Median-Oriented Bat Algorithm for Function Optimization

Limin Zhao and Haifeng Li(✉)

School of Electronic Information and Electrical Engineering, Tianshui Normal University, Tianshui 741001, China
lihaifeng8848@gmail.com

Abstract. The bat algorithm is easily trapped into local optima, the population diversity is poor, and the optimizing precision is bad. In order to overcome these disadvantages, this paper presents a median bat algorithm (MBA) to avoid local optima and carry out a global search over entire search space. The proposed algorithm adopts the median position of the bats. And the median and worst bats are combined to the basic bat algorithm to achieve a better balance between the global search ability and local search ability. The simulation results of 10 standard benchmark functions show that the proposed algorithm is effective and feasible in both low-dimensional and high-dimensional case. Compared to the basic bat algorithm, particle swarm optimization and CLSPSO, the proposed algorithm can get high precision and can almost reach the theoretical value.

Keywords: Bat algorithm · Median-oriented · Function optimization

1 Introduction

The nature-inspired optimization algorithms derived from the simulation of biological group behaviors in natural world. With a simple and parallel implement, strong robustness, good optimization results and so on, the nature-inspired optimization algorithms have become the focus of study. In recent years, some novel swarm intelligence algorithms have been proposed, such as Artificial Bee Colony Optimization (ABC) [1, 2], Shuffled Frog Leaping Algorithm (SFLA) [3], Artificial Fish Swarm Algorithm (AFSA) [4], Cuckoo Search (CS) [5], Monkey Algorithm (MA) [6], Firefly Algorithm (FA) [7], Glowworm Swarm Optimization algorithm (GSO) [8], Flower Pollination Algorithm (FPA) [9], Wind Driven Optimization (WDO) [10], Charged System Search (CSS) [11] and so on.

First proposed by Yang [12] in 2010, Bat Algorithm (BA) was originated from the simulation of echolocation behavior in bats. Bats use a type of sonar called echolocation to detect prey, and avoid obstacles in the dark. When searching their prey, the bats emit ultrasonic pulses. During flight to the prey, loudness will decrease while the pulse emission will gradually increase, which can make the bat locate the prey more accurately. Applications of BA algorithm span the areas of constrained optimization tasks [13], global engineering optimization [14], multi-objective optimization [15], structural optimization [16], and discrete size optimization of steel frames [17].

D.-S. Huang et al. (Eds.): ICIC 2016, Part I, LNCS 9771, pp. 691–702, 2016.
DOI: 10.1007/978-3-319-42291-6_69

In this paper, we propose a novel median-oriented bat algorithm (MBA) for the function optimization problem. The proposed algorithm adopts the median and worst bat individuals [18] to avoid premature convergence. As a result, the global search ability of MBA is improved and the proposed algorithm can avoid trapping in the local optimum. Simulation results demonstrate the effectiveness and robustness of the proposed algorithm. MBA can get a more accurate solution for the optimization problems. In MBA, the mutation operation in DE is added to the bat algorithm to accelerate the global convergence speed.

The remainder of this paper is organized as follows. Section 2 introduces the basic bat algorithm. In Sect. 3, median-oriented bat algorithm is introduced. The experimental results and comparison results are given in Sect. 4. Finally, some relevant conclusions are presented in Sect. 5.

2 The Basic Bat Algorithm

2.1 The Update of Velocity and Position

Initialize the bat population randomly. Supposed the dimension of search space is n, the position of the bat i at time t is x_i^t and the velocity is v_i^t. Therefore, the position x_i^{t+1} and velocity v_i^{t+1} at time $t+1$ are updated by the following formula:

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

$$v_i^{t+1} = v_i^t + (x_i^t - Gbest)f_i^t \tag{2}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{3}$$

Where, f_i represents the pulse frequency emitted by bat i at the current moment. $f_{\max}$ and $f_{\min}$ represent the maximum and minimum values of pulse frequency respectively. β is a random number in $[0, 1]$ and $Gbest$ represents the current global optimal solution.

