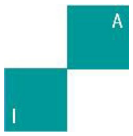
Inteligencia Artificial 26(72), 102-123
doi: 10.4114/intartif.vol26iss72pp102-123



INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

An Improved BAT Algorithm Using Density-Based Clustering

Samraa Adnan Al-Asadi[1], Safaa O. Al-Mamory[2]

[1] Department of Information Networks / College of Information Technology, University of Babylon
Email: samraa.alasadi@uobabylon.edu.iq

[2] College of Business Informatics, University of Information Technology and Communications
Email: salmamory@uoitc.edu.iq

Abstract. BAT algorithm is a nature-inspired metaheuristic algorithm that depends on the principle of the echolocation behavior of bats. However, due to poor exploration, the algorithm suffers from being stuck in the local optima. An improved BAT algorithm based on the density-based clustering technique is proposed to enhance the algorithm's performance.

In this paper, the initial population is improved by generating two populations, randomly and depending on the clusters' center information, and by getting the fittest individuals from these two populations, the initial improved one is generated. The random walk function is improved using chaotic maps instead of the fixed-size movement, so the local search is improved, as well as the global search abilities, by diversifying the solutions. Another improvement is dealing with stagnation by partitioning the search space into two parts depending on the generated clusters' information to obtain the newly generated solution, comparing their quality with the previously generated solution, and choosing the best.

The performance of the proposed improved BAT algorithm is evaluated by comparing it with the original BAT algorithm over ten benchmark optimization test functions. Depending on the results, the improved BAT outperforms the original BAT by obtaining the optimal global solutions for most of the benchmark test functions.

Keywords: Metaheuristic Algorithms, BAT Algorithm, Density-based Clustering, Chaotic Strategies

1 Introduction

Due to technological development, many complicated optimization problems appear in many different fields, like engineering, manufacturing, information technology, and economic management. [1]. These problems are solved using optimization algorithms. Optimization algorithms are categorized as deterministic algorithms and stochastic algorithms. Deterministic algorithms, like the simplex method in linear programming, usually need gradient information. The problems with one global optimum are effectively solved by deterministic algorithms, but for problems with many local optima or problems with unavailable gradient information, they may be invalid. On the other hand, stochastic algorithms require only the information of the objective function. Heuristic and metaheuristic are generally the two types of stochastic algorithms [2]. Heuristic approaches are used to find the optimum or at least near-optimum solutions. These approaches do not guarantee to reach that global optimum solution. A high-level heuristic is called a metaheuristic [3].

Metaheuristic algorithms, for example, Simulated Annealing (SA) [4] and Particle Swarm Optimization (PSO) [5], are very powerful in solving hard optimization problems, so they have been applied in almost all significant areas of engineering and science and industrial applications [6]. For example, the optimum weights of the artificial neural network are found by using PSO instead of using the Backpropagation algorithm due to its better classification accuracy and faster processing time when compared with the Backpropagation algorithm [7], and BAT algorithm is used for attributes and features selection [8]. Generally, some prominent applications of metaheuristic algorithms are [9][10]: ***Combinatorial Optimization,*** where some metaheuristic algorithms like

ISSN: 1137-3601 (print), 1988-3064 (on-line)

Genetic Algorithms (GA) [11], PSO, and Ant Colony Optimization (ACO) [12] are widely used to solve many combinatorial optimization problems, like the Traveling Salesman Problem, Knapsack Problem, and Vehicle Routing Problem. ***Engineering Design and Manufacturing,*** is a field where metaheuristic algorithms are applied to find an optimal or near-optimal solutions for some complex design problems, such as structural optimization and material selection. Another application is ***Machine Learning and Data Mining,*** where metaheuristic algorithms are employed in feature selection and machine learning models' training. ***Scheduling and Resource Allocation,*** is an application where metaheuristic algorithms are applied to solve scheduling and resource allocation problems in various domains, including transportation, manufacturing, project management, and in edge and fog computing. Another application is ***Network Design and Routing,*** where metaheuristic algorithms are used in network design to optimize the placement of network components and routing of data packets in communication networks. Many other fields where metaheuristic algorithms are used, like ***Financial and Economic Applications, Robotics, and Bioinformatics.***

Metaheuristic algorithms can be classified in many ways. Population-based and trajectory-based are one way of classification. The population-based algorithm, like the PSO algorithm, initializes the population randomly. At each iteration, the population will be enhanced and substituted partially or totally by the newly generated population (solutions). On the other hand, simulated annealing uses a single agent or solution that moves through the design space or search space in a piecewise style. A better move or solution is always accepted at each iteration, while a not-so-good move can be accepted with a certain probability. The steps or movements trace a trajectory in the search space, with a non-zero probability that this trajectory can reach the global optimum [2][13]. Another classification is based on whether the algorithm is nature inspired or not. Nature-inspired algorithms are classified as Swarm Intelligence and Evolutionary Algorithms. Swarm Intelligence like PSO, BAT inspired Algorithm, Ant Colony Optimization. Genetic Algorithm is an example of Evolutionary Algorithms. Examples of non-nature-inspired algorithms are Simulated Annealing and Tabu Search [14]. Other classifications are used as whether the algorithm is deterministic or stochastic, memory or memoryless, and whether it is iterative or greedy [13].

The BAT Algorithm is a swarm intelligence-based metaheuristic optimization algorithm introduced by Xin-She Yang in 2010 [15] as a global optimization algorithm that can solve complex optimization problems effectively and be applied to various real-world optimization problems in engineering, science, and economics. The Bat algorithm effectively solves many optimization problems, including continuous, discrete, and multi-objective problems. The algorithm is also known for its simplicity, fast convergence, and ability to handle noisy and dynamic environments.

The use of sound waves and echoes to determine the location and nature of objects is termed echolocation [1]. The algorithm is inspired by the echolocation behavior of bats, a biological phenomenon bats use to navigate in the dark. The BA mimics the behavior of bats in searching for food. Bats use their echolocation system to detect prey and fly toward it. They emit ultrasonic waves and listen to the echoes reflected from their surrounding objects to locate their prey. The BAT algorithm employs the same strategy, where bats represent candidate solutions, and their frequency of emitting pulses corresponds to the quality of their fitness function.

In this paper, an improvement to the BAT algorithm is proposed. The improvement is regarding the algorithm's initialization, exploration, and exploitation processes. The improvement is done with the aid of the clustering technique and chaotic maps. A Density-based clustering approach is used to produce k-dense clusters. Through the information on these clusters (regarding the clusters' center and the similarity between clusters), the BAT algorithm is improved.

The main contributions of this work are:

1- Using of density-based clustering technique to improve the algorithm.
2- Improving the initialization phase of the algorithm by improving the initial population depending on the information of clusters' centers.
3- Balancing the local and global search of the algorithm using two chaotic maps instead of the random walk.
4- Overcoming the stagnation issue, also depending on the clusters, by measuring the similarity between these clusters.

The paper is organized as follows. Section 2 presents related previous improvement researches. Section 3 explains the original BAT algorithms in detail, its flowchart and steps. In Section 4, the improved BAT algorithm is introduced. Section 5 presents the details of the experiments, the performance evaluation test functions, the testing parameters, and the results. Finally, Section 6 concludes this paper.

2 Related Works

Intensification (exploitation) and diversification (exploration) are the two primary components of any metaheuristic algorithm. Diversification means exploring the search space globally to generate diverse solutions. In contrast, intensification means focusing on the search in a local region by exploiting the information of the current good solution that is found in this region. For each swarm intelligence algorithm, the exploration capability should be implemented first to search the whole space globally, while the exploitation capability should be considered later by enhancing the solution's quantity in the local search process [16]. Selecting the best solution ensures that the solutions will converge to optimality. The diversification via randomization avoids the solutions being trapped at local optima and, at the same time, increases the diversity of the solutions. A good combination of diversification and intensification ensures global optimality [2].

Several modifications have been proposed to enhance the BAT algorithm's performance since it needs a better balance between exploration and exploitation, so it sometimes fails to find the global optimum and easily gets stuck into the local optima [17][18]. This section will briefly describe the modifications and improvements of the BAT algorithm. Table 1 summarizes the main advances aspects regarding exploration, exploitation, and improving the initial population.

Selim Yılmaz et al. enhanced the bat algorithm's local search (exploitation) and global search (exploration) characteristics through two modifications using Inertia weight and modifying the population distribution. The BAT algorithm is hybridized with the invasive weed optimization algorithm [16].

Zaharuddeen Haruna et al. proposed a modified BAT algorithm using Elite opposition–based learning (EOBL) to enhance the diversification of the solution search space and modify the inertia weight to improve its exploitation capability. The main objective of using EOBL is to utilize some elite individuals from the current population to generate a corresponding opposite population by using opposition-based learning and depending on the dynamic search boundary. By evaluating both generated populations (the current population and the opposite population), promising regions which may contain the global optimum will be better reached. This will increase the diversification of the algorithm and prevent premature convergence. Using the number of individuals in the population, the number of iterations, and the current iteration, a weight is generated to utilize with the local search to improve the exploitation, too [17].

Xian Shan et al. proposed a modified search equation with more useful information from the search experiences to generate a candidate solution and incorporate Lévy Flight random walk with the algorithm to avoid being trapped into local optima by improving the exploitation. Furthermore, opposition-based learning is embedded in the algorithm to enhance diversity and convergence capability. The search space is explored using the best solution and its neighbors' solutions using echolocation to balance exploration and exploitation. An opposition-based learning population and the random population are generated and merged, and the individuals will be ranked to generate the initial population [18].

Another modification is the improvement proposed by Min-Rong Chen et al., who suggested an improvement using the Extremal Optimization (EO) algorithm to increase the BAT algorithm's exploitation. The improved update strategy is proposed to obtain the solutions generated from the randomly selected bats to enhance the global search capability (exploration ability) by reducing the dependence on the optimal solution. Besides these improvements, Boltzmann selection and a monitor mechanism are employed to make a balance between exploration and exploitation abilities [19].

S. Yılmaz et al. improved the exploration mechanism of the BAT algorithm by modifying the equation of pulse emission rate and loudness of bats. With the original BAT algorithm, each bat has only one pulse emission rate and loudness. That is, each solution satisfies the condition (rand > ri) will search around the best solution with all dimensions. In the proposed modified BAT, the loudness and pulse rate, which act as a balance, are equalized to the number of problem dimensions, where each dimension j of solution i, which satisfies the condition (randj > rij), will search around the dimension j of the best solution and the rest dimensions of solution i keep on seeking the search space [20].

