An Adaptive Bat Algorithm with Memory for Global Optimization

Chunfeng Wang, Wenxin Song, Lixia Liu

***Abstract*—Bat algorithm (BA) is a relative new method proposed by Yang in 2010. It has been successfully used to solve different global optimization problems. However, it may also lead to premature convergence. For overcoming this disadvantage, this paper presents three improvement strategies, which can be used to enhance the local and global search ability of BA. After that, An adaptive bat algorithm with memory (ABAM) is proposed. In this algorithm, to balance between the local and global search capability, a new search equation is proposed for each bat based on the history optimal position that it memorized; to avoid falling into a local optimal solution, an abandoned mechanism is adopted for each bat based on the number of its position has not been improved; to enhance the local search ability, a dynamic adaptive search equation is presented. To verify the performance of our algorithm, 23 standard test functions are employed. The experimental results show that the algorithm is much more robust and efficient than some state-of-the-art algorithms.**

***Index Terms*—Bat algorithm; Abandoned mechanism; Continuous optimization; Global optimization; Adaptive strategy.**

I. Introduction

We consider the following global optimization problem in this paper:

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & x \in X, \end{array} \tag{1}$$

where $f(x) : X \to R$ is a continuous real-valued function, $x \in X \subset R^n$ is defined by the bound constraint $l_j \leq x_j \leq u_j,\ j = 1, \cdots, n$.

A lot of problems in engineering can be converted into optimization problem (1). Since many objective functions of (1) are multimodal, highly nonlinear, with flat regions and irregularities, it could be a very challenging task to find a global optimal solution. For solving problem (1), many algorithms have been developed, which are generally divided into two algorithms:deterministic and stochastic algorithms. Deterministic algorithms, such as conjugate method, Newton method and steepest decent, do not contain any operators that cause randomness, and produce the same results as long as the initial conditions remain constant. However, with the increasing complexity of problem (1), such as nonconvex, nondifferential etc, deterministic algorithms generally fail to handle them. Therefore, stochastic algorithms have been paid more attention recently, and many effective algorithms have been presented, which are inspired from physical and natural phenomena, include Firefly Algorithm (FA) [1], Glowworm Swarm Optimization (GSO) [2], Invasive Weed Optimization (IWO) [3], Genetic Algorithm(GA)[4,5], Differential Evolution(DE) [6], Particle Swarm Optimization(PSO) [7], Ant Colony Optimization(ACO) [8], Artificial Bee Colony (ABC)[9], Harmony Search (HS) [10], Bat Algorithm(BA) [11], League Championship Algorithm (LCA) [12], Water Cycle Algorithm (WCA) [13], Mine Blast Algorithm (MBA) [14], Whale Optimization Algorithm (WOA) [15], Cockoo Search Algorithm (CSA) [16] and so on.

Among these algorithms, BA is a popular swarm-intelligence-based algorithm inspired from echolocation phenomenon of bat, which was introduced by Xin-She Yang [11]. In BA, by echolocation, each bat can detect the orientation, the type , and the speed of the prey. BA has been demonstrated that it outperform some well-known nature inspired optimization techniques and increasingly applied to different areas.

Although BA has many advantages, it can be trapped in local optima and has a weak convergence rate in a later period. As pointed in [17], its performance diminishes as the dimension of problem increases. Therefore, many variants of BA have been proposed to improve its performance. For example, to enhance the global search behavior of BA, several different chaotic maps were embedded into bat algorithm [18]. By combining BA and harmony search algorithm, a hybrid BA was proposed [19]. To improve the exploration ability of BA, a new bat algorithm was presented based on using complex-value [20]. By using levy flights and chaotic maps, a chaotic bat algorithm was proposed [21]. Recently, to improve the local and global search ability of BA, an Enhance Bat Algorithm (EBA) was developed based on three different techniques [22]. For other research work on bat algorithm, the reader can refer to the literatures [23-32].

The aim of this paper is to present an adaptive bat algorithm with memory (ABAM). In this algorithm, three improvement strategies are presented. The first improvement strategy is that each bat can memorize its history optimal position, which can be used to guide the search process. The second improvement strategy is that an adaptive search equation is used in the best solution of the current iteration, which can be used to enhance the local search ability. The third improvement strategy is that each bat can determine whether a position should be abandoned, which can be used to avoid premature. Based on these improvement strategies, a new bat algorithm is presented. To verify the performance of our algorithm, 23 standard test functions are employed. The experimental results show that the algorithm is much more robust and efficient than some state-of-the-art algorithms.

The remainder of this paper is organized as follows.

Manuscript received Oct. 30, 2017. The research was supported by NSFC(11671122); Henan Normal University National Research Project to Cultivate the Funded Projects (01016400105); the NSF of ShaanXi Province of China(2017JQ1010).

Chunfeng Wang is with College of Mathematics and Information Science, Henan Normal University, Xinxiang, 453007, PR China. E-mail:wangchunfeng10@126.com

Wenxin Song is with College of Mathematics and Information Science, Henan Normal University, Xinxiang, 453007, PR China.

Lixia Liu is School of Mathematics and Statistics, Xidian University, Xi'an, 710126, PR China.

Section 2 describes the original BA. Then our proposed modifications to BA are described in Section 3. Numerical results and discussions are presented in Section 4. Finally, some concluding remarks are provided in Section 5.

II. Basic bat algorithm (BA)

Assume that the search space is n-dimensional, M denotes the size of the swarm population. Let x_i^t be the position of the i-th bat, v_i^t be its velocity, f_i be frequency, A_i^t be loudness, r_i be the emission pulse rate.

In BA, each position denotes a feasible solution. The initial population can be generated randomly by the following equation:

$$x_{ij} = x_j^l + rand(0,1) * (x_j^u - x_j^l),$$

where $i = 1, \cdots, M,\ j = 1, \cdots, n$, x_j^l and x_j^u are lower and upper boundaries of dimension j respectively.

After random initialization, for the i-th bat, if its position and its velocity are x_i^{t-1} and v_i^{t-1} respectively at time step $t-1$, then its new position x_i^t and new velocity v_i^t can be updated according to the following formulas:

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

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

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

where β is a random number in [0,1]; f_{min} and f_{max} are the minimum and maximum frequencies, x^* is the best position that bats have experienced till iteration t.

At iteration t, if $rand > \gamma_i^t$, then a new solution for each bat will be generated by using the following formula:

$$x_{new} = x^* + \epsilon * A^t, \tag{4}$$

where $\epsilon \in [-1, 1]$ is a random vector number, x^* is selected among the best positions that bats have experienced till iteration t, $A^t =< A_i >^t$ is the average loudness of all bats at this time step, γ_i^t is the pulse emission rate value of the i-th bat at iteration t.

In BA, to mimic the feature that when the i-th bat is homing for its prey, the algorithm will decrease its loudness A_i while increase its pulse emission rate γ_i:

$$A_i^{t+1} = \alpha * A_i^t,\ \gamma_i^{t+1} = \gamma_i^0 * (1 - exp(-\gamma * t)), \tag{5}$$

where α and γ are constants. For any $0 < \alpha < 1$ and $\gamma > 0$, we have

$$A_i^{t+1} \to 0,\ \gamma_i^t \to \gamma_i^0, \text{as } t \to \infty, \tag{6}$$

The pseudo code of BA is given in Algorithm 1.