Select a bat from the bat population randomly, and update the corresponding position of the bat according to Eq. (1). This random walk can be understood as a process of local search, which produces a new solution by the chosen solution.

$$x_{new}(i) = x_{old} + \varepsilon A^t \tag{4}$$

Where, x_{old} represents a random solution selected from the current optimal solutions, A^t is the loudness, and ε is a random vector, and its arrays are random values in $[-1, 1]$.

2.2 Loudness and Pulse Emission

Usually, at the beginning of the search, loudness is strong and pulse emission is small. When a bat has found its prey, the loudness decreases while pulse emission gradually increases. Loudness $A(i)$ and pulse emission $r(i)$ are updated according to Eq. (2) and Eq. (3):

$$r^{t+1}(i) = r^{0}(i) \times [1 - \exp(-\gamma t)] \tag{5}$$

$$A^{t+1}(i) = \alpha A^{t}(i) \tag{6}$$

Where, both $0 < \alpha < 1$ and $\gamma > 0$ are constants. $A(i)=0$ means that a bat has just found its prey and temporarily stop emitting any sound. It is not hard to find that when $t \to \infty$, we can get $A^{t}(i) \to 0$ and $r^{t}(i) = r^{0}(i)$.

2.3 The Implementation Steps of Bat Algorithm

Step 1: Initialize the basic parameters: attenuation coefficient of loudness α, increasing coefficient of pulse emission γ, the maximum loudness A^0 and maximum pulse emission r^0 and the maximum number of iterations *Maxgen*;

Step 2: Define pulse frequency $f_i \in [f_{\min}, f_{\max}]$;

Step 3: Initialize the bat population x and v;

Step 4: Enter the main loop. If $rand < r_i$, update the velocity and current position of the bat according to Eqs. (2) and (3). Otherwise, make a random disturbance for position of the bat, and go to ***Step 5***;

Step 5: If $rand < A_i$ and $f(x_i) < f(x^*)$, accept the new solutions, and fly to the new position;

Step 6: If $f(x_i) < f_{\min}$, replace the best bat, and adjust the loudness and pulse emission according to Eqs. (5) and (6);

Step 7: Evaluate the bat population, and find out the best bat and its position;

Step 8: If termination condition is met (i.e., reach maximum number of iterations or satisfy the search accuracy), go to step 9; Otherwise, go to step 4, and execute the next search.

Step 9: Output the best fitness values and global optimal solution.

Where, *rand* is a uniform distribution in $[0, 1]$.

3 Median-Oriented Bat Algorithm

In this section, a novel median-oriented bat algorithm (MBA) is presented to enhance the performance of the basic bat algorithm [19–22]. In BA, each bat moves toward good solutions based on the best solution. MBA is a global search algorithm.

$$stepnow = (iterMax - iter)^3 * (step_{ini} - step_{final})/(iterMax)^3 + step_{final} \tag{7}$$

$$v_i^{t+1} = v_i^t + f_i^t * (x_i^t - Gbest) + stepnow * rand * (x_i^t - Gmedian - Gworst) \tag{8}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{9}$$

Due to the proposed algorithm considering the best bat individual, the median bat individual and the worst bat individual, this is equivalent to adopt a compromise solution. The coordination of the bat population of individuals is conducive to cover a wider range of bat population and increase the diversity of bat population.

Pseudo code of MBA

```
1: BEGIN
2:     Initialize the bat population
3:     For all $X_i$ do
4:         Calculate fitness $f(X_i)$
5:     End for
6:     Get the best solution $Gbest$, median solution $Gmedian$ and worst
solution $Gworst$
7:     $iter \leftarrow 1$
8:     While $iter <= iterMax$
9:         Calculate $stepnow$
10:        Update the position and velocity as follows:
11:        $v_i^{t+1} = v_i^t + f_i^t * (x_i^t - Gbest) + stepnow * rand * (x_i^t - Gmedian - Gworst)$
12:        If $rand > r_i$ then
13:            Generate a local solution around the best solution
14:        End If
15:        Generate a new solution randomly
16:        If $rand < p_m$ then
17:            Implement the differential evolution mutate operation
18:        End if
19:        Calculate fitness $f(X_i)$
20:        If $(rand < A_i$ and $f(X_i) < f(X_*))$ then
21:            Accept the new solutions
22:            Reduce $A_i$ and increase $r_i$
23:        End if
24:        Get the best solution
25:        $iter \leftarrow iter + 1$
26:    End While
27:    Memorize the best solution $Gbest$
28: END
```

