RAIRO-Oper. Res. 57 (2023) 2659–2685
https://doi.org/10.1051/ro/2023135

RAIRO Operations Research
www.rairo-ro.org

A NEW MODIFIED BAT ALGORITHM FOR GLOBAL OPTIMIZATION

Nouhaila Adil* and Halima Lakhbab

Abstract. Bat Algorithm, is an evolutionary computation technique based on the echolocation behaviour of microbats while looking for their prey. It is used to perform global optimization. It was developed by Xin-She Yang in 2010. Since then, it has extensively been applied in various optimization problems because of its simple structure and robust performance. Continuous, discrete, or binary, many variants were proposed over the last few years, with applications to solve real-world cases in different fields. Yet, it has one major drawback: its premature convergence due to a lack in its exploration ability. In this paper, we introduce a selection-based improvement and three other modifications to the standard version of this metaheuristic in order to enhance the diversification and intensification capabilities of the algorithm. The newly proposed method has been then tested on 20 standard benchmark functions and the CEC2005 benchmark suit. Some non-parametric statistical tests were also used to compare the New Bat algorithm with other algorithms, and results indicate that the new method is very competitive and outperforms some of the state-of-the-art algorithms.

Mathematics Subject Classification. 90C26, 90C59, 90C06.

Received July 23, 2022. Accepted September 6, 2023.

1. Introduction

Metaheuristics are a class of approximate methods designed to solve complex optimization problems. Most of these methods imitate certain strategies or metaphors from nature. Their goal is to provide a good enough solution in a reasonable time for hard problems in science and engineering and they are problem independent. Two major components of any metaheuristic algorithm are exploration and exploitation, also known as diversification and intensification. The diversification strategy aims at extending the search onto different parts of the search space looking for new solutions. While intensification guides the method to deeply exploit a promising part of the search space using information from existing solutions.

The advantage of metaheuristic methods is that they don't depend on the characteristics of the problem being optimized (no need to look for derivatives, linearity, or other) and can search very large spaces of candidate solutions. The disadvantage is the number of parameters that can be important and should be fine-tuned to guarantee a good optimization performance. Besides, those methods give only approximate solutions. Metaheuristic algorithms can be classified in many ways. A distinction can be made between single-individual methods and

Keywords. Bat algorithm, metaheuristics, swarm intelligence, roulette wheel selection.

Department of Mathematics and Computer Sciences, Faculty of Sciences Ain Chock, HASSAN II University, Casablanca, Morocco.
*Corresponding author: adil.nouhaila@gmail.com

population-based methods. Single-based methods solutions are also called trajectory methods and improve a single solution along iterations, like Tabu Search [13], and Simulated Annealing (SA) [17].

In population-based methods, the search process starts with multiple candidate solutions rather than one. Then, a new population of solutions is generated using the characteristics of the current one. Finally, this new population is integrated into the current one depending on the method's defined selection conditions. The search process is stopped when a termination criterion set in the beginning is met. Genetic Algorithms [19] and Particles Swarm Optimization (PSO) [16] are famous examples of this type of method.

The bat algorithm BA was proposed by Yang [33]. The method grabbed the researcher's attention due to its easy implementation and high performance. Several variants of BA were proposed in the last 13 years (see related work in Sect. 2), in an attempt to improve the algorithm and overcome its premature convergence problem.

In this paper, we introduce a new variant of the bat algorithm we call NeBA (**Ne**w **B**at **A**lgorithm), where, we include new modifications, to increase the diversification and enhance intensification in the algorithm. The robustness and efficiency of the NeBA method are tested on 20 Classical benchmark test functions, in addition to the CEC2005 benchmark suite. The latter consists of 25 generic test problems. Those problems present real-valued functions to optimize and specify a common termination criterion, size of problems, initialization scheme, linkages/rotation, etc. so that the evaluation of new methods can be more systematic. Therefore, the paper is organized as follows; BA Related work is presented in Section 2. The standard algorithm of BA is described in Section 3. Our new modified bat algorithm (NeBA) is introduced in Section 4, followed by the results and discussion in Section 5, and finally the conclusion and perspectives are given in Section 6.

2. Related work

Since its development, the bat algorithm attracted the researchers' attention due to its simplicity, good performance, and ability to produce high-quality solutions, especially on low-dimensional functions. It was seen as a promising method, that needs, nonetheless, to be studied more deeply to overcome its deficiency in solving high-dimensional problems. Consequently, several extensions and techniques were proposed in the last few years.

Actually, finding an equilibrium between exploration and exploitation in an algorithm is a difficult task. One way to address this is to add factors to adjust and adapt the search along iterations. Yu Li *et al.* [18] for example, proposed an improved BA (LAFBA) where they first added a dynamically decreasing inertia weight to improve the speed and accuracy of the algorithm. Secondly, they introduced Lévy flight to the bat's position update equation to enable bats to move in the search space without being trapped in local optima (in other words to jump out of an unpromising area). Lastly, they used a speed adjustment factor, to reduce the impact of velocity on position updates in the last iterations. The resulting algorithm thus was given a certain balance between global and local search capabilities and has shown great performance. Gan *et al.* [9] also included a stochastic inertia weight in their new proposed bat algorithm based on iterative local search (ILSSIWBA), and as reported in their results, the stochastic inertia weight helped balance the relationship between the global and local convergence and it showed a great improvement in the optimization precision. The authors also used an iterative local search metaheuristic known for its ability to avoid being trapped in a local optimum. Some other authors have focused on the local search component. Zhang *et al.* [38] reported that the local search step in BA does not perform well in the early search stage. Therefore, they designed an individual local search technique to enhance the exploitation of BA and to effectively search the bat's neighbourhood during the first iterations. Bangyal *et al.* [3] have designed an improved version of BA by applying torus walk instead of the classical local search, to improve diversity and convergence. Chakri *et al.* [4] is among the works that proposed a new extension of BA with several modifications. In this new extension, they call the directional BA (dBA), they used the idea that bats should be directed by other bats not only the best bat of the swarm, using the directional nature of echolocation inspired by real bats' echolocation nature. This idea was used to enhance the exploration of the algorithm by diversifying the movement directions of the bats in the population. They also introduced a parameter w that regulates the step size of the random walk equation in order to reinforce

the local search, especially in the final iterations. The other modification consists of changing the loudness and pulse rate update equations to monotonically decreasing, increasing equations to guarantee the equilibrium between the exploration and exploitation of the algorithm. Meng *et al.* [20], also thought of a new adaptation of the algorithm inspired by real bats' comportment while looking for food, by incorporating the bats' habitat selection and Doppler effect in echoes. The goal was to further mimic bat's behaviours and reflect their self-adaptive ability, that the original BA misses (as it simplified the social behaviour of bats) and can be considered as one of the reasons for its imperfection.

Different from the cited works above, some papers addressed the BA drawbacks differently. Topal *et al.* [31] for instance, have developed a role-based search to improve the Bat Algorithm. In their algorithm, there are only two bats, one is the explorer and the other is the exploiter, besides, they exchange their roles depending on their position during the search process. Wang *et al.* [32] in another hand, have proposed a new bat algorithm based on a novel topology, defining new velocity formula and a new local update equation to reduce the impact of the parameter setting of the algorithm. Other works also consider the hybridization of BA with other algorithms, like the hybrid system (MBADE) proposed by Yildizdan and Baykan [36]. Their work involves the use of a modified version of BA. They first proposed to improve the exploitation and exploration abilities of the algorithm (inspired by the work of Chakri *et al.* [4]), and then combined it with the Differential Evolution (DE) algorithm to build an effective hybrid system with improved potential exploitation. The work of He *et al.* [15] also, is among the hybrid algorithms of the BA category. They proposed a bat algorithm based on simulated annealing (SA) and Gaussian perturbations. It can be considered as a simple hybridization with SA, where the best solution in the population is replaced by a new one given by the SA algorithm, and in the local search step Gaussian perturbation was performed instead of the classical local search, thus speeding up global convergence and improves the accuracy of the algorithm. Yilmaz *et al.* [37] considered the hybridization of BA with invasive weed optimization, in addition to two more modifications in the velocity update equation. Invasive weed optimization is a metaheuristic that is known to have a good exploration. Chaotic maps often used in the study of dynamical systems have also shown good results in enhancing BA performance. The maps exhibit some sort of chaotic behaviour. Gandomi and Yang [10] and Jordehi [24] both incorporated Chaos in BA to increase the diversity of the algorithm. A recent work of Rauf *et al.* [23], proposed an ASF-BA algorithm that incorporates two major adjustments to the original BA, the first adjustment is the use of an inertia weight based on the Sine map (One of the Chaotic maps) to boost the velocity of bats. Secondly, they replaced the local search equation with Sugeno Function (used in fuzzy search), in order to provide bats with the ability to adjust their fitness dynamically using their own experience and their neighbours.