Daranat Tansui et al. enhanced the BAT algorithm's exploitation power by introducing two new random walk processes that made its local search more thorough. The decision on what random walk to use depends on the generated random number. The authors also improved the exploration power by introducing inertia weight to intensify its global search near the end of the optimization process [21].

Xiaowei Wang et al. proposed an improved BAT algorithm called an Adaptive Bat Algorithm (ABA). Each bat has the ability to adjust its flight speed and direction dynamically and adaptively while searching for food. It uses the hunting approach of combining random search with shrinking search to provide better global convergence property and effectively avoid premature convergence [22].

Jianqiang Huang and Yan Ma proposed a novel BAT algorithm based on an integration strategy with the aim of enhancing the Algorithm's global search ability. The proposed bat disturbs the local optimum through a linear combination of Gaussian functions with different variances to avoid being stuck in the local optima. An adaptive weight is used with the velocity equation to balance the exploration and exploitation [23].

Sha-Sha Guo al. proposed an improved BAT algorithm based on a chaotic map and the algorithm of Levy flight search strategy and contraction factor to improve the search performance and the BAT algorithm's convergence speed and optimization precision [24].

A. Rezaee Jordehi proposed a chaotic-based BAT swarm optimization algorithm to alleviate the premature convergence problem of the original BAT algorithm. The authors use the chaotic map function in the loudness updating by multiplying a linearly decreasing function by the chaotic map function [25].

Table 1: A summary of the main improvements of the related works

Authors	Ref.	Initial population	Exploration	Exploitation
Selim Yılmaz et al.	[16]		✓	✓
Zaharuddeen Haruna et al.	[17]	✓	✓	✓
Xian Shan et al.	[18]	✓	✓	✓
Min-Rong Chen et al.	[19]		✓	✓
S. Yılmaz et al.	[20]		✓	
Daranat Tansui et al.	[21]		✓	✓
Xiaowei Wang et al.	[22]			✓
Jianqiang Huang and Yan Ma	[23]		✓	✓
Sha-Sha Guo et al.	[24]	✓	✓	✓
A. Rezaee Jordehi	[25]		✓	✓

3 Original BAT Algorithm

Bats can forage and can accurately avoid obstacles through their echolocation ability. The BAT algorithm can be explained by doing three phases, as shown in Figure 1. These phases are the initialization of the bats, the movement of the bats, and the update of the loudness and pulse emission rate. In detail, these three phases are as the following steps [15]:

Step 1: Initializing bat position, velocity, frequency, pulse rate, and loudness:

The algorithm starts by randomly initializing the population of bats within the search space, where each bat represents a potential solution to the optimization problem. The position and velocity of each bat are randomly generated within the search space. Then, the bats are sorted based on their fitness values, and the best bat in the population is identified as the global best solution.

The generation counter is initialized to 1; the positions of each D dimension of every bat in the population are initialized randomly to take a value within the available value range for each dimension as described in Equation (1) [15], along with their velocity v_i; frequency f_i, loudness A_0, and pulse rate r_i.

$$x_{ij} = x_{min} + Rand(0,1) * (x_{max} - x_{min}) \quad \dots \text{Eq. (1)}$$

Where (i = 1, 2, …., n) (j = 1, 2,…., D) denotes n bats in the population, each with D dimensions, each dimension within the range of the available boundaries x_{min} and x_{max}.

The initial loudness A^0 is randomly generated within the range of [1,2], while the initial pulse rate is generated randomly within the range [0,1].

Step 2: Adjusting the frequency, updating velocities and positions of the candidate solutions to generate new solutions:

Each bat searches for the optimal solution by using echolocation. The bat's position is updated based on its current position, velocity, and random walk component. The frequency of emitting pulses, the loudness of the emitted pulse, and the pulse emission rate are the three primary parameters that control the bat's position update. The frequency of emitting pulses determines the step size of the bat's motion, while the loudness of the emitted pulse corresponds to the magnitude of the step. The pulse emission rate controls the exploration-exploitation balance of the algorithm, where a high rate leads to more exploration and a low rate leads to more exploitation.

Each bat is assigned a frequency and loudness value that controls its movement. The loudness decreases over time, and the frequency increases, allowing the bats to explore the search space effectively.

The frequency is adjusted, the velocity is updated, and a new solution is generated as described in Equations (2), (3), and (4), respectively [15].

Algorithm1 Original BAT Algorithm

Begin
Objective function f(x), x = (x1, ..., xd)^T
Step 1: Initialization.
- Set the generation counter t = 1;
- Initialize the positions of the population x_i (i = 1, 2, ..., n) Eq. (1), and their velocities v_i
- Initialize frequency f_i, loudness *Ap*, and pulse rate *ri*

Step 2: **While** the termination criteria are not satisfied or *t* < Maximum Generation **do**
- Adjusting the frequency updating velocities and positions of the candidate solutions to generate new solutions Eq. [(2) – (4)],

- **if** (random number > *ri*), **then**
  - Select a solution among the better solutions;
  - Generate a local solution around the selected solution as in Eq. (5),

  end if
- Generate a new solution by flying randomly
-
- i**f** (random number < *Ai* & *f(xi)* < *f(x_)*), **then**
  - Accept the new solution
  - Increase *ri* and reduce *Ai* according to Eq. (6) and Eq. (7),

  end if
- Rank the bats and find the current best x*
- *t* = *t*+ 1;

Step 3: **end while**
Step 4: Post-processing the results
End

Figure 1. The pseudo code of the original BAT Algorithm [15]

$$f_i = f_{min} + (f_{max} - f_{min})\,\beta \quad \ldots \text{Eq. (2)}$$

$$V_i^t = V_i^{t-1} + \left(x_i^t - x_*\right) f_i \quad \ldots \text{Eq. (3)}$$

$$x_i^t = x_i^{t-1} + v_i^t \quad \ldots \text{Eq. (4)}$$

Where f_i is the pulse frequency that affects the i_{th} bat's velocity, f_{max} and f_{min} are the maximum and minimum available values of f_i. β represent a random number within the range [0,1]. x_* is the best position found within the population.

Step 3: Applying local search

Some bats perform a local search by randomly adjusting their positions to efficiently explore the search space using the random walk represented in Equation (5) [15].

$$x_{new}^i = x_{old}^i + \varepsilon\, A^t \quad \ldots \text{Eq. (5)}$$

After generating a new solution and if the ith bat's pulse emission rate is less than a randomly generated value, local search using random walk is used to create a solution using one solution x_{old}^i among the current population and generating new one x_{new}^i. A^t is the average loudness of all the bats, while ε is a random variable.

Step 4: Accepting the solution and updating the loudness and the pulse rate:

The loudness and frequency of each bat's echolocation pulse are updated based on the quality of the solution found in the previous iteration. If the generated random variable is greater than the Loudness and the fitness value of the current solution is better than the previous one, then accept the solution and increase the pulse rate and decrease the loudness as described in Equations (6) and (7), respectively [15].

$$A_i^{t+1} = \alpha A_i^t \quad \ldots \text{Eq. (6)}$$

$$r_i^t = r_i^o\,(1 - e^{\gamma t}) \quad \ldots \text{Eq. (7)}$$

Where α and γ are two constants.

Step 5: Update:

The best solution found by the bats is updated. The algorithm iteratively updates the position and velocity of each bat until the stopping criteria are met. The algorithm's stopping criteria can be defined based on the number of iterations or the convergence of the fitness values of the bats, so Steps 2-4 will be repeated until a termination criterion is met.

The standard parameters' minimum and maximum values used in the BAT algorithm are shown in Table 2 below.

Table 2: Standard values of BAT algorithm's parameters [15]

Parameter	Min value	Max value
Frequency	0	100 (depending on the domain size of the problem of interest)
Velocity	0 or random value between [0, v_max]	v_max depends on the problem being solved
Position	Min and Max values depend on the domain size of the problem of interest.	
Pulse rate	0	1
Loudness (2 examples)	1	100 (it is the initial value)
	0	1 (it is the initial value)

4 The Proposed Enhancements

This section describes the proposed improved BAT Algorithm based on the clustering approach. Generally, clustering is an unsupervised learning task that aims to find distinct groups in data called clusters. These clusters are more similar than others [26][27]. The main clustering algorithms are hierarchical, partitioning, and density-based clustering [28]. Density-based clustering is one of the prominent paradigms for clustering large data sets [27], and it is the clustering type used to improve the BAT algorithm.

Ester et al. [29] first introduced the DBSCAN algorithm, which depends on a density-based notion of clusters. Clusters are identified by looking at the density of points. Regions with a high density of points depict clusters, whereas regions with a low density indicate clusters of noise or outliers. This algorithm is particularly suited to deal with large, noisy datasets and can identify clusters of different sizes and shapes [28].

The original BAT algorithm has good exploitation but poor exploration [21], so it can easily get trapped at a local minimum of most multimodal test functions. A modification of the original BAT algorithm is applied to overcome this problem. The proposed improved algorithm is shown in the flowchart of Figure 2. The modification is about the following phases:

- Improving population initialization to improve the global search ability.
- Balancing the algorithm's local and global search abilities.
- Overcome stagnation.

The following subsections describe these phases in detail.