4 Simulation Experiments and Discussion

4.1 Simulation Platform

All the algorithms are implemented in Matlab R2012 (a). The test environment is set up on a computer with AMD Athlon (tm) II X4 640 Processor, 3.00 GHz, 4 GB RAM, running on Windows 7.

4.2 Benchmark Functions

In order to verify the effectiveness of the proposed algorithm, we select 10 standard benchmark functions [23] in Table 1 to detect the searching capability of the proposed

Table 1. Benchmark functions

No	D	Name	Benchmark function	Scope
F_1	30/100	Sphere	$f(x) = \sum_{i=1}^{n} x_i^2$	[−100, 100]
F_2	30/100	Step	$f(x) = \sum_{i=1}^{n-1} (\lfloor xi + 0.5 \rfloor)^2$	[−100, 100]
F_3	30/100	Quartic	$f(x) = \sum_{i=1}^{n} x_i^4 + random[0,1)$	[−1.28, 1.28]
F_4	30/100	Rastrigin	$f(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	[−5.12, 5.12]
F_5	30/100	Rosenbrock	$f(x) = \sum_{i=1}^{n-1} [(x_i - 1)^2 + 100(x_i^2 - x_{i+1})^2]$	[−2.048, 2.048]
F_6	30/100	Ackley	$f(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i)) + 20 + e$	[−32.768, 32.768]
F_7	30/100	Griewank	$f(x) = \frac{1}{4000}\sum_{i=1}^{n}(x_i^2) - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$	[−600, 600]
F_8	30/100	Weierstrass	$f(x) = \sum_{i=1}^{n}(\sum_{k=0}^{k\max}[a^k\cos(2\pi b^k(x_i + 0.5))]) - n\sum_{k=0}^{k\max}[a^k\cos(2\pi b^k \times 0.5)]$ $a = 0.5,\ b = 3,\ k\max = 20$	[−0.5, 0.5]
F_9	30/100	Cosine mixture	$f(x) = \sum_{i=1}^{n} x_i^2 - 0.1\sum_{i=1}^{n}\cos(5\pi x_i)$	[−1, 1]
F_{10}	30/100	Alpine	$f(x) = \sum_{i=1}^{n} \lvert x_i\sin(x_i) + 0.1x_i \rvert$	[−10, 10]

algorithm. The proposed algorithm in this paper (i.e., MBA) is compared with PSO, CLSPSO and BA.

4.3 Parameter Setting

In PSO, we use linear decreasing inertia weight is $\omega_{\max} = 0.9$, $\omega_{\min} = 0.4$, and learning factor is $C_1 = C_2 = 1.4962$. In CLSPSO, inertia weight and learning factor are the same as in PSO. The times of chaotic search is $MaxC = 10$. In BA, the parameters are generally set as follows: pulse frequency range is $f_i \in [-1,\ 1]$, the maximum loudness is $A^0 = 0.9$, minimum pulse emission is $r^0 = 0.5$, attenuation coefficient of loudness is $\alpha = 0.95$, increasing coefficient of pulse emission is $\gamma = 0.05$. In MBA, the basic parameters are the same as in BA, and $stepLen\_ini = 5 step\ Len\_final = 0$.

4.4 Experimental Results

In order to evaluate the performance of MBA, sixteen benchmark functions are adopted in this paper. In this section, the population size $popsize = 50$ and maximum number of iterations $iterMax = 2000$, and MBA is compared to DE and BA. The test results are get from 50 independent run times.