It is worth noting that the application horizon of BA is very large, and not limited to continuous optimization problems. The algorithm was adapted to solve many discrete and engineering problems, including, Routing problems [1, 22], scheduling [21], Reliability-based design [5], classification [14], image processing [27], energy systems [26] in which, the algorithm gives competitive results. A multi-objective version of the algorithm was also considered [30, 34].

In all of these works, the algorithms, have been effectively tested on different benchmarks and problems proving the strength of BA as a base strategy that can be improved in many ways to enhance its global convergence. Thus, in this paper, we introduce a new modified BA, where we include the use of metropolis criterion and roulette wheel selection along with other modifications, in an attempt to increase the diversification in the population and to prevent the algorithm from being trapped on local optima.

3. Standard BA

As discussed in the previous section, bat algorithm is one of the stochastic nature-inspired algorithms, that proved to be very efficient in solving optimization problems. Whether it was a continuous or discrete problem, the algorithm was adapted to the different versions and it performed well in both cases.

It was first developed by Yang in 2010, inspired by the way bats look for their prey using what we call echolocation. Reference [33] contains a detailed description of this natural behavior and its modeling to form the bat algorithm.

Generally, the implementation was based on three main rules described as follows:

(1) All bats use echolocation to know the distance and location of their prey and can distinguish between food and obstacles. The position of a bat x_i is encoded as a solution to an optimization problem under consideration [33].
(2) Bats fly randomly from position x_i with velocity v_i, a varying frequency f_i, in addition to an adjustable loudness A_i and pulse rate r_i depending on the proximity of food [33].
(3) The loudness varies from (positive) value $A_{\max}$ to a minimum value $A_{\min}$ [33].

Each bat represents a solution in a d-dimensional search space. The bat at iteration t is provided with a velocity v_i^t and a position x_i^t. Those two parameters are updated depending on the bat's adjusted frequency Q_i^t. The best bat in the swarm is kept as x_* during the iterative process. In the following, the update equations as given in [33]:

$$Q_i^t = Q_{\min} + (Q_{\max} - Q_{\min}) * rand \tag{1}$$
$$v_i^t = v_i^{t-1} + \left(x_* - x_i^{t-1}\right) Q_i^t \tag{2}$$
$$x_i^t = x_i^{t-1} + v_i^t \tag{3}$$

where t is the current iteration number and $rand$ is a random number drawn uniformly from $[0, 1]$.

After that and with a probability r_i (pulse rate), the current bat can perform a local search, using a random walk, and the position is updated by:

$$x_{i_{(\text{new})}} = x_{i_{(\text{old})}} + \epsilon \langle A^t \rangle \tag{4}$$

where ϵ is random from $[-1, 1]$ and $\langle A^t \rangle$ is the average loudness at each iteration t.

As the bat approaches its target, the loudness decreases while the pulse rate increases, thus allowing a certain balance between the exploration and exploitation aspects of the algorithm. The update equations are as follows:

$$A_i^t = \alpha A_i^{t-1} \tag{5}$$
$$r_i^t = r_i^0 \left(1 - e^{-\gamma t}\right) \tag{6}$$

where γ, α in $[0, 1]$. α is like the cooling coefficient in SA [17]. Usually we take $\alpha = \gamma = 0.9$, and $r_i^0 \in [0, 1]$ is the initial emission rate.

In brief, how the algorithm works can be summarized with these three steps:
The first step is where we actually update the position and velocity of bats using equations (1)–(3). This one is considered the diversification phase of the algorithm.
The second is the local search component. This one is controlled by the increasing pulse rate, and the higher it gets during iteration, the more likely the local search will be performed (favored in the last iterations).
The last step is called the selection phase, where the algorithm chooses if the current bat should be updated in the swarm by the resulting bat of previous steps or not, depending on this bat's fitness and its loudness A. This step is what ensures the balance in diversification and intensification techniques used in the algorithm. The pseudocode of the algorithm is depicted in Algorithm 1.

Algorithm 1: Standard BA pseudo-code.

```
Randomly initialize the bat population X = (x_1, x_2, ..., x_n);
Evaluate fitness for each bat F(x_i)
for each bat x_i in the population do
    Initialize the pulse rate r_i, velocity v_i and loudness A_i;
end
repeat
    for each bat x_i in the population do
        Generate new solutions s_i by adjusting frequency, and updating velocities and locations/solutions
        using (Eqs. (1)–(3))
        if rand > r_i then
            Select one solution among the best ones;
            Generate a local solution around the selected one (Eq. (4));
        if rand < A_i and F(s_i) < F(x_*) then
            Accept the new solution: x_i := s_i;
            Increase r_i and decrease A_i (Eqs. (5) and (6));
    end
    Rank the bats and update the current best bat of the population;
until termination criterion reached;
```

4. New modified Bat Algorithm

The New Bat Algorithm (NeBA) we introduce, combines a set of modifications. Those modifications mainly try to change the fact that in BA, the search is always oriented toward the elite solutions, which makes the population loses its diversity along iterations and leads the BA to converge prematurely. Thus, a set of mechanisms were added in the proposed algorithm with the goal of overcoming this drawback. Also, a modification in the local search component is included to enhance the intensification in the algorithm.

4.1. First modification: velocity update

The first modification was applied to the velocity update equation.

We know that velocity plays the role of a rate at which bats change their position. Besides, the velocity as it was defined in [33], provides the bats with a range, a pace, and an orientation using both the frequency and the direction of the best bat in the swarm. The latter being the global solution in the current iteration, it seems only right for bats to follow its path and look for food in the neighboring area. This is what makes the algorithm lacking in terms of diversification aspect and the social behavior of bats is not fully considered. So, to take advantage of this behavior, we assume that the bat selects the direction to take adaptively, based on the information it gathers from other individuals in the population. The idea was inspired by the Adaptive Large Neighborhood Search algorithm (ALNS) [25]. The ALNS is generally used for solving discrete optimization problems like the well-known Vehicle Routing Problem (VRP)[1]. The main idea of ALNS is that it explores a set of neighborhoods and improves a solution by alternating the use of destroy and repair operators (those operators can be problem specific and they are used to destroy a solution and rebuild it differently in an attempt to have a better solution). For more details about the method, readers can revert to the paper [25].

In our case, instead of always using information from the best bat in the velocity update equation, we alternate between three possible choices: the best bat's information x_*, the current bat's best past information x_p or randomly chosen one's information x_r. This mechanism allows a broader neighborhood search. Thus the updating rule becomes:

$$v_i^{t+1} = \omega^t v_i^t + \left(b_s - x_i^t\right) * Q_i^t \tag{7}$$

[1]A well-known problem in combinatorial optimization, where a set of products must be delivered to a set of customers, object to various constraints. The objective of the resolution of such problems is to minimize transportation costs.

where b_s is one of the three possibilities cited above, chosen using roulette wheel selection, and ω^t is the inertia weight at iteration t, which will be described in the third modification Section 4.3.

For simplicity, we denote C as the set of possible choices we are selecting from. In our case $C = \{x_*, x_p, x_r\}$.

Roulette selection is a stochastic selection method, where the probability of selecting an individual is proportional to his fitness, and the fitness should be positive. In our work, the algorithm is supposed to be general and problem independent as we have diverse testing benchmarks. So, instead of using fitness values to compute the probability of selecting a bat from C, we adopted the method presented in the paper [2], in which the authors use the notion of scores and weights instead. Details of the method are described as follows.

Initially, identical weights w_i and nil scores π_i are assigned to elements of the set C. After that, the score of the bat from C that was chosen as b_s will be incremented after each position update by σ_j, j in $\{1, 2, 3\}$, depending on the derived new solution quality. Moreover, the inequality $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq 0$ is to be ensured for reasonable adjustments. In this paper, the authors' choice is:

- If the new solution is the new best solution, $\sigma_1 = 20$.
- If it is better than the incumbent solution, $\sigma_2 = 10$.
- If it is worse than the incumbent solution, $\sigma_3 = 2$.

Then, the score and the weight of the used element are updated according to the equations (8) and (9):

$$\pi_i^{\text{new}} = \pi_i + \sigma_j \tag{8}$$

$$w_i^{\text{new}} = (1 - \alpha) * w_i + \alpha * \frac{\pi_i}{Nu_i} \tag{9}$$

where, Nu_i is the number of uses of bat i from set C, α is a parameter between 1 and 0 that determine the degree of influence of the last score and the previous weight on the new obtained weight w_i^{new}.

Lastly, the probability of being selected is computed using the formula:

$$P_i = \frac{w_i}{\sum_{k=1}^{m} w_k}, \qquad m = |C|. \tag{10}$$

Hence, at the beginning of the first iteration, there is a probability as high as equation (10) for element i to be chosen as the b_s in equation (7). This modification increases the diversity of the movement in different directions, so as to avoid trapping into local optima.

4.2. Second modification: population update criterion

The second modification we made to the original BA, concerns the condition of acceptance of a newly founded solution. In the original BA, the acceptance of a new solution depends on two conditions:

(1) The solution has to produce an objective value lower than the current one $F_{\text{new}} < F_i$.
(2) A randomly generated number has to be lower than the current corresponding loudness, $rand < A_i$.

