. (21)–(23), they provide an exploration capability. Therefore, to avoid the premature convergence, $\vec{x}_m(t)$ and $\vec{x}_u(t)$ are two randomly selected positions, and we call it random triangle-flipping strategy.

3. Hybrid triangle-flipping strategy

Generally, for an evolutionary algorithm, the global search capability and local search capability should be balanced dynamically. In the first period, the algorithm mainly focuses on exploring the potential area with more probability to contain the optimal solutions, while in the late period, the algorithm prefers to improve the solution quality. With this manner, we divide the total search period into two sections:

If $t < \lambda \cdot L\arg est\_Generation$

The random triangle-flipping strategy is performed;

Else

One special directing triangle-flipping strategy is performed as follows:

$$\vec{v}_i(t+1) = (\vec{p}(t) - \vec{x}_m(t)) \cdot fr_i(t), \tag{24}$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1). \tag{25}$$

End

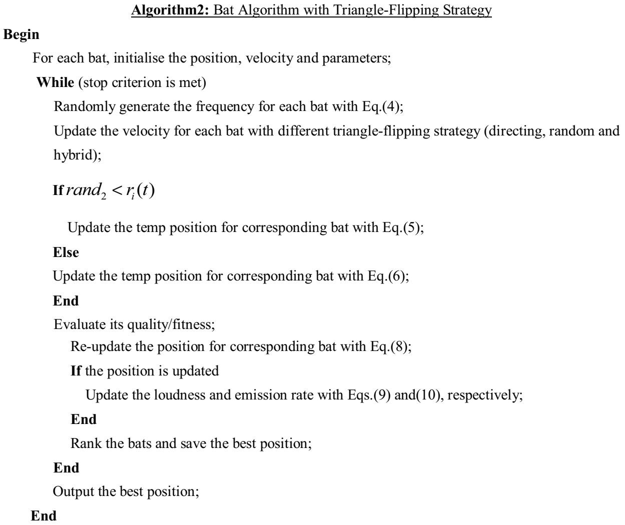
where λ is a threshold parameter used to divide the total generation. With this manner, only one random position $\vec{x}_m(t)$ should be required.

Note 1 In above mentioned three triangle-flipping strategies, all of them will occur in those bats with their positions which are not updated in previous generation. For other bats, their velocities and positions are still updated by Eqs. (3) and (5). Is it possible for all bats to update their velocities and positions with triangle-flipping strategies? In the following experiment, we will consider such cases.

Note 2 Due to the performance of the global historical best position $\vec{p}(t)$, the triangle-flipping manners will provide a global search manner because only Eqs. (18)–(20) are satisfied. Therefore, to provide a large exploration capability, the following manner is considered:

$$x_{gk}(t+1) = x_{\min} + (x_{\max} - x_{\min}) \cdot rand, \tag{26}$$

where *rand* is a random number generated by uniform distribution.



```
Algorithm2: Bat Algorithm with Triangle-Flipping Strategy
Begin
    For each bat, initialise the position, velocity and parameters;
    While (stop criterion is met)
        Randomly generate the frequency for each bat with Eq.(4);
        Update the velocity for each bat with different triangle-flipping strategy (directing, random and
        hybrid);
        If $rand_2 < r_i(t)$
            Update the temp position for corresponding bat with Eq.(5);
        Else
        Update the temp position for corresponding bat with Eq.(6);
        End
        Evaluate its quality/fitness;
            Re-update the position for corresponding bat with Eq.(8);
            If the position is updated
                Update the loudness and emission rate with Eqs.(9) and(10), respectively;
            End
            Rank the bats and save the best position;
        End
        Output the best position;
    End
```

For a given problem *f*, we assume that *O*(*f*) is the computational complexity of its fitness evaluation function. Therefore, the computational complexity of the standard BA is $O(G_{max} \times N \times f)$, where G_{max} is the maximum number of generations, and *N* is the population size. Compared to the standard BA, our approach does not increase extra loop operations. So, both of them have the same computational time complexity.

5 Numerical comparison

The simulation is performed on the CEC2013 benchmark set [54], which contains 28 functions for real-parameter optimization (please refer to Table 1). These functions can be divided into three groups:uni-modal functions (F1–F5), multi-modal functions (F6–F20), and composition functions (F21–F28).

The simulation experiments is performed in Matlab 2011 environment. The population size, the maximum number of fitness evaluations, and the dimensional size are set to 100, 3.0E+05, 30, respectively. For each test function, each algorithm will run 51 times.

5.1 Determine of threshold parameter

To test the performance of our proposed hybrid triangle-flipping strategy, firstly, we focus on the threshold parameter value of λ. The parameters used in bat algorithm with hybrid triangle-flipping strategy are listed in Table 2.

To provide a deep insight, the Golden section method is employed. The initial interval is [0.0,1.0], and 13 round experiments are conducted. The rankings achieved by the Friedman test show the corresponding index (the smallest ranking value means the best performance of the corresponding algorithm) [41]. Table 3 presents the ranking values under different values of λ. It can be seen that the final threshold parameter λ is obtained by the 13th round test and the best λ is equal to 0.258.

5.2 Comparison of different triangle-flipping strategies

In this section, we will compare the performance of different triangle-flipping strategies:

- Bat algorithm with directing triangle-flipping strategy (BA-DTFS);
- Bat algorithm with directing triangle-flipping strategy for all bats without conditions (see Note 1) (BA-DTFS1);
- Bat algorithm with random triangle-flipping strategy (BA-RTFS);
- Bat algorithm with random triangle-flipping strategy for all bats without conditions(BA-RTFS1);
- Bat algorithm with hybrid triangle-flipping strategy (BA-HTFS);

Table 4 provides the mean error function values achieved by BA-DTFS, BA-DTFS1, BA-RTFS, BA-RTFS1 and BA-HTFS, where *w*/*t*/*l* means that BA-HTFS wins in *w* functions, ties in *t* functions, and loses in *l* functions, compared with its competitors. From the results, BA-HTFS performs better than BA-DTFS and BA-DTFS1 on 26 functions and 27 functions, respectively, while BA-RTFS and BA-RTFS1 outperforms BA-HFTS on 0 function and 8 functions.

For uni-modal functions (F1–F5), the hybrid triangle-flipping strategy performs better than the directing triangle-flipping strategy and the random triangle-flipping strategy. Especially for F1 and F4, BA-HTFS achieves much better solutions than other four algorithms. For multi-modal functions (F6–F20), though the hybrid triangle-flipping strategy can help BA find more accurate solutions than the directing triangle-flipping strategy and the random triangle-flipping strategy, these improvements are not obvious on most test cases. For F10 and F16, BA-HTFS can obtain reasonable solutions, while other four algorithms fall into local minima (except for BA-RTFS1 on F16). For composition functions (F21–F28), all BAs hardly search reasonable solutions, but BA-HTFS can find more accurate solutions than other four algorithms.