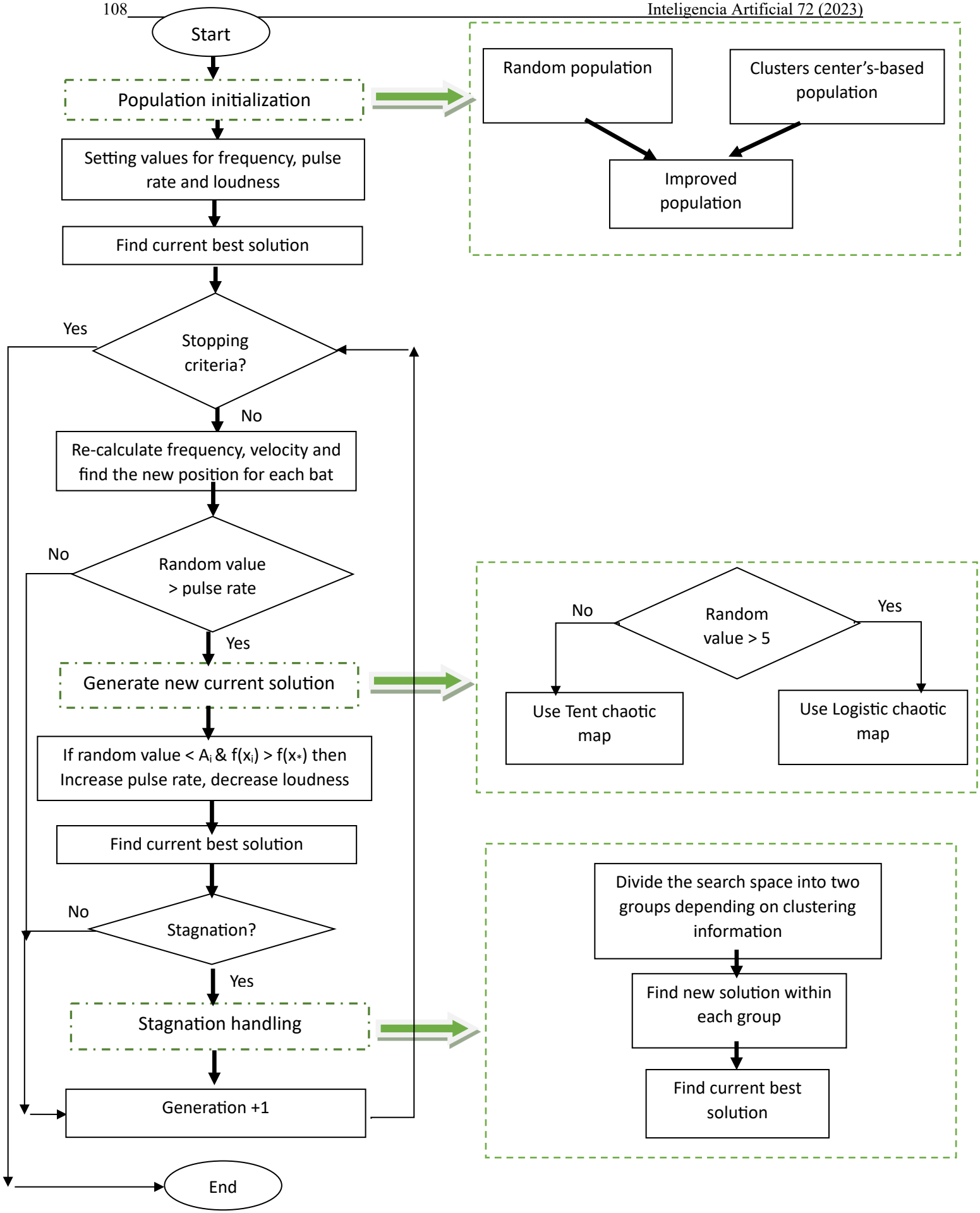


Figure 2. Improved BAT algorithm

4.1 Improving the population's initialization

Metaheuristic algorithms work by initializing the population of candidate solutions, then iteratively updating these candidate solutions until convergence occurs or the stopping criteria are met. The initial population's quality mainly affects the algorithm's convergence and has a role in the quality of the produced solution [30].

A method to improve the population depends on the information obtained from density-based clustering is proposed. Density-based clustering is used to produce several clusters. These clusters are almost the most important features. For example, if the clustering is applied to a dataset of base stations for a large smart city, then, depending on the workload and users' requests to these base stations, the base stations will be grouped depending on the workload heaviness grades. So, each cluster will represent a degree of importance. For each cluster, the center will be computed. Four regions are calculated for each cluster depending on the closeness to the centers; the two closest areas to the center will get a higher probability of producing individuals to the population.

Algorithm 2 Proposed Cluster-based method for Bat Algorithm population initialization

Begin

Input: the total number of individuals in the population N_P

Output: initial population "*Population"*

1: Generate a uniformly random population "1^{st} *Population*"

2: Generate density-based k clusters, then get the (k centers) $(c_1, c_2, \ldots\ldots, c_k)$, (k lower bounds) $(a_1, a_2, \ldots\ldots, a_k)$, and (k upper bounds) $(b_1, b_2, \ldots\ldots, b_k)$

3: Weight each cluster depending on the number of points within each.

$$w_i = p_i / sum$$

w_i is the weight of cluster i, p_i is the no. of points within cluster i, and the *sum* is the total number of points within all clusters

4: Depending on the cluster's weight, get the number of population's individuals within each cluster N_P_i

$$N\_P_i = w_i * N\_P$$

5: For each cluster where cluster center $c_i \in (c_1, c_2, \ldots\ldots, c_k)$, lower bound $a_i \in (a_1, a_2, \ldots\ldots, a_k)$, and upper bound $b_i \in (b_1, b_2, \ldots\ldots, b_k)$ do:

generate four regions' boundaries (R1, R2, R3, R4) from the available variables range

$R1:(a_i , a_i + (c_i - a_i) / 2)$

$R2: (a_i + (c_i - a_i) / 2 , c_i)$

$R3: (c_i , c_i + (b_i - c_i) / 2)$

$R4: (c_i + (b_i - c_i) / 2 , b_i)$

generate a $2*N\_P_i/3$ individual for the two closest regions to the center, and $N\_P_i/3$ individuals for the two other regions and insert all generated individuals into the "2^{nd} *Population"*

end for

6: Evaluate both populations (1^{st} *Population and* 2^{nd} *Population*) to get the final initial population (*Population*) by selecting the best individuals from both populations

End

Figure 3. The pseudo code of the population initialization

A uniformly random population is initialized and called the 1st population, and a cluster-center-based population is initialized too and called the 2nd population. The final improved initial population is generated by taking the fittest individuals from the two populations. Figure 3 shows a pseudo-code for the population initialization.

After clustering the search space, the center and upper and lower boundaries are extracted for each cluster. Equation (8) represents the calculation of the cluster's center.

$$\text{center} = \text{lower bound} + (\text{upper bound} - \text{lower bound}) / 2 \quad \ldots \text{Eq. (8)}$$

From each cluster, the range of four regions will be computed as the following Equations (9), (10), (11), and (12), where 'a' is the cluster's lower bound, 'b' is the cluster's upper bound:

$$\text{Region1 (R1) domain: } [a, a+(\text{center}-a)/2] \quad \ldots \text{Eq. (9)}$$

$$\text{Region2 (R2) domain: } [a+(\text{center}-a)/2, \text{center}] \quad \ldots \text{Eq. (10)}$$

$$\text{Region3 (R3) domain: } [\text{center}, \text{center}+(b-\text{center})/2] \quad \ldots \text{Eq. (11)}$$

$$\text{Region4 (R4) domain: } [\text{center}+(b-\text{center})/2, b] \quad \ldots \text{Eq. (12)}$$

For each cluster, and depending on the number of points within, the ratio of the population's individuals that are produced from that cluster is computed. The larger the cluster, the higher the participation ratio in the population initialization.

As shown in Figure 4, four regions are computed for each cluster. The two closest regions to the cluster's center will produce a higher ratio of individuals than the other two regions since the cluster's center will have the most important and valuable concentrated points than the two farthest isolated regions. So, the regions R2 and R3 are more important than R1 and R4.

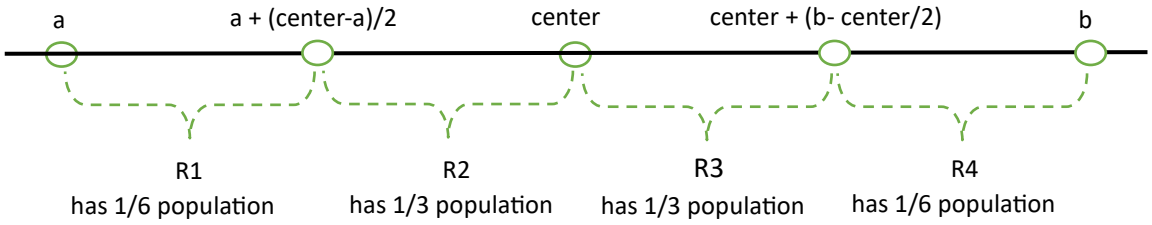


Figure 4. The four regions of each cluster

Generating the cluster's center-based population will improve the initial population used by the algorithm. As a result, the convergence will be enhanced, and the final solution will be improved too.

When using only the randomly generated population, the individuals may be far from the best optimal solution, so that the convergence will worsen.

4.2 Balancing the algorithm's local and global search

As mentioned previously, the BAT algorithm needs better exploration. To find other good solutions around the current global solution and explore the search space more thoroughly, the BAT algorithm uses a random walk, which is a random process consisting of taking a series of consecutive random steps. The following mathematical Equation (13) [31] represents the random walk.

$$X_{t+1} = X_t + \omega_t \quad \ldots \text{Eq. (13)}$$

X_t is the current location or state at t, and ω_t is a step or random variable with a known distribution [31]. The BAT algorithm uses ω_t within step size +1 or -1.

In the proposed improved BAT algorithm, a Logistic chaotic map is used instead of the random walk, and the algorithm's behavior is improved.

Chaotic functions are used to balance the exploration and the exploitation of the algorithm due to its non-repetitive nature and, as a result, will tackle premature convergence problems.

The random-based optimization algorithms can use a chaotic dynamic instead of randomness. Optimization algorithms that use chaotic (chaos optimization algorithms (COAs)) can easily escape from local minima than other stochastic optimization algorithms. Stochastic optimization algorithms have to accept some bad solution to escape from the local minima, while COA searches on the regularity of chaotic motion to escape from local minima [32].

The main property of chaos is "ergodicity," which means uniformly and randomly visiting all the parts of the space that the system moves in [25][32]. Some one-dimensional maps or chaotic variable generators are Circle map, Sine map, Gaussian map, Bernoulli shift map, Chebyshev map, Logistic and Tent maps [32]. Logistic and Tent maps are used in the improved BAT algorithm:

- ***Logistic map:*** a one-dimensional polynomial map that leads to chaotic dynamics. The following Equation (14) defines the logistic map [32].

$$X_{n+1} = \lambda X_n (1 - X_n) \quad \dots \text{Eq. (14)}$$

  Where $0 < \lambda \leq 4$

- ***Tent map:*** it is also a one-dimensional polynomial map; it is like the logistic map and displays some particular chaotic effects. It is expressed as the following Equation (15) [32]

$$X_{n+1} = \begin{cases} \mu X_n & X_n < 0.5 \\ \mu (1 - X_n) & X_n \geq 0.5 \end{cases} \quad \dots \text{Eq. (15)}$$

  Where $\mu = 2$

4.3 Stagnation handling

Like many other metaheuristic algorithms (PSO, for example), the BAT algorithm suffers from stagnation during the search process. To enhance the search process of the BAT algorithm when stagnation occurs, the search space will be divided into two parts depending on the produced clusters. The two closest clusters (depending on the similarity measurement) will be the first part; the other clusters will be the other.