4.4.1 Experimental Results of Low-Dimension Case

The comparison results of all the algorithms on all the functions are recorded in Table 2. The best, mean, worst and std represent the optimal value, mean value, worst value and standard deviation, respectively. We can see that the performance of MBA exhibits significantly better than that of other algorithms. For the functions F_1, F_2, F_4, F_7, F_8, F_9, F_{10}, the MBA algorithm can obtain the theoretical optimal values in all runs. For the function F_3, compared with the mean value, the quality of MBA algorithm is far better than PSO, CLSPSO, BA with at least higher 5, 5, and 3 orders of magnitude, respectively. For function F_5, the best value and the mean value of MBA are both better than those of other algorithms. For function F_6, compared with the mean value, the quality of MBA algorithm is far better than PSO, CLSPSO, BA with at least higher 16, 15, and 17 orders of magnitude, respectively. By Comparison with standard deviation, MBA is also better than PSO, CLSPSO, BA, we can see that the MBA algorithm has strong robustness. And for functions F_1, F_2, F_4, F_6, F_7, F_8, F_9, F_{10}, the standard deviations are 0. That is, the MBA algorithm obtains the same global optimal value in all runs. For the basic BA algorithm, the best values are inferior to those of MBA, even to the worse values of MBA on all functions. The results of CLSPSO are a little better than those of PSO, but obvious worse than those of MBA. Figure 1 shows the mean fitness of four algorithms on the function F_1 to F_{10}, when the mean value is not 0, MBA has a longer and downward column. When the mean value is 0, we cannot find the bar for MBA algorithm.

Table 2. Experimental results for function from F_1 to F_{10} (D = 30)

Fun	Algorithm	Best	Worst	Mean	Std
F_1	PSO	0.0115	0.2843	0.0979	0.0702
	CLSPSO	2.5669e−12	8.2722e−04	5.7519e−05	1.3171e−04
	BA	8.4173e−04	0.00133	0.0011	1.1398e−04
	MBA	***0***	***0***	***0***	***0***
F_2	PSO	24	226	96.48	41.1172
	CLSPSO	0	5	0.1	0.7071
	BA	34605	61144	48338	6398.5881
	MBA	***0***	***0***	***0***	***0***
F_3	PSO	0.0099	2.7316	0.2126	0.5970
	CLSPSO	0.0012	2.7361	0.1303	0.5374
	BA	0.0021	0.0108	0.0050	0.0019
	MBA	***8.0885e−07***	***5.6234e−05***	***1.3090e−05***	***1.1655e−05***
F_4	PSO	33.3947	1.4977e+02	79.0019	28.1500
	CLSPSO	4.2064e−12	1.61300e+02	27.5633	41.6538
	BA	1.3447e+02	2.5789e+02	1.9631e+02	30.3170
	MBA	***0***	***0***	***0***	***0***
F_5	PSO	28.1084	8.8179e+02	56.3213	1.2249e+02
	CLSPSO	3.2750e−06	1.2833e+02	6.5676	20.1953
	BA	21.4382	83.3127	27.2583	8.3434
	MBA	***1.0689e−05***	***3.9982***	***0.2400***	***0.9589***
F_6	PSO	2.0649	6.2396	3.8940	0.8686
	CLSPSO	1.9754e-06	0.2757	0.0481	0.0562
	BA	18.3337	19.4259	19.0089	0.2248
	MBA	***8.8818e−16***	***8.8818e−16***	***8.8818e−16***	***0***
F_7	PSO	0.5668	2.0102	1.2231	0.2560
	CLSPSO	5.9094e−07	1.0303	0.2590	0.2940
	BA	1.8765e+02	4.1058e+02	3.2020e+02	45.7655
	MBA	***0***	***0***	***0***	***0***
F_8	PSO	10.7982	26.1507	18.2713	3.8755
	CLSPSO	0.0023	25.9101	3.2922	5.9277
	BA	22.5364	41.3711	31.5940	3.4232
	MBA	***0***	***0***	***0***	***0***
F_9	PSO	−2.6592	−0.3120	−1.7452	0.5675
	CLSPSO	−2.999999	−0.1170	−2.7675	0.7129
	BA	0.1125	3.7779	1.6354	0.7540
	MBA	***−3***	***−3***	***−3***	***0***
F_{10}	PSO	0.4080	8.0998	3.9332	1.9486
	CLSPSO	3.8542e−06	7.8113	0.5980	1.6200
	BA	1.9301	16.4262	6.4830	3.1697
	MBA	***0***	***0***	***0***	***0***

