Parameterless Bat Algorithm and Its Performance Study

Iztok Fister Jr., Uroš Mlakar, Xin-She Yang and Iztok Fister

Abstract A parameter-free or parameterless bat algorithm is a new variant of the bat algorithm which was recently introduced. Characteristic of this algorithm is that user does not need to specify the control parameters when running this algorithm. Thus, this bat algorithm variant can have wide usability in solving real-world optimization problems. In this chapter, a preliminary study of the proposed parameterless bat algorithm is presented.

Keywords Bat algorithm · Control parameters · Optimization · Swarm intelligence

1 Introduction

As the global market becomes more competitive, it necessitates that companies should design their products and services in a more cost-effective and sustainable way so as to maximize their profits and performance and to minimize the costs and errors. All these must also meet the highest standards subject to various design constraints. This requires researchers and designers to follow the latest trends, to use the most cost-effective technologies and to use innovative optimization techniques. In recent years, artificial intelligence (AI) has started to become one of the most useful tools in industries and manufacturing engineering in the sense that artificial intelligence can help to design, develop, manufacture products more intelligently,

I. Fister Jr. (✉) · U. Mlakar · I. Fister
Faculty of Electrical Engineering and Computer Science,
University of Maribor, Smetanova 17, 2000 Maribor, Slovenia
e-mail: iztok.fister1@um.si

U. Mlakar
e-mail: uros.mlakar@um.si

I. Fister
e-mail: iztok.fister@um.si

X.-S. Yang
School of Science and Technology, Middlesex University, London NW4 4BT, UK
e-mail: x.yang@mdx.ac.uk

X.-S. Yang (ed.), *Nature-Inspired Computation in Engineering*,
Studies in Computational Intelligence 637, DOI 10.1007/978-3-319-30235-5_13

and such applications include robotics, car production, information technologies and obviously computer games.

Over the past few decades, many different optimization methods have been developed, and it is estimated there may be more than 100 different algorithms in the literature including the many variants of nature-inspired algorithms [7]. In fact, nature-inspired algorithms have become promising and powerful for solving the real-world optimization problems [14] and these algorithms often mimic the behavior of natural and biological systems.

In this chapter, we focus on the bat algorithm (BA) [13] and its extension to a parameter-free variant. In spite of its simplicity, this is a very efficient algorithm. The original BA has five parameters that represent a potential problem for users which usually do not know how to specify them properly. Therefore, this work introduces a new parameterless BA (i.e., PLBA) that eliminates this drawback by proposing techniques for a rational and automated parameter setting on behalf of the user. This study is an extension of our conference paper [6] where we presented a new variant of bat algorithm called parameterless BA (PLBA) that was based on the main idea of Lobo and Goldberg [11]. Here, the performance study has been extended in order to show the behavior of PLBA on dimensions higher than $D = 10$.

Therefore, the organization of this chapter is as follows. Section 2 provides the fundamentals of the bat algorithm. Section 3 discusses control parameters in the bat algorithm. Section 4 focuses on the experiments and presents the results obtained from experiments. Then the chapter concludes with the discussion of the performed work and some directions for the future are outlined.

2 Fundamentals of the Bat Algorithm

The standard bat algorithm was developed by Xin-She Yang in [13], and three idealized rules were used to capture some of the characteristics of microbats in nature:

- All bats use echolocation to sense the distance to target objects.
- Bats fly with the velocity v_i at position x_i, the frequency $Q_i \in [Q_{min}, Q_{max}]$ (also the wavelength λ_i), the rate of pulse emission $r_i \in [0, 1]$, and the loudness $A_i \in [A_0, A_{min}]$. The frequency (and wavelength) can be adjusted depending on the proximities of their targets.
- The loudness varies from a large (positive) A_0 to a minimum constant value A_{min}.

