Performance Evaluation of Bat Algorithm to Solve Deterministic and Stochastic Optimization Problems

Ratnaji Vanga, Manu K. Gupta and J. Venkateswaran
Industrial Engineering and Operations Research, IIT Bombay

February 8, 2013

Abstract

This paper focuses on applicability of bat algorithm in various deterministic and stochastic optimization problems. A modified bat algorithm is proposed by changing the structure of original algorithm. Performance of this algorithm is tested for Rosenbrock and Rastrigin functions with different dimensionality and stochasticity. An interface between Anylogic and Java is developed for implementation. Various measure of performance are discussed. Different plots have been shown to get further insight on various problems. It is concludes, based on results, that dimensionality has more effect in convergence of algorithm than stochasticity. It is also observed that on adding stochasticity to non linear functions improves the performance of algorithm.

Keywords: Bat Algorithm, Deterministic and Stochastic Optimization, Measure of Performance, Meta-heuristics.

1 Introduction

Bat Algorithm is nature inspired meta-heuristic optimization algorithm for solving engineering optimization task. This algorithm was first proposed in Yang (2010). Bat Algorithm is based on the echolocation behaviour of bats. Most bats use echolocation while microbats use it extensively. Echolocation is a type of sonar used to avoid close obstacles in the dark, detect prey, and locate places to sleep over Altringham et al. (1998). During echolocation, these microbats emit a very loud sound pulse with different frequencies and listen for the echo that bounces back from the surrounding objects. Bat Algorithm is designed for continuous optimization problems by Parpinelli and Lopes (2011). One of it's extended version is used to solve multi objective optimization problems by Yang (2011). Musikapun and Pongcharoen (2012) applied Bat Algorithm in multi-stage multi-machine multi-product scheduling. This algorithm has also found application in numerical optimization as shown by Tsai et al. (2011).

In the seminal paper on Bat Algorithm by Yang (2010), Bat Algorithm is compared with Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) for ten test functions including Rosenbrock and Rastrigin. It turned out that for all test functions Bat Algorithm performs better that GA and PSO. Yang and Gandomi (2012) applied Bat Algorithm for eight engineering non linear optimization problems and found that Bat Algorithm performs better as compare to existing algorithm.

Meta-heuristics and evolutionary algorithms have been extensively used to solve non linear optimization problems. Bat algorithm is one such algorithm. We modified bat algorithm so that it can search in wider range. We made two modifications in bat algorithm. First modification is on location updating formulae and another is on the way of calculating best objective. We considered deterministic and stochastic version of Rosenbrock and Rastrigin function to test this modified algorithm. We also tested these functions in higher dimension. Here stochasticity is created using uniform distribution. We also define three measure of performance, i.e., quality of solution, number of iteration and execution time to compare various cases. All functions are modelled in Anylogic and modified bat algorithm is implemented in Java through interface to Anylogic. Ten different seed values are implemented to reduce the effect of random number in performance measure for each function. Graph of iteration vs best objective and time vs best objective is plotted to obtain further insights. Some observations are also listed.

This paper is organised as follows. Section 2 describes original and modified Bat algorithm. Section 3 presents different optimization problems. Overview of implementation details is given in Section 4. Section 5 describes results and various plots to get insights. Finally conclusion and future plans are discusses in Section 6.

2 Description of Bat Algorithm

A Bat Algorithm is developed using the following rules Komarasamy and Wahi (2012)

1. All bats use echolocation to sense distance, and they also know the difference between food/prey and background barriers in some magical way.

2. Bats fly randomly with velocity v_i at position x_i with a fixed frequency f varying wavelength λ and loudness A_0 to search for prey. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission r in the range $[0, 1]$, depending on the proximity of their target.

3. Although the loudness can vary in many ways, assume that the loudness varies from a large (positive) A_0 to a minimum constant value A_{min}.

Based on these approximations, the basic steps of the Bat Algorithm (BA) can be summarized as the pseudo code described in Parpinelli and Lopes (2011).

Frequency f is assumed to be within $[0, f_{max}]$ (for simplicity) and the value of f_{max} depends on problem size. The rate of pulse can be in the range of $[0, 1]$ where 0 means no pulses at all, and 1 means the maximum rate of pulse emission. β is a random vector drawn from $(0, 1)$. x_i^t and v_i^t are solutions and velocities at time step t. x^* is the current global best solution which is located after comparing all the solutions among all the n bats. For local search part, A new solution for each bat is generated locally using a local random walk:

$$x_{new} = x_{old} + \epsilon A^t$$

where $\epsilon \in [-1, 1]$ is a random number while A^t is the average loudness of all the bats at this time step.