Algorithm 1: Pseudo code of BA

01: Initialize bat population x_i and velocity v_i, pulse rate γ_i and loudness A_i, $i = 1, \cdots, M$.
02: Define pulse frequency f_i at x_i.
03: **While** $t < Itermax$ **do** % $Itermax$ is the maximum number of iteration.
04: Generate new positions by adjusting frequency, and updating velocities and location.
05: If $rand > \gamma_i$
06: Select a solution among the best solutions, and generate a local solution around the selected best solution.
07: End if
08: If $rand < A_i$ and $f(x_i) < f(x^*)$
09: Accept new position, increase γ_i and reduce A_i.
10: End if
11:Find the current best x^*.
12:End while
13: Output the optimal results

III. The new bat algorithm (ABAM)

By analyzing the main strategies used in the basic bat algorithm, it is not difficult to find that BA has difficulty in keeping balance between exploration and exploitation, and it may suffer from premature convergence problem. To tackle with these problems, three improvement strategies are proposed.

A. The first improvement strategy

From (2), it can be seen that, only the information of the global best solution x^* of the bats swarm is used. As pointed by [22], this velocity equation may cause the algorithm to fall into the local optimal solution. So, to balance global and local search intensity, a modified equation was proposed by [22]:

$$v_i^{t+1} = \omega * v_i^t + (x_i^t - x^*) * f_i,$$

where ω is a inertia weight factor. The modified equation improved the performance of the basic BA, however, just like human cognitive process, the history experience of individual may play important role, therefore, if the information of the history experience of individual is adopted, then the search behavior may be greatly improved. Based on such consideration, we propose a modified velocity equation as follows:

$$v_i^{t+1} = \omega * v_i^t + (x_i^t - \frac{x^* + x_{pi}^*}{2}) * f_i, \tag{7}$$

where ω is a inertia factor that balances global and local search intensity of the i-th solution, which is linearly decreased along with the iteration of the algorithm, and can be set as follows:

$$\omega = a * exp(b * t^2),\ a = w_{\max} * exp(-b),\ b = \frac{1}{(Itermax)^2 - 1};$$

x_{pi}^* is the history optimal position of the i-th bat.

From (7), we can see that, the information of the history best solution of the i-th bat is considered, which represents the memory of the bat. This equation will allow bat to explore broadly the whole of the search space, enhance diversity of the swarm and alleviating premature convergence.

B. The second improvement strategy

In BA, the role of the equation (4) is to provide local search with guidance of the best solution. Generally speaking, at the early stage of the algorithm, the current optimal solution x^* may be far away from the true optimal solution, and the current optimal solution x^* should be more and more close to the true optimal solution, so the search range at x^* should be smaller and smaller. However, the search equation (4) can not be very good at achieving this goal. To achieve such a purpose, a new search equation for local search is proposed as follows:

$$x_{new} = x^* + \varphi * (x^* - x_i), \tag{8}$$

where φ is a random number, and generated as follows:

$$\varphi = (2 * rand - 1) * (2 - t * \frac{2}{Itermax}), \tag{9}$$

here, t denotes the current number of iterations, $Itermax$ is the maximum number of iterations.

From (9), it can be seen that, φ is a random number in the interval $[-r, r]$, where r is linearly decreased from 2 to 0 over the course of iterations. This strategy can guarantee the local search of the algorithm is carried out near the current optimal solution x^* in the late stage of the algorithm.

C. The third improvement strategy

Inspired by the bee colony algorithm, in our algorithm, an abandoned mechanism is proposed to avoid premature. The detail is given below.

Assume that "L" is a predetermined number, which is used to determine the position x_i of the i-th bat whether should be abandoned. If the position x_i can not be improved anymore when L is achieved, this means that x_i may be a local optimal solution, then the position should be abandoned, and a new position x_{new} should be generated to replace it. To enable the algorithm to explore broadly the whole of the search space, and use the information of the current best solution at the same time, a new position x_{new} can be generated by using the following equation

$$x_{new} = (1 - \omega) * x^* + \phi * (x^* - x_i), \qquad (10)$$

where ϕ is a random number in the range of [-1,1]; ω is defined as (7).

From (10), it can be seen that, when x_{new} is generated, to avoid too much randomness, the information of the current best solution x^* is utilized. At the same time, according to the change of ω, x_{new} is obtained by a large random in the early stage, so it can explore the search space broadly; with the current best optimal solution x^* more and more close to the true optimal solution, it is obtained in the near of the global best position x^*.

Based on the above mentioned explanation, the pseudo code of the ABAM algorithm is given in Algorithm 2.

Algorithm 2: Framework of ABAM

01:Initialize a population of M bats with random positions x_i in the search space, and random velocities v_i; the maximum iteration $maxcycle$; ω_{min}, ω_{max};f_{min}, f_{max};the limit L of a position abandoned.
02: Set $x^*_{pi} = x_i (i = 1, \cdots, M)$ and find the current best solution x^*.
03: **While** $t \leq Itermax$ **do** % $Itermax$ is the maximum number of iteration.
04: For $i = 1$ to M **do**
05: For $j = 1$ to n **do**
06: By (1) and (7), update the velocity of each bat.
07: By (3), update the position of each bat.
08: End for
09: If $rand > r_i$
10: Select a solution among the best solutions
11: By (8), generate a local solution around the selected best solution.
12: End if
13: If $rand < A_i$ and $f(x_i) < f(x^*)$
14: Set $x^* = x_i$;
15: Increase r_i, reduce A_i.
16: End if
17: If $f(x_i) < f(x^*_{pi})$
18: $x^*_{pi} = x_i$, set $L_i = 0$;
19: Else
20: Set $L_i = L_i + 1$.
21: End if
22: End for
23: For $i = 1$ to M **do**
24: If $L_i = L$
25: By (10), to generate a new position, and replace x_i.
26: End if
27: End for
28: $t = t + 1$
29: End while

IV. Experimental results and discussion

A. Experiment 1-comparisons among BAs

In experiment 1, to evaluate the performance of proposed ABAM algorithm, it is applied to minimize 23 benchmark functions with different characteristics, including formulations, bounds of search space, global minimum values as seen in Table I. These benchmark functions are widely used to test the performance of global optimization algorithms. They cover different types, include unimodal , multimodal and rotated. Generally speaking, unimodal functions are used to test the convergence speed, while multimodal functions are used to detect whether the algorithm faces the problem of premature convergence.

TABLE I: Benchmark functions used in experiment 1.