A temporal new population is initialized for each part following the same improved population initialization process. After that, the best part's solution will be derived by picking the fittest one. By evaluating the produced best solutions from both parts, the best one will be set as the current best solution, and the process will be continued.

In this case, the search process will become diversified, so stagnation will likely occur less.

5 Experimental Results

The goal of the experiments is to evaluate the performance of the improved BAT algorithm and compare it with the original one.

Both original and improved BAT algorithms are implemented using C#, and the experiments are conducted on a computer with 11th Gen Intel® Core™ i9-11900H CPU @2.50GHz and 16.0 GB RAM in Windows 10 Pro environment.

5.1 Benchmark test functions

To evaluate the performance of the new or improved metaheuristic algorithms, benchmark numerical test functions are used. Metaheuristic algorithms with good performance on these functions can solve real-life hard optimization problems. Benchmark test functions are fundamentally optimization problems presented as mathematical numerical functions [33][34][35]. These functions are optimized with a set of best suitable parameter values that help achieve the best solution. There are large amount of sub-optimal solutions, and the best solution is hidden within. These solutions are spread all over the problem landscape with various numbers and types of hills and valleys. Metaheuristic algorithms tend to find the best solution as quickly as possible, but they have no guarantee of obtaining it. The performance efficiency of any metaheuristic algorithm is measured by its global and local search and convergence ability. The algorithms with good global search are hard to be stuck in

local minima or maxima locations. At the same time, it is hard for any metaheuristics with efficient convergence ability to miss any best solution within the neighborhoods [35].

The benchmark test functions have different properties. Modality, dimensionality, valleys, Basins, and separability are the main properties [36]. Multimodal functions are functions with more than one local optimum. They are used to test the algorithm's ability to escape from any local minimum. These functions are among the most difficult problems for many algorithms. To search the function landscape effectively, the algorithm's exploration process needs to be better designed.

Since the function's flatness feature does not provide any information to the algorithm for directing the search process toward the minima, functions with flat surfaces are complex [36]. The multimodal test function tests the algorithm's local and global search ability and convergence speed stability. The unimodal test function tests the algorithm's exploitation capability and convergence.

5.2 Used benchmark test functions

The proposed improved BAT algorithm is evaluated against the original algorithm using ten benchmarked test functions. Since the proposed improvement of the BAT algorithm mainly focuses on enhancing the algorithm's diversification, multimodal test functions are used for the evaluation process. The following multimodal test functions are used:

Ackley: is one of the most commonly used test functions to evaluate the metaheuristic algorithm. It has one global optimal solution within a numerous local minimum. This global optimal solution is found in the middle within a deep narrow basin. The value of the best solution is 0, and it is found at f(x*) = [0, 0, …, 0] within the [-32, 32] domain. The function is mathematically written as Equation (16) [35].

$$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x^2}\right) - \exp\left(\frac{1}{D}\sum_{i-1}^{D}\cos 2\pi\, x_i\right) \quad \text{… Eq. (16)}$$

***Rastrigin Function*:** A function that presents numerous local minima locations. The function has only one global best solution 0 that is found at f(x*) = [0, 0, …, 0] within the domain of [-5.12, 5.12]. The function is mathematically written as Equation (17) [35].

$$f(x) = \left|\sum_{i=1}^{D}(x_i^2 - 10\,\cos(2\pi x_i) + 10)\right| \quad \text{… Eq. (17)}$$

***Griewank Function*:** A function with widespread suboptimal solutions spread throughout the search environment, with only one global optimum solution. The value of the global best solution is 0, which is found at f(x*) = [0, 0, …, 0] within the domain [-600, 600]. The function is mathematically written as Equation (18)[35] .

$$f(x) = \frac{1}{4000}\sum_{i=1}^{D}x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad \text{… Eq. (18)}$$

***Sphere Function*:** A continuous, differentiable, separable, scalable, multimodal function. It is subject to $0 \leq$ xi ≤ 10. The global minimum solution is located at f(x∗) = f (0, · · ·, 0), f(x∗) = 0. It is mathematically written as Equation (19) [36].

$$f(x) = \sum_{i=1}^{D}x_i^2 \quad \text{… Eq. (19)}$$

Alpine Function*:* A multimodal function with global minima value 0 that is found at f(x*) = [0, 0, …, 0] within the domain [-10, 10]. The function is mathematically expressed as Equation (20) [35].

$$f(x) = \sum_{i=1}^{D-1}|x_i\,\sin(x_i) + 0.1\,x_i| \quad \text{… Eq. (20)}$$

***Himmelblau*:** A function that is solved with continuous values in the domain [-6,6]. The value of the best solution is 0, and it can be found at four locations: f(x*) = [3.2, 2.0], f(x*) = [-2.805118, 3.131312], f(x*) = [-3.779310, -3.283186], and f(x*) = [3.584428, -1.848126] in 2-dimensional space. The function is defined as Equation (21) [35].

$$f(x) = \frac{1}{D}\sum_{i=1}^{D}\left(x_i^4 - 16x_i^2 + 5x_i\right) \quad \text{… Eq. (21)}$$

***Schwefel**:* It is a difficult-to-solve function containing several local minima locations. The value of the global minima is 0, and it is located at f(x*) = [1, 1, …, 1] in the domain of [-500, 500], This function is expressed Mathematically as Equation (22) [35].

$$f(x) = \sum_{i=1}^{D} -x_i \sin\left(\sqrt{|x_i|}\right) \qquad \text{… Eq. (22)}$$

Besides evaluating the capability of diversification (finding a global solution), the algorithm must be checked to determine whether it has good exploitation (local search ability and good convergence). Unimodal test functions are used for this purpose. The following unimodal test functions are used:

SumSquare: A function that is also known as Axis Parallel Hyper-Ellipsoid function. It maintains no local optima but one global optima f(x*) = [0, 0, …, 0 within the range of [-10, 10] of continuous values. The function is written as Equation (23) [35].

$$f(x) = \sum_{i=1}^{D} ix_i^2 \qquad \text{… Eq. (23)}$$

***Sphere Function*:** An easy-to-solve unimodal and continuous function. It is evaluated using a domain [-5.12, 5.12], its minimum solution value is 0, and it is located at f(x*) = [0, 0, …, 0]. The function is mathematically written as Equation (24) [35].

$$f(x) = \sum_{i=1}^{D} x_i^2 \qquad \text{… Eq. (24)}$$

***Step Function*:** A flat surface unimodal function that is often considered difficult to solve as no proper direction towards the globally optimum location is easily found. The value of the global minimum solution is 0, and it is located at f(x*) = [0, 0, …, 0] within the [-100, 100] range. It is mathematically represented as Equation (25) [35].

$$f(x) = \sum_{i=1}^{D} (x_i + 0.5)^2 \qquad \text{… Eq. (25)}$$

5.3 Testing parameters

Many testing parameters must be specified as follows:

1. *Population size:* the total number of individuals in the population; for example, populations of 100, 200, 300, and 1000 individuals are used in the experiments.
2. *Dimensions:* the total number of design variables; for example, 10, 20, and 30 dimensions. In the experiments, the dimension used is 30 (30 is the average dimension with almost all benchmark test functions).
3. *Total runs:* represent the number of times the algorithm runs, for example, 6, 8, or 10 independent runs for each algorithm. Within the experiments, the results are documented within 30 runs.
4. *Maximum iterations:* the total number of iterations, for example, 100.
5. Convergence speed: the computation time that is taken by the algorithm.
6. Statistical results: some information is calculated to evaluate the performance of both original and modified BAT algorithms like:
    - *Best value:* the minimum value among all obtained values from all runs.
    - *Worst value:* the maximum value among all obtained values from all runs.
    - *Mean value:* the average value obtained from all runs.
    - *Standard deviation (STD):* to measure how the results spread out from its mean.

The experiments are run over ten benchmark test functions represented in Section 5.2, where each function is implemented 30 times for both original and the improved BAT algorithms. All the results are documented; the statistical results (best, worst, mean, and standard deviation values) are computed and shown in Tables 2, 3, 4, and 5 for different population sizes, such as 100, 200, 300, and 400 individuals, respectively.

Table 3: Results of the comparison between the original and improved BAT algorithms with (population size = 100, dimension = 30)

	Original BAT Algorithm			Modified BAT Algorithm		
Test Function	Best Value	Worst Value	Mean	Best Value	Worst Value	Mean
Ackley	0.966	1.241	1.071± 0.060	0.780	0.988	0.882 ± 0.049
Rastrigin	367.871	460.830	403.134 ± 21.025	0	0	0
Sphere Multimodal	4E-121	0.0246	0.001 ± 0.006	0	0	0
Schwefel	4.684E-14	11.831	3.1959297 ± 5.008	0	5.68E-14	1.14E-14 ± 2.94E-14
Alpine	49.445	67.874	57.462 ± 4.349	2.7E-194	1.1E-176	6.4E-178
Himmelblau	6.155E-195	3.361E-104	1.120E-105 ± .032E-105	0.012	9.253	1.700 ± 2.636
Griewank	440.483	655.143	529.789 ± 56.617	108.101	158.8376	133.8413± 14.204
SumSquare	5217.17	9560.65	7423.77 ± 1071.4	0	0	0
Sphere Unimodal	125.835	190.252	154.885 ± 15.782	0	0	0
Step	48749.03	72786.59	57759.02 ± 5772.236	5884.260	17904.083	13147.230 ± 2999.498

Table 4: Results of the comparison between the original and improved BAT algorithms with (population size = 200, dimension = 30)