We have found that those conditions are somehow conservative. In our work, alternatively and based on the SA algorithm, first we introduce the notion of temperature T, then we modify condition (1) to include the metropolis criterion. So we accept a solution if:

(1) The solution produces an objective value lower than the current one ($F_{\text{new}} < F_i$) or a randomly generated number has to be lower than metropolis criterion $\left(rand < \exp\left(-\frac{(F_{\text{new}} - F_i)}{T}\right)\right)$.
(2) A randomly generated number has to be lower than the current corresponding loudness, $rand < A_i$.

Consequently, we allow with probability, the solution to be updated in the population even if it is a non-improving one. With this new configuration, we provide the bats with a non-monotone behavior (see Fig. 5 in the Numerical Results section), so that they can explore more research areas, increasing the probability of escaping

from sub-optimal areas to find a more promising region, yielding to even better results. This modification is complementary to the first one assuring the increase of population diversity.

In this work, the maximum Temperature $T_{\max}$ is assumed to be equal to the absolute value of the objective function of the first designed best bat at the beginning $T_{\max} = abs(F_{\text{best}})$ and to decrease at each iteration as the following:

$$T^t = T_{\max} - \frac{(T_{\max} - T_{\min}) * t}{N_{\max}} \tag{11}$$

$T_{\min}$ is the minimum value to be reached, set to 0.01. t and $N_{\max}$ are the current iteration and the maximum number of iterations respectively.

The update equations of the loudness A and pulse rate r, remain the same. Thus, if a new solution is better than the incumbent one and it was accepted, A and r are updated using equations (5) and (6).

4.3. Third modification: inertia weight

Inertia weight factor originally was introduced in PSO [28]. The factor generally plays the role of a speed adjustor that balance the search capabilities and efficiency of the algorithm as it has been confirmed in other works.

So, in order to enhance the balance between the exploration and the exploitation in the algorithm, and to allow the search process to exploit some specific areas, we use a linear decreasing inertia weight into the velocity equation (7). At each iteration, the inertia weight is updated as follows:

$$\omega^t = \omega_{\max} - \frac{(\omega_{\max} - \omega_{\min}) * t}{N_{\max}}. \tag{12}$$

In this strategy, ω is decreased linearly from $\omega_{\max}$ at the early iterations to $\omega_{\min}$ at the later ones.

4.4. Forth modification: local search component

The local search component in the Bat Algorithm is what allows to exploit areas around the best solution (or one among several chosen elite solutions).

In order to enhance this exploitation capability, we adopted a modification that has been introduced in [4]. In the latter, a parameter ζ was included in the random walk equation:

$$x_{i_{(\text{new})}} = x_* + \epsilon \cdot \langle A^t \rangle \cdot \zeta^t \tag{13}$$

ζ is a parameter that regulates the scale of the search, and as it has been reported, the technique effectively improves the quality of the Local Search, as it allows, in the last iterations, the bat to move randomly in a reduced search region around the best solution and thus the exploitation capability of the algorithm is also enhanced.

In our work, the parameter is updated as the following:

$$\zeta^t = \zeta_{\max} - \frac{(\zeta_{\max} - \zeta_{\min}) * t}{N_{\max}} \tag{14}$$

where $\zeta_{\max}$ and $\zeta_{\min}$ are the initial and final values, respectively.

To sum up all the modifications above, the flowshart of NeBA is presented in Figure 1 and the pseudo-code can be presented as follows:

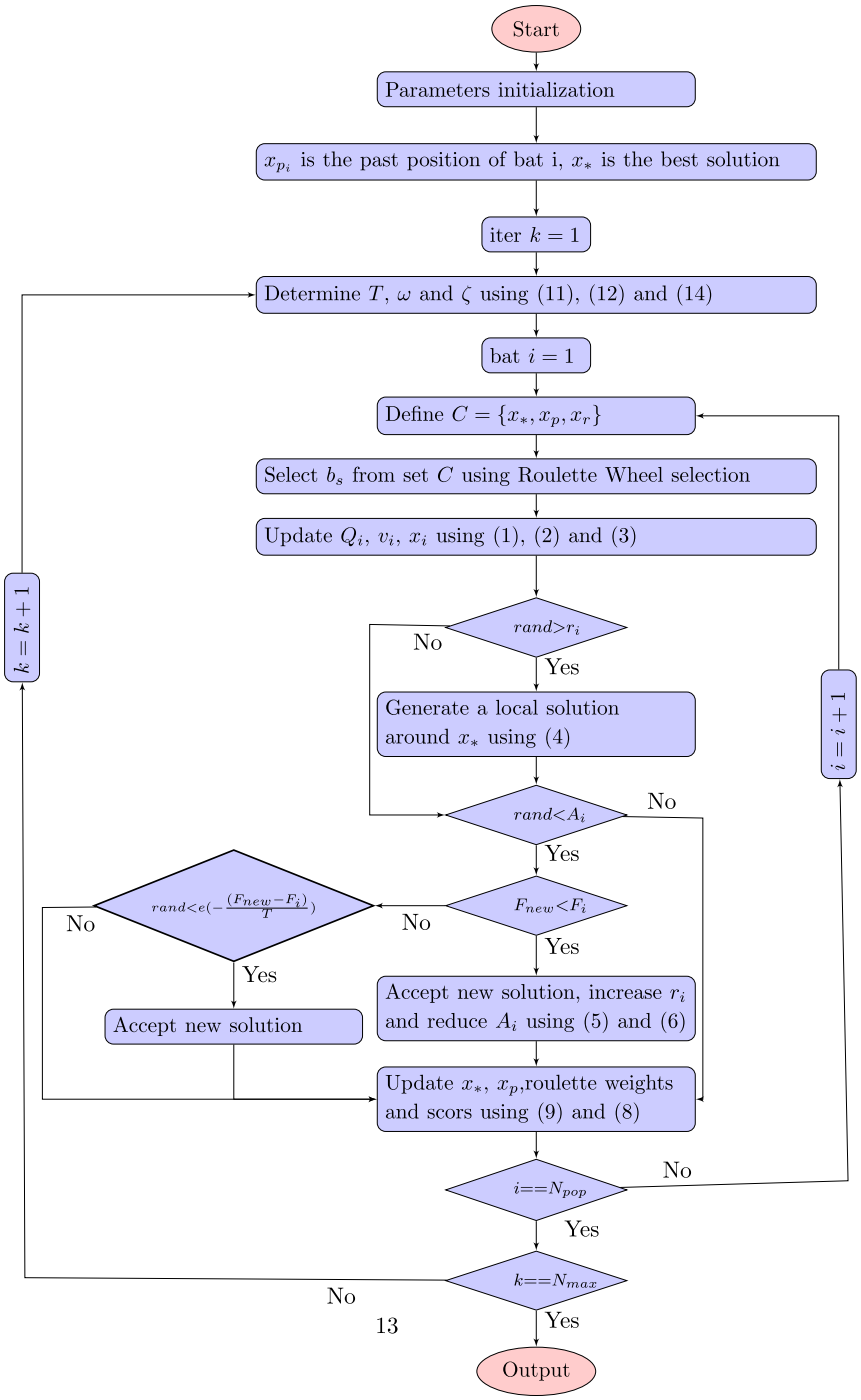


FIGURE 1. Flowchart of NeBA.

Algorithm 2: NeBA pseudo-code.

Randomly initialize the bat population $X = (x_1, x_2, \ldots, x_n)$;
Evaluate the fitness for each bat $F(x_i)$;
for *each bat* x_i *in the population* **do**
  Initialize the pulse rate r_i, velocity v_i and loudness A_i;
end
Initialize Scores π_j, weights w_j and maximum temperature $T_{\max}$;
repeat
  Update temperature T, inertia weight w and local search adjustor ζ (Eqs. (11), (12) and (14));
  for *each bat* x_i *in the population* **do**
    Define set of choices $C = \{x_*, x_p, x_r\}$;
    Select b_s using Roulette wheel selection;
    Generate new solutions by adjusting frequency, and updating velocities and locations/solutions (Eqs. (1), (7) and (3));
    if $rand > r_i$ **then**
      Generate a local solution around the best bat (Eq. (13));
    if $rand < A_i$ **then**
      if $F_{\text{new}} < F(x_i)$ **then**
        Accept the new solution;
        Increase r_i and decrease A_i (Eqs. (6) and (5));
      else if $rand < \frac{\exp(-(F_{\text{new}} - F(x_i)))}{T}$ **then**
        Accept the new solution;
    Update scores and weights;
    Update x_p and x_*;
  end
  Return the current best bat of the population;
until *termination criterion reached*;

5. Numerical results and discussion

The performance of the proposed NeBA was investigated by carrying out four numerical experiments. The first test was performed on 20 classical Benchmark functions and compared with the results of standard BA. The second experiment shows the impact of each modification in our NeBA. In the third test, a comparison was done between the NeBA and other famous algorithms from literature on the same 20 functions to efficiently assess the performance of NeBA. At last, the CEC2005 benchmark functions were used to test the algorithm as well. The details are given in the sections that follow.

