Communications on Applied Electronics (CAE) – ISSN : 2394-4714
Foundation of Computer Science FCS, New York, USA
Volume 5 – No.8, August 2016 – www.caeaccess.org

An Empirical Comparison between the Artificial Bee Colony and Bat Algorithms on Continuous Function Optimization Problem

Shifat Sharmin Shapla
Stamford University Bangladesh
51, Siddeswari Road, Dhaka-1217, Bangladesh

Tanveer Ahmed Belal
Ahsanullah University of Science and Technology
Dhaka-1208, Bangladesh

Mohammad Shafiul Alam
Ahsanullah University of Science and Technology
Dhaka-1208, Bangladesh

ABSTRACT
Swarm intelligence is the collective and collaborative behavior of self-organized systems, natural or artificial. Swarm intelligence algorithms basically come from nature or biological behavior of nature. In this paper we have conducted an experimental comparison between two Swarm intelligence algorithms — the Artificial Bee Colony (ABC) algorithm and basic Bat algorithm on both unimodal and multimodal high dimensional continuous functions. The ABC algorithm has a well-balanced exploration and exploitation ability which is based on the intelligent food foraging behavior of honey bee swarm, proposed by Karaboga in the year 2005, while the Bat algorithm was inspired by the echolocation behavior of bats found in nature. The experimental results show that the ABC algorithm performs better than the BAT algorithm.

Keywords
Swarm intelligence, artificial bee colony algorithm, bat algorithm, continuous function optimization.

1. INTRODUCTION
ABC is a swarm intelligence based algorithm that mimics the candidate solutions as a swarm of bees which forage across a search space for continuously better quality food sources (i.e., candidate solutions). In ABC algorithm there are three types of bees — the employed bee, onlooker bee and scout bee. The employed and onlooker bees exploit the sources by local searches in the neighbor of the solutions and the scout bees abandon the solutions that are not beneficial anymore for the search progress and new random solutions are inserted instead of them to explore new regions of the search space.

The Bat Algorithm is a recent swarm intelligence algorithm which is based on the echolocation behavior of microbats with varying pulse rates of emission and loudness. After that a few improvements of Bat algorithm have been proposed in the literature, e.g., [1] [2] [3]. In this paper an experimental comparison between ABC algorithm and BAT algorithm have shown. The rest of the paper is organized as follows. Sections 2 and 3 describe the ABC algorithm and BAT algorithm, respectively. Section 4 presents the experimental results and comparison between these two algorithms. Finally, section 5 concludes the paper with a discussion and a few suggestions on future research.

2. ARTIFICIAL BEE COLONY (ABC) ALGORITHM
The ABC algorithm is motivated by the intelligent behavior of honey bees. ABC, as an optimization tool, provides a population based search procedure in which individuals called foods positions are modified by the artificial bees with tie and the bee's aim is to discover the places of food sources with high nectar amount and finally the one with the highest nectar. In ABC system, artificial bees fly around in a multidimensional search space and some choose food sources depending on the experience of themselves and their nest mates, and adjust their positions. Some (scout bees) fly and choose the food sources randomly without using experience. If the nectar amount of a new source is higher than that of the previous one in their memory, they memorize the new position and forget the previous one [4]. Thus, ABC system combines local search methods, carried out by employed and onlooker bees, with global search methods, managed by onlookers and scouts, in an attempt to balance between explorative and exploitative characteristics of the search process. In the original implementation of the ABC algorithm [5], half of the colony consists of employed bees, while the other half is the onlooker bees. The number of food sources (i.e., candidate solutions being exploited) is kept equal to the number of employed bees in the colony. Scout bees are created only when it is necessary, i.e., when a particular food source/candidate solution fails to improve for an unacceptably long period of time, indicating possible stagnation at some locally optimal point. However, in each cycle (i.e., iteration), no more than one scout bee is initiated, which limits the degree of random explorations of the algorithm. After a scout bee discovers a food source with sufficiently good quality, it turns into an employed bee. The detailed pseudo code is given below.