Table 5 presents the results of Wilcoxon test between BA-HTFS and other four algorithms [55, 56]. The *p* values for BA-DTFS, BA-DTFS1 and BA-RTFS are less than the significant level 0.05. It means the performance of BA-HTFS is significantly better than BA-DTFS, BA-DTFS1 and BA-RTFS.

Table 6 shows the mean rankings achieved by Friedman test for five algorithms [57]. A smaller ranking value means that the corresponding algorithm is better. From the results, the performances of five BAs are ranked as follows: BA-DTFS, BA-DTFS1, BA-RTFS, BA-RTFS1, and BA-HYFS.

Table 1 Summary of the CEC2013 benchmark set

	No.	Function	Optimal fitness
Uni-modal functions	1	Sphere function	–1400
	2	Rotated high conditioned elliptic function	–1300
	3	Rotated Bent Cigar function	–1200
	4	Rotated discus function	–1100
	5	Different powers function	–1000
Basic multi-modal functions	6	Rotated Rosenbrock's function	–900
	7	Rotated Schaffers F7 function	–800
	8	Rotated Ackley's function	–700
	9	Rotated Weierstrass function	–600
	10	Rotated Griewank's function	–500
	11	Rastrigin's function	–400
	12	Rotated Rastrigin's function	–300
	13	Non-continuous rotated Rastrigin's function	–200
	14	Schwefel's function	–100
	15	Rotated Schwefel's function	100
	16	Rotated Katsuura function	200
	17	LunacekBi_Rastrigin function	300
	18	Rotated LunacekBi_Rastrigin function	400
	19	Expanded Griewank'splusRosenbrock's function	500
	20	Expanded Scaffer's F6 function	600
Composition functions	21	Composition function 1 ($n=5$, rotated)	700
	22	Composition function 2 ($n=3$, unrotated)	800
	23	Composition function 3 ($n=3$, rotated)	900
	24	Composition function 4 ($n=3$, rotated)	1000
	25	Composition funaction 5 ($n=3$, rotated)	1100
	26	Composition function 6 ($n=5$, rotated)	1200
	27	Composition function 7 ($n=5$, rotated)	1300
	28	Composition function 8 ($n=5$, rotated)	1400

Table 2 Parameter settings for bat algorithm

Search Domain	$[-100, 100]^D$
Frequency	[0.0, 5.0]
$A(0)$	0.95
$r(0)$	0.9
α	0.99
γ	0.9

The highest ranking demonstrates that BA-HTFS is the best algorithm among five algorithms.

5.3 Comparison of BA-HTFS with other bat algorithms

To further verify the performance of BA-HTFS, we compare BA-HTFS with four other algorithms. The involved algorithms are listed as follows.

- Standard bat algorithm (SBA);
- Chaotic bat algorithm (CBA) [25];
- Bat algorithm with lévy distribution (LBA1) [31];
- Bat algorithm with lévy distribution (LBA2) [42];
- Bat algorithm with hybrid triangle-flipping strategy (BA-HTFS).

For SBA, CBA, LBA1 and LBA2, we use the same parameter settings as in their corresponding references. Table 7 provides the mean error function values achieved by SBA, CBA, LBA1, LBA2 and BA-HTFS (the dynamic comparison can be viewed in Fig. 3). From the results, BA-HTFS performs better than SBA and CBA on 26 functions and 22 functions, while LBA1 and LBA2 outperforms BA-HTFS on 4 functions and 5 functions, respectively.

Table 3 Experimental results for threshold $\lambda$with golden section

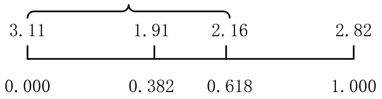
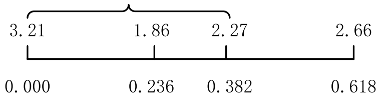
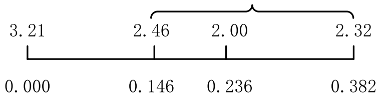
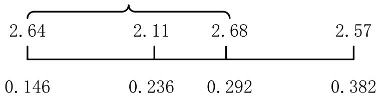
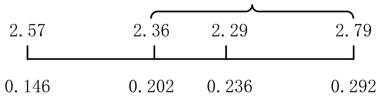
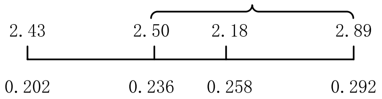
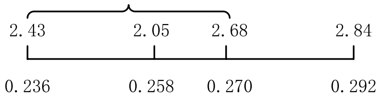
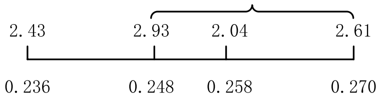
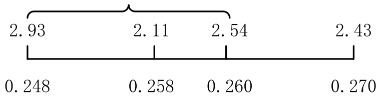
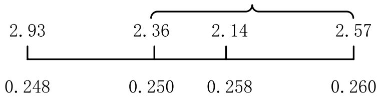
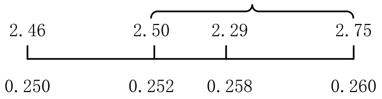
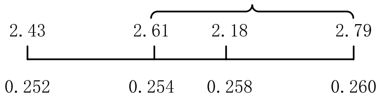
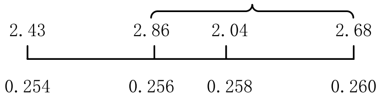
Round	Value	Rankings	Figure
1	0.000 **0.382** 0.618 1.000	3.11 **1.91** 2.16 2.82	 3. 11 1. 91 2. 16 2. 82 0. 000 0. 382 0. 618 1. 000
2	0.000 **0.236** 0.382 0.618	3.21 **1.86** 2.27 2.66	 3. 21 1. 86 2. 27 2. 66 0. 000 0. 236 0. 382 0. 618
3	0.000 0.146 **0.236** 0.382	3.21 2.46 **2.00** 2.32	 3. 21 2. 46 2. 00 2. 32 0. 000 0. 146 0. 236 0. 382
4	0.146 **0.236** 0.292 0.382	2.64 **2.11** 2.68 2.57	 2. 64 2. 11 2. 68 2. 57 0. 146 0. 236 0. 292 0. 382
5	0.146 0.202 **0.236** 0.292	2.57 2.36 **2.29** 2.79	 2. 57 2. 36 2. 29 2. 79 0. 146 0. 202 0. 236 0. 292
6	0.202 0.236 **0.258** 0.292	2.43 2.50 **2.18** 2.89	 2. 43 2. 50 2. 18 2. 89 0. 202 0. 236 0. 258 0. 292
7	0.236 **0.258** 0.270 0.292	2.43 **2.05** 2.68 2.84	 2. 43 2. 05 2. 68 2. 84 0. 236 0. 258 0. 270 0. 292
8	0.236 0.248 **0.258** 0.270	2.43 2.93 **2.04** 2.61	 2. 43 2. 93 2. 04 2. 61 0. 236 0. 248 0. 258 0. 270
9	0.248 **0.258** 0.260 0.270	2.93 **2.11** 2.54 2.43	 2. 93 2. 11 2. 54 2. 43 0. 248 0. 258 0. 260 0. 270
10	0.248 0.250 **0.258** 0.260	2.93 2.36 **2.14** 2.57	 2. 93 2. 36 2. 14 2. 57 0. 248 0. 250 0. 258 0. 260
11	0.250 0.252 **0.258** 0.260	2.46 2.50 **2.29** 2.75	 2. 46 2. 50 2. 29 2. 75 0. 250 0. 252 0. 258 0. 260
12	0.252 0.254 **0.258** 0.260	2.43 2.61 **2.18** 2.79	 2. 43 2. 61 2. 18 2. 79 0. 252 0. 254 0. 258 0. 260
13	0.254 0.256 **0.258** 0.260	2.43 2.86 **2.04** 2.68	 2. 43 2. 86 2. 04 2. 68 0. 254 0. 256 0. 258 0. 260