These idealized rules can be outlined as the pseudocode presented in Algorithm 1. The BA is a population-based algorithm with five parameters [6], and its population size is Np. The main loop of the algorithm (lines 4–16 in Algorithm 1) starts after the initialization (line 1), the evaluation of the generated solutions (line 2) and determination of the best solutions (line 4).

Algorithm 1 Bat algorithm

Input: Bat population $\mathbf{x_i} = (x_{i1}, \ldots, x_{iD})^T$ for $i = 1 \ldots Np$, MAX_FE.
Output: The best solution $\mathbf{x}_{best}$ and its corresponding value $f_{min} = \min(f(\mathbf{x}))$.
1: init_bat();
2: $eval$ = evaluate_the_new_population;
3: f_{min} = find_the_best_solution($\mathbf{x}_{best}$); {initialization}
4: **while** termination_condition_not_met **do**
5: **for** $i = 1$ **to** Np **do**
6: $\mathbf{y}$ = generate_new_solution($\mathbf{x}_i$);
7: **if** rand(0, 1) $< r_i$ **then**
8: $\mathbf{y}$ = improve_the_best_solution($\mathbf{x}_{best}$)
9: **end if** { local search step }
10: f_{new} = evaluate_the_new_solution($\mathbf{y}$);
11: $eval = eval + 1$;
12: **if** $f_{new} \leq f_i$ **and** N(0, 1) $< A_i$ **then**
13: $\mathbf{x}_i = \mathbf{y}$; $f_i = f_{new}$;
14: **end if** { save the best solution conditionally }
15: f_{min}=find_the_best_solution($\mathbf{x}_{best}$);
16: **end for**
17: **end while**

Briefly speaking, the BA algorithm consists of the following elements:

- generating a new solution (line 6),
- improving the best solution (lines 7–9),
- evaluating the new solution (line 10),
- saving the best solution conditionally (lines 12–14),
- determining the best solution (line 15).

The generation of a new solution obeys the following equations:

$$\begin{aligned} Q_i^{(t)} &= Q_{min} + (Q_{max} - Q_{min})N(0, 1), \\ \mathbf{v}_i^{(t+1)} &= \mathbf{v}_i^t + (\mathbf{x}_i^t - \mathbf{x}_{best})Q_i^{(t)}, \\ \mathbf{x}_i^{(t+1)} &= \mathbf{x}_i^{(t)} + \mathbf{v}_i^{(t+1)}, \end{aligned} \tag{1}$$

where $N(0, 1)$ is a random number drawn from a uniform distribution, and $Q_i^{(t)} \in [Q_{min}^{(t)}, Q_{max}^{(t)}]$ is the frequency determining the magnitude of the velocity change. The improvement of the current best solution is performed according to the following equation:

$$\mathbf{x^{(t)}} = \mathbf{x}_{best} + \epsilon A_i^{(t)} N(0, 1), \tag{2}$$

where $N(0, 1)$ denotes the random number drawn from a Gaussian distribution with a zero mean and a standard deviation of one. In addition, ϵ is the scaling factor and $A_i^{(t)}$ the loudness. The improvement of the current best solution is controlled by a parameter r_i.

It is worth pointing out that these parameters can somehow balance the exploration and exploitation components of the BA search process, where exploitation is governed by Eq. (1) and exploitation by Eq. (2).

The evaluation function models the characteristics of the problem to be solved. The archiving of the best solution conditionally is similar to that used in simulated annealing (SA) [10], where the best solution is taken into the new generation according to a probability controlled by the parameter A_i. In fact, the parameter A_i prevents the algorithm to get stuck into a local optimum.

The bat algorithm has attracted a lot of interests in the literature and has been applied to many applications. For a relatively comprehensive review, please refer to Yang and He [15].