Functions	Range	Optimal value
$f_1 = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$	[-10,10]	0
$f_2 = 4x_1^2 - 2.1x_1^4 + (x_1^6)/3 + x_1x_2 - 4x_2^2 + 4x_2^4$	[-5,5]	-1.0316
$f_3 = \sum_{i=1}^{n} x_i^2$	[-100,100]	0
$f_4 = \sum_{i=1}^{n} \mid x_i \mid + \prod_{i=1}^{n} \mid x_i \mid$	[-10,10]	0
$f_5 = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	[-100,100]	0
$f_6 = \max_i \{\mid x_i \mid, 1 \leq i \leq n\}$	[-100,100]	0
$f_7 = \sum_{i=1}^{n} ix_i^2$	[-10,10]	0
$f_8 = \sum_{i=1}^{n} ix_i^4$	[-1.28,1.28]	0
$f_9 = \sum_{i=1}^{n} \mid x_i \mid^{i+1}$	[-1,1]	0
$f_{10} = \sum_{i=1}^{n} (10^6)^{\frac{i-1}{n-1}} x_i^2$	[-100,100]	0
$f_{11} = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	[-1.28,1.28]	0
$f_{12} = \sum_{i=1}^{n} ix_i^4 + random[0, 1)$	[-1.28,1.28]	0
$f_{13} = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	[-5.12,5.12]	0
$f_{14} = -20\exp(-0.2 * \sqrt{\sum_{i=1}^{n} x_i^2/n}) - \exp(\sum_{i=1}^{n} \cos(2\pi x_i/n) + 20 + e$	[-32,32]	0
$f_{15} = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1$	[-600,600]	0
$f_{16} = 0.5 + \frac{sin^2(\sqrt{\sum_{i=1}^{n} x_i^2}) - 0.5}{(1+0.001(\sum_{i=1}^{n} x_i^2))^2}$	[-100,100]	0
$f_{17} = \frac{\pi}{n} \sum_{i=1}^{n} (x_i^4 - 16x_i^2 + 5x_i)$	[-5,5]	-78.3323
$f_{18} = \sum_{i=1}^{n} \mid x_i sin(x_i) + 0.1x_i \mid$	[-10,10]	0
$f_{19} = \sum_{i=1}^{n} (y_i^2 - 10\cos(2\pi y_i) + 10)$, $if \mid x_i \mid < \frac{1}{2}, y_i = x_i; else\ y_i = \frac{round(2x_i)}{2}$	[-5.12,5.12]	0
$f_{20} = \frac{1}{n}[10sin^2(pi * y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2(1 + 10sin^2(pi * y_{i+1})) + (y_n - 1)^2] + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$, where $y_i = 1 + \frac{1}{4}(x_i + 1)$		
$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a \leq x_i \leq a, \\ k(-x_i - a)^m, & x < -a. \end{cases}$	[-50,50]	0
$f_{21} = \sum_{i=1}^{n} (20^{\frac{i-1}{n-1}} z(i))^2,\ Z = x * M$	[-100,100]	0
$f_{22} = \sum_{i=1}^{n} (1000 * z(1))^2 + \sum_{i=2}^{n} z_i^2,\ Z = x * M$	[-100,100]	0
$f_{23} = \sum_{i=1}^{n} (z_i^2 - 10\cos(2\pi z_i) + 10)\ Z = x * M$	[-5.12,5.12]	0

Functions $f_1 - f_2$ are two lower dimensional, and $f_3 - f_{22}$ are scalable problems. Functions $f_3 - f_{10}$ are continuous and unimodal. Function f_{11} is a discontinuous step function.

Function f_{12} is a quartic function with noise. $f_{13} - f_{20}$ are multimodal functions and the number of their local minima increases exponentially with the problem size. $f_{21} - f_{23}$ are three rotated functions.

In order to comprehensively verify the effectiveness of ABAM, its performance is compared with the basic BA [11] and other two state-of-the-art algorithms, i.e. IS1 [22], ABA [33]. The proposed algorithm (ABAM) and other three algorithms are coded in Matlab 7.0, and the experiments platform is a personal computer with Pentium 4, 3.06GHz CPU, 512M memory and Windows XP.

For all the algorithms, the stop criterion is that maximum number of iterations reaches $maxcycle$. For each benchmark function, each algorithm independently run with each algorithm 30 times.

The parameters of algorithms are given as below. For all algorithms, the population size $M = 50$, the maximum number of iterations $maxcycle = 2500$. The common parameters of BA, ABA,IS1 and ABAM: $f_{min} = 0$, $f_{max} = 1$, $\alpha = 0.9$, $\gamma = 0.85$. The parameter of ABAM: the limit L of a position abandoned is set to $M * n$ [34].

ABAM is compared with BA, ABA and IS1, in terms of the mean value(Mean), minimum value(Min), and the standard deviation(SD) of the solutions obtained in 30 independents runs by each algorithm. The comparison results are presented in Table II. Note that, the mean and the standard deviation of solutions indicate the solution quality of the algorithms and the stability of the algorithms, respectively.