Table 4 Comparison results for different triangle-flipping strategies

Functions	BA-DTFS	BA-DTFS1	BA-RTFS	BA-RTFS1	BA-HTFS
F1	1.80E+00	1.90E+00	2.38E+00	6.38E−03	9.63E−05
F2	1.64E+06	1.68E+06	1.31E+06	7.16E+04	4.22E+04
F3	3.57E+08	2.77E+08	1.70E+08	2.12E+07	7.75E+06
F4	2.16E+04	1.35E+04	8.51E+02	1.97E+00	8.90E−02
F5	3.54E−01	3.68E−01	6.76E−01	1.27E−02	1.11E−03
F6	4.78E+01	6.12E+01	4.90E+01	1.91E+01	6.72E+00
F7	3.34E+02	1.79E+04	2.13E+02	1.06E+02	1.06E+02
F8	2.09E+01	2.09E+01	2.09E+01	2.10E+01	2.09E+01
F9	3.55E+01	3.65E+01	3.53E+01	2.95E+01	2.98E+01
F10	1.17E+00	1.20E+00	1.32E+00	5.25E−01	7.79E−02
F11	6.83E+02	6.24E+02	4.64E+02	3.74E+02	3.45E+02
F12	6.55E+02	6.33E+02	4.75E+02	3.69E+02	3.52E+02
F13	6.32E+02	7.25E+02	5.18E+02	3.44E+02	3.58E+02
F14	4.74E+03	4.70E+03	4.96E+03	4.50E+03	4.27E+03
F15	4.80E+03	4.77E+03	4.78E+03	4.34E+03	4.44E+03
F16	2.11E+00	2.16E+00	2.04E+00	9.75E−01	3.50E−01
F17	4.60E+02	7.70E+02	4.22E+02	2.12E+02	1.72E+02
F18	4.19E+02	7.61E+02	3.92E+02	2.06E+02	1.41E+02
F19	2.92E+01	5.81E+01	2.35E+01	9.86E+00	7.14E+00
F20	1.46E+01	1.47E+01	1.46E+01	1.31E+01	1.26E+01
F21	3.90E+02	3.58E+02	3.37E+02	3.40E+02	3.33E+02
F22	6.00E+03	5.79E+03	5.93E+03	5.40E+03	5.49E+03
F23	6.09E+03	5.79E+03	6.10E+03	5.60E+03	5.55E+03
F24	3.35E+02	3.52E+02	3.18E+02	2.93E+02	2.95E+02
F25	3.52E+02	3.42E+02	3.48E+02	3.24E+02	3.27E+02
F26	2.04E+02	2.08E+02	2.18E+02	2.03E+02	2.04E+02
F27	1.34E+03	1.41E+03	1.32E+03	1.16E+03	1.13E+03
F28	4.84E+03	4.77E+03	3.91E+03	1.94E+03	2.25E+03
w/t/l	26/2/0	27/1/0	27/1/0	19/1/8	

Table 5 Wilcoxon Test for different triangle-flipping strategies

BA-HTFS vs.	*p* value
BA-DTFS	**0.000**
BA-DTFS1	**0.000**
BA-RTFS	**0.000**
BA-RTFS1	0.093

Table 6 Friedman test for different triangle-flipping strategies

Algorithm	Rankings
BA-DTFS	4.05
BA-DTFS1	4.20
BA-RTFS	3.54
BA-RTFS1	1.84
BA-HTFS	**1.38**

The standard BA can hardly find reasonable solutions on all test functions. For CBA, it only achieves promising solutions on F16. LBA1 and LBA2 obtain good solutions on two functions F1 and F5. BA-HTFS can find reasonable solutions on five functions F1, F4, F5, F10, and F16. For composition functions, all algorithms fall into local minima.

Tables 8 and 9 present the statistical results achieved by Wilcoxon and Friedman tests. From the results, the performance of the five algorithms ranks as follows: SBA, LBA1, CBA, LBA2and BA-HTFS. The highest ranking demonstrates that BA-HTFS is the best algorithm among five algorithms. The *p* values show that BA-HTFS is significantly better than SBA, CBA, LBA1 and LBA2.

Table 7 Comparison results for BA-HTFS and other four bat algorithms

Function	SBA	CBA	LBA1	LBA2	BA-HTFS
F1	1.96E+00	2.30E+00	8.42E−01	3.59E−01	9.63E−05
F2	3.69E+06	4.48E+06	3.54E+06	2.23E+06	4.22E+04
F3	3.44E+08	6.71E+08	4.78E+08	3.57E+08	7.75E+06
F4	3.20E+04	3.00E+04	1.45E+04	6.85E+03	8.90E−02
F5	5.86E−01	1.73E+00	4.74E−01	2.76E−01	1.11E−03
F6	5.63E+01	6.29E+01	5.07E+01	4.85E+01	6.72E+00
F7	2.16E+02	2.42E+02	1.77E+02	2.00E+02	1.06E+02
F8	2.09E+01	2.10E+01	2.09E+01	2.10E+01	2.09E+01
F9	3.57E+01	3.52E+01	3.40E+01	3.62E+01	2.98E+01
F10	1.32E+00	1.48E+00	1.23E+00	1.07E+00	7.79E-02
F11	4.07E+02	4.27E+02	1.49E+02	3.16E+01	3.45E+02
F12	4.06E+02	4.30E+02	7.42E+02	7.18E+02	3.52E+02
F13	4.37E+02	4.36E+02	5.59E+02	5.11E+02	3.58E+02
F14	4.78E+03	2.62E+03	3.17E+03	1.15E+03	4.27E+03
F15	4.89E+03	3.87E+03	4.76E+03	4.83E+03	4.44E+03
F16	2.16E+00	6.06E−01	1.33E+00	1.54E+00	3.50E−01
F17	8.92E+02	2.74E+02	3.36E+02	1.61E+02	1.72E+02
F18	9.44E+02	2.61E+02	3.28E+02	3.35E+02	1.41E+02
F19	6.07E+01	4.35E+01	1.89E+01	1.28E+01	7.14E+00
F20	1.44E+01	1.44E+01	1.47E+01	1.49E+01	1.26E+01
F21	3.38E+02	3.27E+02	3.22E+02	3.05E+02	3.33E+02
F22	5.94E+03	3.15E+03	3.32E+03	1.20E+03	5.49E+03
F23	5.77E+03	5.03E+03	6.03E+03	5.82E+03	5.55E+03
F24	3.15E+02	2.92E+02	3.22E+02	3.23E+02	2.95E+02
F25	3.49E+02	3.32E+02	3.53E+02	3.54E+02	3.27E+02
F26	2.00E+02	2.83E+02	3.54E+02	3.36E+02	2.04E+02
F27	1.28E+03	1.19E+03	1.33E+03	1.35E+03	1.13E+03
F28	3.42E+03	2.89E+03	4.68E+03	4.34E+03	2.25E+03
w/t/l	26/1/1	22/0/6	23/1/4	23/0/5	