2.1 How to Control Parameters in Bat Algorithm?

As is the case for all evolutionary algorithms (EA) [4] and swarm-intelligence based (SI-based) algorithms [1], they are stochastic algorithms and all these algorithms have algorithm-dependent parameters. The values of these parameters will largely affect the performance of the algorithm under consideration. To control these parameters is equivalent to guiding the algorithm's search process during the exploration of the search space of an optimization algorithm and hence may have a huge influence on the quality of obtained solutions. Thus, parameter tuning and control become an important topic for active research, though the setting of these parameters may depend on the type of problems and may be well problem-specific. For example, the bat algorithm is guided by five parameters, and the following parameters are the part of basic bat algorithm:

- the population size NP,
- loudness A_i,
- pulse rate r_i,
- minimum frequency Q_{min} and
- maximum frequency Q_{max}.

In order to avoid the tedious task for tuning parameters, a parameterless variant of the bat algorithm has been proposed here and will be presented in detail in the rest of this chapter.

3 Design of a Parameterless Bat Algorithm

In order to develop a new parameterless BA (PLBA), the influence of these algorithm-dependent parameters was studied and extensive studies revealed that some algorithm parameters can be rationally set, while the optimal setting of another parameters needs to be found automatically. For example, the parameter $Q_i \in [Q_{min}, Q_{max}]$

determines the magnitude of the change and settings of parameters Q_{min} and Q_{max} depend on the problem of interest. However, the rational setting of this parameter can be approximated with the lower x_i^{Lb} and upper x_i^{Ub} bounds of the particular decision variables as follows:

$$Q_i^{(t)} = \frac{x_i^{(Ub)} - x_i^{(Lb)}}{Np} \cdot N(0, 1), \tag{3}$$

where $N(0, 1)$ has the same meaning as in Eq. (1). For example, when the $x_i^{(Lb)} = -100.0$ and $x_i^{(Ub)} = 100.0$, the frequency is obtained in the interval $Q_i^{(t)} \in [0, 2]$.

The rationality for setting the values of parameters r_i and A_i is based on the following consideration. Parameter r_i controls the exploration/exploitation components of the BA search process. The higher the value, the more the process is focused on the exploitation. However, the higher r_i also means that the modified copies of the best solution are multiplied in the neighborhood of the current population. As a result of high r_i, premature convergence to the local optimum may occur. Therefore, the rational setting of this parameter is to set r_i properly, and extensive numerical experiments suggested $r_i = 0.1$ is a good value that improves on average one in ten solutions in the current population.

Similar consideration is valid also for parameter A_i. The lower the parameter, the less time the best current solution is preserved in the new generation. Therefore, the rational selection of this parameter is $A_i = 0.9$ that preserves the 90 % of the best solutions and ignores the remaining 10 %.

The population size is also a crucial parameter in the BA. The lower population size may suffer from the lack of diversity, while a higher population size may cause slow convergence. However, the rational setting of the population size depends on the problem of interest. Therefore, an automatic setting of this parameter is proposed in the PLBA, where the population size is varied in the interval $Np \in [10, 1280]$ such that each population size is multiplied by two in each run starting with $Np = 10$. As a result, eight instances of the PLBA denoted as PL-1 to PL-8 are obtained and a user needs to select the best results among the instances generated.

4 Experiments and Results

The purpose of our experimental work was to show that the results of the proposed PLBA are comparable if not better than the results of the original BA. In line with this, the original BA was compared with eight instances of the PLBA using the rational and automatic parameter setting; i.e., Q_i was calculated according to Eq. (3) (e.q., $Q_i \in [0.0, 2.0]$), pulse rate and loudness were fixed as $r_i = 0.1$ and $A_i = 0.9$, while the population size was varied in the interval $Np = \{10, 20, 40, 80, 160, 320, 640, 1280\}$. In contrast, the original BA was run using the following parameters: $Np = 100$, $r_i = 0.5$, $A_i = 0.5$, and $Q_i \in [0.0, 2.0]$.