The loudness usually decreases once a bat has found its prey, while the rate of pulse emission increases which is justified by the functions taken in algorithm for the same. We can take $A_0 = 1$ and $A_{min} = 0$ where $A_{min} = 0$ means that a bat has just found the prey and temporarily stopped emitting any sound. Constant α is similar to the cooling factor of a cooling schedule in the simulated annealing. Here $0 < \alpha < 1$ and $\gamma > 0$, For simplicity one can use $\alpha = \gamma = 0.9$. Each bat should have different values of loudness and pulse emission rate, and this can be achieved by randomization. The initial loudness A_i^0 can typically be $[0,1]$ while the initial emission rate r_i^0 can be any value between $[0,1]$. Their loudness and emission rates will be updated only if the new solutions are improved, which means that these bats are moving towards the optimal solution.

2.1 Modified Bat Algorithm

In this section, we will describe some modifications made to original Bat Algorithm. In Bat Algorithm, frequency, velocity and location is generated for every iteration as shown in algorithm. It is clear that algorithm is moving in only one direction due to distance being positive. Firstly, We modified the location updating formulae as follows

$$\vec{x}^t = \vec{x}_i^{t-1} + \vec{v}_i^t * e,\ e \in [-1,1]$$

Secondly, since we are keeping track of only one best solution, selection among the best solution is eliminated (Step 16 of Algorithm in Parpinelli and Lopes (2011)). As there is inherent randomness involved in algorithm, it is not required to generate a new solution by flying randomly inside algorithm (Step 19 of Algorithm in Parpinelli and Lopes (2011)). Another modification in the condition to update r_i and A_i are made in the sense: Instead of comparing fitness of newly generate solution with current best we are comparing with previous fitness of that bat sothat bats will locally move in right direction (Step 20 of Algorithm in Parpinelli and Lopes (2011)). We are updating current best after each evaluation. An algorithm based on above modifications is shown in 2.1.

3 Description of Optimization Problems

We are considering following deterministic and stochastic optimization problems

1. **Rosenbrock:** D dimensional Rosenbrock function is defined by

$$f(x_1, x_2, \cdots, x_D) = \sum_{d=1}^{D-1} 100(x_{d+1} - x_d^2)^2 + (x_d - 1)^2.$$

The global minimum is inside a long, narrow, parabolic shaped flat valley. To find the valley is trivial. To converge to the global minimum, however, is difficult. It has a global minimum at $(x_1, x_2, \cdots, x_D) = (1, 1, \cdots, 1)$ where $f(x_1, x_2, \cdots, x_D) = 0$. The above function is for deterministic case and stochastic version of given function is as follows:

$$f(x_1, x_2, \cdots, x_D) = \sum_{d=1}^{D-1} 100r(x_{d+1} - x_d^2)^2 + (x_d - 1)^2.$$

2. **Rastrigin:** D dimensional Rastrigin function is defined by

$$f(x_1, x_2, \cdots, x_D) = 10D + \sum_{i=1}^{D} \left[x_i^2 - 10\cos 2\pi x_i\right]$$

Rastrigin function has many local minima i.e. the "valleys" in the plot. However, the function has just one global minimum, which occurs at the point $[0, 0, \cdots, 0]$ where the value of the