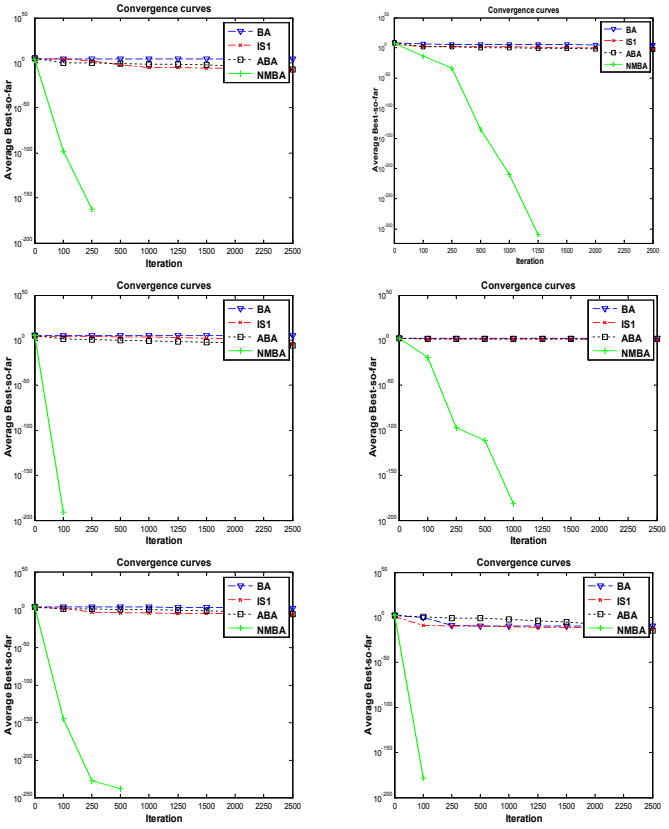

Fig. 1 Convergence rates of $f_3 - f_8$ functions

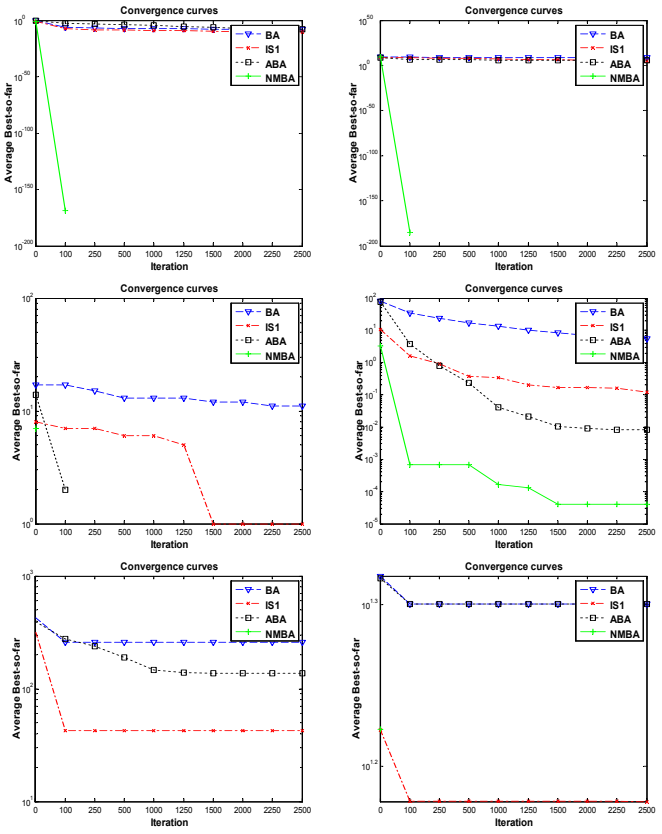

Fig. 2 Convergence rates of $f_9 - f_{14}$ functions

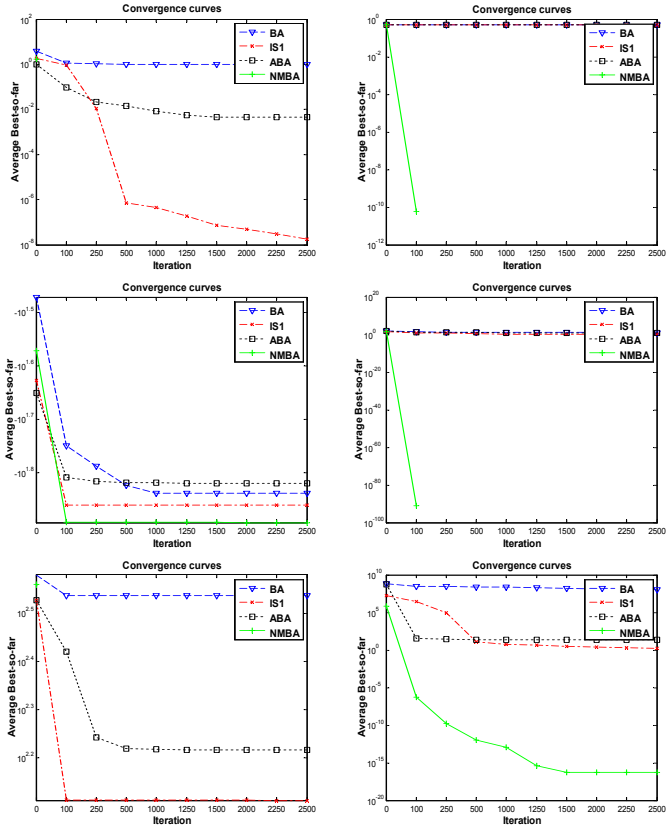

Fig. 3 Convergence rates of $f_{15} - f_{20}$ functions

For statistical comparison, a well known nonparametric statistical hypothesis test-the two-tailed wilcoxon signed-rank test is conducted to compare the significance between the ABAM algorithm and other algorithms at $\alpha = 0.05$ significance level in Table II. The sign "+" indicates that

ABAM is better than its competitor, "=" means that the two algorithms achieve the same accuracy results. And then, to show the convergence speed of ABAM superiority over contenders more clearly, the convergence graphs of BA, ABA, IS1 and ABAM are shown in Figs.1-3.

From Table II, it can be seen that ABAM is significantly better than BA, IS1 and ABA on almost all the test functions except for the functions f_2 and f_{11}. For the function f_2, solution accuracy obtained by ABAM is worse than IS1, but better than BA and ABA. For the function f_{11}, solution accuracy obtained by ABAM is the same as that found by ABA, however, the convergence speed of ABAM is more faster that of ABA as shown in Figs. 1-3. In addition, the performance of BA, ABA and IS1 decreases as their dimensions increase, but ABAM does not exist such defect.

Overall speaking, the ABAM exhibits the extremely convergence performance on almost all the benchmark functions. And the experimental results show that it outperforms conspicuously than BA, ABA and IS1.

B. Experiment 2-comparisons with MuABC, and DEs

To further test the efficiency of the ABAM algorithm, ABAM algorithm is compared with other five well known population based algorithms, i.e. MuABC [35], DE [36], jDE [37], JADE [38], SaDE [39]. The stop criteria, population size and the number of independent runs are kept same as in [35]

The test functions $f_3,\ f_4,\ f_{11},\ f_{13},\ \ f_{14},\ f_{15},\ f_{18}$ are chosen from the Table I. The experimental results are given in Table III. The results of the state-of-the-art algorithms except ABAM, are taken from [35] directly.

From Table III, it can be seen that, except for function f_{11}, ABAM is significantly better than other five algorithms on the other test functions. For function f_{11}, although the results of ABAM is worse than that of MuABC, better than that of DE, jDE, JADE and SaDE.

All in all, ABAM performs better than MuABC, DE, jDE, JADE and SaDE in terms of obtaining better solution on almost all cases.

C. Experiment 3-comparisons with PSOs

In this experiment, ABAM was compared with some PSOs, including FIPS-PSO [40], CLPSO [41], APSO [42], SSG-PSO, SSG-PSO-BFGS and SSG-PSO-PS [43]. 21 benchmark functions are chosen from [43] to evaluate the performance of proposed ABAM algorithm. The formulations, bounds of search space, global minimum values are provided in Table IV.

For this experiment, the population size M is set to 50 as used in [43], the other parameters of ABAM are defined as same as experiment 1. The comparison results are presented in Table V, which includes the mean value(Mean) and the standard deviation(SD). We performed 30 trails of running ABAM algorithm for each test problem. Except the results obtained ABAM algorithm, the rest of the data are taken from [43] directly.

From Table V, it is observed that ABAM is far better than FIPS-PSO, CLPSO, APSO, SSG-PSO, SSG-PSO-BFGS and SSG-PSO-PS for functions $f_2 - f_5$, f_8, f_{10} and $f_{13} - f_{21}$. For functions $f_6,\ f_7$ and f_{11}, ABAM has the same accuracy results as CLPSO, SSG-PSO, SSG-PSO-BFGS and SSG-PSO-PS, but has better results than FIPS-PSO and APSO. For function f_9, ABAM has the same accuracy results as CLPSO, APSO, SSG-PSO, SSG-PSO-BFGS and SSG-PSO-PS, but has better results than FIPS-PSO. For function f_1, all the algorithms have the same accuracy results. In summary, ABAM ranks on the top according to the overall performance shown in Table V.

V. Conclusion

In this paper, by using three improvement strategies, we presented a new bat algorithm. The performance of ABAM was compared with the standard BA, and other algorithms. The results showed that ABAM presents promising results for considered problems.

In the future, the adaption of the parameters in ABAM can be studied to improve its performance.

References

[1] X. S. Yang, "Natureinspired Metaheuristic Algorithms," Luniver Press, 2008.

[2] K. Krishnanand, D. Ghose, "Detection of multiple source locations using a glowworm metaphor with applications to collective robotics," in In Proceedings of the IEEE Swarm Intelligence Symposium, 2005.

[3] A. R. Mehrabian, C. Lucas, "A novel numerical optimization algorithm inspired from weed colonization," *Ecological Informatics*, vol. 1, pp 355-366, 2006.

[4] K. Deep, M. Thakur, "A new mutation operator for real coded genetic algorithms," *Applied Mathematics and Computation*, vol. 193, pp 211-230, 2007.

[5] P. Kaelo, M. M. Ali, "Integrated crossover rules in real coded genetic algorithms," *European Journal of Operational Research*, vol. 176, pp 60-76, 2007.

[6] R. Storn, K. Price, "Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 2, pp 341-359, 1997.

[7] J. Kennedy, R. Eberhart, "Particle swarm optimization," in IEEE International Conference on Neural Networks, 1995.

[8] M. Dorigo, T. Stutzle, "Ant colony optimization," MIT Press, Cambridege, MA, 2004.

[9] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Erciyes University, Technical Report-TR06, Kayseri, Turkey, 2005.

[10] Z. W. Geem, J. H. Kim, G. V. Loganathan, "A new heuristic optimization algorithm:harmony search," *Simulation*, vol. 76, pp 60-68, 2001.

[11] X. Yang, "A new metaheuristic bat-inspired algorithm", in Nature Inspired Cooerative Strategies For Optimization (NISCO 2010), Berlin, Germany.

[12] A. Husseinzadeh Kashan, "An efficient algorithm for constrained global optimization and apllication to mechaniical engineering design league championship algorithm(LCA)," *Computer-Aided Design*, vol. 43, pp 1769-1792, 2011.