6 Conclusions and future work

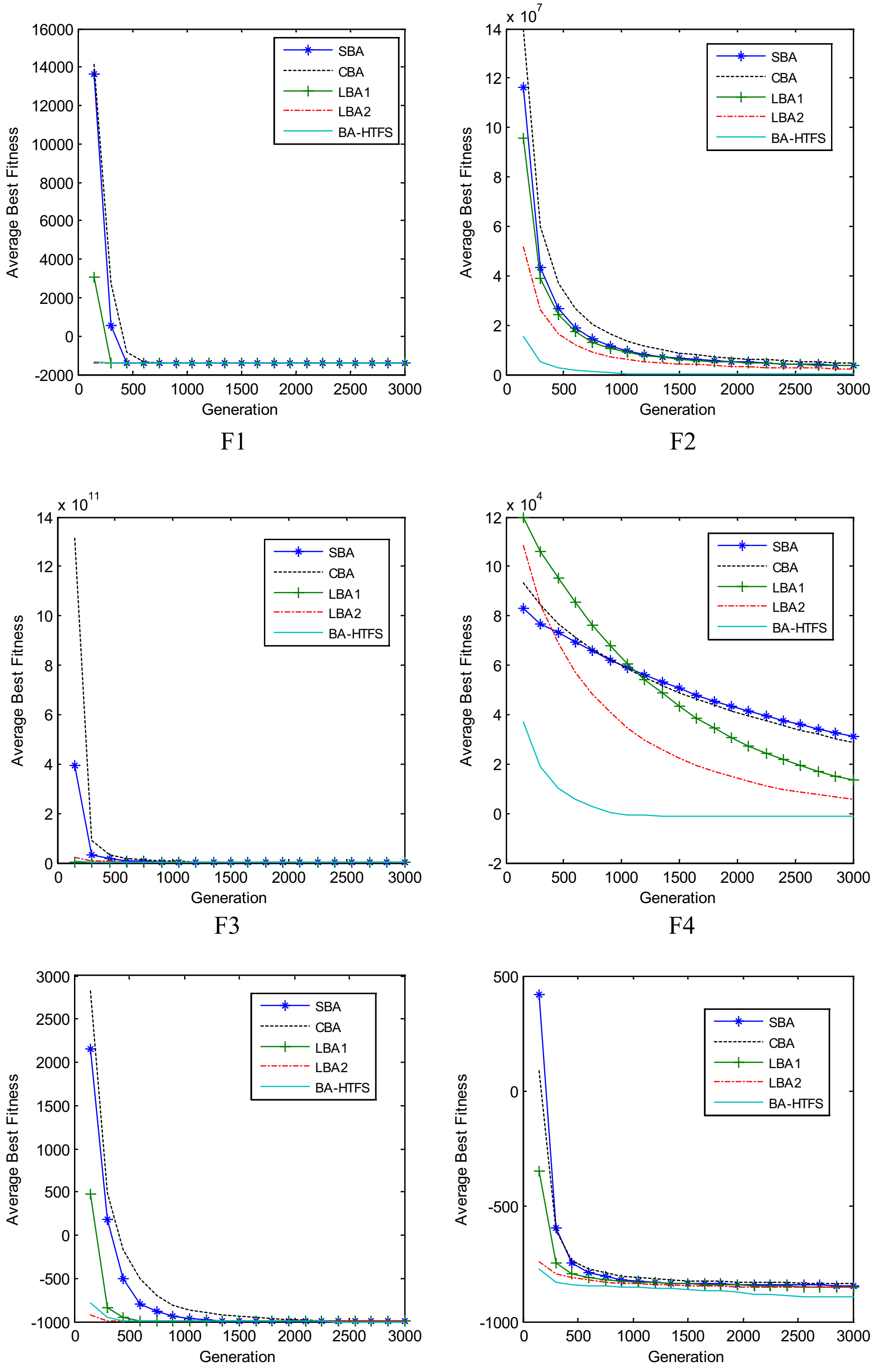
Swarm intelligence is a phenomenon to describe the emergent behaviors for some species. How to utilize this emergent intelligence is one crucial problem for swarm intelligence. Different from particle swarm optimization and ant colony optimization, BA can be viewed as a new attempt. In BA, the position is updated only when a better position is found. Under this rule, the bats will be suspended when a better position is not found. To tackle this issue, this paper proposes several triangle-flipping strategies to improve the performance of BA. Simulation results show the hybrid triangle-flipping strategy is superior to other algorithms.

Future research topic includes self-adaptive search strategy and other practical applications.

In this paper, bat algorithm with hybrid triangle-flipping strategy achieves the best performance, this phenomenon implies that how to hybrid the different triangle-flipping strategies may influence the performance significantly. According to the No Free Lunch Theorem, the self-adaptive hybrid strategy should be considered to further improve the robustness.

How to apply BA to solve some hot research topics is very important. For example, the Internet of Things is a network in cyber-physical-social space connected with massive number of physical devices, sensors and buildings. There are many optimization problems required to be handled. Wang et al. [58] designed a back propagation neural network model to establish the relationship between solar radiation signal and air temperature error. Zhang et al. [59] proposed optimal cluster-based mechanisms for energy conservation with multiple mobile sinks. For Barrier coverage, relocation of sensors with minimum number of mobile sensors and formation of k-barrier coverage with minimum energy cost were optimized [60]. How to ensure the security of data sharing within a group and how to efficiently share the outsourced data in a group manner are formidable challenges? By taking advantage of the symmetric balanced incomplete block design, Shen et al. [61] presented a novel block design-based key agreement protocol that supports multiple participants. In our future work, how to modify BA to optimize these problems will be considered.