5.1. First experiment: BA *vs.* NeBA

In the first test, twenty popular benchmark functions as shown in Table 1, have been used to measure the performance of the new algorithm. The first column presents the name of the function, the second presents its number that will be used in the discussion, the next column defines the formula and in the last column, we have the interval where the search is performed.

To note that, these functions have different characteristics such as separability, modality, and scalability as shown in Table 2. Abreviations are described in the footnote[2].

In this experiment, the results obtained are compared with the results of the original BA that was implemented by the authors. Besides, in order to keep the fairness in comparison and so that we can carry out meaningful statistical tests, we run the two algorithms 51 times (like used in the work [4] as we are using it as reference for results compared with other algorithms), using the same common parameters, and then, values for the specific parameters of our algorithm were chosen experimentally and can be summarized in Table 3 as follows:

[2]MultiM: Multimodal; UniM: Unimodal; S:Separable; NS:Non-Saparable; Sc: Scalable; NSc: Non-Scalable.

TABLE 1. Benchmark functions F1–F20.

Function name	N°	Formula	Bounds
Ackley	F1	$-20e^{-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}} - e^{\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)} + 20 + e^1$	$[-100, 100]^d$
Alpine I	F2	$\sum_{i=1}^{d} \lvert x_i \sin(x_i) + 0.1x_i \rvert$	$[-10, 10]^d$
Bent Cigar	F3	$x_1^2 + 10^6 \sum_{i=2}^{d} x_i^2$	$[-10, 10]^d$
Dixon price	F4	$(x_1 - 1)^2 + \sum_{i=2}^{d} i\big(2x_i^2 - x_{i-1}\big)^2$	$[-10, 10]^d$
Griewank	F5	$1 + \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos\Big(\frac{x_i}{\sqrt{i}}\Big)$	$[-600, 600]^d$
Lévy	F6	$\sin^2(\pi w_1) + \sum_{i=1}^{d-1}(w_i - 1)^2\big(1 + 10\sin^2(\pi w_i + 1)\big)$ $+(w_d - 1)^2\big(1 + \sin^2(2\pi w_d)\big)$ **where** $w_i = 1 + \frac{x_i - 1}{4}$	$[-5.12, 5.12]^d$
Michalewicz	F7	$-\sum_{i=1}^{d} \sin(x_i) \sin\Big(\frac{ix_i^2}{\pi}\Big)^{2m}$	$[0, \pi]^d$
Powell	F8	$\sum_{i=1}^{d/4}\Big[(x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2$ $+(x_{4i-2} - 2x_{4i-1})^4 + 10(x_{4i-3} + x_{4i})^4\Big]$	$[-10, 10]^d$
Rastrigin	F9	$10d + \sum_{i=1}^{d}\big(x_i^2 - 10\cos(2\pi x_i)\big)$	$[-5.12, 5.12]^d$
Rosenbrock	F10	$\sum_{i=1}^{d}\Big[100\big(x_{i+1} - x_i^2\big)^2 + (1 - x_i)^2\Big]$	$[-10, 10]^d$
Rotated hyper ellipsoid	F11	$\sum_{i=1}^{d}\sum_{j=1}^{i} x_j^2$	$[-65, 65]^d$
Salomon	F12	$1 - \cos\Big(2\pi\sqrt{\sum_{i=1}^{d} x_i^2}\Big) + 0.1\sqrt{\sum_{i=1}^{d} x_i^2}$	$[-100, 100]^d$
Schwefel	F13	$418.9829d - \sum_{i=1}^{d} x_i \sin\Big(\sqrt{\lvert x_i \rvert}\Big)$	$[-500, 500]^d$
Schaffer F7	F14	$\frac{1}{d-1}\sum_{i=1}^{d-1}\Big[\big(x_i^2 + x_{i+1}^2\big)^{0.25}$ $+\big(x_i^2 + x_{i+1}^2\big)^{0.25}\sin^2\Big(50\big(x_i^2 + x_{i+1}^2\big)^{0.1}\Big)\Big]$	$[-100, 100]^d$
Sphere	F15	$\sum_{i=1}^{d} x_i^2$	$[-100, 100]^d$
Styblinski Tang	F16	$0.5\sum_{i=1}^{d}\big(x_i^4 - 16x_i^2 + 5x_i\big) + 39.16599d$	$[-10, 10]^d$
Sum of different power	F17	$\sum_{i=1}^{d} \lvert x_i \rvert^{i+1}$	$[-100, 100]^d$
Trid	F18	$\sum_{i=1}^{d}(x_i - 1)^2 - \sum_{i=2}^{d} x_i x_{i-1}$	$[-d^2, d^2]^d$
Zakharov	F19	$\sum_{i=1}^{d} x_i^2 + \Big(\sum_{i=1}^{d} 0.5ix_i\Big)^2 + \Big(\sum_{i=1}^{d} 0.5ix_i\Big)^4$	$[-5, 10]^d$
Weirstrass	F20	$\sum_{i=1}^{d}\big(\sum_{k=0}^{20}\big[0.5^k\cos\big(2\pi\cdot 3^k(x_i + 0.5)\big)\big]\big)$ $-d\sum_{k=0}^{20}\big[0.5^k\cos\big(2\pi\cdot 3^k\cdot 0.5\big)\big]$	$[-0.9, 0.9]^d$

TABLE 2. Benchmark functions characteristics.

	Separability	Scalability	Modality
F1	NS	Sc	MultiM
F2	S	NSc	MultiM
F3	NS	Sc	UniM
F4	NS	Sc	UniM
F5	NS	Sc	MultiM
F6	NS	Sc	MultiM
F7	NS	Sc	MultiM
F8	NS	Sc	UniM
F9	S	Sc	MultiM
F10	NS	Sc	UniM
F11	NS	NSc	UniM
F12	NS	Sc	MultiM
F13	S	Sc	UniM
F14	NS	Sc	UniM
F15	S	Sc	UniM
F16	NS	NSc	MultiM
F17	NS	Sc	UniM
F18	NS	NSc	MultiM
F19	NS	Sc	MultiM
F20	S	Sc	MultiM

TABLE 3. Test parameters.

Common parameters	Specific parameters of NeBA
Population size $N_{\text{pop}} = 30$	Min and Max inertia weight and ζ parameter:
Max iterations $N_{\max} = 500$	$w_{\min} = \zeta_{\min} = 0.1$
Dimension $D = 30$	$w_{\max} = \zeta_{\max} = 0.8$
Pulse rate $r \in [0, 1]$ and Loudness $A \in [0, 2]$	The parameter λ used in selection:
Max and Min frequency $Q_{\min} = 0$, $Q_{\max} = 2$	$\lambda = 0.4$
$\alpha = \gamma = 0.9$	

It is worth mentioning that all the presented results were generated using Python as the programming language, and the test was performed on an Intel Core i7 laptop, with 2.80 GHz and RAM of 16 GB.

Next, the best solution (Min), the mean value (Mean), the standard deviation (Std), and the worst solution (Max) values were calculated and presented in Table 4 (Best results in bold) to show the performance of our algorithm.

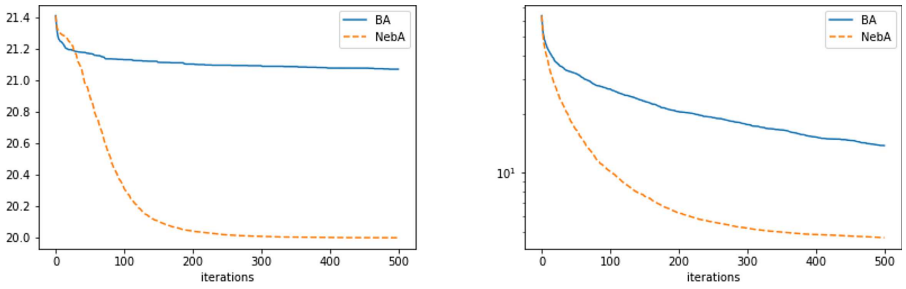
From the results shown in Table 4, our NeBA clearly outperforms BA in 14 out of 20 functions considering the four criteria. Besides, in terms of the quality of the best solution obtained, the NeBA gives a better minimum value than BA in all functions but F13, and a better Mean for all functions but F13 and F16.

Next, Figures 2–4 represent the convergence aspect of both algorithms, the mean of the global minimum at each iteration over the 51 runs was computed to efficiently assess the convergence of the two algorithms.