	Original BAT Algorithm			Modified BAT Algorithm		
Test Function	Best Value	Worst Value	Mean	Best Value	Worst Value	Mean
Ackley	0.923	1.183	1.074 ± 0.060	0.79482	0.976	0.892 ± 0.042
Rastrigin	338.404	437.766	404.402 ± 23.512	0	0	0
Sphere Multimodal	0	0	0	0	0	0
Schwefel	0	9.929	0.594 ± 2.315	0	5.684E-14	3.790E-15 ± 1.468E-14
Alpine	39.426	63.022	54.140 ± 4.935	8.9E-193	7.2E-179	4.9E-180
Himmelblau	0.030	6.266	1.473 ± 2.052	5.5E-193	3.21E-97	1.07E-98 ± 5.77E-98
Griewank	411.545	578.960	514.568 ± 46.258	94.1358	162.9374	131.514 ± 16.331
SumSquare	6042.286	8681.658	7536.947 ± 687.924	0	0	0
Sphere Unimodal	117.970	173.404	152.717 ± 15.185	0	0	0
Step	40642.83	69701.07	59886.71 ± 6889.308	5736.577	16852.91	13474.76 ± 2433.201

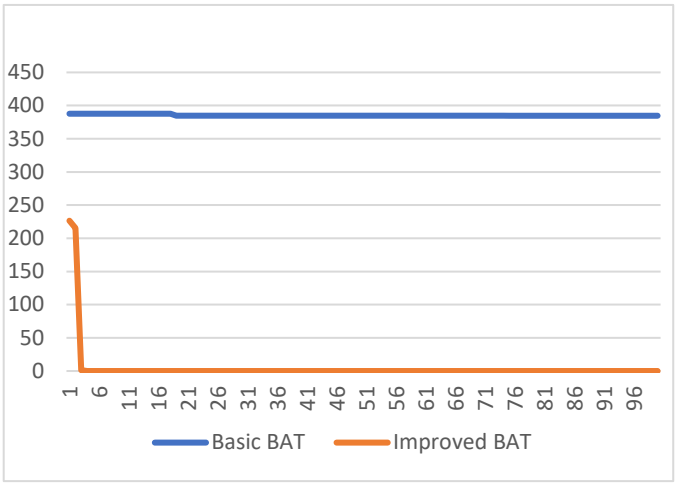
Table 5: Results of the comparison between the original and improved BAT algorithms with (population size = 300, dimension = 30)

	Original BAT Algorithm			Modified BAT Algorithm		
Test Function	Best Value	Worst Value	Mean	Best Value	Worst Value	Mean
Ackley	0.9439	1.126	1.059 ± 0.049	0.814	0.967	0.888 ± 0.042
Rastrigin	324.708	433.280	398.981 ± 21.847	0	0	0
Sphere Multimodal	0	0	0	0	0	0
Schwefel	0	0	0	0	0	0
Alpine	40.131	58.485	51.873 ± 4.530	2.1E-193	5.5E-178	1.9E-179
Himmelblau	0.005	3.482	0.896 ± 1.205	3.3099E-193	7.6512E-83	2.550E-84 ± 1.373E-83
Griewank	367.932	592.541	506.638 ± 43.759	104.0246	151.7316	128.4485 ± 12.01705
SumSquare	5351.231	7926.316	6962.806 ± 744.079	0	0	0
Sphere Unimodal	114.690	174.366	147.325 ± 15.717	0	0	0
Step	43421.28	64068.7	55745.03 ± 5739.469	6897.456	16386.44	12465.72 ± 2044.637

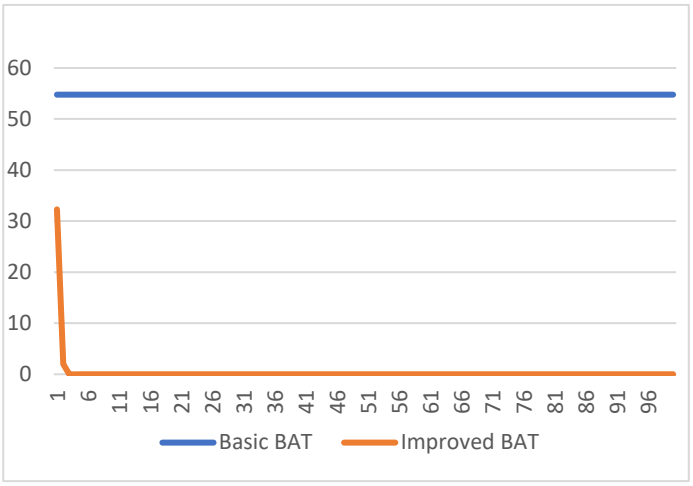
Table 6: Results of the comparison between the original and improved BAT algorithms with (population size = 1000, dimension = 30)

	Original BAT Algorithm			Modified BAT Algorithm		
Test Function	Best Value	Worst Value	Mean	Best Value	Worst Value	Mean
Ackley	0.934	1.090	1.018 ± 0.039	0.772946	0.885841	0.837585 ± 0.031687
Rastrigin	318.810	413.542	373.44 ± 23.477	0	0	0
Sphere Multimodal	0	0	0	0	0	0
Schwefel	0	0	0	0	0	0
Alpine	37.275	55.146	48.340 ± 4.624	0	2.65E-174	8.85E-176
Himmelblau	0.034	1.088	0.381 ± 0.480	0	7.525E-76	2.5.8E-77 ± 1.351E-76
Griewank	342.572	533.830	458.124 ± 38.226	88.541	139.992	116.454 ± 11.946
SumSquare	4721.069	8134.941	6357.135 ± 841.033	0	0	0
Sphere Unimodal	96.855	154.901	131.470 ± 12.208	0	0	0
Step	41465.39	58794.17	50709.8 ± 4928.865	5193.702	15241.46	11655.64 ± 2293.18

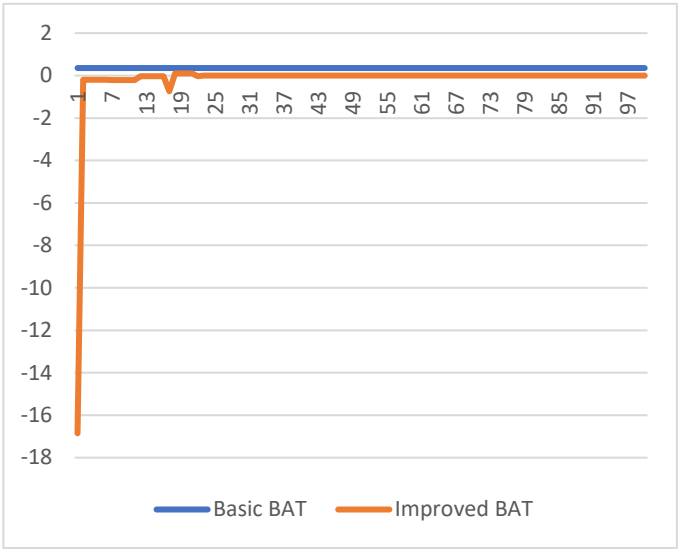
As shown from the previous results in Tables 2, 3, 4, and 5, the improved BAT algorithm outperforms the original BAT algorithm, especially as the population grows, and reaches much better values on all the ten benchmarks, especially for Rastrigin, Sphere (both multimodal and unimodal), Schwefel, Alpine, Himmelblau, and SumSquare functions where it reaches the optimal value (which is 0) compared to the original BAT algorithm. The original BAT gets the optimal value for Schwefel and the multimodal sphere functions only (0 value), and it is close to the optimal value in the Himmelblau function, while with the other functions, it produces worse values.



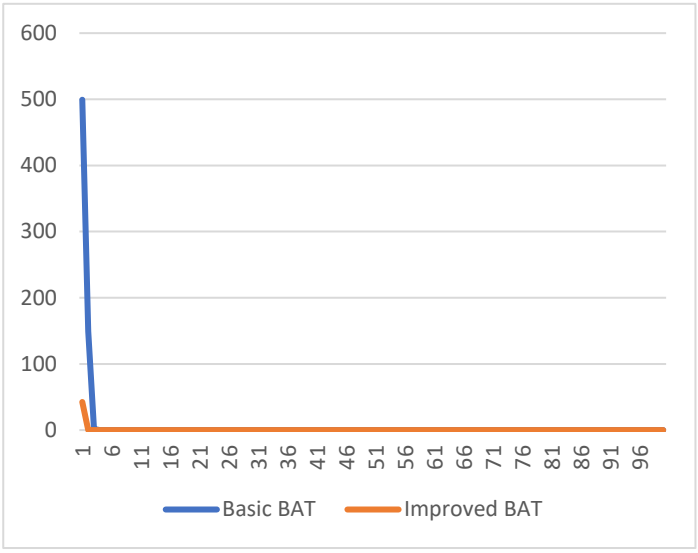
(a) Using the Rastrigin test function



(b) Using the Alpine test function

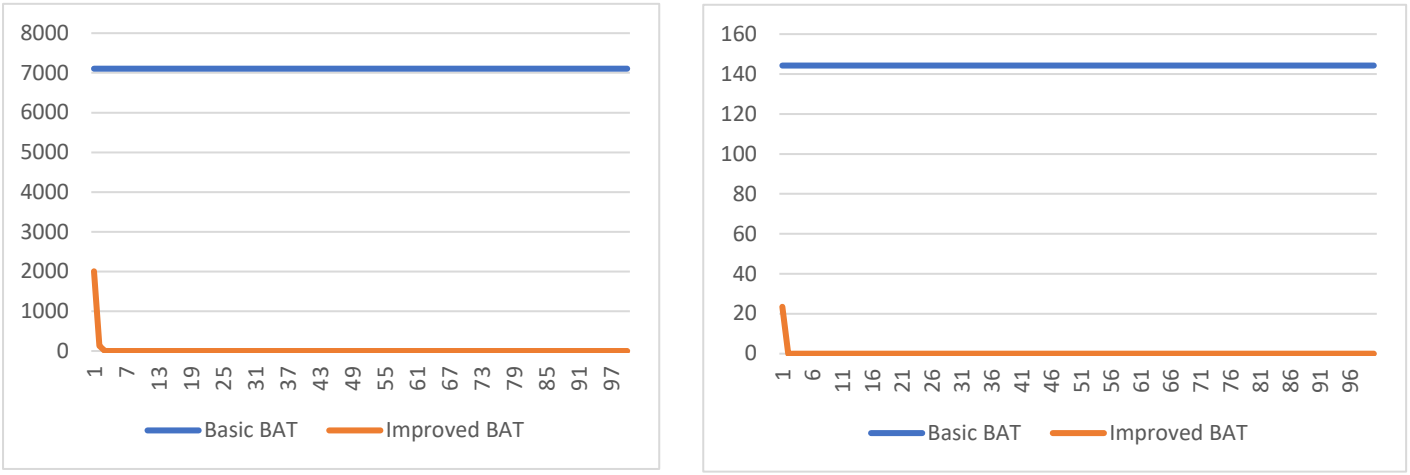


(c) Using the Himmelblau test function



(d) Using the Multimodal Sphere test function

Figure 5. Convergence curves using four multimodal benchmark test functions (a) Rastrigin, (b) Alpine, (c) Himmelblau, (d) Multimodal Sphere



(a) Using SumSquare test function **(b) Using Unimodal sphere test function**

Figure 6. Convergence curves using two unimodal benchmark test functions (a) SumSquare, (b) Unimodal Sphere

The improved algorithm has a great mean value near or equal to the optimal value, except for the Griewank and Step test functions. In contrast, the original BAT algorithm mean values are far from the optimal value, except for the Himmelblau and Ackley test functions, where the mean value is near the optimal one, and it reaches the optimal value for both Schwefel and the multimodal sphere functions.

