OPSEARCH (2017) 54:684–705
DOI 10.1007/s12597-017-0302-0

APPLICATION ARTICLE

Multi-directional bat algorithm for solving unconstrained optimization problems

Mohamed A. Tawhid[1,2] · **Ahmed F. Ali**[3,4]

Accepted: 26 January 2017 / Published online: 9 February 2017

Abstract In this paper, we propose a new hybrid algorithm for solving unconstrained global optimization problems by hybridizing the bat algorithm with multi-directional search algorithm (MDS). We call the proposed algorithm by multi-directional bat algorithm (MDBAT). In MDBAT algorithm, we try to overcome the slow convergence of the bat algorithm as a metaheuristic algorithm by invoking one of the promising direct search algorithm which is called MDS algorithm. The bat algorithm has a good ability to make exploration and exploitation search while the MDS has a good ability for accelerating convergence on the region of optimal response. In the beginning, the standard bat algorithm starts the search for number of iterations then the MDS algorithm starts its search from bat algorithm found so far. The combination between the standard bat algorithm and the MDS algorithm helps the MDS algorithm to start the search from a good solution instead of the random initial solution. The MDS algorithm can accelerate the search of the proposed algorithm instead of letting the algorithm running for more iterations without any improvement. We investigate the general performance of the MDBAT algorithm by applying it on 16 unconstrained global optimization problems and

✉ Mohamed A. Tawhid
Mtawhid@tru.ca

Ahmed F. Ali
ahmed_fouad@ci.suez.edu.eg

[1] Department of Mathematics and Statistics, Faculty of Science, Thompson Rivers University, Kamloops, BC V2C 0C8, Canada

[2] Department of Mathematics and Computer Science, Faculty of Science, Alexandria University, Moharam Bey, Alexandria 21511, Egypt

[3] Department of Computer Science, Faculty of Computers and Informatics, Suez Canal University, Ismailia, Egypt

[4] Department of Mathematics and Statistics, Faculty of Science, Thompson Rivers University, Kamloops, BC V2C 0C8, Canada

comparing it against 8 benchmark algorithms. The experimental results indicate that MDBAT is a promising algorithm and outperforms the other algorithms in most cases.

Keywords Bat algorithm · Multi-directional search algorithm · Global optimization · Unconstrained optimization problems

1 Introduction

We consider the unconstrained optimization problem

$$\min f(x), \quad x \in R^n \tag{1}$$

where $f : R^n \to R$ is an objective function and x represents a decision variable vector. Also, $x \in S \subset R^n$ and S denotes the search space, which is n dimensional and bounded by parametric constraints as follows:

$$l_i \leq x_i \leq u_i \quad i = 1, 2, \ldots, n \tag{2}$$

where l_i and u_i are the lower and upper bounds of the decision variables x_i, respectively.

Many researchers have solved the problem in (1) via deterministic algorithms such as steepest descent, Newton, and conjugate gradient methods, see, e.g., [37]. These methods require the differentiability of the objective function. However, computing the Jacobian (derivative of the objective function) is a difficult and expensive operation. Also, the objective function might be nonsmooth. This is a motivation for many researchers to develop stochastic global optimization algorithms such as swarm intelligence (SI) algorithms. SI take inspiration from the behavior of a group of social organisms. These algorithms are applied to solve global optimization problems and their applications such as Ant Colony Optimization (ACO) [8], Artificial Bee Colony [12], Particle Swarm Optimization (PSO) [13], Bacterial foraging [18], Bat algorithm [35], Bee Colony Optimization (BCO) [28], Wolf search [25], Cat swarm [6], Cuckoo search [34], Firefly algorithm [33], Fish swarm/school [15], etc. Some old metaheuristic algorithms such as genetic algorithms (GA) have some drawbacks when they are dealing with multimodal optimization problems.

Recently, Yang [35] proposed a novel metaheuristic search algorithm, called bat algorithm (BA). Preliminary studies show that it is very promising and could outperform existing algorithms. Bats are captivating animals, are the only mammals with wings and have advanced capability of echolocation. Micro bats use a type of sonar, called, echolocation, to detect prey, avoid obstacles, and locate their roosting crevices in the dark.

BA has a good capability to balance the global exploration and the local exploitation during the search process. Since BA has a powerful performance, is easy to implement, and has fast convergence, many researchers have attracted and applied BA on their works to solve various applications, for example, Yang [36] applied BA to solve multi-objective optimization and benchmark engineering

problems. Zhang and Wang [38] improved the diversity of solutions by using the mutation with bat algorithm for image matching. Komarasamy and Wahi [14] gave a combination of K-means and bat algorithm (KMBA) for efficient clustering. Lin et al. [16] carried out parameter estimation in dynamic biological systems using a chaotic bat algorithm by combining Levy flights and chaotic maps. Nakamura et al. [17] developed a discrete version of bat algorithm to solve classifications and feature selection problems. Wang and Guo [31] combined bat algorithm with harmony search and have produced a hybrid bat algorithm for numerical optimization of function benchmarks. In addition, Xie et al. [32] presented a variant of bat algorithm combining differential operator and Levy flights to solve function optimization problems.

The main objective of this paper is to produce a new hybrid bat algorithm with the multi-directional search algorithm in order to solve unconstrained global optimization problems. The proposed algorithm is called multi-directional bat algorithm (MDBAT). In order to overcome the slow convergence of the bat algorithm, we invoke the multi-directional search algorithm. The multi-directional search can accelerate the search instead of letting the algorithm running for more iterations without any improvement. The bat algorithm running for number of iterations then the multi-directional search starts the search from the best obtained solution from the bat algorithm.

The rest of this paper is organized as follows. In Sect. 2, we overview the multi-directional search algorithm. In Sect. 3, we present the standard bat algorithm and describe he main concepts of the proposed MSBAT algorithm. In Sect. 4, we present our numerical experimental results. Finally, in Sect. 5 we end up with our conclusion and future work.

2 Multi-directional search algorithm

The multi-directional search algorithm was proposed by Dennis and Torczon [30] as a step towards a general-purpose optimization algorithm with promising properties for parallel computation. The multi-directional search algorithm is a direct search algorithm.

According to Dennis and Torczon [7], the Nelder-Mead Simplex algorithm can converge to non-minimizers when the dimension of the problem becomes large enough.

One of the advantages of the multi-directional search is that unlike the Nelder-Mead simplex algorithm, it is backed by convergence theorems that numerical testing also indicate are borne out in practice.

Since the multi-directional search is backed by convergence theorems, it has a promising behavior when applied with the high dimensional problems.

In Algorithm 1 and Fig. 1, we present the main steps of the multi-directional search algorithm.

We can summarize the main steps of the multi-directional search as follows.