Algorithm 1 Modified Bat Algorithm

```
Require: parameters $n, \alpha, \gamma$, Number of iterations (N), lb, ub
 1: Initialise the bats population $\vec{x_i}$ randomly, $\forall i = 1, 2, \dots n$, $t = 0$
 2: for i = 1 to n do
 3:     Define pulse frequency $f_i$ at every $\vec{x_i}$.
 4:     Initialise pulse rates $r_i^0$ and velocity $v_i^0$
 5:     Randomly generate loudness $A_i$ from $unif(0, 1)$
 6: end for
 7: Compute fitness of each bat $F(\vec{x_i})$, $\forall i = 1, 2, \dots n$
 8: find the current best $\vec{x}_*$
 9: while (t < N) do
10:     Itebest ← large value
11:     for i = 1 to n do
12:         Generate new solutions by adjusting:
13:         Frequency: $f_i = f_{min} + (f_{max} - f_{min})\beta, \beta \in [0, 1]$
14:         Velocity: $v_i^t = v_i^{t-1} + (\vec{x}^{t-1} - \vec{x}_*)f_i$
15:         Location: $\vec{x}_i^t = \vec{x}_i^{t-1} + v_i^t * e$, $e \in [-1, 1]$
16:         if $(rand > r_i)$ then
17:             Generate a local solution around the selected best solution:
18:             $\vec{x_i}^t = \vec{x}_* + \epsilon\bar{A}$, $\epsilon \in unif[-1, 1]$
19:         end if
20:         if $(\vec{x_i}^t \notin [lb, ub])$ then
21:             Generate a random solution in the range [lb, ub]
22:         end if
23:         if $rand < A_i$ & $F(\vec{x}_i^t) < F(\vec{x}_i^{t-1})$ then
24:             Increase $r_i$: $r_i = r_i^0(1 - exp(-\gamma t))$
25:             Decrease $A_i$: $A_i = \alpha A_i$
26:         end if
27:         Update iteration best (itebest)
28:     end for
29:     Find the current best $\vec{x}_*$
30:     $t = t + 1$
31: end while
32: Post process results and visualisation
```

function is 0. At any local minimum other than $[0, 0, \cdots, 0]$, the value of Rastrigin's function is greater than 0. We consider stochastic version of Rastrigin function as follows:

$$f(x_1, x_2, \cdots, x_D) = 10D + \sum_{i=1}^{D} r\left[x_i^2 - 10\cos 2\pi x_i\right]$$

Here r is a random variable in both stochastic equations and is taken as *unif(0,1)* and *unif(0,10)* in our experiments.

Experiments: We studied following 12 cases.

$$\left\{ \begin{array}{l} Rosenbrock \\ Rastrigin \end{array} \right\} \times \left\{ \begin{array}{l} Deterministic \\ Stochastic - \epsilon \in U[0,1] \\ Stochastic - \epsilon \in U[0,10] \end{array} \right\} \times \left\{ \begin{array}{c} 2-D \\ 6-D \end{array} \right\}$$

4 Implementation Details

Algorithm 2 B1. Implementation

```
Require:
1: Initial_experiment_setup
2: Call initializeAlgo()
3: while stopping condition not met, say, iterations $\leq$ 100*M do
4:     for current_replication = 1 ... Max_replication do
5:         Obtain the new solution by calling setVariable() /* Before simulation run */
6:         Update the solution in the Anylogic Model
7:         Simulate the model
8:         Record the objective value obtained from simulation model/* After_simulation_run */
9:         After_Iteration
10:    end for
11:    Send the mean objective value using functionEval(objective)
12: end while
```

Algorithm B2: Java code which implements the metaheuristic (Bat Algorithm).

Global variables: n, α, γ, *bestobjective*, *lb*, *ub*, N, f_{max}, v_i, r_i, *bestX*, A, *iteration_count*, *batno*, *fitness*, *bat_best*

1. initializeAlgo()

    1. Input parameters: n, α, γ, *lb*, *ub*, f_{max}, v_i, r_i, *seed*
    2. Randomly generate a initial bat population, fitness of each bat and loudness (A_i)
    3. *iteration_count* $\leftarrow 0$
    4. *batno* $\leftarrow 0$
    5. *bestobjective* $\leftarrow \infty$ for maximization or $-\infty$ for minimization

2. setVariable()

    (a) $x \leftarrow$ assign current bat position

    (b) return x

3. functionEval(objective)

    (a) if (current objective < *bestobjective*) // For minimization problem.
    1.1 *bestX* $\leftarrow x$

    (b) *batNo* $\leftarrow$ *batNo* +1

    (c) if (*batNo* == n)
    3.1 Generate n new solutions using different equations (refer to steps 15, 16, 17 in algo 2)
    3.2 *batNo* $\leftarrow 0$

    (d) if (*iteration_count* == N)
    4.1 Stop; return *bestobjective*

5 Results and Discussion

All functions are tested based on following measure of performance.

1. **Quality of the solution:** Quality of solution is calculated in terms of the gap between known optimal and solution coming out of algorithm.
2. **Number of iterations:** Total number of iterations are calculated to get the final objective as well as objective hits 0.1 and 0.05.
3. **Execution time:** Total execution time is observed to get the final objective as well as objective hits 0.1 and 0.05.