Regarding the convergence speed, the convergence to the optimal value of the improved BAT algorithm is much faster and better than the original BAT algorithm as shown in Figures 5 and 6. As a general result, the performance of the enhanced BAT algorithm is significantly better than the original BAT algorithm.

All the previous results demonstrate the performance behavior of the improved BAT algorithm with all three improvements together (improved initial population, improved local search, and stagnation handling) against the original one.

An incremental analysis approach is made to diagnose which improvement has the bulk effect. This analysis is based on evaluating the performance of the algorithm over six cases, three cases of them are with only one improvement as follows:

Case 1: Partially Improved BAT algorithm WITH ONLY improved initial population.

Case 2: Partially Improved BAT algorithm WITH ONLY improved local search.

Case 3: Partially Improved BAT algorithm WITH ONLY stagnation handling.

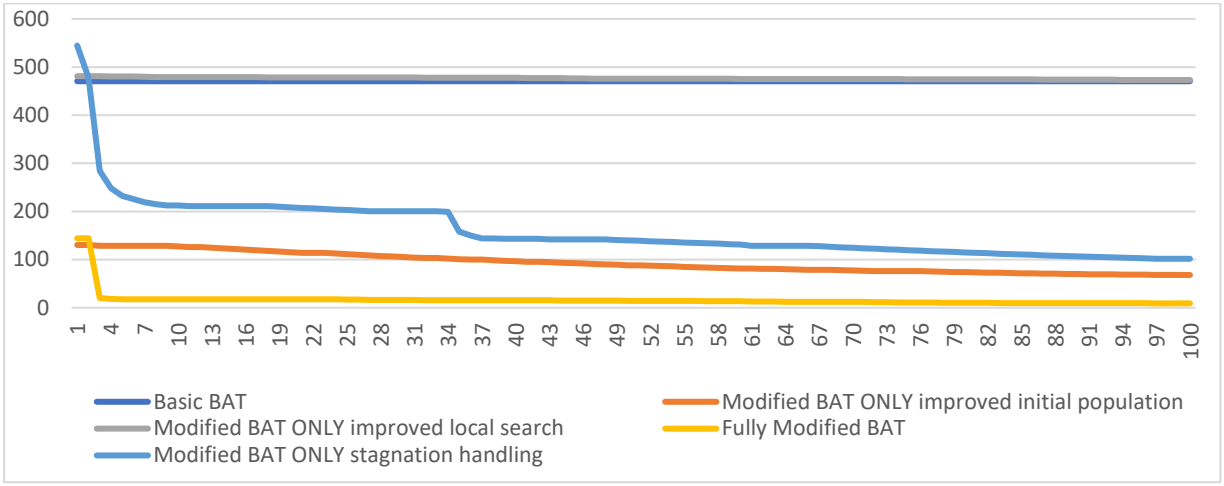
The other three cases are with two improvements as follows:

Case 4: Partially Improved BAT algorithm but (WITHOUT improved initial population)

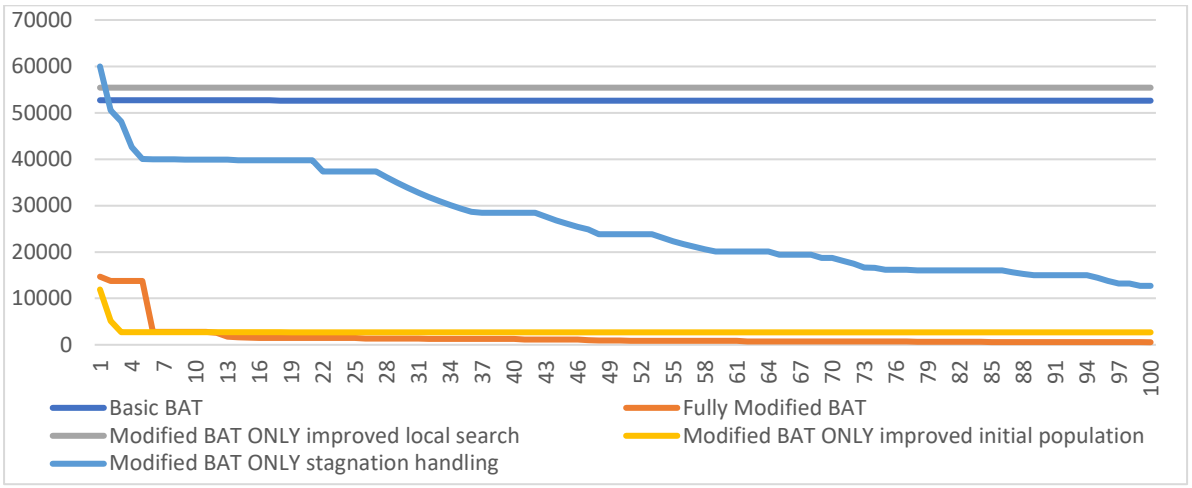
Case 5: Partially Improved BAT algorithm but (WITHOUT improved local search)

Case 6: Partially Improved BAT algorithm but (WITHOUT stagnation handling)

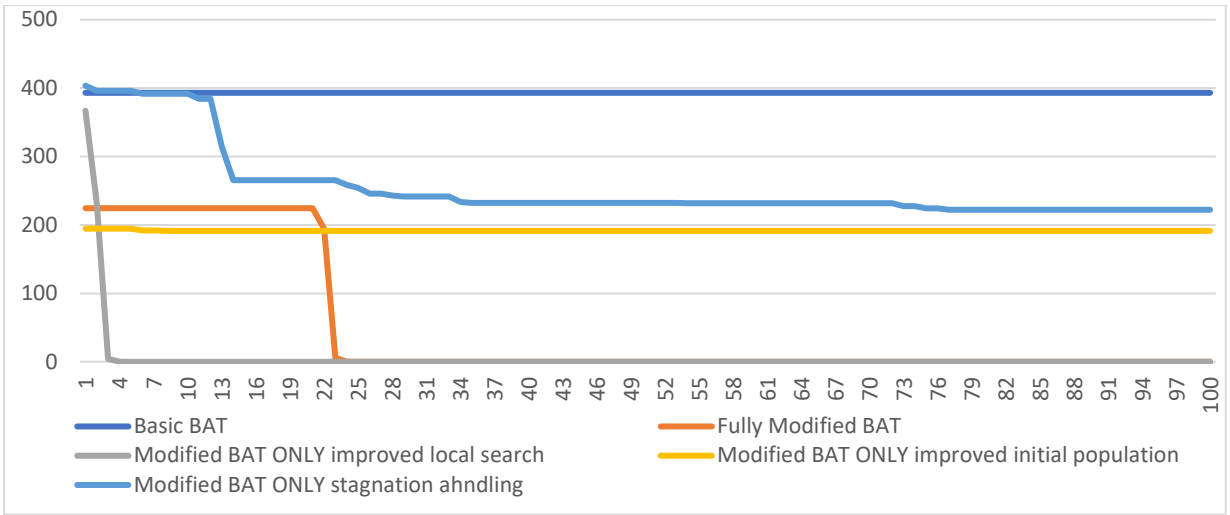
Figure 7 shows the performance of the partially improved BAT (first three cases) against the original BAT and fully improved BAT (with all three improvements together), while Figure 8 shows the performance of the partially improved BAT (second three cases) against the original BAT and fully improved BAT (with all three improvements together)



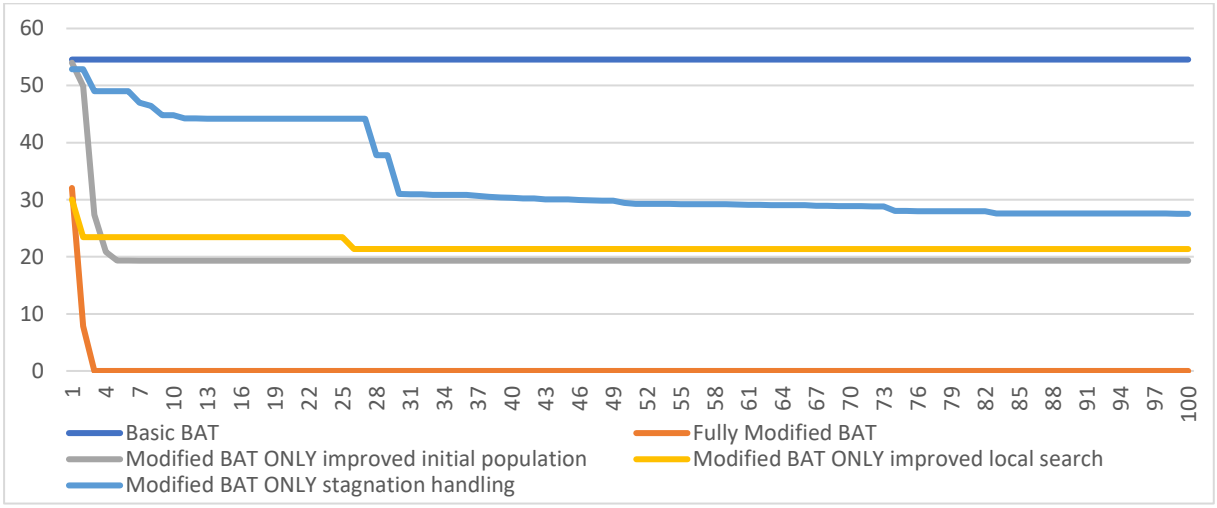
(a) Using Griewank multimodal benchmark test functions