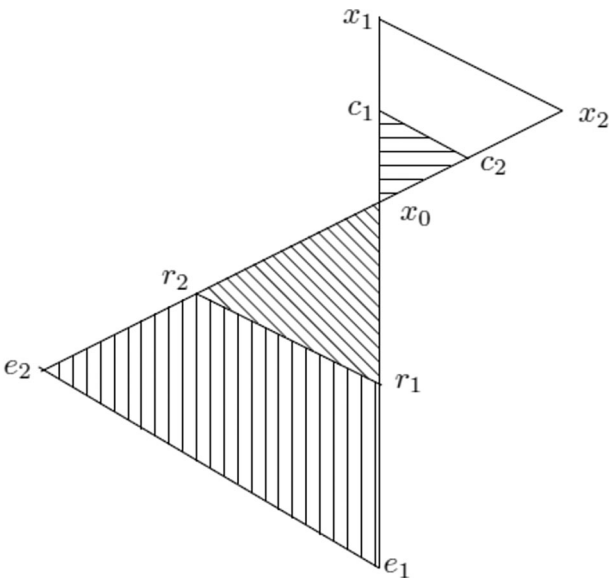

Fig. 1 Multi-directional search

- *Step 1* The algorithm starts by setting the initial values of the expansion factor μ, contraction factor θ and the maximum number of iterations parameter Max_{itr1}. (Line 1)
- *Step 2* The multi-directional search algorithm begins with a simplex S with verities x_i^0, where $i = 0, 1, \ldots, n$. (Line 2)
- *Step 3* The vertices are sorted in ascending order where $f(x_0^0) \leq f(x_1^0) \leq \cdots \leq f(x_n^0)$. (Line 3)
- *Step 4* The iteration counter is initialized and the main algorithm loop is initialized. (Lines 4–5)
- *Step 5* The reflected step is started by reflecting the vertices $x_1, x_2, \ldots, x_n$ through the best vertex x_0 and the new reflected vertices are evaluated. (Lines 6–9)
- *Step 6* If a reflected vertex succeeds and its value is better than the current best vertex, then the algorithm starts the expansion step. (Line 10)
- *Step 7* The expansion process starts to expand each reflected edge by using the expansion factor μ, where $\mu = 2$ to create new expanded vertices. The new expanded vertices are evaluated in order to check the success of the expansion step. (Lines 10–14)
- *Step 8* If the expanded vertex is better than the all reflected vertices, the new simplex will be the expanded simplex. (Lines 15–16)
- *Step 9* If the expansion and reflection steps fail, then the contracted simplex starts by changing the size of the step via using contraction factor θ, which reduces the reflected simplex to the half in the next iteration. (Lines 18–21)
- *Step 10* The new vertices are evaluated and the vertices are sorted according to their evaluation function value and the new simplex is constructed. (Lines 24–27)

- *Step 11* The iteration counter is increased and the overall process are repeated till the termination criterion is satisfied which is by default the maximum number of iterations Max_{itr1}. Finally, the best solution is produced. (Lines 28–30)

3 The basic bat algorithm

In this section, we overview of the main concepts and structure of the bat algorithm as follows.

3.1 Main concepts

Bat algorithm (BA) is a novel metaheuristic optimization algorithm proposed by Yang [35]. Because the bat algorithm is simple to understand, its adjustment parameters are few, it is easy to implement and its convergence speed is fast at a very initial stage by switching from exploration to exploitation, however, switching from exploration to exploitation quickly may lead to stagnation after some initial stage.

Bats use sonar called as echolocation to detect prey and to avoid obstacles. Micro-bats are able to recognize positions of the objects by spreading short and high audio signals and by reflection and collision of these spread signals.

Xin-She Yang idealized the following rules to model Bat algorithm:

- All bats use echolocation to sense distance, and they also know the difference between food/prey and background barriers in some magical way.
- Each bat randomly moves with velocity v_i at a position x_i with a frequency f_{min} varying loudens A_0 and pulse emission rate r.
- Although the loudness can vary in many ways, Xin-She Yang assumes that the loudness varies from a large value A_0 to a minimum value A_{min}.

The bat algorithm is a population based method, where the population size consist of bats (solutions). Each bat (solution) in the population is randomly moving with velocity v_i and a location x_i. Also each bat is randomly assigned a frequency drawn uniformly from $[f_{min}, f_{max}]$. The position of each bat in the population is updated as shown in the following equations.

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{3}$$

$$v_i^t = v_i^{t-1} + \left(x_i^{t-1} - x^*\right)f_i, \tag{4}$$

$$x_i^t = x_i^{t-1} + v_i^t, \tag{5}$$

where $\beta \in [0, 1]$ is a random vector drawn from a uniform distribution.

The loudness A_i and the pulse rate emission r_i are very important to let the algorithm switch between exploration and exploitation process. When the bat has

found its pray, the loudness decreases and the rate of pulse emission increases. The bat algorithm starts with an initial value of the loudness A_0 and the rate of pulse emission r_0, then these values will be updated as shown in the following equations.

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

$$r_i^{(t)} = r_i^{(0)}[1 - exp(-\gamma t)] \tag{7}$$

where α and γ are constants, the α parameter plays a similar role as the cooling factor in the simulated annealing algorithm, where $\alpha \in [0, 1]$ and $\gamma > 0$.

For the local search part, a solution is selected among the current best solutions, for each bat, a new solution is generated locally using a local random walk as in (8)

$$x_{new} = x_{old} + \lambda A^t \tag{8}$$

where λ is a random number, $\lambda \in [-1, 1]$ and A^t is the average loudness of all the bats at this iterations.

3.2 Bat algorithm

In Algorithm 2, we show the main steps of the standard bat algorithm and can summarize these steps as follows.

- *Step 1* The algorithm starts by setting the initial values of its parameters and the main iteration counter is set to zero. (Lines 1–2)
- *Step 2* The initial population is randomly generated by generating the initial position x^0 and the initial velocity v^0 for each bat (solution) in the population, the initial frequency f_i is assigned to each solution in the population, where f is randomly generated from $[f_{min}, f_{max}]$. The initial population is evaluated by calculating the objective function for each solution in the initial population $f(x_i^0)$. The values of pulse rate r_i and loudness A_i are initialized, where $r \in [0, 1]$ and A_i varies from a large A_0 to A_{min}. (Lines 3–9)
- *Step 3* The new population is generated by adjusting the position x_i and the velocity v_i for each solution in the population is given in (3), (4), and (5). (Lines 12–13)
- *Step 4* The new population is evaluated by calculating the objective function for each solution and the best solution x^* is selected from the population. (Lines 14-15)
- *Step 5* The local search method is applied in order to refine the best found solution at each iteration. (Lines 16–19)
- *Step 6* The new solution is accepted with some proximity depending on parameter A_i, the rate of pulse emission increases and the loudness decreases. The values of A_i and r_i are updated as shown in (6) and (7).
- *Step 7* The new population is evaluated and the best solution is selected from the population. The operations are repeated until termination criteria are satisfied and the overall best solution is produced. (Lines 25–28)

3.3 The proposed MDBAT algorithm

In Algorithm 3, we present the algorithm of the proposed MDBAT algorithm. The proposed algorithm applies the same steps of the standard bat algorithm as shown in Algorithm 2, then the best found bat solution is refined by applying Algorithm 1 for a number of iterations Max_{itr2} till termination criteria are satisfied.