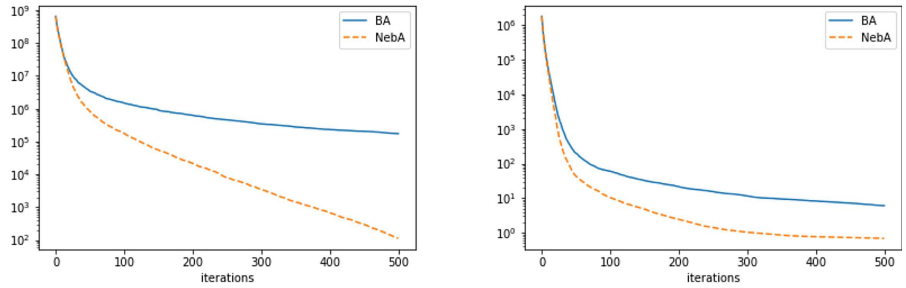
The figures show that NeBA is superior in terms of convergence compared with the BA in several benchmarks. In the beginning, both algorithms allow a certain level of diversity in the swarm. But at some point, the BA starts losing its diversity, as we start gathering with a probability only the good candidates, unlike our NeBA, which gives room to the population to accept even low-quality candidates, this is what we call a non-monotone property, that helps to explore new possible sub-optimal regions, considering that we keep records anyways,

TABLE 4. First experiment: classical benchmark BA *vs.* NeBA.

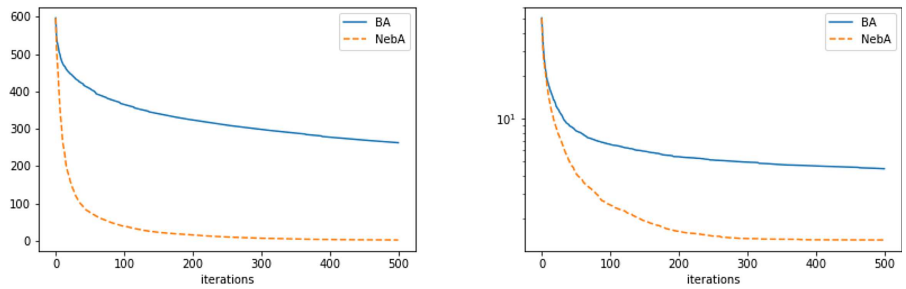
		Mean	Std	Min	Max
F1	BA	2.107E+01	6.456E−02	2.084E+01	2.117E+01
	NeBA	**2.000E+01**	**1.081E−03**	**1.999E+01**	**2.000E+01**
F2	BA	1.378E+01	5.525E+00	4.461E+00	2.840E+01
	NeBA	**4.642E+00**	**2.286E+00**	**8.006E−01**	**1.149E+01**
F3	BA	1.739E+05	6.907E+04	4.247E+04	3.865E+05
	NeBA	**1.122E+02**	**3.036E+01**	**4.379E+01**	**1.786E+02**
F4	BA	6.112E+00	4.706E+00	1.827E+00	2.473E+01
	NeBA	**6.925E−01**	**5.292E−02**	**6.679E−01**	**9.110E−01**
F5	BA	2.630E+02	4.425E+01	6.947E+01	3.436E+02
	NeBA	**2.475E+00**	**1.549E+00**	**3.483E−01**	**6.950E+00**
F6	BA	4.469E+00	2.249E+00	1.219E+00	1.316E+01
	NeBA	**1.416E+00**	**9.934E−01**	**8.958E−02**	**5.081E+00**
F7	BA	−1.006E+01	**6.707E−01**	−1.184E+01	−9.042E+00
	NeBA	**−1.762E+01**	2.096E+00	**−2.150E+01**	**−1.292E+01**
F8	BA	1.070E+01	6.629E+00	2.324E+00	2.988E+01
	NeBA	**2.808E−02**	**1.237E−02**	**4.718E−03**	**5.465E−02**
F9	BA	2.817E+02	2.340E+01	2.226E+02	3.319E+02
	NeBA	**8.873E+01**	**2.068E+01**	**4.380E+01**	**1.294E+02**
F10	BA	8.955E+01	8.905E+01	3.449E+01	3.650E+02
	NeBA	**3.427E+01**	**2.199E+01**	**1.726E+01**	**1.125E+02**
F11	BA	1.797E+00	1.265E+00	3.995E−01	7.667E+00
	NeBA	**3.247E−03**	**1.513E−03**	**1.214E−03**	**8.869E−03**
F12	BA	2.781E−02	1.007E−02	1.528E−02	5.688E−02
	NeBA	**9.150E−04**	**1.376E−04**	**5.379E−04**	**1.298E−03**
F13	BA	**5.003E+03**	6.933E+02	**3.218E+03**	**6.042E+03**
	NeBA	5.429E+03	**6.685E+02**	4.391E+03	7.167E+03
F14	BA	7.755E+00	**4.502E−01**	6.496E+00	8.658E+00
	NeBA	**6.443E+00**	6.670E−01	**4.252E+00**	**7.465E+00**
F15	BA	8.540E−02	5.160E−02	2.628E−02	3.072E−01
	NeBA	**8.641E−05**	**2.509E−05**	**4.139E−05**	**1.342E−04**
F16	BA	**−1.005E+03**	**3.584E+01**	−1.074E+03	**−9.449E+02**
	NeBA	−1.000E+03	4.095E+01	**−1.076E+03**	−9.064E+02
F17	BA	9.511E+20	4.794E+21	3.090E+08	3.323E+22
	NeBA	**1.721E+08**	**7.099E+08**	**7.771E+01**	**4.768E+09**
F18	BA	−1.668E+07	**8.081E+05**	−1.821E+07	−1.492E+07
	NeBA	**−1.939E+07**	8.374E+05	**−2.108E+07**	**−1.723E+07**
F19	BA	4.345E+00	1.661E+00	1.819E+00	8.424E+00
	NeBA	**1.447E−03**	**6.123E−04**	**5.219E−04**	**3.216E−03**
F20	BA	−1.278E+01	1.414E+00	−1.695E+01	−1.092E+01
	NeBA	**−3.739E+01**	**4.219E+00**	**−4.671E+01**	**−2.569E+01**



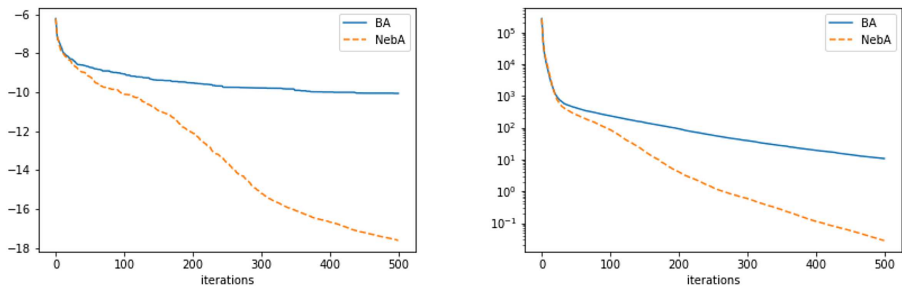
F1 F2



F3 F4



F5 F6



F7 F8

FIGURE 2. Convergence of the objectif function F1–F8.