[13] H. Eskandar, A. Sadollahb, A. Bahreininejad, et al, "Water cycle algorithm-a novel metaheuristic optimization method for solving constrained engineering optimization problems," *Computers and Structures*, vol. 110-111, pp 151-166, 2012.

[14] A. Sadollaha, A. Bahreininejada, H. Eskandarb, et al, "Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems," *Applied Soft Computing*, vol. 13, no. 5, pp 2592-2612, 2013.

[15] S. Mirjalili, A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp 51-67, 2016.

[16] S. X. Li, J. S. Wang, "Improved cockoo search algorithm with novel searching mechanism for solving unconstrained function optimization problem," *IAENG International Journal of Computer Science*, vol. 44, no. 1, pp 08-12, 2017.

[17] I. F. Jr, D. Fister, X. S. Yang," A hybrid bat algorithm," CORR abs/1303.6310.

[18] A. H. Gandomi, X. S. Yang, "Chaotic bat algorithm," *Journal of Computational Science*, vol. 5, no. 2, pp 224-232, 2014.

[19] G. Wang, L. Guo, "A novel hybrid bat algorithm with harmony search for globalnumerical optimization," *Journal of Applied Mathematics*, vol. 2013, pp 1-21, 2013.

[20] L. Li, Y. Zhou, " A novel complex-valued bat algorithm," *Neural Computing and Applications*, vol. 25, no. 6, pp 1369-1381, 2014.

[21] J. H. Lin, C. W. Chou, et al, "A chaotic levy flight bat algorithm for parameter estimation in nonlinear dynamic biological systems," *Journal of Computing and Information Technology*, vol. 2, no. 2, pp 56-63, 2012.

[22] S. Yilmaz, E. U. Kucuksille," A new modification approach on bat algorithm for solving optimization problems," *Applied Soft Computing*, vol. 28, pp 259-275, 2015.

[23] A. H. Gandomi, X. S. Yang, A. H. Alavi, S. Talatahari, "Bat algorithm for constrained optimization tasks," *Neural Computing and Applications*, vol. 22, no. 6 pp 1239-1255, 2013.

[24] D. Rodriguesa, L. A. M. Pereira, R. Y. M. Nakamura, et al. "A wrapper approach for feature selection based on Bat Algorithm and Optimum-Path Forest," *Expert Systems with Applications*, vol. 41, no. 5, pp 2250-2258, 2014.

[25] K. Khan, A. Sahai, "A comparison of BA, GA, PSO, BP and LM for training feed forward neural networks in e-learning context," *International Journal of Intelligent Systems and Applications*, vol. 4, no. 7, pp 23-29, 2012.

[26] M. K. Marichelvam, T. Prabaharan, X. S. Yang, M. Geetha, "Solving hybrid flow shop scheduling problems using bat algorithm," *International Journal of Logistics Economics and Globalisation*, vol. 5, no. 1, pp 15-29, 2013.

[27] A. P. Song, M. b. Li, X. H. Ding, et al,"Community detection using discrete bat algorithm," *IAENG International Journal of Computer Science*, vol. 43, no. 1, pp 37-43, 2016.

[28] X. Cai, W. Z. Li, L. Wang, et al, "Bat algorithm with Gaussian walk for directing orbits of chaotic systems," *International Journal of Computing Science and Mathematics*, vol. 5, no. 2, pp 198-208, 2014.

[29] J. H. Lin, C. W. Chou, et al., "A novel bat algorithm based on differential operator and lvy filights trajectory," *Journal of Computational Neuroscience*, vol. 2013, pp 1-13, 2013.

[30] P. W. Tsai, J. S. Pan, et al. "Bat algorithm inspired algorithm for solving numerical optimization problems," *Applied Mechanics and Materials*, vol. 148, pp 134-137, 2012.

[31] S. Yilmaz, E. U. Kucuksille, Y. Cengiz, "Modified bat algorithm," *Elektron Elektrotech*, vol. 20, no. 2, pp 71-78, 2014.

[32] X. Cai, L. Wang, et al., "Bat algorithm with Gaussian walk," *International Journal of Bio-Inspired Computation*, vol. 6, no. 3, pp 166-174, 2014.

[33] X. W. Wang, W. Wang, Y. Wang, "An adaptive bat algorithm," *Intelligent Computing Theories and Technology*, vol. 7996, pp 216-223, 2013.

[34] D. Karaboga, B. Akay, "A comparative study of artificial bee colony algorithm," *Applied Mathematics and Computation*, vol. 214, pp 108-132, 2009.

[35] W. F. Gao, L. L. Huang, S. Y. Liu, "Artificial bee colony algorithm with multiple search strategies," *Applied Mathematics*, vol. 271, pp 269-287, 2015.

[36] R. Storn, K. Price, "Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp 341-359, 1997.

[37] J. Brest, S. Greiner, B. Boskovic, et al, "Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, pp 646-657, 2006.

[38] J. Zhang, A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 5, pp 945-958, 2009.

[39] A. K. Qin, V. L. Huang, P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, pp 398-417, 2009.

[40] J. J. Liang, A. K. Qin, P. N. Suganthan, et al, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolution*, vol. 10, pp 281-295, 2006.

[41] R. Member, J. Kennedy, J. Neves, "The fully informed particle swarm:simpler, maybe better," *IEEE Transactions of Evolutionary Compution*, vol. 8, pp 204-210, 2004.

[42] Z. H. Zhan, J. Zhang, Y. Li, et al, "Adaptive partical swarm optimization, IEEE Transactions on Systems," *Man and Cybernetics, Part B:Cybernetics*, vol. 39, pp 1362-1381, 2009.

[43] G. H. Wu, D. S. Qiu, Y. Yu, et al, "Superior solution guided particle swarm optimization combined with local search techniques," *Expert Systems with Applications*, vol. 41, pp 7530-7548, 2014.