4 Numerical experiments

In this section, we evaluate the efficiency of the MDBAT algorithm by presenting the general performance of it with various benchmark functions and comparing the results of the proposed algorithm against various algorithms. In the following subsections, we present the parameter setting of the proposed algorithm and the properties of the applied test functions. Also, we present the performance analysis of the proposed algorithm with the comparative results between it and the other algorithms.

4.1 Parameter setting

We summarize the parameters of the MDBAT algorithm summarized with their assigned values in Table 1. We select these values based on the common setting in the literature or our preliminary numerical experiments.

- *Population size P* The experimental tests show that the best population size is $P = 20$, increasing this number will increase the evaluation function values without any improvement in the obtained results.

Table 1 Parameter setting

Parameters	Definitions	Values
P	Population size	20
f_{min}	Minimum frequency	0
f_{max}	Maximum frequency	2
A_0	The initial loudness	1
r_0	The initial pulse rate	0.5
α	The loudness constant	0.9
γ	The rate of pulse emission constant	0.9
λ	Step size for random walk	1
μ	Expansion factor	2
θ	Contraction factor	0.5
Max_{itr1}	Maximum iterations number for bat algorithm	10
Max_{itr2}	Maximum iterations number for multi-directional search algorithm	100
N_{elite}	Number of best solution for multi-directional search	1

- *Frequency parameter f* Bat movement is based on the value of the frequency parameter *f*. In MDBAT algorithm, it turns out that the quality of the solution is related to the value of *f* parameter. The experimental tests show that the best maximum value of *f* is $f_{max} = 2$ and the minimum value of *f* is $f_{min} = 0$.
- *Loudness parameters A and* α Loudness parameter *A* is one of the most important parameter in the bat algorithm. The acceptance of the new generated solutions is depending on the value of *A*. The α parameter plays a similar role as the cooling factor in the simulated annealing algorithm. We set the initial value of *A* to $A = 1$ and the value of α is set to $\alpha = 0.9$.
- *Pulse emission rate r* The value of the rate of pulse emission parameter *r* is very important to apply the local search method in the algorithm. The experimental tests show that, the best value of *r* is 0.9 and the rate of pulse emission constant is $\gamma = 0.9$.
- *Step size for random walk* λ The experimental results show that the best step size λ in the random walk is equal to 1. Figure 2 shows the general performance of the standard bat algorithm with different values of step size at $\lambda = 0.1, 0.01, 0.001, 1$ for functions f_1, f_2, f_4, f_5, f_6, f_8, f_9, f_{16} (randomly picked). Figure 2 shows that the best value of λ is equal to 1.
- *Multi-directional search parameters* There are two main parameters in the multi-directional search algorithm, which are the expansion factor parameter μ and the contraction factor parameter θ. We take the selected values for μ and θ from lecture, where $\mu = 2$ and $\theta = 0.5$.
- *Stopping condition parameters* MDBAT terminates the search when the number of iterations reaches to the desired maximum number of iterations or any other termination depends on the comparison with other algorithms. In our experiment, we set the value of the maximum iteration number for bat algorithm $Max_{itr1} = 10$ before starting the multi-directional search algorithm and the maximum iteration number for multi-directional search algorithm $Max_{itr2} = 100$.
- *Final intensification* We collect the best obtained solutions from the bat algorithm in a list in order to apply the multi-directional search algorithm on them, the number of the solutions in this list is called N_{elit}. We set $N_{elit} = 1$ in order to avoid increasing the value of the function evaluation value.

4.2 Test unconstrained optimization problems

We test MDBAT algorithm on 16 unconstrained optimization functions with various properties (multi-model, uni-model) functions. We list these functions in Table 2 and report their properties in Table 3.

4.3 The efficiency of applying the multi-directional search in MDBAT algorithm

In order to investigate the efficiency of invoking the multi-directional search algorithm, we apply the standard bat algorithm without combining it with the multi-

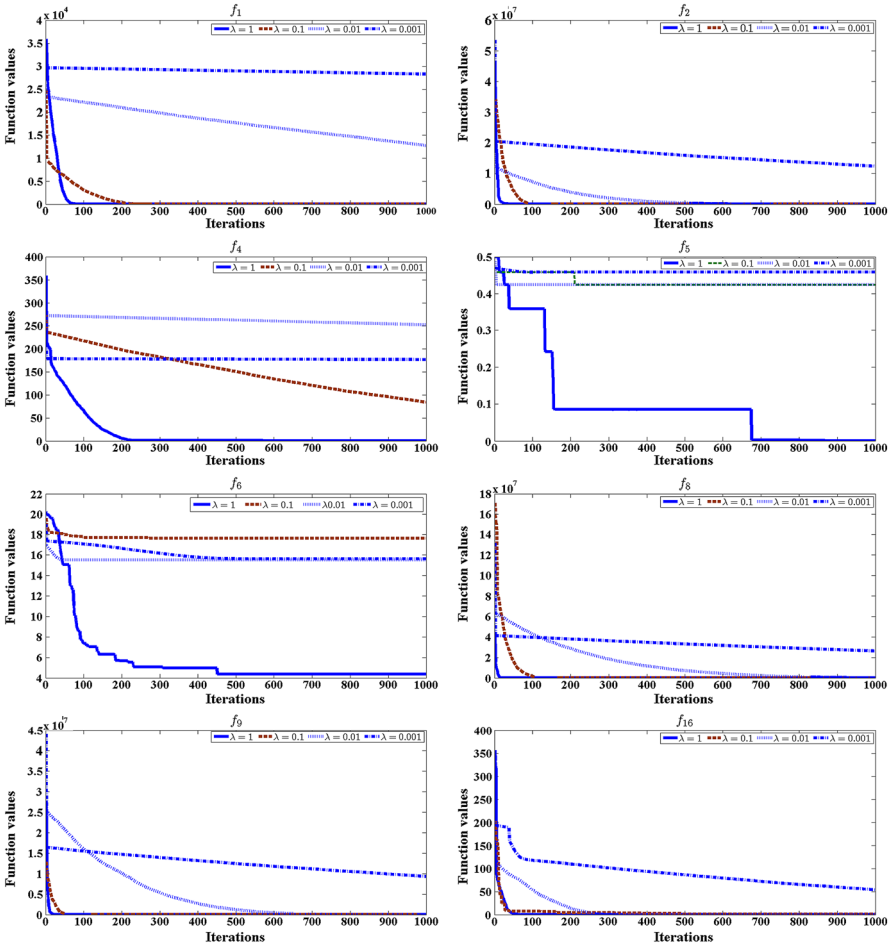

Fig. 2 The general performance of the standard bat algorithm with different values of step size λ

directional search algorithm and the proposed algorithm on six functions which are randomly picked. In Table 4, we report the mean and the standard deviation over 50 runs. The run is successful if the algorithm reaches to the global minimum of the solution within an error of 10^{-4} before the 20,000 function evaluation value. If any algorithm fails to obtain the desired function value, then we report the value of best obtained function value in parentheses. The results in Table 4 show that the combination between the standard bat algorithm and the multi-directional search can accelerate the search and obtain the desired function values for all test functions faster than the standard bat algorithm which needs more iterations in order to obtain the desired function values.