Table 1 Summary of the benchmark functions

Tag	Function name	Definition
f_1	Ackley's	$-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$
f_2	Griewank	$1 + \frac{1}{4000}\sum_{i=1}^{n}x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})$
f_3	Rastrigin	$10n + \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i))$
f_4	Sphere	$\sum_{i=1}^{n}x_i^2$
f_5	Whitley	$\sum_{i=1}^{n}\sum_{j=1}^{n}\left[\frac{(100(x_i^2-x_j)^2+(1-x_j)^2)^2}{4000} - \cos(100(x_i^2 - x_j)^2 + (1 - x_j)^2) + 1\right]$
f_6	Exponential_function	$\sum_{i=1}^{n}e^{ix_i} - \sum_{i=1}^{n}e^{-5.12i}$
f_7	Quartic	$\sum_{i=0}^{n}ix_i^4 + \text{random}[0, 1)$
f_8	Ridge	$\sum_{i=1}^{n}(\sum_{j=1}^{i}x_j)^2$
f_9	Schwefel	$\sum_{i=1}^{n}(-x_i\sin(\sqrt{\lvert x_i\rvert})) + 418.982887 \cdot n$
f_{10}	Double_Sum	$\sum_{i=1}^{n}(\sum_{j=1}^{i}(x_j - j)^2)$

Table 2 Function domain

Function	Domain
f_1	$[-32, 32]$
f_2	$[-512, 512]$
f_3	$[-5.12, 5.12]$
f_4	$[-100, 100]$
f_5	$[-10.24, 10.24]$
f_6	$[-5.12, 5.12]$
f_7	$[-1.28, 1.28]$
f_8	$[-64, 64]$
f_9	$[-512, 512]$
f_{10}	$[-10.24, 10.24]$

Interestingly, the setting of frequency in the original BA matches the rational setting in PLBA according Eq. (3).

Experiments were run on the benchmark suite consisting of ten functions presented in Table 1.

Test functions and the domains of parameters shown in Table 2 were mainly based on the Neal Holtschulte website [9].

The dimensions of functions were set to $D = 10$, $D = 20$, $D = 30$. All algorithms stopped after $FE = 1000 * D$ of fitness function evaluations and each algorithm was run 25 times. Three tests have been conducted according to each of the dimension used. It is worth pointing out that there are different suggestions concerning the number of function evaluations in the literature and most studies use far more numbers of evaluations such as 100,000D. However, as our purpose here is to mainly show this

parameter-free bat algorithm works, we will use far fewer numbers of evaluations. Obviously, the results will in general improve as the number of function evaluations increases.

In the remainder of the chapter, the results of the benchmark functions are presented in detail. In addition, statistical tests are illustrated in order to show the differences and to see how each algorithm performs. The results for the case of $D = 10$ suggested that the results are improved by increasing the population size. The best results were obtained by the biggest population size (i.e., $D = 1280$).

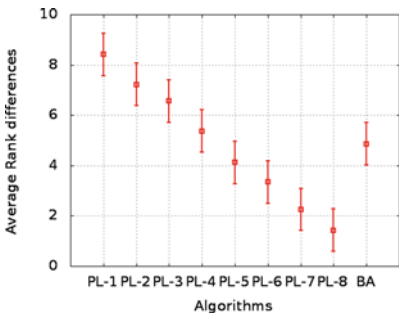
In order to estimate the results statistically, Friedman tests [8] were conducted. In essence, this Friedman test is a two-way analysis of variances where the test statistic is calculated and then converted to ranks, followed by post-hoc tests using the calculated ranks. Loosely speaking, a low value of rank means a better algorithm [3]. The post-hoc tests are performed only if a null hypothesis of the Friedman test is rejected. In short, the so-called null hypothesis corresponds to the case that medians between ranks of all algorithms are equal. According to Demšar [2], the Friedman test is a safe and robust non-parametric test for comparing more algorithms over multiple data sets, and this test, together with the corresponding Nemenyi post-hoc test, can ensure a neat presentation of statistical results [12]. This test have been conducted using a significance level 0.05 in this study.