TABLE II: ABAM performance comparison with other three algorithms

Fun.	Dim.	BA			IS2			ABA			ABAM		
		Min	Mean	SD	Min	Mean	SD	Min	Mean	SD	Min	Mean	SD
f_1	2	4.097e-013+	3.638e-012	3.156e-012	6.902e-026+	2.092e-013	3.930e-013	1.264e-032+	1.951e-028	4.439e-028	**0**	**0**	**0**
f_2	2	-1.0316+	-0.827	3.625e-001	**-1.0316**	**-1.0316**	**7.337e-013**	-1.0316+	-0.827	3.627e-001	-1.0316+	-1.030	3.837e-003
f_3	10	3.545e+003+	6.910e+003	3.109e+003	7.176e-012+	1.309e-010	1.325e-010	4.872e-023+	5.031e-022	8.834e-022	**0**	**0**	**0**
	30	4.064e+004+	4.636e+004	4.162e+003	2.360e-007+	1.576e-006	1.910e-006	3.082e-017+	9.114e-017	7.305e-017	**0**	**0**	**0**
	50	4.201e+004+	8.973e+004	2.710e+004	4.761e-005+	2.398e-005	1.484e-005	4.185e-016+	1.077e-015	7.855e-016	**0**	**0**	**0**
f_4	10	2.535e+000+	2.014e+000	1.339e+001	1.320e-006+	5.701e-005	4.723e-005	1.241e-010+	1.016e+001	2.273e+001	**0**	**0**	**0**
	30	7.048e+002+	2.513e+006	3.458e+006	3.047e-001+	6.737e-001	4.241e-001	2.873e-007+	2.169e+001	4.851e+001	**0**	**0**	**0**
	50	1.229e+011+	3.323e+016	7.243e+016	1.666e-001+	6.919e-001	4.749e-001	1.449e+001+	3.294e+003	7.180e+003	**0**	**0**	**0**
f_5	10	9.264e-007+	1.187e-006	2.174e-007	6.367e-015+	1.061e-009	1.794e-009	2.384e-018+	4.471e-017	3.346e-017	**0**	**0**	**0**
	30	1.367e+001+	4.156e+001	2.683e+001	3.914e-006+	1.760e-005	1.241e-005	4.018e-009+	7.388e-009	2.781e-009	**0**	**0**	**0**
	50	1.801e+002+	2.445e+002	7.491e+001	1.8720e-004+	3.947e-004	1.340e-004	5.366e-005+	9.082e-005	2.366e-005	**0**	**0**	**0**
f_6	10	4.161e+001+	5.105e+001	8.325e+000	2.852e-005+	8.859e-005	6.078e-005	5.458e-011+	1.634e-010	1.186e-010	**0**	**0**	**0**
	30	7.183e+001+	7.482e+001	3.982e+000	1.346e-002+	4.325e-001	4.002e-001	1.971e+001+	2.165e+001	1.664e+00	**0**	**0**	**0**
	50	8.043e+001+	8.424e+001	3.074e+000	1.797e+000+	6.155e+000	3.316e+000	3.757e+001+	4.622e+001	6.047e+000	**0**	**0**	**0**
f_7	10	3.229e-006+	4.466e-006	1.158e-006	2.772e-010+	5.801e-009	7.377e-009	3.759e-022+	2.705e-020	3.562e-020	**0**	**0**	**0**
	30	3.052e+001+	8.242e+001	7.389e+001	3.750e-007+	2.379e-005	3.535e-005	5.408e-010+	1.573e-009	8.566e-010	**0**	**0**	**0**
	50	7.482e+002+	1.135e+003	2.221e+002	7.818e-004+	1.131e-001	4.320e-004	8.357e-007+	1.131e-005	6.133e-006	**0**	**0**	**0**
f_8	10	1.567e-013+	2.546e-013	1.428e-013	2.867e-028+	1.688e-020	3.168e-020	4.572e-045+	8.013e-043	6.820e-043	**0**	**0**	**0**
	30	1.062e-010+	1.472e-010	3.926e-011	5.426e-014+	2.261e-013	2.560e-013	5.377e-033+	1.250e-031	2.071e-031	**0**	**0**	**0**
	50	5.385e-010+	1.545e-009	5.828e-010	1.131e-011+	1.590e-010	1.121e-010	1.250e-027+	3.826e-027	2.103e-027	**0**	**0**	**0**
f_9	10	6.666e-010+	7.499e-010	6.087e-011	1.732e-011+	6.277e-011	5.299e-011	1.6442e-009+	3.295e-009	1.214e-009	**0**	**0**	**0**
	30	9.104e-009+	1.293e-008	2.455e-009	6.676e-012+	3.816e-010	2.760e-010	3.028e-008+	5.7687e-008	2.104e-008	**0**	**0**	**0**
	50	8.969e-009+	1.475e-008	4.660e-009	6.867e-011+	6.088e-010	7.131e-010	2.469e-008+	5.249e-008	1.770e-008	**0**	**0**	**0**
f_{10}	10	1.470e+007+	3.790e+007	2.421e+007	4.487e-003+	7.180e+001	1.244e+002	5.978e+003+	2.094e+004	1.483e+004	**0**	**0**	**0**
	30	7.949e+008+	9.751e+008	1.124e+008	8.101e+004+	2.863e+005	3.486e+005	2.525e+005+	5.387e+005	1.701e+005	**0**	**0**	**0**
	50	2.487e+009+	3.434e+009	8.989e+008	6.351e+004+	4.946e+005	5.536e+005	1.028e+006+	1.509e+006	4.931e+005	**0**	**0**	**0**
f_{11}	10	0+	1.200e+000	8.366e-001	**0**=	**0**	**0**	**0**=	**0**	**0**	**0**	**0**	**0**
	30	4.000e+000+	8.800e+000	2.774e+000	0+	2.124e+000	1.582e+000	**0**=	0	0	0	0	0
	50	1.416e+001+	1.832e+001	2.915e+000	0+	6.400e+000	4.098e-009	**0**=	**0**	**0**	**0**	**0**	**0**