Fig. 3 The convergence curves of different algorithms on the benchmark set



F1

F2

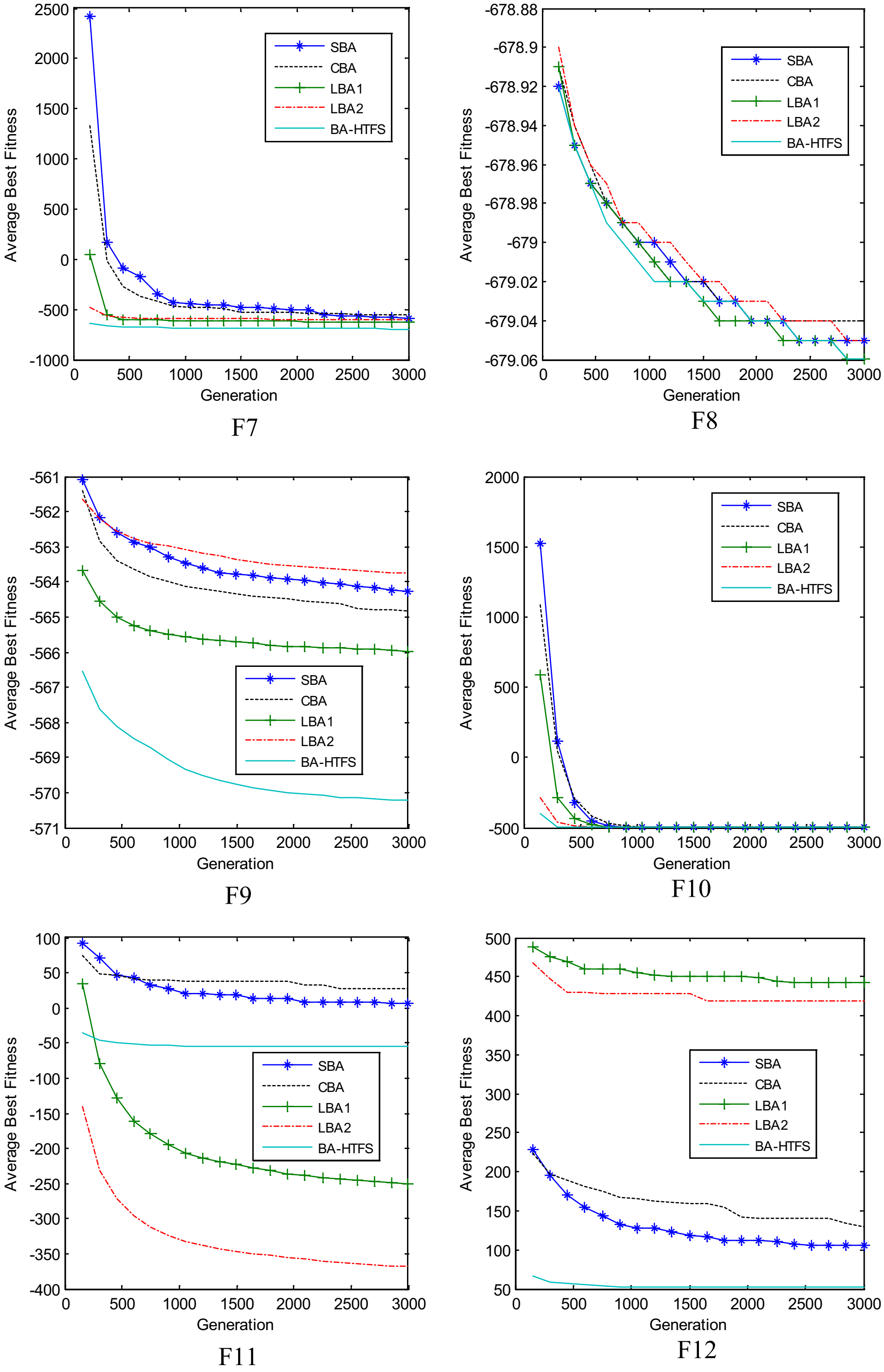
F3

F4

F5

F6

Fig. 3 (continued)