Step 1: Generate an initial population of N individuals as food sources (i.e. candidate solutions). Each individual has D attributes, where D is the dimensionality of problem.

Step 2: Calculate the fitness of each individual.

Step 3: Each employed bee searches the neighbor to find a better food source. For each employed bee generates a new solution, $\boldsymbol{v_i}$ around its current position $\boldsymbol{x_i}$ using following rule (1).

$$\boldsymbol{v_{ij}} = \boldsymbol{x_{ij}} + \varphi_{ij}\,(\boldsymbol{x_{kj}} - \boldsymbol{x_{ij}}) \qquad (1)$$

Here, $k \in \{1, 2, \ldots, N_{emp}\}$ and $j \in \{1, 2, \ldots, D\}$ are randomly chosen indices, N_{emp} is the number of employed bees (or food sources) and φ_{ij} is a uniform random value produced from [-1, 1].

Step 4: Compute the fitness of both $\boldsymbol{x_i}$ and $\boldsymbol{v_i}$ and apply greedy selection scheme to pick the better one to be included into the population and discard the others.

Step 5: Calculate and normalize the selection probability value p_i for each food source $\boldsymbol{x_i}$ using (2)

$$p_i = \frac{fitness(\boldsymbol{x}_i)}{\sum_{j=1}^{SN} fitness(\boldsymbol{x}_j)} \tag{2}$$

Step 6: Assign each onlooker bee to a food source position $\boldsymbol{x_i}$ at random with probability p_i.

Step 7: Produce new food positions (i.e. solutions), v_i for each onlooker bee using the employed bee $\boldsymbol{x_i}$ by using (1).

Step 8: Evaluate the fitness of $\boldsymbol{x_i}$ and $\boldsymbol{v_i}$. Apply greedy selection between then to keep the one with better fitness and discard the other.

Step 9: If a particular solution has not been improved over the past *limit* cycle then select it for abandonment. Replace it by placing a scout bee at a food source placed uniformly at random over the entire search space by using (3), i.e., for j = 1, 2, ……, D

$$\boldsymbol{x_{ij}} = min_j + rand(0,1) * (max_j - min_j) \tag{3}$$

Step 10: Set the iteration counter C = C+1 and also keep track of the best food source position.

Step 11: Check for termination. If the best solution found is acceptable or a predefined maximum number of the cycles have elapsed, then stop and return the best solution found so far. Otherwise go back to step 2 and repeat again.

3. BAT ALGORITHM

Bat algorithm is a meta-heuristic algorithm which was inspired by the echolocation behavior of microbats, with varying pulse rates of emission and loudness. [6]. Bats are born with the appealing innovative capability of echolocation. Bats emit a high sound frequency to listen the echo that bounces back from the neighboring objects [7] [8]. Signal frequencies of bats varies depends on the species [9] [10]. Such echolocation characteristics of microbats emphasize on some of approximate rules, which are given below [11].

i. Distance: Bats use echolocation to sense distance. They acknowledge the ranges/spaces between prey and surrounded barriers.

ii. Frequency: Bats fly randomly with velocity v_i at position x_i with a fixed frequency f_{min}, wavelength λ and loudness A_0 to search for prey. They can automatically adjust the wavelength of their emitted pulses and adjust the rate of pulse emission in the range of [0, 1], depending on the proximity of their target [12].

iii. Loudness: Though loudness can vary in many ways. Here assume that the loudness differs from a large A_0 to a minimum constant value A_{min}.

3.1 Initialization of Bat Algorithm

The initial population is generated randomly. Each individual of the population consists of real valued vectors with dimension D and n number of bats. For generating the initial population.

$$x_{ij} = x_{min\ j} + rand\ (0,1)(x_{\max\ j} - x_{min\ j}) \tag{4}$$

Where i = 1,2, ……., n; j = 1,2, ……, D; $x_{max\ j}$ and $x_{min\ j}$ are the upper and lower boundaries for dimension.