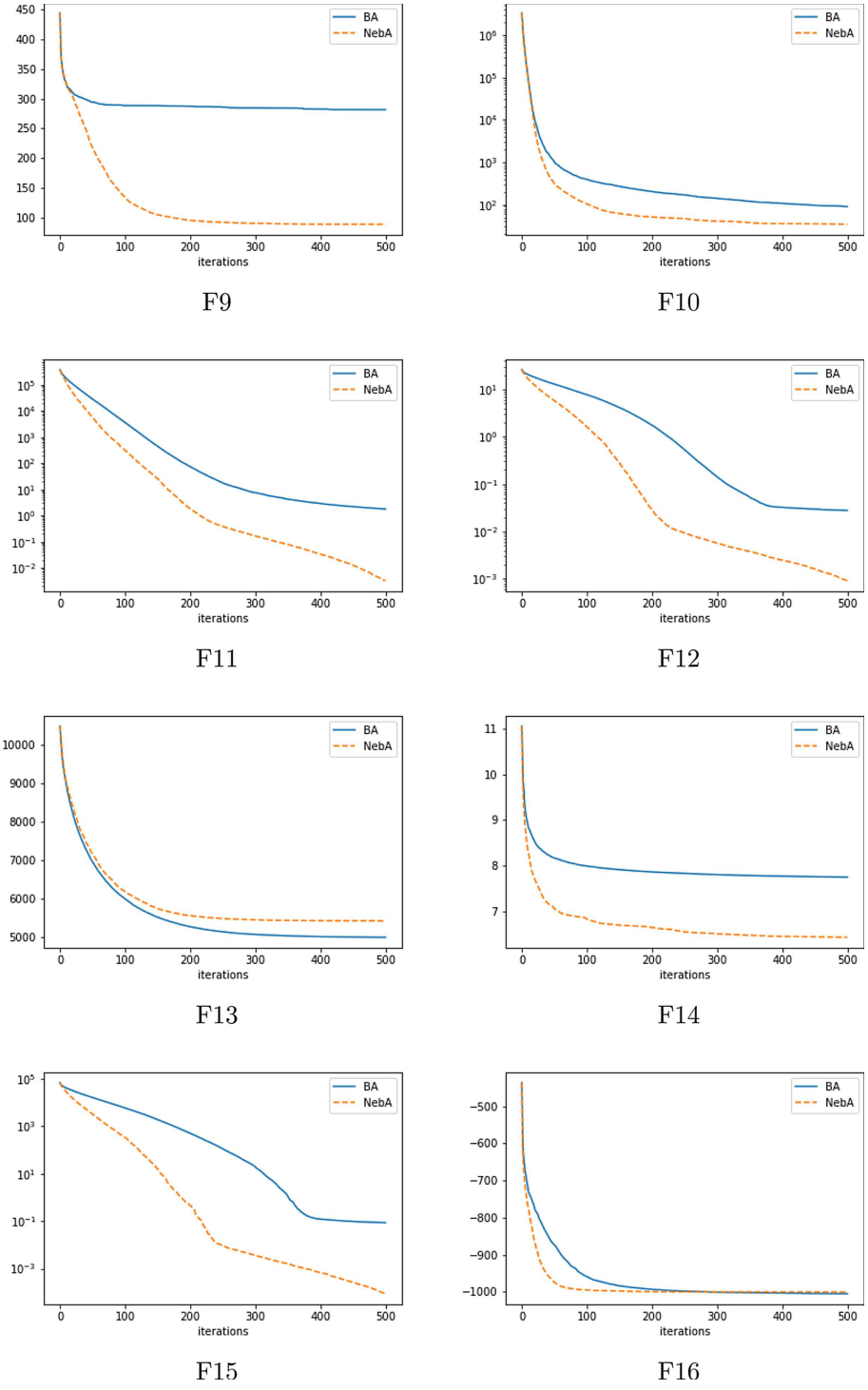


FIGURE 3. Convergence of the objectif function F9–F16.

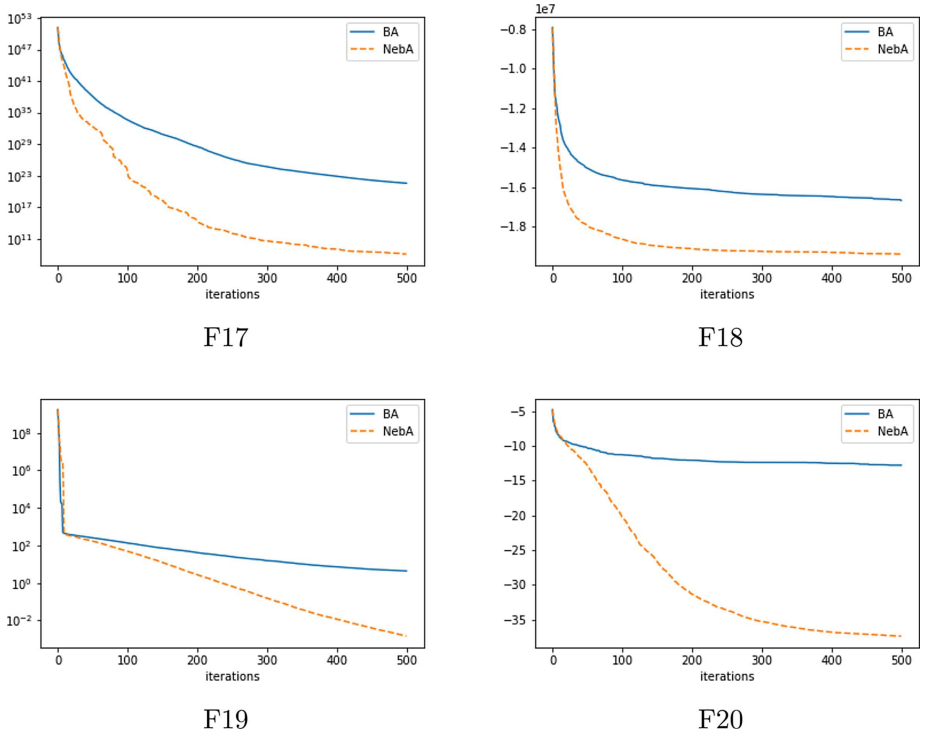


FIGURE 4. Convergence of the objectif function F17–F20.

of the met global best solution in each iteration. Thus, the search area becomes wider, and bats are given the possibility to learn from each other.

The non-monotone property allows, in general, escaping from local optima. Figure 5 represents the non-monotone aspects of the bats update for 4 benchmark functions. A random bat was selected, and its fitness evolution over iterations is represented to show how the bat can explore its surroundings more effectively, by sometimes accepting non-improving solutions, to find better search areas.

The graphs in Figure 5 show that providing the algorithm with the non-monotone ability, lets the bats jump out of their current positions, sometimes to really bad positions, to afterward jump out of this same bad position to a good one. It can be seen as a way to find hidden promising search areas. In the case of Rastrigin, Michalwicz, and Ackley for example, the evolution of the chosen bat from BA's swarm, stopped after a certain number of iterations, while the one from NeBA's swarm, with the new ability, escapes and manage to find a better position. Generally, although, in the example of the function F13, the chosen bat's graph shows that BA hits minimum values better than NeBA. The values given by NeBA are indeed in a wider range, thus it explores more neighborhoods, providing more information to be used by the bats in the swarm.

To show NeBA's performance over BA, non-parametric pairwise comparisons were used, namely the sign test and Wilcoxon test. Table 5, presents the result of the two tests. The first row presents the number of wins and losses of NeBA against BA. The algorithm is considered a winner on a given problem if the mean of the best of 51 runs is better than the BA.

Now as we have 20 problems, and with a level of significance $\alpha = 0.05$, our algorithm outperforms its standard version (an algorithm is considered better if it has at least 15 wins, see [7]). The remaining rows present the p-value of the sign test and Wilcoxon's respectively. Both show that NeBA has the upper hand with a level of significance of 0.05.

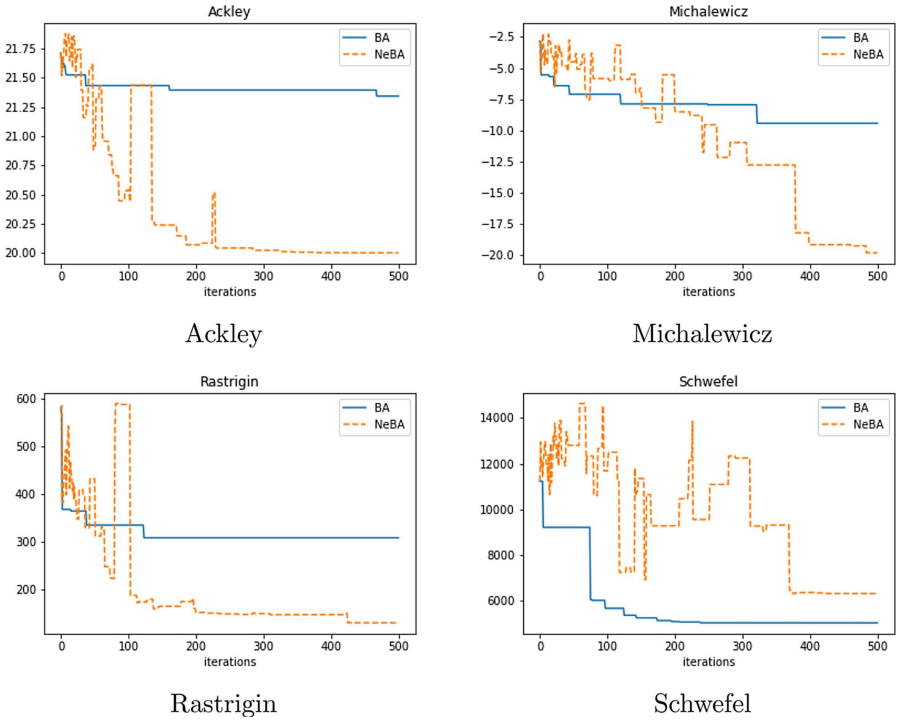


FIGURE 5. Fitness evolution of a random bat by BA and NeBA for F1, F7, F9, F13.

TABLE 5. Pairwise comparison – first experiment.

NeBA *vs.*	BA
Wins/Losses	18/02
Sign test	4.02E−04
Wilcoxon test	1.69E−03

5.2. Second experiment: impact of each modification

In order to analyze the impact of each modification on the enhancement of the BA, we compare the standard BA with BA where we integrated each modification alone. The parameters settings are the same as used in the first experiment and the comparison algorithms are described as follows:

- BA: the standard bat algorithm.
- NeBA1: the Bat algorithm where we integrated the first modification.
- NeBA2: the Bat algorithm where we integrated the second modification.
- NeBA1+2: the Bat algorithm where we integrated the first and second modifications.
- NeBA3: the Bat algorithm where we integrated the third modification.
- NeBA4: the Bat algorithm where we integrated the fourth modification.
- NeBA: the new proposed BA with all the modifications.