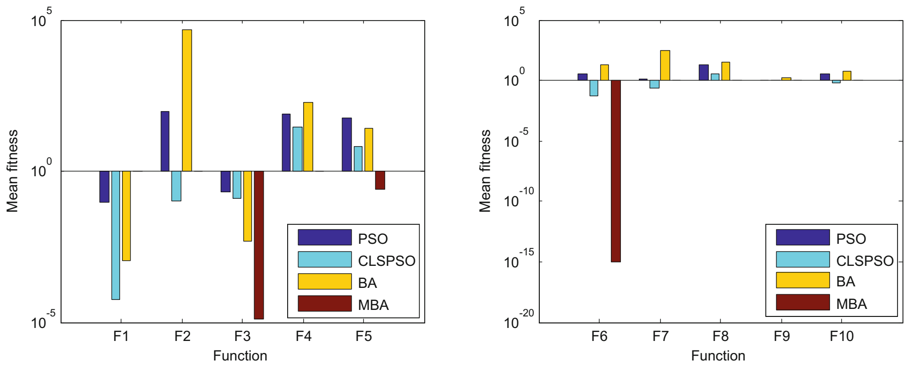

Fig. 1. Mean fitness of four algorithms for F_1 to $F_{10}(D = 30)$. (Color figure online)

4.4.2 Experimental Results of High-Dimension Case

In order validate the performance of the proposed algorithm further, we implement the experiment on 100 dimensions for the all algorithms and keep the parameters unchanged.

The comparison results for high-dimension case are shown in Table 3. Seen from the results, the optimization performance of MBA is the best. For the functions F_1, F_2, F_4, F_7, F_8, F_9, F_{10}, the MBA algorithm still obtains the theoretical optimal values in all runs without a doubt. Only the precision of the best value descends for solving the F5

Table 3. Experimental results for function from F_1 to F_{10} (D = 100)

Fun	Algorithm	Best	Worst	Mean	Std
F_1	PSO	0.8546	45.5550	10.4782	11.0198
	CLSPSO	1.9934e−16	30.1745	0.6042	4.2672
	BA	8.1807e−04	0.0013	0.0011	1.2315e−04
	MBA	***0***	***0***	***0***	***0***
F_2	PSO	99	3369	1257.86	7.5513e+02
	CLSPSO	0	0	0	0
	BA	25644	62332	47257	6.6565e+03
	MBA	***0***	***0***	***0***	***0***
F_3	PSO	0.1522	1.4610e+02	23.3675	38.9468
	CLSPSO	0.0017	1.4303e+02	10.1692	31.2951
	BA	0.0012	0.00992	0.0046	0.0016
	MBA	***3.6753e−06***	***4.3172e−04***	***9.7659e−05***	***9.3351e−05***
F_4	PSO	4.2493e + 02	7.4633e + 02	5.9333e + 02	73.6489
	CLSPSO	1.5224e-04	6.9392e+02	2.7107e+02	2.7534e+02
	BA	1.3754e+02	2.6587e+02	2.0306e+02	25.5434

(*Continued*)

Table 3. (*Continued*)