Table 2 Unconstrained benchmark functions

Function	Name	Definition
f_1	Sphere	$f(x) = \sum_{i=1}^{D} x_i^2$
f_2	Rosenbrock's	$f(x) = \sum_{i=1}^{D-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$
f_3	Rastrigin's	$f(x) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right).$
f_4	Griwank's	$f(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
f_5	Schaffer's f6	$f(x) = 0.5 + \frac{sin(\sqrt{x_1^2+x_2^2})}{(1+0.001(x_1^2+x_2^2))^2}$
f_6	Ackley's	$f(x) = 20 + exp(1) - 20exp\left(-\frac{1}{5}\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right)$ $- exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right)$
f_7	Corana	$f(x) = \sum_{i=1}^{4} \begin{cases} 0.15(z_j - 0.05sign(z_j))^2 d_j, & if\lvert x_j - z_j\rvert < 0.05, \\ d_j x_j^2, & otherwise, \end{cases}$ $(d_1, d_2, d_3, d_4) = (1, 1000, 10, 100)$, and $z_j = \lfloor \lvert \frac{x_j}{0.2} \rvert + 0.49999 \rfloor sign(x_j) 0.2$
f_8	Levy1	$f(x) = 0.1\Big\{\sin^2(3\pi x_1) + \sum_{i=1}^{D-1}(x_i - 1)^2\left[(1 + \sin^2(3\pi x_{i+1}))\right]$ $+(x_D - 1)^2[1 + sin^2(2\pi x_n)]\Big\} + \sum_{i=1}^{D} u(x_i, 5, 100, 4),$ $u(z, a, k, m) = \begin{cases} k(z-a)^m, & z > a; \\ 0, & -a \le z \le a; \\ k(-z-a)^m, & z < -a. \end{cases}$
f_9	Levy2	$f(x) = \frac{\pi}{D}\{10\sin^2(\pi x_1) + \sum_{i=1}^{D-1}\left[(x_i - 1)^2(1 + 10\sin^2(\pi x_i + 1) + (x_D - 1)^2\right]$ $\sum_{i=1}^{D} u(x_i, 10, 100, 4),$
f_{10}	Goldstein and price	$f(x) = \left\{1 + (x_0 + x_1 + 1)^2(19 - 14x_0 + 3x_0^2 - 14x_1 + 3x_1^2)\right\}$ $\left\{30 + (2x_0 - 3x_1)^2(18 - 32x_0 + 12x_0^2 + 48x_1 - 36x_0x_1 + 27x_1^2\right\}$
f_{11}	Branin	$f(x) = (x_1 - \frac{5.1}{4\pi^2}x_0^2 + \frac{5}{\pi}x_0 - 6)^2 + 10(1 - \frac{1}{8\pi})cos(x_0) + 10$
f_{12}	Shubert	$f(x) = \sum_{j=1}^{5} jcos(j+1)x_1 + j) \sum_{j=1}^{5} jcos((j+1)x_2 + j)$
f_{13}	Easom	$f(x) = -cos(x_1)cos(x_2)exp(-((x_1 - \pi)^2 + (x_2 - \pi)^2))$
f_{14}	Hartmann	$f(x) = -\sum_{i=1}^{4} \alpha_i exp(-\sum_{j=1}^{3} A_{ij}(x_j - P_{ij})^2)$ $\alpha = [1 \quad 1.2 \quad 3 \quad 3.2]$ $A = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}$ $P = \begin{bmatrix} 0.3689 & 0.117 & 0.2673 \\ 0.4699 & 0.4387 & 0.747 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{bmatrix}$
f_{15}	B2	$f(x) = x_1^2 + 2x_2^2 - 0.3cos(3\pi x_1) - 0.4cos(4\pi x_2) + 0.7$
f_{16}	Zakharov	$f(x) = \sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} 0.5ix_i)^2 + (\sum_{i=1}^{n} 0.5ix_i)^4$

Table 3 Properties of classical functions

Function	Bounds	Optimal
f_1	$[-100, 100]^D$	0
f_2	$[-30, 30]^D$	0
f_3	$[-5.12, 5.12]^D$	0
f_4	$[-600, 600]^D$	0
f_5	$[-100, 100]^D$	0
f_6	$[-32, 32]^D$	0
f_7	$[-1000, 1000]^D$	0
f_8	$[-50, 50]^D$	0
f_9	$[-50, 50]^D$	0
f_{10}	$[-2, 2]^D$	3
f_{11}	$[-10, 10]^D$	0.397887
f_{12}	$[-10, 10]^D$	−186.7309
f_{13}	$[-100, 100]^D$	−1
f_{14}	$[0, 1]^D$	−3.86278
f_{15}	$[-100, 100]^D$	0
f_{16}	$[-10, 10]^D$	0

Table 4 The efficiency of applying the multi-directional search in MDBAT algorithm

Function	D	Standard bat	MDBAT
f_1			
Mean	30	(0.1720)	**1745**
SD		15.48	5.14
f_2			
Mean	30	(42.8809)	**4520**
SD		145.23	7.58
f_6			
Mean	30	(17.5345)	**5560**
SD		50.45	25.36
f_{10}			
Mean	2	8002	**815.25**
SD		35.65	$2e^{-5}$
f_{14}			
Mean	2	(−3.8619)	**1120.25**
SD		78.45	$3.45e^{-6}$
f_{16}			
Mean	2	320	**220.25**
SD		1.25	$1.15e^{-4}$

4.4 MDBAT and other algorithms

We give the first test in order to investigate the powerful performance of the proposed MDBAT algorithm by comparing it with four benchmark algorithms (particle swarm optimization with its variants). Before discussing the comparison results of all algorithms, we present a brief description about the comparative four algorithms [21] as follows.

- *RWMPSOg* RWMPSOg is Random Walk Memetic Particle Swarm Optimization (with global variant), which combines the particle swarm optimization with random walk as a direction exploitation.
- *RWMPSOl* RWMPSOl is Random Walk Memetic Particle Swarm Optimization (with local variant), which combines the particle swarm optimization with random walk as a direction exploitation.
- *PSOg* PSOg is standard particle swarm optimization with global variant without local search method.
- *PSOl* PSOl is standard particle swarm optimization with local variant without local search method.