Table 6 compares the mean value obtained over 10 runs, of BA with the other algorithms. Overall, from the results obtained, we can see that NeBA combining all the modifications gives better results in 14 out of the 20 test functions, while the others were better improved by one of the modifications rather than the combination.

TABLE 6. Mean values – second experiment.

	BA	NeBA1	NeBA2	NeBA3	NeBA4	NeBA1+2	NeBA
F1	2.11E+01	2.11E+01	2.00E+01	2.05E+01	2.00E+01	2.00E+01	**2.00E+01**
F2	1.14E+01	1.55E+01	4.50E+00	8.26E+00	**4.50E+00**	4.51E+00	4.50E+00
F3	1.53E+05	1.48E+05	8.91E+03	6.53E+03	1.35E+03	2.48E+04	**9.20E+01**
F4	6.35E+00	7.96E+00	1.22E+00	1.13E+00	8.37E−01	1.72E+00	**7.23E−01**
F5	2.67E+02	2.55E+02	2.64E+02	**8.88E−03**	3.44E+02	2.01E+02	1.86E+00
F6	3.08E+00	3.66E+00	4.15E+00	2.77E+00	5.17E+00	5.93E+00	**1.99E+00**
F7	−9.74E+00	−9.73E+00	−1.30E+01	−1.48E+01	−1.05E+01	−1.12E+01	**−1.82E+01**
F8	1.25E+01	1.46E+01	3.05E+00	5.43E−01	7.84E−01	3.91E+00	**3.07E−02**
F9	2.78E+02	2.75E+02	1.16E+02	1.07E+02	1.25E+02	1.18E+02	**8.90E+01**
F10	6.89E+01	9.76E+01	4.53E+01	7.05E+01	6.20E+01	**3.95E+01**	4.46E+01
F11	1.75E+00	2.05E+00	1.63E+02	1.35E−01	2.89E+02	5.20E+00	**4.43E−03**
F12	2.53E−02	2.55E−02	5.45E+00	7.48E−03	3.90E+00	4.01E−01	**9.97E−04**
F13	5.44E+03	**5.02E+03**	6.14E+03	5.61E+03	5.65E+03	5.28E+03	5.28E+03
F14	7.62E+00	7.45E+00	7.55E+00	7.05E+00	7.66E+00	7.38E+00	**6.43E+00**
F15	8.06E−02	7.97E−02	1.88E+03	4.00E−03	1.70E+03	8.65E+01	**9.99E−05**
F16	**−1.01E+03**	−1.00E+03	−9.99E+02	−1.01E+03	−9.90E+02	−1.00E+03	−9.93E+02
F17	6.19E+14	1.48E+16	2.05E+20	**2.09E+02**	1.76E+28	1.32E+17	5.14E+07
F18	−1.60E+07	−1.59E+07	−1.60E+07	−1.91E+07	−1.63E+07	−1.66E+07	**−1.97E+07**
F19	4.95E+00	5.51E+00	1.78E−01	9.23E−02	6.99E−02	4.49E−01	**1.59E−03**
F20	−1.29E+01	−1.33E+01	−3.38E+01	−2.43E+01	−2.81E+01	−2.68E+01	**−3.82E+01**

TABLE 7. Wilcoxon test – second experiment.

vs. BA	NeBA1	NeBA2	NeBA3	NeBA4	NeBA1+2	NeBA
Wins/Losses	07/13	**12/08**	**17/03**	**11/09**	**14/06**	**19/01**
Wilcoxon test	2.94E−01	9.85E−01	**3.65E−03**	9.56E−01	**8.97E−02**	**1.34E−04**

TABLE 8. Average Rankings of the algorithms – Friedman second experiment.

Algorithm	Ranking
BA	5.075
NeBA1	5.25
NeBA2	4.5
NeBA1+2	4.074999999999999
Neba4	4.675
NeBA3	2.8250000000000006
NeBA	1.6000000000000005
p-value	4.085E−8

The Wilcoxon test and Friedman tests were conducted between BA and each algorithm to show the influence of each modification. Results are represented in Tables 7 and 8.

As we can see from the Table 7, adding the second, third, and fourth modifications has given better results than BA in the majority of functions. As for the first modification, BA still gives better results than NeBA1 in more functions. This can be explained by the fact that the purpose of the first modification is mainly to increase the algorithm's diversification and produce different quality solutions. But the diversification of the

BA algorithm is also highly dependent on the selection phase, where the new solutions are accepted with a probability only if they have better fitness than the old ones. This had the authors combine the first and the second modifications (NeBA1+2). From the table, we see that this combination gives a better impact and we can see from Friedman's ranking in Table 8 that it comes in third place. On the other hand, NeBA3 is the one that improved most of the functions; 17 out of 20 test functions; ranking second in the Friedman's test. This means that the third modification introduced in BA has a high impact on the performance of the algorithm. Adding the dynamically decreasing inertia weight helps balance between local and global search ability of the algorithm, by allowing a strong global search at early iterations that switches gradually to a strong local search in the final iterations. Lastly combining the total modifications has enhanced the performance significantly and gives the best result in all functions excluding F16, where BA still has the upper hand, in addition to F2, F5, F10, F13, and F17, where NeBA still improves the output in comparison with BA, but the best result is given by NeBA4, NeBA3, NeBA1+2, NeBA1 and NeBA3 respectively.

As we can see from the Table 7, adding the second, third, and fourth modifications has given better results than BA in the majority of functions. As for the first modification, BA still gives better results than NeBA1 in more functions. This can be explained by the fact that the purpose of the first modification is mainly to increase the algorithm's diversification and produce different quality solutions. But the diversification of the BA algorithm is also highly dependent on the selection phase, where in fact the new solutions are accepted with a probability only if they have better fitness than the old ones. This had the authors combine the first and the second modifications as (NeBA1+2) to access this statement.

To effectively conduct a multiple comparison of the algorithms, the Friedman ranks as has been presented in the tutorial on the use of non-parametric tests in [7], are used. The rankings are presented in Table 8 and were computed using the CONTROLTEST java package proposed by Derrac *et al.* [7]. Among all the algorithms, NeBA has the lowest rank, which means it performs better than the other ones. The combined NeBA1+2 comes in third place and from the Table 6, we clearly see that this combination has a better impact on the performance of the algorithm than only one of the modifications. On the other hand, NeBA3 is the one that improved most of the functions; 17 out of 20 test functions; ranking second in the Friedman's test. This means that the third modification introduced in BA has a high impact on the performance of the algorithm. Adding the dynamically decreasing inertia weight helps balance between local and global search ability of the algorithm, by allowing a strong global search at early iterations that switches gradually to a strong local search in the final iterations. Lastly combining the total modifications has enhanced the performance significantly and gives the best result in all functions excluding F16, where BA still has the upper hand, in addition to F2, F5, F10, F13, and F17, where NeBA still improves the output in comparison with BA, but the best result is given by NeBA4, NeBA3, NeBA1+2, NeBA1 and NeBA3 respectively.

5.3. Third experiment: NeBA *vs.* other algorithms

The third experiment consists of a comparison of NeBA with famous algorithms from the literature. Results of dBA, PSO, HS, CS, and GA were obtained from the paper [4], and the details of the parameters set are described as follows:

- **dBA**: it's the directional bat algorithm from Chakri *et al.* [4]. The parameters setting are $r_0 = 0.1$, $r_\infty = 0.7$, $A_0 = 0.9$, $A_\infty = 0.6$, $f_{\min} = 0$ and $f_{\max} = 2$.
- **PSO**: the classical particle swarm optimization is considered [8]. The parameter settings are $c_1 = 1.5$, $c_2 = 1.2$ and the inertia coefficient w is a monotonically decreasing function from 0.9 to 0.4.
- **HS**: the considered harmony search algorithm is the standard one from Geem *et al.* [12] with the following setting BW = 0.2, HMCR = 0.95, PAR = 0.3.
- **CS**: the cuckoo search *via* Lévy flights described in Yang and Deb [35] is considered with the probability of the alien eggs discovery $p_a = 0.25$.
- **GA**: the standard genetic algorithm is considered [6] with a crossover probability = 0.95 and mutation probability = 0.05.

TABLE 9. Third experiment: classical benchmark NeBA *vs.* literature algorithms.