It has been observed that following four parameters are affecting the performance of the algorithm. So these parameters has to be tuned according to the problem under consideration. Parameters are number of bats (n), maximum frequency of bat (f_{max}), velocity of each bat ($\bar{v}_i^0$) and pulse rate of bat i (r_i^0). We conducted the initial trial runs for each function and selected parameters as shown in following table.

	n	f_{max}	$\bar{v}_i^0$	r_i^0
Rosenbrock - Deterministic	1	0.01	0.4	0.4
Rastrigin - Deterministic	2	0.01	0.5	0.5
Rosenbrock - Stochastic	1	0.01	0.4	0.4
Rastrigin - Stochastic	2	0.01	0.5	0.5

Table 1: Parameter Setting

Each function is tested with 10 different seeds (17, 802, 1729, 19249, 65536, 2101869, 6012012, 15081947, 18967282, 31415926) and average, minimum and maximum performance is tabulated. We have plotted graph of iteration Vs best objective and time Vs best objective for illustration purpose.

5.1 Rosenbrock - Deterministic - 2D

Parameter setting: $n = 1,\ f_{max} = 0.01,\ \bar{v}_i^0 = 0.4, r_i^0 = 0.4$.

						Z≤0.05			Z≤0.1		
	Iteration	Time	Zbest	X	Y	Iteration	Time	Z	Iteration	Time	Z
Min	37401	152	0.0004	0.9686	0.9379	28	0	0.0056	28	0	0.028
Max	99982	377	0.0058	1.0762	1.1588	9845	42	0.0464	1059	14	0.0993
Avg	72610	289.6	0.0024	1.00504	1.0187	1213.5	12	0.02951	252.2	8.5	0.06402

Observations: Above results show that BA takes less than 10000 iterations to achieve an accuracy level of 0.05. Average time taken to reach this level is 12 ms. Average objective with 1 lakh iterations is 0.0024 which is quite close to optimal solution. For few seeds (802, 1729, 15081947), Algorithm achieves relatively good solution in less number of iterations.

5.2 Rastrigin - Deterministic - 2D

Parameter setting: $n = 2,\ f_{max} = 0.01,\ \bar{v}_i^0 = 0.5, r_i^0 = 0.5$.

						Z ≤ 0.05			**Z ≤ 0.1**		
	Iteration	**Time**	**Zbest**	**X**	**Y**	**Iteration**	**Time**	**Z**	**Iteration**	**Time**	**Z**
Min	12607	53	0	-0.0185	-0.0263	35	5	0.0118	35	5	0.0134
Max	98346	385	0.2735	0.0263	0.0185	60841	240	0.0492	60735	240	0.1
Avg	68868.9	275.2	0.07365	-0.00233	0.00233	21517.8	91	0.0237	24886.71	103.28	0.06

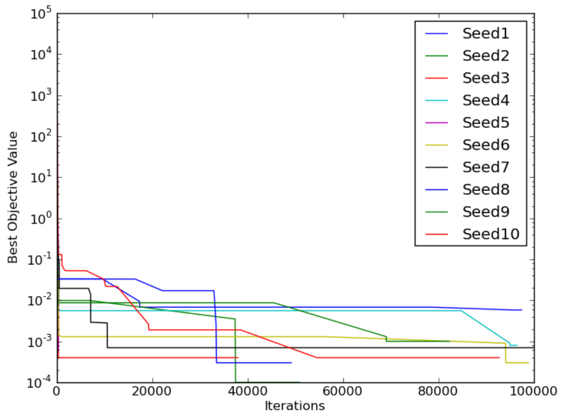

Figure 1: Iterations-vs-best objective

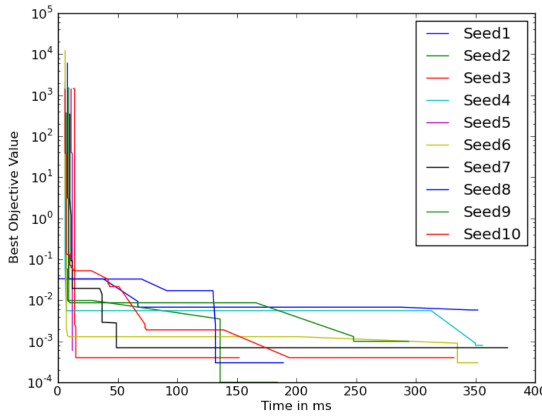

Figure 2: Time-vs-best objective

5.3 Rosenbrock - Stochastic(unif(0,1)) - 2D

Parameter setting: $n = 1,\ f_{max} = 0.01,\ \bar{v}_i^0 = 0.4, r_i^0 = 0.4$.