4.4.1 Comparison between RWMPSOg, RWMPSOl, PSOg, PSOl and MDBAT

We present the comparison results between our MDBAT algorithm and the other particle swarm optimization algorithms. We test the five comparative algorithms on 9 benchmark functions $f_1 - f_9$ and report the results in Table 2. We take the results of the other comparative algorithms from their original papers [21]. In Tables 5, 6 and 7, we report the minimum (min), maximum (max), average (Mean), standard deviation (SD) and Success rate (%Suc) of the evaluation function values over 50 runs with different population size (SS) 15, 30 and 60, respectively. The run is successful if the algorithm reaches to the global minimum of the solution within an error of 10^{-4} before the 20,000 function evaluation value. We report the best results between the comparative algorithms in boldface text. The results in Tables 5, 6 and 7 show that the proposed MDBAT algorithm succeeds and obtains the desired objective value of each function faster than the other algorithms in most cases.

4.4.2 Comparison results between GA-PSO, DE-PSO, AMPSO and MDBAT

We give the second comparison test by comparing the proposed MDBAT with other four various hybrid PSO algorithms. The first algorithm is a hybrid genetic algorithm with particle swarm optimization, which is called GA-PSO. The second algorithm is a hybrid deferential evolution algorithm with PSO algorithm, the hybrid algorithm is called DE-PSO algorithm. The third and fourth algorithms are a modified version of PSO including EP based adaptive mutation operator AMPSO1, AMPSO2 [9]. Before presenting the comparative results between the proposed MDBAT algorithm and the other four hybrid PSO algorithms, we give a brief description of each algorithm as follows.

Table 5 Comparison results of MDBAT and other PSO-based algorithms for problems $f_1 - f_9$ at $SS = 15$

Function	RWMPSOg	RWMPSOl	PSOg	PSOl	MDBAT
f_1					
Best	5324	6526	6585	6195	1471
Mean	6009.7	7318.3	10,824.8	8467.5	**1476.02**
Worst	6713	8746	24,060	10,335	1480
SD	342.9	458.6	3408.4	907.7	2.63
Suc	50	50	43	50	50
f_2					
Best	3233	3814	4830	3795	3452
Mean	9275.5	7679.0	14,898.3	8004.9	**3481**
Worst	62,928	24,288	84,735	38,940	3490
SD	11,272.3	3846.9	15,417.8	6441.3	3.245
Suc	50	50	36	50	50
f_3					
Best	4042	2213	2370	2820	1468
Mean	14,212.2	8999.1	3282.3	10,759	**1478.88**
Worst	43,767	139,085	6105	81,195	1486
SD	9229.6	19,172.8	1109.8	12,830.5	3.47
Suc	33	49	11	45	45
f_4					
Best	5115	5370	6030	6105	1874
Mean	5956.5	6588.2	8785.9	8006.4	**1884.4**
Worst	6602	7698	18,885	10,890	1894
SD	344.5	465.4	2567.2	1042.3	484.2
Suc	50	50	29	50	50
f_5					
Best	4166	2628	1230	45	5453
Mean	17,962.7	21,386.1	7727.4	27,170	**5506.86**
Worst	70,755	86,842	59,250	117,975	5653
SD	17,727.2	21,344.6	11,973.9	28,745.1	41.85
Suc	50	50	31	45	50
f_6					
Best	33,726	9844	–	9420	4470
Mean	42,746.1	12,978.2	–	12,733.5	**4490**
Worst	69,092	27,297	–	39,990	4500
SD	7086.1	2487.5	–	4123.7	6.06
Suc	42	50	0	50	50
f_7					
Best	1560	1922	1365	2175	2115.23
Mean	2094.3	2685.8	**1896.7**	2686.2	2303
Worst	3428	3290	2685	3540	2550
SD	458.0	314.6	244.8	312.1	223.45
Suc	50	50	47	50	45

Table 5 continued

Function	RWMPSOg	RWMPSOl	PSOg	PSOl	MDBAT
f_8					
Best	57145	27995	17,325	15840	3137
Mean	74,845.6	38,248.6	29,893.8	19,353.8	**5130.4**
Worst	110,259	60,191	41,955	27,060	5276
SD	108,48.2	6308.8	7954.4	2027.4	460.37
Suc	47	49	13	43	30
f_9					
Best	12,882	12,116	2070	4080	2424
Mean	20,300.4	175,79.2	8546	140,11.7	**5176.13**
Worst	41,456	28,755	48,555	58,575	5276
SD	5209.8	3315.0	10,426.9	13,167.1	486.45
Suc	49	50	30	45	35

- *GA-PSO* The GA-PSO algorithm was proposed by Kao and Zahara [11]. In GA-PSO, in order to solve a D dimensional problem, the hybrid approach takes 4D individuals which are randomly generated. These individuals may be regarded as chromosomes in the case of GA, or as particles in the case of PSO. These 4D individuals are evaluated and sorted by their fitness, and the top 2D individuals are fed into the real-coded GA to create 2D new individuals by crossover and mutation operations. They proposed a random mutation operator for the real-coded GA to modify an individual with a random number in the problems domain with a 20% probability. The created new 2D individuals from real-coded GA are used to adjust the remaining 2D particles by the PSO method.
- *DE-PSO* The DE-PSO algorithm was proposed by Pant et al. [19]. It starts as a usual DE algorithm till the trial vector is generated. If the trial vector is better than the corresponding target vector, then it is included in the population, otherwise the algorithm enters the PSO phase and generates a new candidate solution using PSO velocity and position update equations. The method is iteratively repeated till the optimum value is reached.
- *AMPSO* The AMPSO algorithm was proposed by Pant et al. [20]. The algorithm is called adaptive mutation operator particle swarm optimization. Two versions of AMPSO called AMPSO1 and AMPSO2 are proposed. In AMPSO1, the personal best particle is mutated, while in AMPSO2 the global best particle is mutated.

In Table 8, we report the comparative results between the proposed MDBAT algorithm, GA-PSO, DE-PSO, AMPSO1 and AMPSO2. We take the results of algorithm GA-PSO, DE-PSO, AMPSO1 and AMPSO2 from [29]. In order to make a fair comparison, we use the same parameters values as in other four algorithms. We report the average function evaluation results and the rate of success over 100 runs. We report the best values in boldface text.

Table 6 Comparison results of MDBAT and other PSO-based algorithms for problems $f_1 - f_9$ at $SS = 30$