(b) Using Step unimodal benchmark test functions

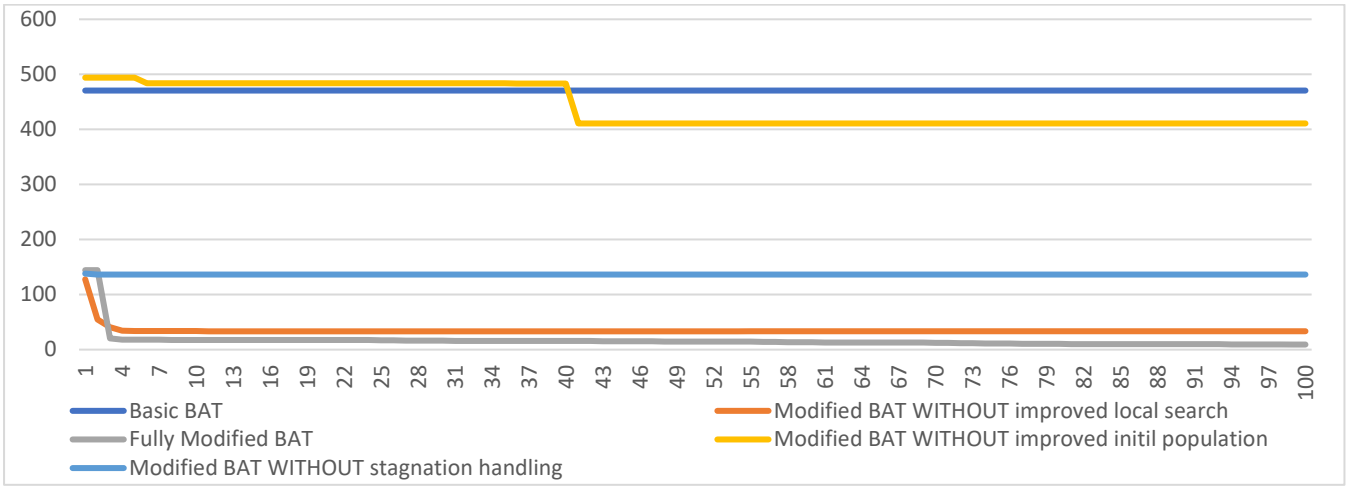


(c) Using Rastrigin multimodal benchmark test function

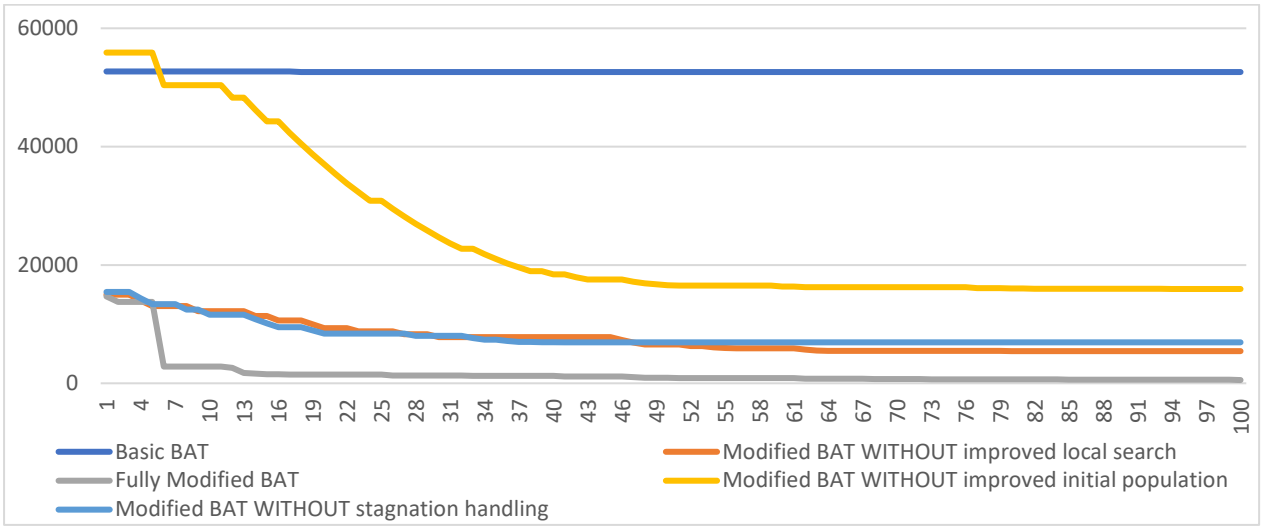


(d) Using Alpine multimodal benchmark test function

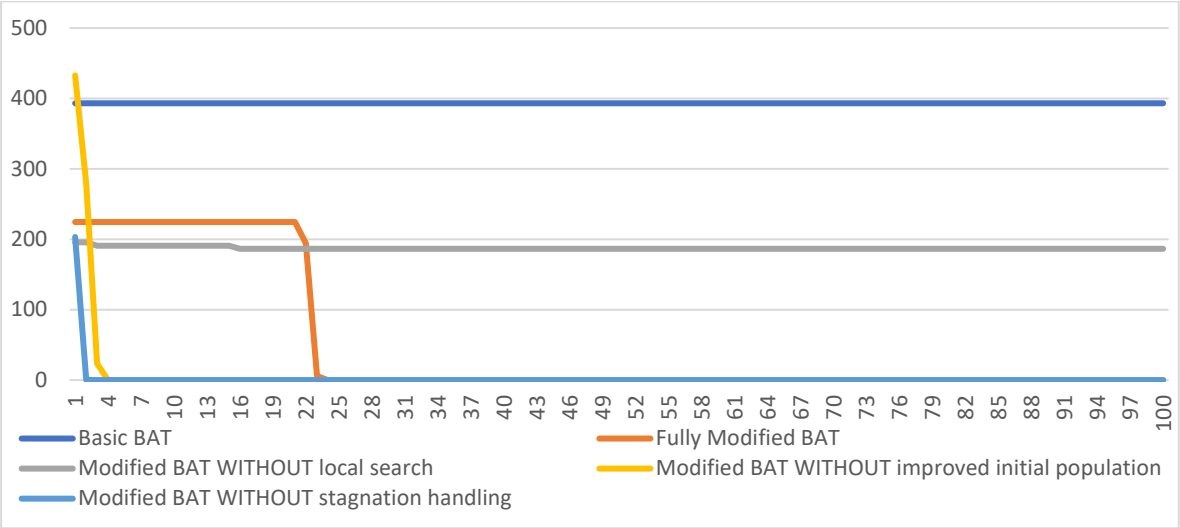
Figure 7. Convergence curves using four benchmark test functions over original BAT, fully improved BAT, and cases 1,2, and 3



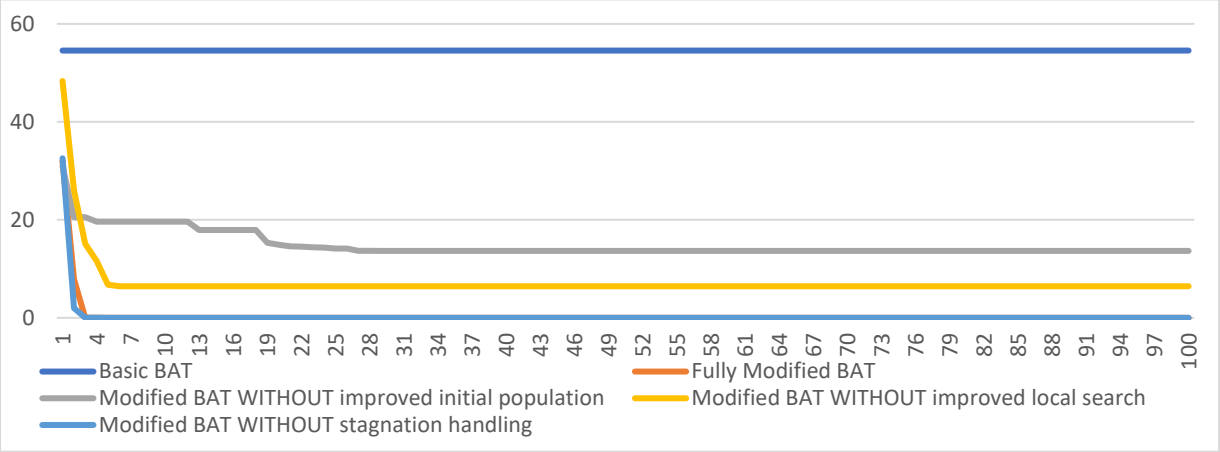
(a) Using Griewank multimodal benchmark test functions



(b) Using Step unimodal benchmark test functions



(c) Using Rastrigin multimodal benchmark test function



(d) Using Alpine multimodal benchmark test function

Figure 8. Convergence curves using four benchmark test functions over original BAT, fully improved BAT, and cases 4,5, and 6

As noticed clearly from Figure 7, the most significant role in the improvement is given to the population initialization phase, where the performance of the BAT algorithm with only an improved initial population is the first better performance after the fully improved BAT algorithm in three out of four test functions (Griewank, Step, and Alpine). The second significant role is equally due to improved local search and stagnation handling. In the Alpine test function, the performance of the BAT with only improved local search is the second better performance after Full improved BAT algorithm and BAT with only improved initial population. In the Rastrigin test function, the performance of the BAT with only improved local search is better than even the BAT algorithm with the improved initial population.
BAT algorithm with only stagnation handling has the second better performance after Full improved BAT algorithm and BAT with only improved initial population in two out of four test functions.

From another perception, the same thing is noticed in Figure 8, the improved BAT WITHOUT improved initial population has the worst performance after the Basic BAT algorithm in three out of four test functions (Griewank, Step, and Alpine), which means that the improved initial population has the most significant role in the improvement of BAT algorithm.

The convergence is improved in case there is an improvement in the initial population (in whatever case, with handling the stagnation or not, and with improving the local search or not).

The second significant role or effect is using chaotic maps to improve the local search (balancing the exploration and exploitation).

The improved BAT algorithm will be used in the Edge and Fog computing resource allocation problems, specifically regarding edge and fog nodes' placement issues.