F7

F8

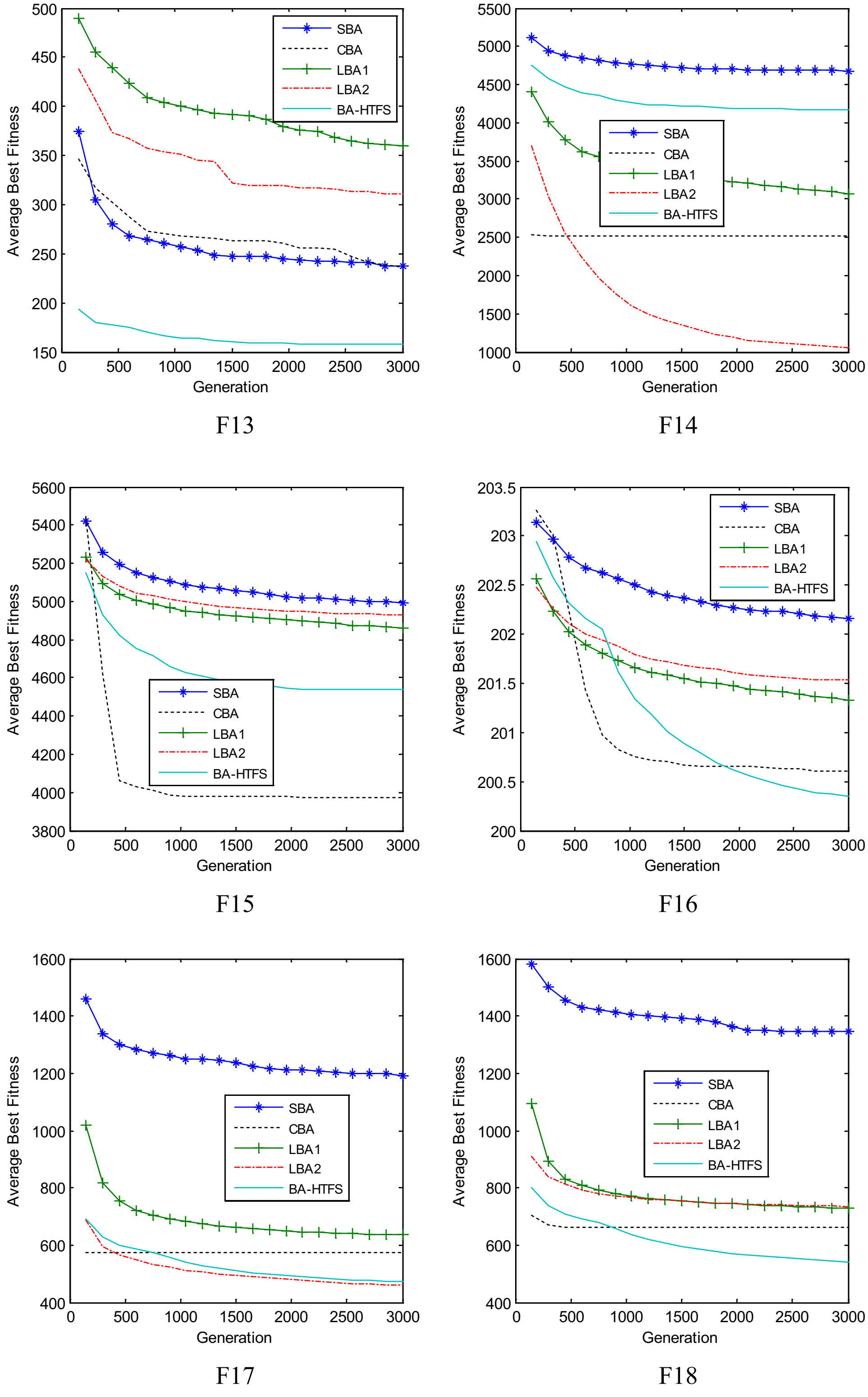
F9

F10

F11

F12

Fig. 3 (continued)



F13

F14

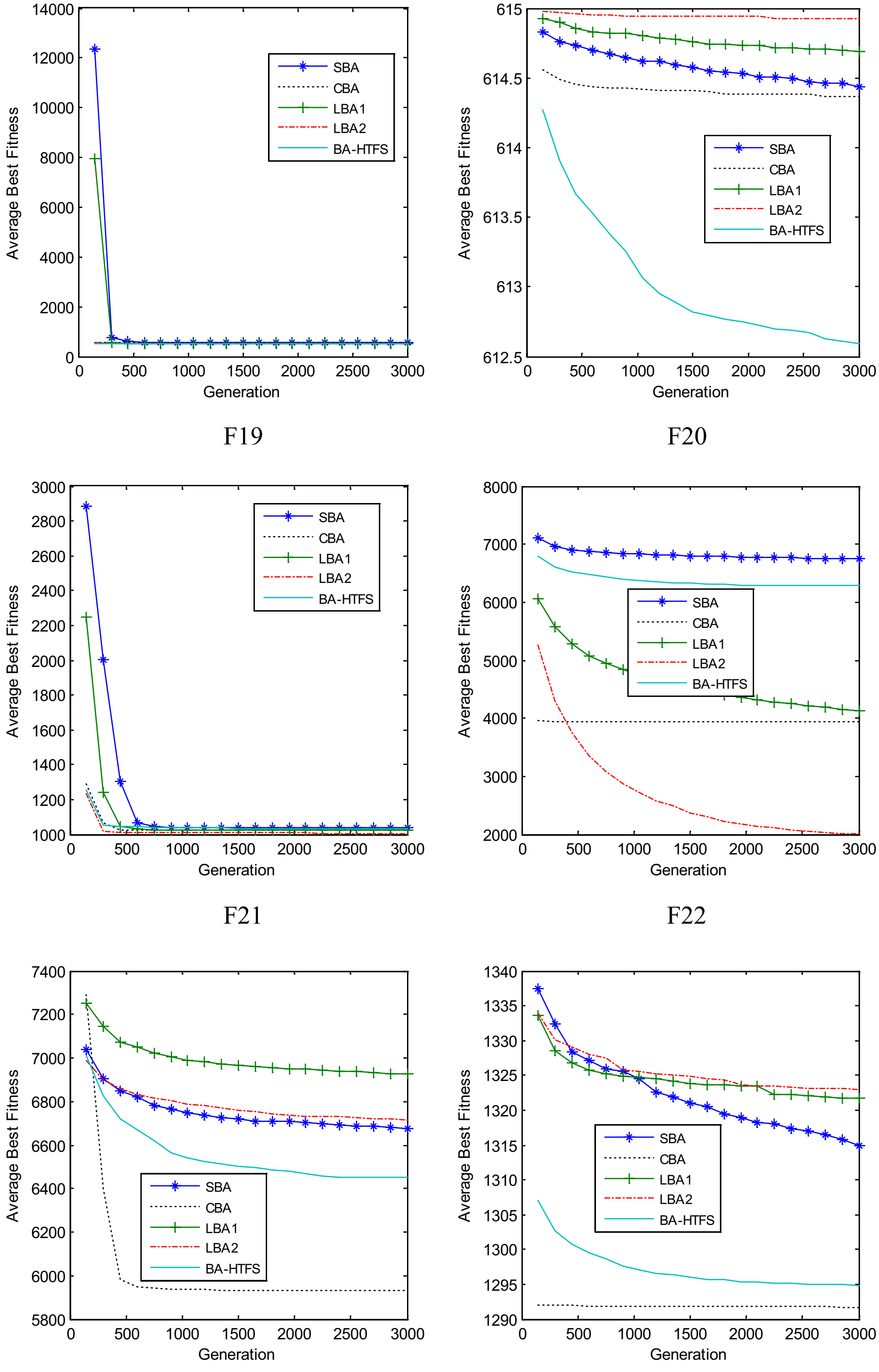
F15

F16

F17

F18

Fig. 3 (continued)