3.2 Solution, Frequency and Velocity

Step size of a new solution is defined by the frequency in Bat algorithm. For each solution, the frequency is set to a value which ranges between upper and lower boundaries f_{min} and f_{max}. The bat positon and velocity is updated by using the frequency. Following equations are used to update velocity and positions [13].

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{5}$$

$$V_i^t = V_i^{t-1} + (x_i^t - x^*)f_i \tag{6}$$

$$x_i^t = x_i^t + V_i^t \tag{7}$$

Where $\beta \in [0,1]$ indicates randomly generated number, x*represents the global best solutions in the population, f_i is the frequency for the solution i, V_i represents the new velocity for the solution i. A solution is selected among the best solution and random walk is applied in order to increase exploration. Thus a new candidate solution is generated.

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

$\overline{A^t}$ is the average loudness of all the bats and $\varepsilon \epsilon [0,1]$ is uniform random number that represents the directions and intensity of random walk.

3.3 Loudness and Pulse Emission Rate

Loudness and pulse emission rate must be adjusted and updated as iterations proceed. When bat gets closer to its prey, the loudness A usually decrease and pulse emission rate increases [14] [15]. Loudness A and pulse emission rate r are updated by using following equations:

$$A_i^{t=1} = \alpha A_i^t \tag{9}$$

$$r_i^{t+1} = r_i^0[1 - e^{(-\gamma t)}] \tag{10}$$

Where α and γ are constants which having the determined values for these equations which is set to 0.9 in this simulation. Here the initial loudness A_i^0 can typically be [1, 2], while the initial emission rate r_i^0 can be [0, 1] normally [16].

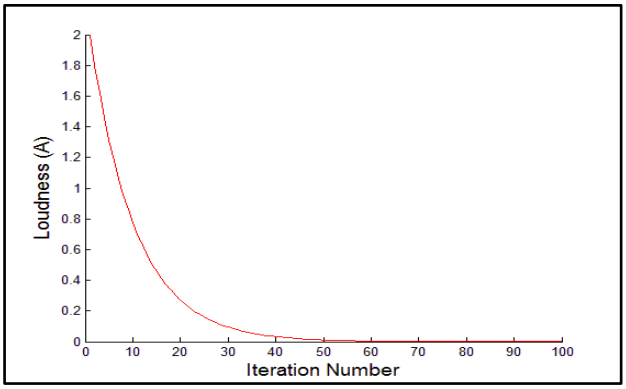

Fig.1: Loudness(A) in Bat Algorithm

Communications on Applied Electronics (CAE) – ISSN : 2394-4714
Foundation of Computer Science FCS, New York, USA
Volume 5 – No.8, August 2016 – www.caeaccess.org

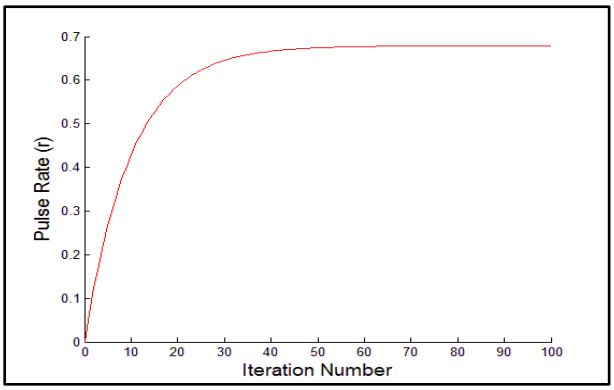

Fig.2: Pulse Emission Rate (r) in Bat Algorithm

3.4 Pseudo Code of Bat Algorithm

The initial population is generated randomly. Each individual of t

1. Objective function: $f(x),\ x = (x_1, x_2, x_3, \dots\dots\dots, x_d)^T$
2. Initialize bat population x_i and velocity v_i ; $i = (1,2, \dots\dots\dots, n)$
3. Define pulse frequency $f_i\ at\ x_i$
4. Initialize pulse rate $r_i$and loudness A_i
5. **while** (t < maximum number of iterations)
6. Generate new solutions by adjusting frequency,
7. and updating velocities and locations/solutions
8. **if** (rand >r_i)
9. Select a solution among the best solutions
10. Generate a local solution around the selected best solution
11. **end if**
12. **if** (rand < A_i) **and** $f(x_i) < f(x^*)$
13. Accept new solutions
14. increase r_i, reduce A_i
15. **end if**
16. Rank the bats and find current best x^*
17. **end while**
18. Display results.