Fun.	Dim.	BA			IS2			ABA			ABAM		
		Min	Mean	SD	Min	Mean	SD	Min	Mean	SD	Min	Mean	SD
f_{12}	10	1.091e+000$^+$	1.550e+000	3.698e-001	4.677e-003$^+$	1.395e-002	1.235e-002	4.667e-004$^+$	8.847e-004	6.079e-004	**4.029e-007**	**2.216e-005**	**2.960e-005**
	30	5.112e+000$^+$	6.579e+000	1.272e+000	5.104e-002$^+$	8.486e-002	2.838e-002	1.252e-002$^+$	1.941e-002	5.738e-003	**3.921e-007**	**2.129e-005**	**2.702e-005**
	50	8.636e+000$^+$	9.647e+000	9.166e-001	1.152e-001$^+$	2.242e-001	6.982e-002	1.751e-002$^+$	3.259e-002	1.569e-002	**3.224e-006**	**1.755e-005**	**1.839e-005**
f_{13}	10	4.477e+001$^+$	6.029e+001	1.480e+001	8.954e+000$^+$	2.248e+001	1.109e+001	1.989e+001$^+$	4.218e+001	1.882e+001	**0**	**0**	**0**
	30	1.552e+002$^+$	1.997e+002	2.907e+001	4.775e+001$^+$	7.462e+001	2.512e+001	1.482e+002$^+$	1.918e+002	2.623e+001	**0**	**0**	**0**
	50	2.716e+02$^+$	3.484e+02	5.594e+001	7.362e+001$^+$	1.064e+000	3.676e+001	2.716e+02$^+$	3.259e+02	5.834e+001	**0**	**0**	**0**
f_{14}	10	1.835e+001$^+$	1.918e+001	7.605e-001	2.813e+000$^+$	6.218e+000	2.854e+000	1.597e+001$^+$	1.752e+001	9.456e-001	**-8.881e-016**	**-8.881e-016**	**0**
	30	1.976e+001$^+$	1.996e+001	1.857e-003	1.361e+001$^+$	1.490e+001	1.197e+000	1.881e+001$^+$	1.939e+001	5.379e-001	**-8.881e-016**	**-8.881e-016**	**0**
	50	1.995e+001$^+$	1.996e+001	7.041e-004	1.154e+001$^+$	1.488e+001	1.897e+000	1.885e+001$^+$	1.943e+001	5.028e-001	**-8.881e-016**	**-8.881e-016**	**0**
f_{15}	10	7.569e-008$^+$	3.053e-001	4.160e-001	1.226e-012$^+$	2.455e-002	4.252e-002	0$^+$	3.437e-002	3.723e-002	**0**	**0**	**0**
	30	1.015e-006$^+$	4.571e-001	5.050e-001	6.271e-008$^+$	4.967e-001	4.929e-001	6.661e-016$^+$	3.298e-001	4.684e-001	**0**	**0**	**0**
	50	1.000e+000$^+$	1.126e+000	2.829e-001	3.817e-007$^+$	6.411e-001	4.967e-001	9.999e-001$^+$	1.000e+000	3.912e-014	**0**	**0**	**0**
f_{16}	10	4.888e-001$^+$	4.967e-001	4.459e-003	1.782e-001$^+$	3.223e-001	1.233e-001	4.970e-001$^+$	4.982e-001	8.548e-004	**0**	**0**	**0**
	30	4.996e-001$^+$	4.998e-001	1.200e-004	4.662e-001$^+$	4.870e-001	1.232e-002	4.997e-001$^+$	4.998e-001	8.405e-005	**0**	**0**	**0**
	50	4.998e-001$^+$	4.999e-001	1.308e-005	4.888e-001$^+$	4.931e-001	4.163e-003	4.998e-001$^+$	4.999e-001	3.424e-006	**0**	**0**	**0**
f_{17}	10	-69.850$^+$	-65.892	3.223e+000	-72.677$^+$	-66.457	3.686e+000	-72.677$^+$	-68.719	3.223e+000	**-78.332**	**-75.493**	**4.897e+000**
	30	-67.022$^+$	-64.949	2.149e+000	-67.965$^+$	-65.138	2.308e+000	-68.907$^+$	-66.268	1.548e+000	**-78.332**	**-75.703**	**5.873e+000**
	50	-68.153$^+$	-65.552	1.629e+000	-69.850$^+$	-66.118	2.927e+000	-67.022$^+$	-65.552	1.172e+000	**-78.332**	**-78.330**	**2.799e-003**
f_{18}	10	2.036e+000$^+$	4.091e+000	2.147e+000	3.433e-003$^+$	4.220e-002	6.662e-002	2.389e-007$^+$	4.720e-002	3.649e-002	**0**	**0**	**0**
	30	5.690e+000$^+$	1.316e+001	5.550e+000	6.110e-002$^+$	1.523e+000	1.480e+000	1.700e+000$^+$	3.776e+000	1.376e+000	**0**	**0**	**0**
	50	1.834e+001$^+$	2.886e+001	6.276e+000	2.172e+000$^+$	4.038e+000	3.429e+000	1.937e+000$^+$	4.217e+000	2.497e+000	**0**	**0**	**0**
f_{19}	10	6.125e+001$^+$	7.535e+001	1.661e+001	1.800e+001$^+$	2.660e+001	8.648e+000	3.300e+001$^+$	5.620e+001	2.156e+001	**0**	**0**	**0**
	30	2.655e+002$^+$	3.534e+002	6.104e+001	6.325e+001$^+$	1.659e+002	7.283e+001	1.980e+002$^+$	2.232e+002	1.920e+001	**0**	**0**	**0**
	50	4.802e+002$^+$	5.767e+002	6.722e+001	2.475e+002$^+$	3.357e+002	9.772e+001	3.250e+002$^+$	3.706e+002	5.412e+001	**0**	**0**	**0**
f_{20}	10	2.287e+001$^+$	2.067e+004	2.693e+004	6.825e-011$^+$	2.772e+000	2.066e+000	8.415e+000$^+$	2.510e+001	1.681e+001	**1.717e-015**	**4.268e-004**	**8.007e-004**
	30	1.608e+007$^+$	9.662e+007	5.544e+007	1.194e+000$^+$	5.920e+000	4.440e+000	1.535e+001$^+$	2.089e+001	6.093e+000	**1.147e-014**	**1.154e-004**	**1.757e-004**
	50	3.049e+008	3.540e+008	4.585e+007	1.517e+000$^+$	5.454e+000	2.891e+000	1.214e+001$^+$	1.666e+001	3.299e+000	**7.432e-016**	**2.201e-004**	**4.823e-004**
f_{21}	10	1.084e+005$^+$	2.008e+005	6.965e+004	3.052e-007$^+$	7.371e-006	6.655e-006	3.042e-009$^+$	8.527e-009	5.263e-009	**0**	**0**	**0**
	30	1.350e+006$^+$	1.917e+006	3.415e+005	1.488e+002$^+$	8.626e+002	5.117e+002	2.775e-004$^+$	6.691e-004	3.391e-004	**0**	**0**	**0**
	50	3.284e+006	4.834e+006	9.560e+005	2.298e+003$^+$	3.595e+003	8.322e+002	1.464e-002$^+$	6.641e-002	4.847e-002	**7.432e-016**	**2.201e-004**	**4.823e-004**
f_{22}	10	1.616e+004$^+$	2.703e+004	9.115e+003	2.776e-006$^+$	4.875e-005	5.556e-005	7.939e+003$^+$	1.218e+004	3.879e+003	**1.717e-015**	**4.268e-004**	**8.007e-004**
	30	7.234e+004$^+$	9.610e+004	2.817e+004	3.409e-003$^+$	2.116e-001	2.325e-001	4.432e+004$^+$	6.341e+004	1.274e+004	**1.147e-014**	**1.154e-004**	**1.757e-004**
	50	1.446e+005	1.741e+005	3.592e+004	4.974e-001	2.310e+000	2.139e+000	8.397e+004	1.123e+005	2.496e+004	**7.432e-016**	**2.201e-004**	**4.823e-004**
f_{23}	10	2.189e+001$^+$	4.438e+001	2.210e+001	1.393e+001$^+$	2.965e+001	1.225e+001	1.293e+001$^+$	3.880e+001	1.979e+001	**0**	**0**	**0**
	30	1.751e+002$^+$	1.932e+002	1.674e+001	3.880e+001$^+$	6.129e+001	1.625e+001	1.612e+002$^+$	1.867e+002	4.610e+001	**0**	**0**	**0**
	50	2.875e+002$^+$	3.256e+002	3.309e+001	9.054e+001$^+$	1.248e+002	3.261e+001	2.030e+002$^+$	3.194e+002	7.121e+001	**0**	**0**	**0**

TABLE III: Comparison of ABAM, MuABC and DEs on optimizing 30-dimensional functions

Fun.	Max.FEs	DE		jDE		JADE		SaDE		MuABC		ABAM	
		Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
f_3	150,000	9.8e-014	8.4e-014	1.46e-028	1.78e-028	2.69e-056	1.41e-055	3.28e-020	3.63e-020	1.81e-076	2.37e-076	**0**	**0**
f_4	200,000	1.6e-009	1.1e-009	9.02e-024	6.01e-024	3.18e-025	2.05e-024	3.51e-025	2.74e-025	2.48e-055	2.82e-055	**0**	**0**
f_{11}	10,000	4.7e+003	1.1e+003	6.13e+002	1.72e+002	5.62e+000	1.87e+000	5.07e+001	1.34e+001	**0**	**0**	1.160e+000	2.130e+000
f_{13}	100,000	1.8e+002	1.3+001	3.32e-004	6.39e-004	1.33e-001	9.47e-002	2.43e+000	1.60e+000	**0**	**0**	**0**	**0**
f_{14}	50,000	1.1e-001	3.9e-002	2.37e-004	7.10e-005	3.35e-009	2.84e-009	3.81e-006	8.26e-007	3.35e-012	6.02e-013	**-8.881e-016**	**0**
f_{15}	50,000	2.0e-001	1.1e-001	7.29e-006	1.05e-005	1.57e-008	1.09e-007	2.52e-009	1.24e-008	8.09e-010	1.78e-010	**0**	**0**
f_{18}	300,000	2.3e-004	1.7e-004	6.08e-010	8.36e-010	2.78e-005	8.43e-006	2.94e-006	3.47e-006	9.12e-093	1.25e-092	**0**	**0**

(Advance online publication: 28 May 2018)

TABLE IV: Benchmark functions used in experiment 3.