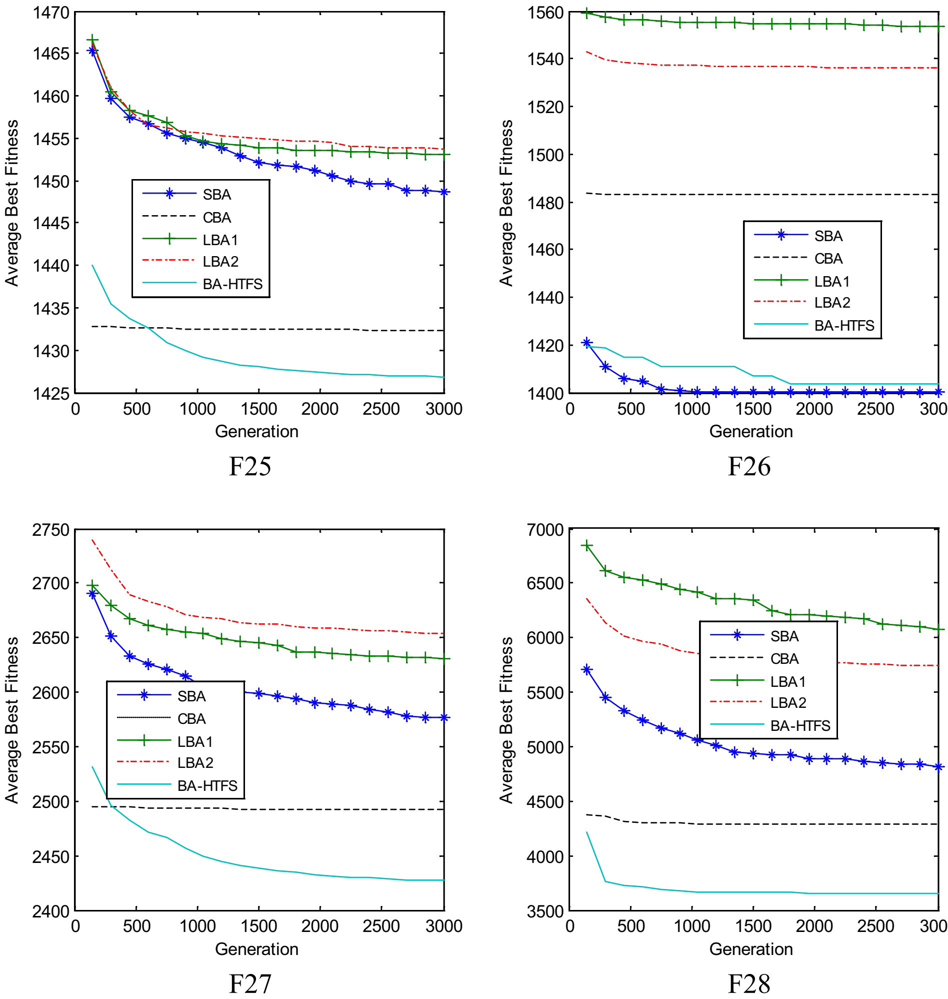


F19 F20

F21 F22

F23 F24

Fig. 3 (continued)



F25

F26

F27

F28

Table 8 Wilcoxon test for five bat algorithms

BA-HTFS vs.	*p* value
SBA	**0.000**
CBA	**0.032**
LBA1	**0.004**
LBA2	**0.010**

Table 9 Friedman test for five bat algorithms

Algorithm	Rankings
SBA	3.73
CBA	3.18
LBA1	3.39
LBA2	3.09
BA-HTFS	**1.61**

Acknowledgements This work is supported by the National Natural Science Foundation of China under no. 61663028, Natural Science Foundation of Shanxi Province under Grant no. 201601D011045, and International Science & Technology Cooperation Program of China under Grant no. 2014DFR70280.

References

1. Yang XS, Cui ZH, Xiao RB, Gandomi AH, Karamanoglu M (2013) Swarm intelligence and bio-inspired computation: theory and applications. Elsevier, London
2. Eberhart RC, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings of the sixth international symposium on micromachine and human science, Nagoya, Japan, pp 39–43
3. Wang H, Sun H, Li CH, Rahnamayan S, Pan JS (2013) Diversity enhanced particle swarm optimization with neighborhood search. Inf Sci 223:119–135
4. Jia Z, Duan H, Shi Y (2016) Hybrid brain storm optimisation and simulated annealing algorithm for continuous optimisation problems. Int J Bio Inspir Comput 8(2):109–121
5. Dorigo M (1992) Optimization, learning and natural algorithms, PhD thesis, Politecnico di Milano, Italy