						Z ≤ 0.05			**Z≤0.1**		
Seed	**Iteration**	**Time**	**Zbest**	**X**	**Y**	**Iteration**	**Time**	**Z**	**Iteration**	**Time**	**Z**
Min	18762	562	0	0.9694	0.9387	19	0	0.0022	19	0	0.0112
Max	96483	2961	0.0023	1.0471	1.0975	5792	179	0.0473	4135	128	0.098
Avg	71314.9	2167.3	0.00057	1.00153	1.0036	1158.7	42.9	0.02693	640.5	27.4	0.05225

5.4 Rastrigin - Stochastic(unif(0,1)) - 2D

Parameter setting: $n = 2,\ f_{max} = 0.01,\ \bar{v}_i^0 = 0.5, r_i^0 = 0.5$.

						Z≤0.05			**Z≤0.1**		
Min	37290	1269	0.0003	-0.0231	-0.0155	2852	103	0.0327	171	17	0.0327
Max	96968	3201	0.4239	0.0154	0.9829	78519	2557	0.0499	78395	2557	0.9501
Avg	76105.8	2513.3	0.08878	-0.00511	0.10189	36040.16	1206.67	0.04	27880.87	937	0.06

5.5 Rosenbrock - Stochastic(unif(0,10)) - 2D

Parameter setting: $n = 2,\ f_{max} = 0.01,\ \bar{v}_i^0 = 0.5, r_i^0 = 0.5$.

						Z≤0.05			**Z≤0.1**		
Seed	**Iteration**	**Time**	**Zbest**	**X**	**Y**	**Iteration**	**Time**	**Z**	**Iteration**	**Time**	**Z**
Min	54142	2168	0.0003	0.8952	0.7935	14	95	0.0032	14	95	0.0032
Max	99266	4189	0.011	1.0731	1.1681	27591	1237	0.05	7791	395	0.0999
Avg	84844.4	3439.7	0.00213	1.00126	1.00797	8240.5	411.9	0.02858	2474.3	183.3	0.06827

5.6 Rastrigin - Stochastic(unif(0,10)) - 2D

Parameter setting: $n = 2,\ f_{max} = 0.01,\ \bar{v}_i^0 = 0.5, r_i^0 = 0.5$.

						Z≤0.05			**Z≤0.1**		
Seed	**Iteration**	**Time**	**Zbest**	**X**	**Y**	**Iteration**	**Time**	**Z**	**Iteration**	**Time**	**Z**
Min	7122	373	0.001	-3.0076	-1.9918	539	81	0.004	505	78	0.004
Max	96104	4211	0.2743	0.9931	3.0221	57411	2640	0.0477	57411	2640	0.0965
Avg	63672.9	2841.7	0.05187	-0.10462	0.403	13038.42	605.42	0.02	11036.11	521.44	0.0539

5.7 Rastrigin - Deterministic - 6D

	Iteration	Time	Zbest	X1	X2	X3	X4	X5	X6
Min	33146	1494	9.3022	-2.0136	-1.0213	-1.9055	-2.0876	-0.9896	-1.1051
Max	99682	4398	24.8441	0.116	3.0064	2.0453	1.9865	2.0732	3.0607
Avg	72546.46	2928.3	17.56	-0.74	0.65	-0.001	0.15	1.03	0.86

5.8 Rosenbrock - Deterministic - 6D

	Iteration	Time	Zbest	X1	X2	X3	X4	X5	X6
Min	12638	257	16.1776	-1.0324	-0.2668	-0.1811	-0.1081	-0.5097	-0.082
Max	91342	2459	75.9857	0.9107	0.9238	1.1567	1.3006	1.0588	1.4416
Avg	53402.45	1210.5	50.83	-0.27	0.36	0.53	0.44	0.21	0.51

5.9 Rastrigin - Stochastic(unif(0,1)) - 6D

	Iteration	Time	Zbest	X1	X2	X3	X4	X5	X6
Min	48141	1004	2.3829	-3.0671	-3.9507	-4.1082	-5.049	-4.9925	-2.1029
Max	96694	2235	7.4419	4.9459	1.9838	2.0526	5.0895	5.0467	4.0834
Avg	80018.45	1835.01	4.51	1.042	-0.68	0.13	-1.07	0.71	1.091