Function	RWMPSOg	RWMPSOl	PSOg	PSOl	MDBAT
f_1					
Best	7507	10,021	9060	13,200	2917
Mean	8615.2	11,129.9	11,242.3	16,716	**2922.4**
Worst	9616	12,834	17,850	20,250	2927
SD	492.0	658.5	1508.1	1573.7	2.635
Suc	50	50	47	50	50
f_2					
Best	4944	7879	6450	7440	5148
Mean	10,534.2	12,220.5	12,469.7	14,337.6	**6154**
Worst	95,361	20,581	35,100	38,100	6875
SD	13,234.9	2554.3	5877.1	6673.5	739.02
Suc	50	50	29	50	50
f_3					
Best	6669	4050	3270	3720	2921
Mean	18,234.5	13,815.3	5097.3	16,848	**2929.08**
Worst	50,466	155,611	8250	141,600	2935
SD	10,670.7	21787.9	1276.8	22,935.6	3.21
Suc	46	50	22	50	46
f_4					
Best	7062	8379	8280	11,850	3917
Mean	7954.6	9881.4	9718.1	16,132.8	**3932.5**
Worst	9188	11,880	11,910	20,940	3940
SD	446.5	726.2	914.8	2203.4	3.99
Suc	50	50	47	50	50
f_5					
Best	265	174	2100	90	8909
Mean	12,425.4	18,300.2	18,210.0	28,363.2	**8910**
Worst	63,487	113,044	271,350	201,450	8915
SD	9827.5	17,947.4	44,027	38,115.6	5.86
Suc	50	50	37	50	50
f_6					
Best	50,051	19,879	15,090	17,670	9921
Mean	58,797.6	24,600.1	16,395	24,231.6	**9936.98**
Worst	69,792	30,381	17,700	45,810	9948
SD	4651.9	2378.5	1305	3791.3	6.47
Suc	48	50	2	50	50
f_7					
Best	2110	3738	2370	3180	4253
Mean	3412.2	4710.6	**3272.4**	4657.8	4484.75
Worst	6635	5759	4350	5550	4687
SD	825.8	562.9	336.7	520.9	178.28
Suc	50	50	49	50	46

Table 6 continued

Function	RWMPSOg	RWMPSOl	PSOg	PSOl	MDBAT
f_8					
Best	63,590	32,514	19,110	30,300	2523
Mean	78,764.5	39,916.5	24,277.9	36,608.6	**5815.82**
Worst	93710	44,470	35,370	42,300	6026
SD	6701.0	2958.3	4173.8	2478.8	752.96
Suc	49	50	19	49	40
f_9					
Best	22,377	11,422	4740	9570	5654
Mean	34,710.7	19,030.9	17,416.6	21,287.1	**6013.6**
Worst	50,848	56,181	179,490	97,830	6040
SD	5818.2	8005.8	29,615.7	19,328.5	62.62
Suc	50	50	38	49	40

The results in Table 8 show that, the proposed MDBAT are better in 9 test cases out of 11 cases. We can conclude from Fig. 2, that the proposed algorithm is a promising algorithm and faster than the other algorithms.

4.5 Wilcoxon signed-ranks test

Wilcoxons test is a nonparametric procedure employed in a hypothesis testing situation involving a design with two samples [10, 24, 39]. It is a pairwise test that aims to detect significant differences between the behavior of two algorithms. ρ is the probability of the null hypothesis being true. The result of the test is returned in $\rho < 0.05$ indicates a rejection of the null hypothesis, while $\rho > 0.05$ indicates a failure to reject the null hypothesis. The R^+ is the sum of positive ranks, while R^- is the sum of negative ranks.

We apply the Wilcoxon signed-ranks test on the results in Tables 5, 6 and 7. In Tables 9, 10 and 11, we report the comparison by the test between the MDBAT algorithm and RWMPSOg, RWMPSOl, PSOg and PSOl. Also, we apply the Wilcoxon signed-ranks test on the results in Table 8 and report the comparison by test between the MDBAT algorithm and GA-PSO, DE-PSO, AMPSO1, AMPSO2 in Table12.

The statistical results in Tables 9, 10 and 11 show that the proposed algorithm is a promising algorithm and outperform the other algorithms.

5 Conclusion and future work

We propose a novel hybrid bat algorithm and multi-directional search algorithm in order to solve unconstrained global optimization problems. We call the proposed algorithm by multi-directional bat algorithm (MDBAT). In MDBAT, the standard bat algorithm starts the search for number of iterations then the multi-directional

Table 7 Comparison results of MDBAT and other PSO-based algorithms for problems $f_1 - f_9$ at $SS = 60$

Function	RWMPSOg	RWMPSOl	PSOg	PSOl	MDBAT
f_1					
Best	11,757	16,116	14,280	24,780	5816
Mean	13,604.3	18,208.7	16,360.0	34,054.8	**5819.78**
Worst	15,153	20,323	19,140	40,920	5827
SD	660	947.2	933.3	3769.3	2.15
Suc	50	50	48	50	50
f_2					
Best	6915	10,458	8880	14,040	8145
Mean	15,515.3	20,687.0	264,200	26,443.2	**8503.25**
Worst	54,052	37,905	156,900	76,740	9145
SD	20,835.7	5662.9	27,337.6	10,755	450.06
Suc	50	50	39	50	50
f_3					
Best	6812	4622	5340	10,020	5825
Mean	19,494.8	16,489.6	10,042.5	24,465.6	**5827.28**
Worst	80,063	75,139	22,860	58,560	5833
SD	15,016.4	12,050.7	3103.7	9765	3.54
Suc	50	50	40	50	44
f_4					
Best	10,488	14,520	12,240	23,520	5821
Mean	12,142.2	16,858.4	14,891.0	31,506	**5832**
Worst	13,876	21,845	18,360	39,600	5383
SD	786.7	1417.6	1176.4	3796.8	4.09
Suc	50	50	49	50	40
f_5					
Best	4826	271	3120	180	10,809
Mean	11,019.3	18,080.2	11,877.3	27,240	**10,810**
Worst	37,069	56,900	66,600	135,960	10,815
SD	6057.7	12,403.9	12,168.4	27,689.4	1.90
Suc	50	50	44	50	50
f_6					
Best	69,607	36,395	21,840	39,120	15,882
Mean	79,473.6	48,110.9	25,116	47,902.8	**15,835.12**
Worst	95,133	59,773	30,300	60,420	15,843
SD	5080.0	5710.3	2171.2	4924.5	5.52
Suc	50	50	20	50	50
f_7					
Best	4569	7230	4620	7020	8005
Mean	5781.5	8765.1	5616	8769.6	**8010.1**
Worst	8238	10001	6720	10800	8045
SD	629.4	739.1	401.1	714.3	8.29
Suc	50	50	50	50	45

Table 7 continued

Function	RWMPSOg	RWMPSOl	PSOg	PSOl	MDBAT
f_8					
Best	77,465	62,821	26,100	64,080	5914
Mean	89,876.7	74,476.9	30,172.7	74,986.8	**7305.08**
Worst	106,207	83,195	33,840	120,360	7526
SD	7017.4	5460.9	1770.7	7833	382.89
Suc	49	50	33	50	42
f_9					
Best	34,611	8951	6300	17,280	7247
Mean	54,114.9	20,441.7	39,581.1	25,918.8	**7520.42**
Worst	144,452	74,149	427,680	59,520	7526
SD	19,734.5	13,287.7	91,879.1	6634.4	39.45
Suc	50	50	38	50	43

search starts the search from the best obtained solution from the bat algorithm. The hybridization between the bat algorithm and multi-directional search can accelerate the convergence instead of letting the algorithm runs for more iterations without any improvements. We apply the standard bat algorithm without combining it with the multi-directional search algorithm and the proposed algorithm on six functions which are randomly selected. Also, we report the mean and the standard deviation over 50 runs. We consider the run is successful if the algorithm reaches to the global minimum of the solution within an error of 10^{-4} before the 20,000 function evaluation value. If any algorithm fails to obtain the desired function value, then we report the value of best obtained function value in parentheses. Further, we show that the combination between the standard bat algorithm and the multi-directional search can accelerate the search and obtain the desired function values for all test functions faster than the standard bat algorithm which needs more iterations in order to obtain the desired function values.