Fun	Algorithm	Best	Worst	Mean	Std
	MBA	***0***	***0***	***0***	***0***
F_5	PSO	1.1518e+02	2.7031e+03	5.9723e+02	6.1093e+02
	CLSPSO	1.0889e−05	2.7735e+03	2.6267e+02	6.9857e+02
	BA	11.0558	87.9935	29.0136	14.1507
	MBA	***2.3174***	***2.5904***	***2.5121***	***0.0664***
F_6	PSO	2.1735	8.1533	5.6369	1.1818
	CLSPSO	2.0846e-06	0.3539	0.0763	0.0912
	BA	18.4867	19.4188	19.0488	0.2185
	MBA	***8.8818e−16***	***8.8818e−16***	***8.8818e−16***	***0***
F_7	PSO	3.2679	25.5472	9.9551	4.7114
	CLSPSO	2.2958e−06	0.9778	0.2083	0.2596
	BA	1.8777e+02	3.7630e+02	3.1151e+02	42.2589
	MBA	***0***	***0***	***0***	***0***
F_8	PSO	31.5448	95.0606	67.5116	12.1533
	CLSPSO	0.0458	88.4343	44.8663	33.5603
	BA	23.6779	39.8344	31.3530	3.9133
	MBA	***0***	***0***	***0***	***0***
F_9	PSO	−7.2753	1.3841	−3.8135	1.6853
	CLSPSO	−9.999999999	0.8495	−7.4278	3.4429
	BA	−0.0349	3.7838	1.5973	0.9812
	MBA	***−10***	***−10***	***−10***	***0***
F_{10}	PSO	12.8953	53.5153	25.7114	7.7965
	CLSPSO	5.3529e−06	48.2857	3.5625	9.8779
	BA	1.2815	13.0499	5.8033	2.3913
	MBA	***0***	***0***	***0***	***0***

function. The standard deviations of MBA are better than those of other algorithms on all functions. It demonstrates that MBA has strong robustness and good global search ability, and MBA does not reduce the accuracy of the solutions as the dimension increases. However, the precision of solutions of other three algorithms, including the best value, the mean value, the worse value, the median value and the standard deviation, will decrease with the increase of the dimension. Figures 2 and 3 show the results of the ANOVA tests for all algorithms on F_1 and F_2. PSO and BA have long tail on the functions F_1, F_2, and more singular points. That is, the methods are not robust and are not acceptable for the function optimization. The precision of BA is inferior to those of PSO and CLSPSO, but better stability and robustness.

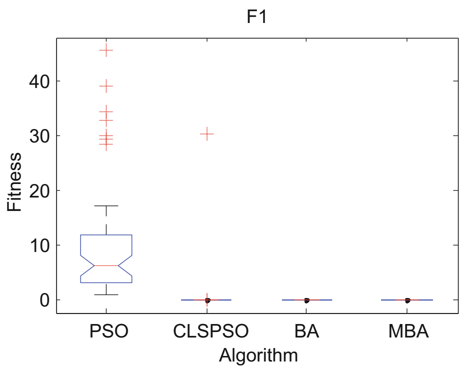

Fig. 2. ANOVA tests for $F_1(D = 100)$. (Color figure online)

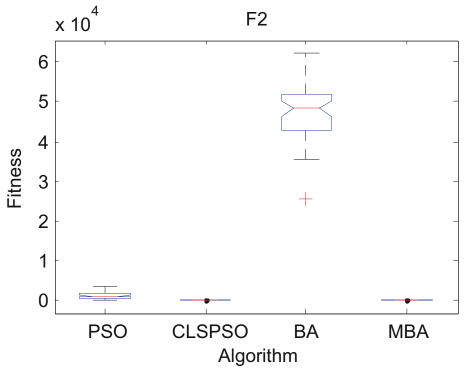

Fig. 3. ANOVA tests for $F_2(D = 100)$. (Color figure online)

5 Conclusions

This paper presented a novel Median-oriented bat algorithm (MBA) for function optimization problem. The proposed algorithm adopts the median and worst bat individuals to avoid premature convergence. MBA has an excellent ability of global search owing to its diversity caused by the probabilistic representation. The simulation experiments show that the proposed algorithm is a feasible and effective way for function optimization. The optimization ability of MBA does not show a significant decline with increasing the dimension. The proposed algorithm is suitable for low - dimensional and high-dimensional case.