6. Stodola P, Mazal J (2016) Applying the ant colony optimisation algorithm to the capacitated multi-depot vehicle routing problem. Int J Bio Inspir Comput 8(4):228–233
7. Zhang YW, Wu JT, Guo X, Li GN (2016) Optimising web service composition based on differential fruit fly optimisation algorithm. Int J Comput Sci Math 7(1):87–101
8. Wang GG, Deb S, Gao XZ, Coelho LdS (2016) A new metaheuristic optimization algorithm motivated by elephant herding behavior. Int J Bio Inspir Comput 8(6):394–409
9. Yang GJ, Zhang XL (2016) Application of extended artificial physics optimisation in product colour harmony design. Int J Comput Sci Math 7(4):350–360
10. Guo ZL, Wang SW, Yue XZ (2016) Enhanced social emotional optimisation algorithm with elite multi-parent crossover. Int J Comput Sci Math 7(6):568–574
11. Yang XS, Deb S (2010) Cuckoo search via Levy flights. In: Proceedings of world congress on nature and biologically inspired computing, India, pp 210–214
12. Cui ZH, Sun B, Wang GG, Xue Y, Chen JJ (2017) A novel oriented cuckoo search algorithm to improve DV-Hop performance for cyber-physical systems. J Parallel Distrib Comput 103:42–52
13. Wang GG, Gandomi AH, Yang XS, Alavi AH (2016) A new hybrid method based on krill herd and cuckoo search for global optimization tasks. Int J Bio Inspir Comput 8(5):286–299
14. Zhang MQ, Wang H, Cui ZH, Chen JJ (2017) Hybrid multi-objective cuckoo search with dynamical local search. Memet Comput. https://doi.org/10.1007/s12293-017-0237-2
15. Wang H, Wang W, Sun H, Rahnamayan S (2016) Firefly algorithm with random attraction. Int J Bio Inspir Comput 8(1):33–41
16. Wang H, Wang WJ, Zhou XY, Sun H, Zhao J, Yu X, Cui ZH (2017) Firefly algorithm with neighborhood attraction. Inf Sci 282/283:374–387
17. Gálvez A, Iglesias A (2016) New memetic self-adaptive firefly algorithm for continuous optimisation. Int J Bio Inspir Comput 8(5):300–317
18. Yu G (2016) An improved firefly algorithm based on probabilistic attraction. Int J Comput Sci Math 7(6):530–536
19. Wang H, Wu ZJ, Rahnamayan S, Sun H, Liu Y, Pan JS (2014) Multi-strategy ensemble artificial bee colony algorithm. Inf Sci 279:587–603
20. Lu Y, Li RX, Li SM (2016) Artificial bee colony with bidirectional search. Int J Comput Sci Math 7(6):586–593
21. Yu G (2016) A new multi-population-based artificial bee colony for numerical optimization. Int J Comput Sci Math 7(6):509–515
22. Yang XS (2010) A new metaheuristic bat-inspired algorithm. In: International workshop on nature inspired cooperative strategies for optimization. Granada, Spain, pp 65–74
23. Li LL, Zhou YQ (2014) A novel complex-valued bat algorithm. Neural Comput Appl 25(6):1369–1381
24. Saha SK, Kar R, Mandal D, Ghoshal SP, Mukherjee V (2013) A new design method using opposition-based BAT algorithm for IIR system identification problem. Int J Bio Inspir Comput 5(2):99–132
25. Gandomi AH, Yang XS (2014) Chaotic bat algorithm. J Comput Sci 5:224–232
26. Jordehi AR (2015) Chaotic bat swarm optimization (CBSO). Appl Soft Comput 26:523–530
27. Xu ZX, Unveren A, Acan A (2016) Probability collectives hybridised with differential evolution for global optimisation. Int J Bio Inspir Comput 8(3):133–153
28. Li C, Zhou C, Li X, Dai G (2017) An improved differential evolution algorithm based on suboptimal solution mutation. Int J Comput Sci Math 8(1):28–34
29. Fister I, Fong S, Brest J, Fister I (2014) A novel hybrid self-adaptive bat algorithm. Sci World J. https://doi.org/10.1155/2014/709738 **(Article ID 709738)**
30. Cai XJ, Gao X, Xue Y (2016) Improved bat algorithm with optimal forage strategy and random disturbance strategy. Int J Bio Inspir Comput 8(4):205–214
31. Xie J, Zhou YQ, Chen H (2013) A bat algorithm based on Levy flights trajectory. Pattern Recognit Artif Intell 26(9):829–837 **(in Chinese)**
32. Khan K, Nikov A, Sahai A (2011) Fuzzy bat clustering method for ergonomic screening of office workplaces. In: Third international conference on software, services and semantic technologies S3T, Bourgas, Bulgaria, pp 59–66
33. Bahmani-Firouzi B, Azizipanah-Abarghooee R (2014) Optimal sizing of battery energy storage for micro-grid operation management using a new improved bat algorithm. Electr Power Energy Syst 5(56):42–54
34. Xue Y, Jiang JM, Zhao BP, Ma TH (2017) A self-adaptive artificial bee colony algorithm based on global best for global optimization. Soft Comput. https://doi.org/10.1007/s00500-017-2547-1
35. Jaddi NS, Abdullah S, andHamdan AR (2015) Multi-population cooperative bat algorithm-based optimization of artificial neural network model. Inf Sci 294:628–644
36. Pongchairerks P, Kachitvichyanukul V (2016) A two-level particle swarm optimisation algorithm for open-shop scheduling problem. Int J Comput Sci Math 7(6):575–585
37. Adewumi AO, Arasomwan MA (2016) On the performance of particle swarm optimisation with(out) some control parameters for global optimisation. Int J Bio Inspir Comput 8(1):14–32
38. Yılmaz S, Kucuksille EU (2013) Improved bat algorithm (IBA) on continuous optimization problems. Lect Notes Softw Eng 1(3):279–283
39. Cui ZH, Li FX, Kang Q (2015) Bat algorithm with inertia weight. In: Proceedings of Chinese automation congress, Wuhan, China, pp 92–796
40. Cai XJ, Li WZ, Kang Q, Wang L, Wu QD (2015) Bat algorithm with oscillation element. Int J Innov Comput Appl 6(3/4):171–180
41. Yilmaz S, Kucuksille EU (2015) A new modification approach on bat algorithm for solving optimization problems. Appl Soft Comput 28:259–275
42. Liu CP, Ye CM (2013) Bat algorithm with the characteristics of Levy flights. CAAI Trans Intell Syst 8(3):240–246 **(in Chinese)**
43. Xie J, Zhou YQ, Chen H (2013) A novel bat algorithm based on differential operator and Lévy flights trajectory. Comput Intell Neurosci. https://doi.org/10.1155/2013/453812 **(Article ID 453812)**
44. Zhu BL, Zhu WY, Liu ZJ, Duan QY, Cao L (2016)A novel quantum-behaved bat algorithm with mean best position directed for numerical optimization. Comput Intell Neurosci. https://doi.org/10.1155/2016/6097484 **(Article ID 6097484)**
45. Cai Q, Ma LJ, Gong MG, Tian DY (2016) A survey on network community detection based on evolutionary computation. Int J Bio Inspir Comput 8(2):84–98
46. Ma TH, Wang Y, Tang ML, Cao J, Tian Y, Al-Dhelaan A, Al-Rodhaan M (2016) LED: a fast overlapping communities detection algorithm based on structural clustering. Neurocomputing 207:488–500
47. Hassan EA, Ibrahem HA, Hassaniem AE, Fahmy AA (2015) A discrete bat algorithm for the community detection problem. In: Proceedings of the 10th international conference on hybrid artificial intelligence systems, Bilbao, Spain, pp 188–199
48. Senthilnath J, Kulkarni S, Benediktsson JA, Yang XS (2016) A novel approach for multispectral satellite image classification based on the bat algorithm. IEEE Geosci Remote Sens Lett 13(4):599–603
49. Gao ML, Shen J, Yin LJ, Liu W, Zou GF, Li HT, Fu GX (2016) A novel visual tracking method using bat algorithm. Neurocomputing 177:612–619

50. Kavousi-Fard A, Niknam T, Fotuhi-Firuzabad M (2016) A novel stochastic framework based on cloud theory and theta-modified bat algorithm to solve the distribution feeder reconfiguration. IEEE Trans Smart Grid 7(2):740–750
51. Talafuse TP, Pohl EA (2016) A bat algorithm for the redundancy allocation problem. Eng Optim 48(5):900–910
52. Li FX, Cui ZH, Sun B (2016) DV-hop localisation algorithm with DDICS. Int J Comput Sci Math 7(3):254–262
53. Lin YH, Wang LJ, Zhong YW, Zhang CP (2016) Control scaling factor of cuckoo search algorithm using learning automata. Int J Comput Sci Math 7(5):476–484
54. Liang JJ, Qu BY, Suganthan PN, Hernndez-Daz AG (2013) Problem definitions and evaluation criteria for the CEC 2013 special session and competition on real-parameter optimization. Technical Report 201212, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore
55. Sun H, Wang K, Zhao Jand Yu X (2016) Artificial bee colony algorithm with improved special centre. Int J Comput Sci Math 7(6):548–553
56. Lv L, Wu LY, Zhao J, Wang H, Wu RX, Fan TH, Hu M, Xie ZF (2016) Improved multi-strategy artificial bee colony algorithm. Int J Comput Sci Math 7(5):467–475
57. Wang H, Cui ZH, Sun H, Rahnamayan S, Yang XS (2017) Randomly attracted firefly algorithm with neighborhood search and dynamic parameter adjustment mechanism. Soft Comput 21(18):5325–5339
58. Wang BW, Gu XD, Ma L, Yan SS (2017) Temperature error correction based on BP neural network in meteorological WSN. Int J Sens Netw 23(4):265–278
59. Zhang J, Tang J, Wang TB, Chen F (2017) Energy-efficient data-gathering rendezvous algorithms with mobile sinks for wireless sensor networks. Int J Sens Netw 23(4):248–257
60. Zhang YH, Sun XM, Wang BW (2016) Efficient algorithm for K-barrier coverage based on integer linear programming. China Commun 13(7):16–23
61. Shen J, Zhou TQ, He DB, Zhang YX, Sun XM, Xiang Y (2017) Block design-based key agreement for group data sharing in cloud computing. IEEE Trans Dependable Secure Comput. https://doi.org/10.1109/TDSC.2017.2725953