We verify the robustness and the effectiveness of the proposed algorithm by applying it on 16 unconstrained global optimization problems and compare it against 8 various particle swarm optimization algorithms, namely, random walk memetic particle swarm optimization (with global and local variants), standard particle swarm optimization with global and variants with local search method, adaptive mutation operator particle swarm optimization, genetic algorithm with PSO, and differential evaluation with PSO.

Finally, the experimental results show that the proposed algorithm is a promising algorithm and has a powerful ability to solve unconstrained optimization problems faster than other algorithms in most cases.

Our future work is concentrated on the following directions:

- Apply the proposed algorithms on solving constrained optimization and engineering problems [1] such as design of a tension/compression spring [5],

Table 8 Mean error of the function values of GA-PSO, DE-PSO, AMPSO1, AMPSO2 and MDBAT algorithms

Function	D	Mean	GA-PSO	DE-PSO	AMPSO1	AMPSO2	MDBAT
f_1	2	Function evaluation	206	938	980	620	**116.74**
		Error	0.0004	$1.67e^{-5}$	$4.23e^{-5}$	$3.34e^{-5}$	$\mathbf{4.84e^{-7}}$
		Success rate%	100	100	100	100	100
f_2	2	Function evaluation	140,894	2351	2668	2750	**120**
		Error	0.00064	$2.46e^{-5}$	$2.16e^{-5}$	$3.03e^{-5}$	**0.00**
		Success rate%	100	100	100	100	100
f_{10}	2	Function evaluation	25,706	1155	1120	1196	**740.88**
		Error	0.00012	$3.67e^{-5}$	$3.81e^{-8}$	$\mathbf{9.89e^{-6}}$	$6.6e^{-6}$
		Success rate%	100	100	100	100	100
f_{11}	2	Function evaluation	8254	1148	754	2778	**662.4**
		Error	0.00009	$2.29e^{-6}$	$1.52e^{-5}$	$1.49e^{-5}$	$\mathbf{2.23e^{-6}}$
		Success rate%	100	100	100	100	100
f_{12}	2	Function evaluation	96,211	3689	2132	2146	**775.2**
		Error	0.00007	$3.03e^{-6}$	$1.72e^{-5}$	$\mathbf{2.54e^{-7}}$	$1.45e^{-6}$
		Success rate%	100	100	100	100	100
f_{13}	2	Function evaluation	809	1792	1258	1432	**623.26**
		Error	0.00003	$1.67e^{-5}$	$4.10e^{-5}$	$3.71e^{-5}$	$\mathbf{4.23e^{-6}}$
		Success rate%	100	100	100	100	100
f_{14}	3	Function evaluation	2117	1059	1020	1132	**981.22**
		Error	0.00020	$1.74e^{-5}$	$1.96e^{-5}$	$2.04e^{-5}$	$\mathbf{2.48e^{-5}}$
		Success rate%	100	100	100	100	100
f_{15}	2	Function evaluation	**174**	928	622	679	791.1
		Error	0.00001	$1.11e^{-5}$	$\mathbf{1.17e^{-6}}$	$2.64e^{-5}$	$7.47e^{-5}$
		Success rate%	100	100	100	100	100
f_{16}	2	Function evaluation	**95**	834	756	780	184
		Error	0.00005	$1.12e^{-5}$	$2.54e^{-5}$	$2.61e^{-6}$	$\mathbf{6.82e^{-9}}$
		Success rate%	100	100	100	100	100
f_{16}	5	Function evaluation	398	2677	1992	2018	**381**
		Error	0.00000	$3.49e^{-5}$	$3.44e^{-5}$	$5.07e^{-5}$	$\mathbf{2.75e^{-10}}$
		Success rate%	100	100	100	100	100
f_{16}	10	Function evaluation	872	7258	5980	6218	**671**
		Error	0.00000	$6.91e^{-5}$	$7.41e^{-5}$	$9.11e^{-5}$	$\mathbf{3.96e^{-11}}$
		Success rate%	100	100	100	100	100

design of a welded beam [22], design of a gear train [23], and design of a pressure vessel [23].

- Modify our proposed algorithm in order to solve other combinatorial problems, large scale integer programming and minimax problems [2, 3, 26].

Table 9 Wilcoxon test for comparison results in Table 5

Compared methods		Solution evaluations			
Method 1	Method 2	R^-	R^+	ρ-value	Best method
MDBAT	RWMPSOg	1	44	0.010862	MDBAT
MDBAT	RWMPSOl	0	45	0.007686	MDBAT
MDBAT	PSOg	1	35	0.017290	MDBAT
MDBAT	PSOl	0	45	0.007686	MDBAT

Table 10 Wilcoxon test for comparison results in Table 6

Compared methods		Solution evaluations			
Method 1	Method 2	R^-	R^+	ρ-value	Best method
MDBAT	RWMPSOg	1	44	0.010862	MDBAT
MDBAT	RWMPSOl	0	45	0.007686	MDBAT
MDBAT	PSOg	1	44	0.010862	MDBAT
MDBAT	PSOl	0	45	0.007686	MDBAT

Table 11 Wilcoxon test for comparison results in Table 7

Compared methods		Solution evaluations			
Method 1	Method 2	R^-	R^+	ρ-value	Best method
MDBAT	RWMPSOg	0	45	0.007686	MDBAT
MDBAT	RWMPSOl	0	45	0.007686	MDBAT
MDBAT	PSOg	0	45	0.007686	MDBAT
MDBAT	PSOl	0	45	0.007686	MDBAT

Table 12 Wilcoxon test for comparison results in Table 8

Compared methods		Solution evaluations			
Method 1	Method 2	R^-	R^+	ρ-value	Best method
MDBAT	GA-PSO	8	58	0.026231	MDBAT
MDBAT	DE-PSO	0	66	0.003346	MDBAT
MDBAT	AMPSO1	10	56	0.040860	MDBAT
MDBAT	AMPSO2	1	65	0.004439	MDBAT

- Modify our proposed algorithm to solve large scale unconstrained global optimization problems and molecular potential energy function as done by the authors of this paper in [4, 27].

Acknowledgements We are thankful to the anonymous referees for thoughtful comments which help improve the manuscript substantially.The research of the Mohamed A. Tawhid is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC). The postdoctoral fellowship of the Ahmed F. Ali is supported by NSERC.