4. SIMULATION AND EXPERIMENTAL RESULT

To compare the performance of ABC and BAT algorithm, we have used eight standard benchmark functions including both unimodal and multimodal functions. The analytical form of each functions, their search space and global minimum values are shown in the following Table 1. These functions are tested with the dimension, D = 30 for both the algorithms.

Here, Table 2 represents the experimental results of ABC and Bat Algorithm on seven benchmark functions. The maximum generations vary from function to function. The mean and standard deviation of the error values found from the simulation are shown in Table 2. The overview of the result is summarized in the following points.

- On all of the seven functions, ABC performs better than BAT algorithm.
- For all these seven functions, ABC reaches very close to the global minimum value, while the BAT algorithm fails to reach sufficiently close for most of these functions.

In summary, ABC is far better than Bat algorithm on both unimodal and multimodal functions. The reason might be the presence of a sound balance between exploitations and explorations in the ABC algorithm, while the Bat algorithm usually gets trapped around some poor local minima, far from the global minimum.

Table 1: Benchmark functions used in the experimental studies. Here, *D*: Dimensionality of the function, *S*: search space, *C*: function characteristics with values — *U*: Unimodal and *M*: Multimodal

No.	Name	*D*	*C*	*S*	Function	f_{min}
f_1	Sphere	30	*U*	$[-5.12, 5.12]^D$	$f_1(x) = \sum_{i=1}^{D} X_i^2$	0
f_2	Step	30	*U*	$[-100, 100]^D$	$f_2(x) = \sum_{i=1}^{d} (\lfloor x_i + 0.5 \rfloor^2)$	0
f_3	Rosenbrock	30	*U*	$[-15, 15]^D$	$f_3(x) = \sum_{i=1}^{d-1} [100\ (x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	0
f_4	Quartic	30	*U*	$[-1.28, 1.28]^D$	$f_4(x) = \sum_{i=1}^{n} i x_i^4$	0

Table 1 (continued): Benchmark functions used in the experimental studies. Here, *D*: Dimensionality of the function, *S*: search space, *C*: function characteristics with values — *U*: Unimodal and *M*: Multimodal.

f_5	Griewank	30	*M*	[-600, 600]^D	$f_3 = \frac{1}{4000}\sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d} cos\frac{x_i}{\sqrt{i}} + 1$	0
f_6	Rastrigin	30	*M*	[-15, 15]^D	$f_6(x) = \sum_{i=1}^{d} [x_i^2 - 10\cos(2\pi x_i) + 10]$	0
f_7	Ackley	30	*M*	[-32, 32]^D	$f_7(x) = -20\exp(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i) + 20 + e$	0

Table 2: Performance comparison of ABC and Bat algorithm on the benchmark functions. Better performance on each function is marked with boldface font.

No.	f_{min}	ABC		BAT		Better Performance by
		Mean	Std. Dev.	Mean	Std. Dev.	
f_1	0.0	6.38e–10	8.30e–11	2.38E+01	6.85E+00	**ABC**
f_2	0.0	0	0	1.08E+04	3.63E+03	**ABC**
f_3	0.0	3.11e+00	1.30e+00	4.29E+05	3.14E+05	**ABC**
f_4	0.0	9.01e–11	2.66e–11	1.29E+01	2.22E+00	**ABC**
f_5	0.0	9.84e–10	4.57e–10	9.09E+01	2.46E+01	**ABC**
f_6	0.0	6.12e–16	9.30e–17	4.64E+02	7.87E+01	**ABC**
f_7	0.0	1.22e–11	7.10e–12	1.52E+01	1.13E+00	**ABC**