Functions	Range	Optimal value
$f_1 = \sum_{i=1}^{n} x_i^2$	[-100,100]	0
$f_2 = \sum_{i=1}^{n-1} (100 * (x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$	[-2.048,2.048]	0
$f_3 = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	[-100,100]	0
$f_4 = \max_i\{\mid x_i \mid, 1 \le i \le n\}$	[-100,100]	0
$f_5 = \sum_{i=1}^{n} \mid x_i \mid + \prod_{i=1}^{n} \mid x_i \mid$	[-10,10]	0
$f_6 = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	[-5.12,5.12]	0
$f_7 = \sum_{i=1}^{n} (y_i^2 - 10\cos(2\pi y_i) + 10)$, $if \mid x_i \mid < \frac{1}{2}, y_i = x_i; else\ y_i = \frac{round(2x_i)}{2}$	[-5.12,5.12]	0
$f_8 = -20\exp(-0.2 * \sqrt{\sum_{i=1}^{n} x_i^2/n})$ $- \exp(\sum_{i=1}^{n} \cos(2\pi x_i/n) + 20 + e$	[-32,32]	0
$f_9 = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1$	[-600,600]	0
$f_{10} = \sum_{i=1}^{n} (100((a_i x_i)^2 - (a_{i+1} x_{i+1}))^2 + (a_i x_i - 1)^2)$, $a_i = 100^{\frac{i-1}{n-1}}$	[-4.196,4.196]	0
$f_{11} = \sum_{i=1}^{n} ((a_i x_i)^2 - 10\cos(2\pi(a_i x_i)) + 10)$, $a_i = 10^{\frac{i-1}{n-1}}$	[-5.12,5.12]	0
$f_{12} = \sum_{i=1}^{n} ((a_i x_i)^2 - 10\cos(2\pi(a_i x_i)) + 10)$, $a_i = 1000^{\frac{i-1}{n-1}}$	[-5.12,5.12]	0
$f_{13} = \sum_{i=1}^{n} z_i^2,\ z = M * x$	[-100,100]	0
$f_{14} = \sum_{i=1}^{n-1} (100 * (z_i^2 - z_{i+1})^2 + (z_i - 1)^2)\ z = M * x$	[-2.048,2.048]	0
$f_{15} = \max_i\{\mid z_i \mid, 1 \le i \le n\},\ z = M * x$	[-10,10]	0
$f_{16} = \sum_{i=1}^{n} (z_i^2 - 10\cos(2\pi z_i) + 10),\ z = M * x$	[-5.12,5.12]	0
$f_{17} = -20\exp(-0.2 * \sqrt{\sum_{i=1}^{n} z_i^2/n})$ $- \exp(\sum_{i=1}^{n} \cos(2\pi z_i/n) + 20 + e,\ z = M * x$	[-32,32]	0
$f_{18} = \frac{1}{4000} \sum_{i=1}^{n} z_i^2 - \prod_{i=1}^{n} cos(\frac{z_i}{\sqrt{i}}) + 1,\ z = M * x$	[-600,600]	0
$f_{19} = \sum_{i=1}^{n} (20^{\frac{i-1}{n-1}} z_i)^2,\ z = M * x$	[-100,100]	0
$f_{20} = (1000 * z(1))^2 + \sum_{i=2}^{n} z_i^2,\ z = M * x$	[-100,100]	0
$f_{21} = \sum_{i=1}^{n} z_i^{2+a_i},\ a_i = 10^{\frac{i-1}{n-1}},\ z = M * x$	[-100,100]	0

TABLE V: Comparison of ABAM and PSOs on optimizing 30-dimensional functions

Fun.	FIPS-PSO		CLPSO		APSO		SSG-PSO		SSG-PSO-BFGS		SSG-PSO-PS		ABAM	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
f_1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_2	2.52e+001	9.08e-001	2.34e+001	3.79e+000	2.38e+001	7.05e+001	2.12e+001	1.84+000	5.73e-011	8.08e-012	6.90e+000	1.25e+000	**0**	**0**
f_3	2.08e+002	8.98e+001	1.34e+002	2.44e+002	9.91e-003	5.03e-002	1.24e+001	1.35e+001	2.57e-014	4.58e-014	4.16e+001	2.31e+000	**0**	**0**
f_4	6.25e-002	4.34e-003	6.61e-003	4.21e-003	2.24e-001	6.85e-001	2.77e-002	1.47e-002	3.77e-006	1.63e-006	1.25e-015	3.25e-015	**0**	**0**
f_5	1.64e-009	5.59-010	7.52e-020	5.69e-019	4.25e-023	7.44e-021	2.75e-025	1.96e-025	1.04e-026	7.54e-026	9.93e-022	1.23e-022	**0**	**0**
f_6	6.39e+001	1.12e+001	0	0	4.25e+000	1.35e+000	0	0	0	0	0	0	**0**	**0**
f_7	5.44e+001	2.63e+001	0	0	3.21e+000	6.32e+000	0	0	0	0	0	0	**0**	**0**
f_8	1.39e-008	2.98e-009	7.77e-014	1.49e-018	6.34e-002	1.43e+000	7.25e-015	1.74e-016	4.97e-015	1.73e-015	1.25e-014	6.23e-015	**-8.8817e-016**	**0**
f_9	2.72e-007	1.18e-006	0	0	0	0	0	0	0	0	0	0	**0**	**0**
f_{10}	7.37e+004	3.14e+005	4.94e+001	4.32e+001	4.06e+006	5.42e+006	3.59e+001	3.22e+001	3.21e+001	3.07e+001	1.81e+000	1.05e+000	**0**	**0**
f_{11}	7.37e+004	9.23e+000	0	0	1.98e+000	2.44e+000	0	0	0	0	0	0	**0**	**0**
f_{12}	4.59e+001	2.38e+001	0	0	0	0	0	0	0	0	0	0	**0**	**0**
f_{13}	7.54e-013	3.26e-013	4.21e-017	6.18e-018	3.24e-020	5.45e-019	5.28e-022	8.71e-022	2.29e-027	3.74e-027	1.97e-024	8.65e-025	**0**	**0**
f_{14}	2.89e+001	4.15e+000	2.64e+001	1.21e+000	7.83e+001	8.24e+001	2.53e+001	6.91e-001	3.98e-010	1.22e-010	2.50e+001	1.41e+001	**0**	**0**
f_{15}	1.36e-004	4.89e-005	1.51e-002	4.64e-003	8.05e-001	1.24e-001	7.25e-004	3.02e-004	8.15e-006	7.15e-006	4.40e-004	2.31e-004	**0**	**0**
f_{16}	1.75e+002	8.97e+000	1.08e+002	1.36e+001	1.02e+002	1.24e+003	4.65e+001	1.12e+001	4.10e+001	1.43e+001	4.44e+001	6.43e+001	**0**	**0**
f_{17}	2.24e-008	5.60e-009	2.76e-003	3.25e-003	3.62e-010	9.94e-010	5.86e-014	1.55e-013	4.59e-015	2.64e-015	6.16e-014	1.95e-013	**0**	**0**
f_{18}	1.14e-003	3.00e-003	2.66e-003	2.12e-003	1.72e-002	2.41e-001	1.02e-005	4.86e-005	1.47e-016	3.03e-016	1.52e-006	2.84e-006	**0**	**0**
f_{19}	1.51e+003	7.14e+002	4.88e+003	1.38e+003	1.25e+003	2.12e+004	7.72e+002	8.86e+002	2.44e-016	4.38e-016	5.13e+001	3.24e+001	**0**	**0**
f_{20}	8.45e+002	2.14e+002	3.35e+002	1.34e+002	7.41e+002	8.44e+002	3.02e+002	1.10e+002	2.23e-012	2.40e-012	2.14e-002	6.54e-002	**0**	**0**
f_{21}	4.54e+010	9.01e+010	3.74e+007	3.45e+007	2.91e+007	4.22e+008	5.42e+006	7.17e+006	8.87e-010	5.79e-010	3.26e+004	2.16e+004	**0**	**0**