5.10 Rosenbrock - Stochastic(unif(0,1)) - 6D

	Iteration	Time	Zbest	X1	X2	X3	X4	X5	X6
Min	1997	22	2.2991	-0.0554	-0.293	-1.4544	-1.2483	-1.1742	-2.2631
Max	98941	2021	17.6838	2.321	1.992	1.402	2.0014	1.5275	2.4789
Avg	51910.6	1062.36	9.17	1.02	0.79	0.13	0.48	0.13	0.20

5.11 Rastrigin - Stochastic(unif(0,10)) - 6D

	Iteration	Time	Zbest	X1	X2	X3	X4	X5	X6
Min	42156	727	5.9953	-3.9302	-5.1182	-2.9709	-1.8903	-1.068	-5.0377
Max	97522	2039	17.0423	3.9736	3.1312	3.9461	5.0135	5.0396	3.1048
Avg	69021.94	1262.82	11.53	0.01	-1.37	1.06	1.90	2.63	-1.09

5.12 Rosenbrock - Stochastic(unif(0,10)) - 6D

	Iteration	Time	Zbest	X1	X2	X3	X4	X5	X6
Min	1997	25	5.5831	-0.8148	-2.1364	-2.8132	-1.6094	-2.3869	-2.1225
Max	96694	1203	30.4146	2.1335	2.4166	0.926	1.799	3.1326	4.3958
Avg	50684.26	634.13	17.10	0.585	0.195	-0.69	0.230	0.305	1.058

5.13 Discussion

On adding stochasticity, the solutions are improving for both the functions. In 2-D case, the solution worsen by adding little stochasticity and then improves by further increasing stochasticity for Rosenbrock function while in case of Rastrigin functions results change in opposite way. In 6-D case, the solution improves by adding little stochasticity (unif(0,1)) but deteriorates on further increasing randomness for both functions. Based on above observation, Optimal level of stochasticity should be added to get the best solution.

Best solution obtained with 6-D case is 4.51 (Rastigin- stochastic(unif(0,1))) which is not close to 0 while in 2-D case we always obtained the convergence level of 0.1. Hence increasing dimensionality deteriorates performance more than the stochasticity.

6 Conclusions and Future Work

We presented implementation of modified bat algorithm for different deterministic and stochastic functions. Results for ten different seeds are shown in table. It can be observed from different graphs that BA algorithm initially converges very fast and after a certain level it's convergence rate becomes slow. So this algorithm can be used for initial convergence. This algorithm is performing worse in case of Rastrigin function in both deterministic and stochastic setting as BA is not achieving 0.1 level of accuracy in all seeds. It is observed from the results that BA takes less computational time. One can study the effect of dimensionality and stochasticity in the convergence of algorithm. Optimizing highly stochastic systems such as queueing models and (s, S) inventory systems using bat algorithm can be another interesting future avenue.

References

Altringham, J., McOwat, T., Hammond, L., 1998. Bats:Biology and Behaviour. Oxford Univesity Press.

Komarasamy, G., Wahi, A., 2012. An optimized k-means clustering technique using bat algorithm. European Journal of Scientific Research 84 (2), 263–273.

Musikapun, P., Pongcharoen, P., 2012. Solving multi-stage multi-machine multi-product scheduling problem using bat algorithm. In: International Conference on Management and Artificial Intelligence. IACSIT press.

Parpinelli, R., Lopes, H., 2011. New inspirations in swarm intelligence: a survey. Int. J. Bio-Inspired Computation.

Tsai, P. W., Pan, J. S., Liao, B. Y., Tsai, M. J., Istanda, V., 2011. Bat algorithm inspired algorithm for solving numerical optimization problems. Applied Mechanics and Materials 148-149, 134–137.

Yang, X. S., 2010. A new mataheuristic bat inspired algorithm. Studies in Computational Intelligence.

Yang, X. S., 2011. Bat algorithm for multiobjective optimization. international journal of Bio-Inspired Computation 3 (5), 267–274.

Yang, X.-S., Gandomi, A. H., 2012. Bat algorithm: a novel approach for global engineering optimization. Engineering Computations 29, 464–483.