Overall, statistical tests can typically reflect the performance of the algorithms against the benchmarks. In summary, nine classifiers (i.e., the results according to different population sizes) were compared, leading to $10 * 5 = 50$ variables, where the first number in the expression denotes the number of functions and the second the number of measures (i.e., minimum, maximum, mean, median and standard deviation). Three Friedman tests were performed regarding the different dimensions.

The results of the Friedman non-parametric test for dimension $D = 10$ are presented in Fig. 1 with a table and a diagram dedicated to the dimension under consideration. In the table, the results of Friedman tests are given together with corresponding Nemenyi post-hoc test. The results of Nemenyi post-hoc test are presented as intervals of critical differences. The best algorithm in Nemenyi post-hoc test is denoted with

Alg.	Friedman	Nemenyi	
		Critical difference	Sign
PL-1	8.41	[7.57,9.25]	†
PL-2	7.22	[6.38,8.06]	†
PL-3	6.56	[5.72,7.40]	†
PL-4	5.37	[4.53,6.21]	†
PL-5	4.12	[3.28,4.96]	†
PL-6	3.34	[2.50,4.18]	†
PL-7	2.25	[1.41,3.09]	
PL-8	1.43	[0.59,2.27]	‡
BA	4.86	[4.02,5.70]	†

(a) $D = 10$



(b) $D = 10$

Fig. 1 Statistical analysis of the results by optimizing functions $D = 10$

‡ symbol in the table, while the significant difference between the control method and corresponding algorithm is depicted by † symbol.

Schematically, the results of the Nemenyi post-hoc test are illustrated in a corresponding diagram. In the diagram, the points show the average ranks and the lines indicate the confidence intervals (critical differences) for the algorithms under consideration. The lower the rank value, the better the algorithm. On the other hand, two algorithms are said to be significantly different when their intervals do not overlap.

From Fig. 1, it can be seen that the PLBA with $Np = 1280$ (i.e., PL-8) is really the best instance according to the Friedman test. This instance outperformed all the other instances of the original BA and the PLBAs except for the case with a population size of $Np = 640$ (i.e., PL-7).

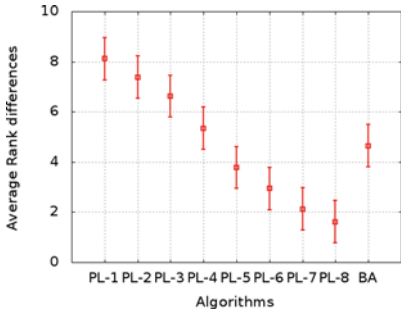
However, it is worth pointing out that the standard BA performs almost equally well, compared to the PLBA-4 with the population size of $Np = 80$. With the actual population size of $Np = 100$ in BA, this means that a moderate size can be sufficient to get very good results, not the necessarily the best results. This result is consistent with our observations and other research in the literature, which suggests that the results usually improve the population size increases for most population-based algorithms. However, there are some algorithms that may not be very sensitive to the population size. For example, the firefly algorithm usually can obtain good results for small and moderate population size $NP = 20$ to $NP = 100$ [5]. Therefore, the effect of the population size needs further investigation for nature-inspired algorithms.

Furthermore, the optimization results of the benchmark functions for $D = 20$ and $D = 30$ show similar trends to the case of dimension $D = 10$. For example, for $D = 30$, the Friedman tests are summarized in Fig. 2 where similar trends are observed.