References

1. Xu, C.F., Duan, H.B., Liu, F.: Chaotic artificial bee colony approach to Uninhabited Combat Air Vehicle (UCAV) path planning. Aerosp. Sci. Technol. **14**(8), 535–541 (2010)
2. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Global Optim. **39**(3), 459–471 (2007)
3. Li, X., Luo, J.P., et al.: An improved shuffled frog-leaping algorithm with extremal optimisation for continuous optimization. Inf. Sci. **192**, 143–151 (2012)
4. Neshat, M., Sepidnam, G., et al.: Artificial fish swarm algorithm: a survey of the state-of-the-art, hybridization, combinatorial and indicative applications. Artif. Intell. Rev. **42**, 965–997 (2012). doi:10.1007/s10462-012-9342-2
5. Gandomi, A.H., Yang, X.S., Alavi, A.H.: Cuckoo search algorithm: a meta-heuristic approach to solve structural optimization problems. Eng. Comput. **29**, 17–35 (2013)
6. Zhao, R.Q., Tang, W.S.: Monkey algorithm for global numerical optimization. J. Uncertain Syst. **2**(3), 164–175 (2008)
7. Gandomi, A.H., Yang, X.S., Talatahari, S., et al.: Firefly algorithm with chaos. Commun. Nonlinear Sci. Numer. Simul. **18**(1), 89–98 (2013)
8. Wu, B., Qian, C.H., et al.: The improvement of glowworm swarm optimization for continuous optimization problems. Expert Syst. Appl. **39**(7), 6335–6342 (2012)
9. Yang, X.-S.: Flower pollination algorithm for global optimization. In: Durand-Lose, J., Jonoska, N. (eds.) UCNC 2012. LNCS, vol. 7445, pp. 240–249. Springer, Heidelberg (2012)
10. Bayraktar, Z., Komurcu M., Werner D.H.: Wind driven optimization (WDO): a novel nature-inspired optimization algorithm and its application to electromagnetics. In: 2010 IEEE International Symposium on Antennas and Propagation Society International Symposium (APSURSI) (2010)
11. Kaveh, A., Talatahari, S.: A novel heuristic optimization method: charged system search. Acta Mech. **213**, 267–289 (2010)
12. Yang, X.-S.: A New metaheuristic bat-inspired algorithm. In: González, J.R., Pelta, D.A., Cruz, C., Terrazas, G., Krasnogor, N. (eds.) NICSO 2010. SCI, vol. 284, pp. 65–74. Springer, Heidelberg (2010)
13. Gandomi, A.H., Yang, X.S., Alavi, A.H., et al.: Bat algorithm for constrained optimization tasks. Neural Comput. Appl. **22**, 1239–1255 (2013)
14. Yang, X.S., Gandomi, A.H.: Bat algorithm: a novel approach for global engineering optimization. Eng. Comput. **29**(5), 464–483 (2012)
15. Yang, X.S.: Bat algorithm for multi-objective optimization. Int. J. Bio-Inspir. Comput. **3**(5), 267–274 (2011)
16. Hasançebi, O., Teke, T., Pekcan, O.: A bat-inspired algorithm for structural optimization. Comput. Struct. **128**, 77–90 (2013)
17. Hasançebi, O., Carbas, S.: Bat inspired algorithm for discrete size optimization of steel frames. Adv. Eng. Softw. **67**, 173–185 (2014)
18. Beheshti, Z., Hj, S.M., Hasan, S.S.: MPSO: median-oriented particle swarm optimization. Appl. Math. Comput. **219**, 5817–5836 (2013)
19. Zhou, Y.Q., Xie, J., Li, L.L., Ma, M.Z.: Cloud model bat algorithm. Sci. World J. **2014**, 11 (2014). doi:10.1155/2014/237102. Article ID 237102
20. Li, L.L., Zhou, Y.Q.: A novel complex-valued bat algorithm. Neural Comput. Appl. **25**, 1369–1381 (2014). doi:10.1007/s00521-014-1624-y

21. Gandomi, A.H., Yang, X.S.: Chaotic bat algorithm. J. Comput. Sci. **5**, 224–232 (2014)
22. Zhou, Y. Q., Li, L.L., Ma, M.Z.: A complex-valued encoding bat algorithm for solving 0–1 knapsack problem. Neural Process. Lett. 1–24 (2015) DOI: 10.1007/s11063-015-9465-y
23. Karaboga, D., Akay, B.: A comparative study of Artificial Bee Colony algorithm. Appl. Math. Comput. **214**, 108–132 (2009)