Many methods and techniques are used for edge and fog nodes' placement problems, like Mathematical modeling, heuristic and metaheuristic algorithms, clustering techniques, and reinforcement learning approaches.

One prominent method is the use of a metaheuristic algorithm along with the clustering technique. The clustering technique is used to define regions with similar features, then use the improved BAT algorithm, which depends mainly on the produced clusters for finding the optimal number and distribution of edge nodes within each cluster. Depending on some objectives and constraints, the improved BAT will get a better placement of edge nodes than the benchmark placement techniques like Random, Top-K, and First-K.

6 Conclusions

This study proposes an enhanced BAT algorithm based on Density-based clustering and chaotic strategies, with three improvements to the original BAT algorithm to improve its exploration and exploitation abilities and enhance its performance.

The improvements are about the population's initialization with the aid of the clusters' centers values; using chaotic map behavior instead of the random walk to balance the local and global search abilities of the algorithm; and finally, the stagnation problem is handled by dividing the search space into two parts also depending on the produced clusters' information. When the search space has been clustered, this means grouping similar points within several clusters; hence, these clusters probably have the most important points.

To evaluate the effectiveness of the improved BAT algorithm, many experiments using ten benchmark test functions are conducted with different population sizes. The results of the simulation experiments show that the proposed enhanced BAT algorithm significantly improves local and global search ability, solution accuracy, and convergence speed compared with the original BAT algorithm.

Acknowledgements

Great thanks to some dear colleagues at the College of the Information Technology/ University of Babylon for their support and help.

References

[1] S. S. Rao, *Engineering optimization: Theory and practice, Fifth Edition*. John Wiley & Sons, Inc, 2020.

[2] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms Second Edition*. Luniver Press, United Kingdom, 2010.

[3] N. Kamel and T. El-Omari, "Sea Lion Optimization Algorithm for Solving the Maximum Flow Problem," *IJCSNS Int. J. Comput. Sci. Netw. Secur.*, vol. 20, no. 8, p. 30, 2020, doi: 10.22937/IJCSNS.2020.20.08.5.

[4] S. Annealing, "Simulated annealing: Theory and applications," *Math. Comput. Simul.*, vol. 30, no. 1–2, pp. 7–15, 1988, doi: 10.1016/0378-4754(88)90140-1.

[5] James Kennedy and Russell Eberhart, "Particle Swarm Optimisation," *Proc. ICNN'95 - Int. Conf. Neural Networks, Perth, WA, Aust.*, vol. 4, pp. 1942–1948, 1995, doi: 10.1109/ICNN.1995.488968.

[6] X. S. Yang, "Engineering Optimization: An Introduction with Metaheuristic Applications," *Hoboken: Wiley*, 2010.

[7] A. H. Alsaeedi, A. H. Aljanabi, M. E. Manna, and A. L. Albukhnefis, "A proactive metaheuristic model for optimizing weights of artificial neural network," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 20, no. 2, pp. 976–984, 2020, doi: 10.11591/ijeecs.v20.i2.pp976-984.

[8] R. Sagban, H. A. Marhoon, and R. Alubady, "Hybrid bat-ant colony optimization algorithm for rule-based feature selection in health care," *Int. J. Electr. Comput. Eng.*, vol. 10, no. 6, pp. 6655–6663, 2020, doi: 10.11591/ijece.v10i6.pp6655-6663.

[9] E.-G. Talbi, *METAHEURISTICS FROM DESIGN TO IMPLEMENTATION*, vol. 6, no. May. John Wiley & Sons, Inc., Hoboken, New Jersey, 2009.

[10] J. E. Smith and A. E. Eiben, *Introduction to evolutionary computing*, vol. 28. Springer Berlin Heidelberg, 2015.

[11] J. H. Holland, “Genetic algorithms,” *Sci. Am.*, vol. 267, no. 1, pp. 66–72, 1992, doi: 10.1038/scientificamerican0792-66.

[12] L. M. G. Dorigo, M, “Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem,” *Belgium TR/IRIDIA/1996-*, vol. 1, no. 1, p. 53, 1997, [Online]. Available: http://people.idsia.ch/~luca/acs-ec97.pdf.

[13] M. Abdel-Basset, L. Abdel-Fatah, and A. K. Sangaiah, *Metaheuristic algorithms: A comprehensive review*. Elsevier Inc., 2018.

[14] J. Doering, R. Kizys, A. A. Juan, À. Fitó, and O. Polat, “Metaheuristics for rich portfolio optimisation and risk management: Current state and future trends,” *Oper. Res. Perspect.*, vol. 6, no. August, p. 100121, 2019, doi: 10.1016/j.orp.2019.100121.

[15] X. S. Yang, “A new metaheuristic Bat-inspired Algorithm,” *Stud. Comput. Intell.*, vol. 284, pp. 65–74, 2010, doi: 10.1007/978-3-642-12538-6_6.

[16] S. Yilmaz and E. U. Küçüksille, “A new modification approach on bat algorithm for solving optimization problems,” *Appl. Soft Comput. J.*, vol. 28, pp. 259–275, 2015, doi: 10.1016/j.asoc.2014.11.029.

[17] Z. Haruna and S. A. T. Mu’azu, Muhammad B., Kabir A. Abubilal, “Development of a Modified Bat Algorithm using Elite Opposition – Based Learning,” *IEEE 3rd Int. Conf. Electro-Technology Natl. Dev. Dev.*, pp. 144–151, 2017.

[18] X. Shan, K. Liu, and P. L. Sun, “Modified Bat Algorithm Based on Lévy Flight and Opposition Based Learning,” *Sci. Program.*, vol. 2016, 2016, doi: 10.1155/2016/8031560.

[19] M. R. Chen, Y. Y. Huang, G. Q. Zeng, K. Di Lu, and L. Q. Yang, “An improved bat algorithm hybridized with extremal optimization and Boltzmann selection,” *Expert Syst. Appl.*, vol. 175, no. March, p. 114812, 2021, doi: 10.1016/j.eswa.2021.114812.

[20] S. Yilmaz, E. U. Kucuksille, and Y. Cengiz, “Modified bat algorithm,” *Elektron. ir Elektrotechnika*, vol. 20, no. 2, pp. 71–78, 2014, doi: 10.5755/j01.eee.20.2.4762.

[21] D. Tansui and A. Thammano, “An Enhanced Bat Algorithm with Random Walk for Solving Continuous Optimization Problems,” *Proc. - 20th IEEE/ACIS Int. Conf. Softw. Eng. Artif. Intell. Netw. Parallel/Distributed Comput. SNPD 2019*, pp. 39–44, 2019, doi: 10.1109/SNPD.2019.8935679.

[22] X. Wang, W. Wang, and Y. Wang, “An Adaptive Bat Algorithm,” *IIntelligent Comput. Theor. Technol. 9th Int. Conf. ICIC 2013, Nanning, China, July 28-31, 2013. Proc. 9. Springer Berlin Heidelberg, 2013.*, pp. 216–223, 2013.

[23] J. Huang and Y. Ma, “Bat algorithm based on an integration strategy and gaussian distribution,” *Math. Probl. Eng.*, vol. 2020, pp. 1–22, 2020, doi: 10.1155/2020/9495281.

[24] S. S. Guo, J. S. Wang, and X. X. Ma, “Improved Bat Algorithm Based on Multipopulation Strategy of Island Model for Solving Global Function Optimization Problem,” *Comput. Intell. Neurosci.*, vol. 2019, 2019, doi: 10.1155/2019/6068743.

[25] A. Rezaee Jordehi, “Chaotic bat swarm optimisation (CBSO),” *Appl. Soft Comput. J.*, vol. 26, pp. 523–530, 2014, doi: 10.1016/j.asoc.2014.10.010.

[26] M. Z. Rodriguez *et al.*, “Clustering algorithms: A comparative approach,” *PLoS One*, vol. 14, no. 1, pp. 1–31, 2019, doi: 10.1371/journal.pone.0210236.

[27] G. Verma, “Chapter-04 D,” *Jaypees Dent. Dict.*, pp. 126–151, 2009, doi: 10.5005/jp/books/10428_4.

[28] T. S. Madhulatha, “An overview of clustering methods,” *IOSR J. Eng.*, vol. 2(4), pp. 719–725, 2012, doi: 10.3233/ida-2007-11602.

[29] X. X. Martin Ester, Hans-Peter Kriegel, Jörg Sander, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,” *Proc. 2nd Int. Conf. Knowl. Discov. Data Min.*, vol. 96, no. 34, pp. 226–231, 1996, doi: 10.11901/1005.3093.2016.318.

[30] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, “A novel population initialization method for accelerating evolutionary algorithms,” *Comput. Math. with Appl.*, vol. 53, no. 10, pp. 1605–1614, 2007,

doi: 10.1016/j.camwa.2006.07.013.

[31] X.-S. S. Yang and M. Karamanoglu, *Nature-Inspired Metaheuristic Algorithms Second Edition*, vol. 4, no. C. Luniver Press, United Kingdom, 2013.

[32] M. S. Tavazoei and M. Haeri, "Comparison of different one-dimensional maps as chaotic search pattern in chaos optimization algorithms," *Appl. Math. Comput.*, vol. 187, no. 2, pp. 1076–1085, 2007, doi: 10.1016/j.amc.2006.09.087.

[33] X.-S. Yang, "Appendix A: Test Problems in Optimization," *Eng. Optim.*, no. 2010, pp. 261–266, 2010, doi: 10.1002/9780470640425.app1.

[34] R. W. Garden and A. P. Engelbrecht, "Analysis and classification of optimisation benchmark functions and benchmark suites," *Proc. 2014 IEEE Congr. Evol. Comput. CEC 2014*, vol. 1, pp. 1641–1649, 2014, doi: 10.1109/CEC.2014.6900240.

[35] K. Hussain, M. N. M. Salleh, S. Cheng, and R. Naseem, "Common benchmark functions for metaheuristic evaluation: A review," *Int. J. Informatics Vis.*, vol. 1, no. 4–2, pp. 218–223, 2017, doi: 10.30630/joiv.1.4-2.65.

[36] M. Jamil and X. S. Yang, "A literature survey of benchmark functions for global optimisation problems," *Int. J. Math. Model. Numer. Optim.*, vol. 4, no. 2, pp. 150–194, 2013, doi: 10.1504/IJMMNO.2013.055204.