5. CONCLUSION

This paper presents a comparative study between two swarm intelligence based algorithms — the Artificial Bee Colony algorithm and BAT algorithm. Both algorithms use collaborative swarm intelligence to find the global optimum value of the continuous optimization problems. The ABC algorithm performs better than the Bat algorithm on all the seven benchmark functions used in our study. The reason behind this might be that — ABC maintains a sound balance between its degree of explorations and exploitations, which provides resilience against premature convergence around the locally optimal points and helps find optimal or near optimal solutions. On the other hand, the Bat algorithm convergences around the locally optimal points, failing to locate the globally optimum point on most of the functions. However, there has been a few research works that try to improve the performance of the original Bat algorithm. An interesting future research direction would be to explore these existing variants of the Bat algorithm, finding their relative strengths and weaknesses and comparing them to other existing swarm intelligence algorithms, e.g., the ABC algorithm.

6. REFERENCES

[1] K. Khan and 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*, pp. 23-29, June 2012.

[2] A Rekaby, "Directed Artificial Bat Algorithm (DABA) – A New Bio-Inspired Algorithm," *Advances in Computing, Communications and Informatics (ICACCI), 2013 International Conference on, Mysore, 2013*, pp. 1241-1246.

[3] Y. Selim and U. K. Ecir, "Improved Bat Algorithm (IBA)on Continuous Optimization Problems," *International Conference on Software and Computer Applications, Lecture Notes on Software Engineering*, vol. 1, no. 3, pp. 279-283, 2013.

[4] V. Tereshko, A. Loengarov, Collective decision-making in honeybee foraging dynamics, Computing and Information Systems Journal 9 (3) (2005).

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

[6] J. D. Altringham, "Bats: Biology and Behaviour", *Oxford University Press, (1996).*

[7] Y. Selim and U. K. Ecir, "Improved Bat Algorithm (IBA) on Continuous Optimization Problems," *Lecture Notes on Software Engineering*, vol. 1, no. 3, pp. 279-283, 2013.

[8] T. Colin, "The Varienty of Life", *Oxford University Press*, 2000.

[9] A. Faritha and C. Chandrasekar, "An optimized approachof modified bat algorithm to record deduplication," *International Journal of Computer Applications*, vol. 62, no. 1, pp. 10-15, 2012.

[10] Y. Xin-She, "Bat Algorithm for MultiobjectiveOptimization," *International Journal Bio-InspiredComputation*, vol. 3, no. 5, pp. 267-274, 2011.

[11] Y. Xin-She, "A New Metaheuristic Bat-Inspired Algorithm, Nature Inspired Cooperative Strategies for Optimization (NISCO)", *Springer*, vol. 284, no. Springer Berlin, pp. 65-74, 2010.

[12] Y. Xin-She, "Bat Algorithm for Multiobjective Optimization," *International Journal Bio-Inspired Computation*, vol. 3, no. 5, pp. 267-274, 2011.

[13] N. Sakib, S. Mustafizur, M. S. Alam, M. W. U. Kabir, "A Novel Adaptive Bat Algorithm to Control Explorations and Exploitations for Continuous Optimization Problems," *International Journal of Computer Applications*, vol. 94, no. 13, 2014.

[14] Y.Xin-She, "A New Metaheuristic Bat-InspiredAlgorithm, Nature Inspired Cooperative Strategies forOptimization (NISCO 2010)," *Springer*, *vol. 284*, *no. Springer Berlin*, pp. 65-74 , 2010.

[15] Y. Xin-She, "Bat Algorithm for MultiobjectiveOptimization," *International Journal Bio-InspiredComputation*, vol. 3, no. 5, pp. 267-274, 2011.

[16] X. S. Yang, "Harmony Search as a MetaheuristicAlgorithm, Music-Inspired Harmony Search Algorithm," *Theory and Applications*, *Studies in ComputationalIntelligence*, vol. 191, pp. 1-14, 2009.