References

1. Ali, A.F., Tawhid, M.A.: A hybrid PSO and DE algorithm for solving engineering optimization problems. Appl. Math. Inf. Sci. **10**(2), 431–449 (2016)
2. Ali, A.F., Tawhid, M.A.: Direct gravitational search algorithm for tackling global optimization problems. East Asian J. Appl. Math. **6**(3), 290–313 (2016)
3. Ali, A.F., Tawhid, M.A.: A hybrid cuckoo search algorithm with nelder mead method for solving global optimization problems. SpringerPlus **5**, 473 (2016)
4. Ali, A.F., Tawhid, M.A.: Hybrid particle swarm optimization and genetic algorithm for minimizing potential energy function. Ain Shams Eng. J. (2016). doi:10.1016/j.asej.2016.07.008
5. Arora, J.S.: Introduction to Optimum Design. McGraw-Hill, New York (1989)
6. Chu, S.C., Tsai, P., Pan, J.S.: Cat swarm optimization. In: Yang, Q., Webb, G. (eds.) PRICAI 2006: trends in artificial intelligence. PRICAI 2006. Lecture notes in computer science, vol. 4099, pp. 854–858. Springer, Berlin (2006)
7. Dennis, J.E., Torczon, V.: Direct search methods on parallel machines. SIAM J. Optim. **1**(4), 448–474 (1991)
8. Dorigo, M.: Optimization, learning and natural algorithms. Ph.D. thesis, Politecnico di Milano, Italy (1992)
9. Firouzi, B., Niknam, T., Nayeripour, M.: A new evolutionary algorithm for cluster analysis. World Acad. Sci. Eng. Technol. **36**, 605–609 (2008)
10. Garcia, S., Fernandez, A., Luengo, J., Herrera, F.: A study of statistical techniques and performance measures for genetics-based machine learning, accuracy and interpretability. Soft Comput. **13**, 959–977 (2009)
11. Kao, Y.T., Zahara, E.: A hybrid genetic algorithm and particle swarm optimization for multimodal functions. Appl. Soft Comput. **8**, 849–857 (2008)
12. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Glob. Optim. **39**(3), 459–471 (2007)
13. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. Proc. IEEE Int. Conf. Neural Netw. **4**, 1942–1948 (1995)
14. Komarasamy, G., Wahi, A.: An optimized K-means clustering technique using bat algorithm. Eur. J. Sci. Res. **84**(2), 263–273 (2012)
15. Li, X.L., Shao, Z.J., Qian, J.X.: Optimizing method based on autonomous animats: fish-swarm algorithm. Xitong Gongcheng Lilun yu Shijian/Syst. Eng. Theory Pract. **22**(11), 32 (2002)
16. Lin, J.H., Chou, C.W., Yang, C.H., Tsai, H.L.: A chaotic Levy flight bat algorithm for parameter estimation in nonlinear dynamic biological systems. J. Comput. Inf. Technol. **2**(2), 56–63 (2012)
17. Nakamura, R.Y.M., Pereira, L.A.M., Costa, K.A., Rodrigues, D., Papa, J.P., Yang, X.S.: BBA: a binary bat algorithm for feature selection. In: 25th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), 22–25, IEEE Publication, pp. 291–297 (2012)
18. Passino, M.K.: Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Syst. **22**(3), 52–67 (2002)
19. Pant, M., Thangaraj, R., Abraham, A.: DE-PSO: a new hybrid meta-heuristic for solving global optimization problems. New Math. Nat. Comput. **7**(03), 363–381 (2011)
20. Pant, M., Thangaraj, R., Abraham, A.: Particle swarm optimization using adaptive mutation. In: Proceedings of 19th International Conference on Database and Expert Systems Application, Italy, pp. 519–523 (2008)
21. Petalas, Y.G., Parsopoulos, K.E., Vrahatis, M.N.: Memetic particle swarm optimization. Ann. Oper. Res. **156**, 99–127 (2007)
22. Rao, S.S.: Engineering Optimization-Theory and Practice. Wiley, New Delhi (1994)
23. Sandgen, E.: Nonlinear integer and discrete programming in mechanical design optimization. J. Mech. Des. (ASME) **112**, 223–229 (1990)
24. Sheskin, D.J.: Handbook of Parametric and Nonparametric Statistical Procedures. CRC Press, Boca Raton (2003)
25. Tang, R., Fong, S., Yang, X.S., Deb, S.: Wolf search algorithm with ephemeral memory. In: Digital Information Management (ICDIM), 2012 Seventh International Conference on Digital Information Management, pp. 165–172 (2012)
26. Tawhid, M.A., Ali, A.F.: Simplex particle swarm optimization with arithmetical crossover for solving global optimization problems. OPSEARCH **53**(4), 705–740 (2016). Springer

27. Tawhid, M.A., Ali, A.F.: A hybrid social spider optimization and genetic algorithm for minimizing molecular potential energy function. Soft Comput. (2016). doi:10.1007/s00500-016-2208-9.
28. Teodorovic, D., DellOrco, M.: Bee colony optimization a cooperative learning approach to complex transportation problems. In: Advanced OR and AI Methods in Transportation: Proceedings of 16th MiniEURO Conference and 10th Meeting of EWGT (13–16 September 2005). Publishing House of the Polish Operational and System Research, Poznan, pp. 51–60 (2005)
29. Thangaraj, R., Pant, M., Abraham, A., Bouvry, P.: Particle swarm optimization: hybridization and experimental illustrations. Appl. Math. Comput. **217**, 5208–5226 (2011)
30. Torczon, V.: Multi-directional Search: A Direct Search Algorithm for Parallel Machines. Department of Mathematical Sciences, Rice University, Houston (1989)
31. Wang, G., Guo, L.: A novel hybrid bat algorithm with harmony search for global numerical optimization. J. Appl. Math. (2013). doi:10.1155/2013/696491
32. Xie, J., Zhou, Y., Chen, H.: A novel bat algorithm based on differential operator and Lévy flights trajectory. Comput. Intell. Neurosci. (2013). doi:10.1155/2013/453812
33. Yang, X.S.: Firefly algorithm, stochastic test functions and design optimization. Int. J. Bio-Inspired Comput. **2**(2), 78–84 (2010)
34. Yang, X.S.: Firefly algorithms for multimodal optimization. In: Watanabe, O., Zeugmann, T. (eds.) Stochastic Algorithms: Foundations and Applications. SAGA 2009. Lecture Notes in Computer Science, vol. 5792, pp. 169–178. Springer, Berlin (2009)
35. Yang, X.S.: A new metaheuristic bat-inspired algorithm. In: González, J.R., Pelta, D.A., Cruz, C., Terrazas, G., Krasnogor, N. (eds.) Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), vol. 284, pp. 65–74. Springer, Berlin (2010)
36. Yang, X.S.: Bat algorithm for multi-objective optimization. Int. J. Bio-Inspired Comput. **3**(5), 267–274 (2011)
37. Yang, W.Y., Cao, W., Chung, T.-S., Morris, J.: Applied Numerical Methods Using MATLAB. Wiley, Hoboken (2005)
38. Zhang, J.W., Wang, G.G.: Image matching using a bat algorithm with mutation. Appl. Mech. Mater. **203**(1), 88–93 (2012)
39. Zar, J.H.: Bio-Statistical Analysis. Prentice Hall, Englewood Cliffs (1999)