In summary, the experiments showed that the original BA algorithm can obtain good results and its performance usually increase as the population size increases. On the other hand, the convergence rate of this algorithm can be very high and thus it may lead to premature convergence if the population size is too low as the algorithm can quickly converge to local optima. It seems that the parameterless BA algorithm

Alg.	Friedman	Nemenyi	
		Critical difference	Sign
PL-1	8.11	[7.27,8.95]	†
PL-2	7.38	[6.54,8.22]	†
PL-3	6.62	[5.78,7.46]	†
PL-4	5.34	[4.50,6.18]	†
PL-5	3.78	[2.94,4.62]	†
PL-6	2.94	[2.10,3.78]	
PL-7	2.13	[1.29,2.97]	
PL-8	1.62	[0.78,2.46]	‡
BA	4.65	[3.81,5.49]	†

(a) $D = 30$



(b) $D = 30$

Fig. 2 Statistical analysis of the results by optimizing functions $D = 30$

can avoid this problem using the higher population sizes. However, selecting the best population size during this automatic search for the proper parameter setting may still depend on the type of problems under consideration.

In addition, though the settings are fixed for A_i and r_i, it seems that a better approach for parameter-free variants should be adaptive; that is to vary A_i and r_i in an adaptive manner so that users need not to worry about the setting of the algorithms, and this adaptivity should also include the population size. More studies are needed to achieve these goals.

5 Conclusion

This chapter presented a performance study of recently proposed parameterless bat algorithm. This variant of the bat algorithm does not need to set the control parameters. Experiments were conducted on a set of ten standard benchmark functions with three different dimensions. In a nutshell, experiments confirmed that parameterless bat algorithm is a very promising variant with the potential to solve the real-world optimization applications.

However, this preliminary study may also suggest that other population-based algorithms can have similar effects of population sizes and the existing literature about particle swarm optimization seemed also to suggest that results can improve significantly as the population size increases. That is partly the reason that it is suggested the population size should be increased for high-dimensional problems.

It is worth pointing out that this parameterless variant of the bat algorithm used fixed parameters A_i and r_i, while allowing the variation of the population size. It can be expected that a more robust approach is to use adaptive settings to set all the parameters including the population size automatically for a given set of problems. But how to achieve this is still an open questions. There are some other open problems regarding automatic parameter tuning and control. Therefore, more detailed studies are highly needed to systematically analyze and study all major population-based metaheuristic algorithms.

References

1. Blum, C., Li, X.: Swarm intelligence in optimization. In: Blum, C., Merkle, D. (eds.) Swarm Intelligence: Introduction and Applications, pp. 43–86. Springer, Berlin (2008)
2. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)
3. Derrac, J., Garca, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol. Comput. **1**(1), 3–18 (2011)
4. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer, Berlin (2003)
5. Fister, I., Yang, X.-S., Brest, J., Fister Jr., I.: Modified firefly algorithm using quaternion representation. Expert Syst. Appl. **40**(18), 7220–7230 (2013)

6. Fister Jr., I., Fister, I., Yang, X.-S.: Towards the development of a parameter-free bat algorithm. In: StuCoSReC: Proceedings of the 2015 2nd Student Computer Science Research Conference, pp. 31–34 (2015)
7. Fister Jr., I., Yang, X.-S., Fister, I., Brest, J., Fister, D.: A brief review of nature-inspired algorithms for optimization. Elektrotehniški vestnik **80**(3), 116–122 (2013)
8. Friedman, M.: A comparison of alternative tests of significance for the problem of m rankings. Ann. Math. Stat. **11**, 86–92 (1940)
9. Holtschulte, N., Moses, M.: Should every man be an island. In: GECCO 2013 Proceedings, 8 pp. (2013)
10. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
11. Lobo, F.G., Goldberg, D.E.: An overview of the parameter-less genetic algorithm. In: Proceedings of the 7th Joint Conference on Information Sciences (Invited paper), pp. 20–23 (2003)
12. Nemenyi, P.B.: Distribution-free multiple comparisons. Ph.D. thesis, Princeton University (1963)
13. Yang, X.-S.: A new metaheuristic bat-inspired algorithm. In: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), pp. 65–74. Springer (2010)
14. Yang, X.-S.: Nature-Inspired Optimization Algorithms. Elsevier (2014)
15. Yang, X.-S., He, X.: Bat algorithm: literature review and applications. Int. J. Bio-inspir. Comput. **5**(3), 141–149 (2013)