		BA	NeBA	dBA	PSO	HS	CS	GA
F1	Mean	2.107E+01	2.000E+01	**5.839E+00**	1.474E+01	1.540E+01	1.209E+01	5.920E+00
	Std	6.456E−02	**1.081E−03**	1.730E+00	1.235E+00	7.839E−01	1.753E+00	2.453E+00
	Min	2.084E+01	1.999E+01	3.214E+00	1.252E+01	1.338E+01	8.691E+00	**2.595E+00**
	Max	2.117E+01	2.000E+01	**8.801E+00**	1.737E+01	1.640E+01	1.750E+01	1.145E+01
F2	Mean	1.378E+01	4.642E+00	3.716E+00	2.690E+01	1.433E+01	1.452E+01	**1.305E+00**
	Std	5.525E+00	2.286E+00	4.409E+00	6.382E+00	2.381E+00	1.746E+00	**1.157E+00**
	Min	4.461E+00	8.006E−01	**3.462E−02**	1.609E+01	9.950E+00	1.112E+01	1.029E−01
	Max	2.840E+01	1.149E+01	2.046E+01	3.970E+01	1.846E+01	1.834E+01	**4.277E+00**
F3	Mean	1.739E+05	**1.122E+02**	4.926E+02	8.307E+07	8.696E+07	3.760E+08	1.803E+07
	Std	6.907E+04	**3.036E+01**	5.304E+02	3.129E+07	1.489E+07	1.192E+08	2.213E+07
	Min	4.247E+04	**4.379E+01**	4.499E+01	3.980E+07	6.110E+07	1.542E+08	1.474E+06
	Max	3.865E+05	**1.786E+02**	2.518E+03	1.557E+08	1.111E+08	6.179E+08	8.335E+07
F4	Mean	6.112E+00	**6.925E−01**	1.911E+01	3.864E+04	7.853E+04	2.611E+02	6.494E+03
	Std	4.706E+00	**5.292E−02**	2.917E+01	2.495E+04	2.813E+04	1.384E+02	9.520E+03
	Min	1.827E+00	**6.679E−01**	7.448E−01	6.914E+03	4.057E+04	1.059E+02	1.207E+01
	Max	2.473E+01	**9.110E−01**	1.044E+02	1.202E+05	1.282E+05	6.159E+02	3.631E+04
F5	Mean	2.630E+02	2.475E+00	**1.405E−01**	7.481E+01	8.040E+01	4.567E+00	1.900E+01
	Std	4.425E+01	1.549E+00	**1.481E−01**	2.717E+01	1.588E+01	9.934E−01	1.828E+01
	Min	6.947E+01	3.483E−01	**5.049E−03**	3.041E+01	4.375E+01	3.026E+00	1.080E−01
	Max	3.436E+02	6.950E+00	**5.630E−01**	1.684E+02	1.201E+02	6.797E+00	5.574E+01
F6	Mean	4.469E+00	**1.416E+00**	4.716E+00	3.979E+01	2.417E+01	5.153E+00	5.675E+00
	Std	2.249E+00	**9.934E−01**	1.826E+00	1.681E+01	5.004E+00	1.865E+00	3.920E+00
	Min	1.219E+00	**8.958E−02**	1.518E+00	2.126E+01	1.366E+01	2.414E+00	1.093E+00
	Max	1.316E+01	**5.081E+00**	9.997E+00	8.057E+01	3.540E+01	8.813E+00	1.562E+01
F7	Mean	−1.006E+01	−1.762E+01	−1.495E+01	−9.620E+00	−1.376E+01	−1.455E+01	**−2.186E+01**
	Std	**6.707E−01**	2.096E+00	3.135E+00	1.732E+00	7.115E−01	7.923E−01	1.793E+00
	Min	−1.184E+01	**−2.150E+01**	−2.094E+01	−1.307E+01	−1.477E+01	−1.673E+01	−2.473E+01
	Max	−9.042E+00	**−1.292E+01**	−1.017E+01	−7.011E+00	−1.208E+01	−1.333E+01	−1.871E+01
F8	Mean	1.070E+01	**2.808E−02**	4.898E+01	9.522E+03	3.091E+04	3.554E+02	2.093E+03
	Std	6.629E+00	**1.237E−02**	5.028E+01	7.144E+03	1.079E+04	1.311E+02	1.975E+03
	Min	2.324E+00	**4.718E−03**	1.344E+00	2.180E+03	1.269E+04	1.778E+02	7.206E+01
	Max	2.988E+01	**5.465E−02**	1.918E+02	3.698E+04	5.492E+04	7.081E+02	6.457E+03
F9	Mean	2.817E+02	8.873E+01	1.193E+02	2.599E+02	1.580E+02	1.366E+02	**5.746E+01**
	Std	2.340E+01	2.068E+01	4.023E+01	3.756E+01	1.558E+01	**1.349E+01**	1.825E+01
	Min	2.226E+02	4.380E+01	6.812E+01	1.707E+02	1.330E+02	1.129E+02	**2.994E+01**
	Max	3.319E+02	1.294E+02	2.471E+02	3.456E+02	1.845E+02	1.644E+02	**9.913E+01**
F10	Mean	8.955E+01	**3.427E+01**	1.645E+02	8.159E+04	1.597E+05	1.073E+03	5.961E+03
	Std	8.905E+01	**2.199E+01**	1.926E+02	6.481E+04	4.048E+04	3.967E+02	9.588E+03
	Min	3.449E+01	**1.726E+01**	2.911E+01	8.566E+03	8.437E+04	6.691E+02	1.048E+02
	Max	3.650E+02	**1.125E+02**	1.011E+03	2.811E+05	2.346E+05	2.290E+03	4.793E+04

Please note that as mentioned in the first experiment, we made sure to use the same common parameters to obtain NeBA results, in order to keep the fairness of the test.

Based on the results of the Table 9, we can see that globally NeBA gives better results than the other algorithms in most cases. But that's not enough to conclude that NeBA is more performant. Next, we represent nonparametric tests applied to perform a rigorous comparison among the algorithms.

To evaluate the performance of NeBA in this experiment, we use both pairwise and multiple comparison tests like it has been presented in the tutorial on the use of nonparametric tests in the paper [7]. For pairwise comparisons, just like in the first experiment, we use the sign test and Wilcoxon's. The test's p-values were computed using the tests from the SciPy library in Python, and are presented in the Table 10.

As depicted in the previous experiment, an algorithm is considered a winner if it performs better in 15 out of the 20 benchmark functions. Thus, it is clear that there is a significant difference between NeBA and PSO, HS, CS, DE, BA, and dBA. In terms of p-value, the sign test shows that NeBA is better than the other

TABLE 9. continued.

		BA	NeBA	dBA	PSO	HS	CS	GA
F11	Mean	1.797E+00	**3.247E−03**	1.461E+01	1.562E+04	5.336E+04	2.138E+03	8.130E+03
	Std	1.265E+00	**1.513E−03**	3.456E+01	7.676E+03	8.132E+03	5.493E+02	8.472E+03
	Min	3.995E−01	**1.214E−03**	1.634E−02	4.828E+03	4.124E+04	1.062E+03	8.280E+01
	Max	7.667E+00	**8.869E−03**	1.256E+02	3.416E+04	7.472E+04	3.409E+03	3.294E+04
F12	Mean	2.781E−02	**9.150E−04**	1.417E+00	1.009E+03	9.074E+02	4.591E+01	2.319E+02
	Std	1.007E−02	**1.376E−04**	4.826E−01	4.147E+02	1.325E+02	1.265E+01	1.940E+02
	Min	1.528E−02	**5.379E−04**	3.554E−01	4.302E+02	6.724E+02	2.426E+01	9.307E+00
	Max	5.688E−02	**1.298E−03**	2.357E+00	2.292E+03	1.277E+03	8.646E+01	7.373E+02
F13	Mean	5.003E+03	5.429E+03	4.357E+03	8.712E+03	**3.722E+03**	5.056E+03	4.208E+03
	Std	6.933E+02	6.685E+02	6.414E+02	5.463E+02	5.060E+02	**1.747E+02**	7.320E+02
	Min	3.218E+03	4.391E+03	2.895E+03	7.293E+03	**2.281E+03**	4.522E+03	2.736E+03
	Max	6.042E+03	7.167E+03	5.646E+03	9.480E+03	**4.624E+03**	5.426E+03	5.993E+03
F14	Mean	7.755E+00	6.443E+00	5.262E+00	6.617E+00	5.946E+00	6.187E+00	**2.064E+00**
	Std	4.502E−01	6.670E−01	7.905E−01	5.041E−01	3.786E−01	**2.475E−01**	9.230E−01
	Min	6.496E+00	4.252E+00	3.861E+00	5.735E+00	5.296E+00	5.729E+00	**7.870E−01**
	Max	8.658E+00	7.465E+00	6.766E+00	7.411E+00	6.838E+00	6.674E+00	**3.680E+00**
F15	Mean	8.540E−02	**8.641E−05**	2.256E−01	2.852E+03	9.618E+03	4.153E+02	1.678E+03
	Std	5.160E−02	**2.509E−05**	4.869E−01	1.105E+03	2.226E+03	9.518E+01	2.032E+03
	Min	2.628E−02	**4.139E−05**	1.920E−03	1.118E+03	5.919E+03	2.340E+02	5.517E+00
	Max	3.072E−01	**1.342E−04**	2.233E+00	5.626E+03	1.568E+04	6.119E+02	7.964E+03
F16	Mean	**−1.005E+03**	−1.000E+03	1.959E+02	6.776E+02	7.305E+02	3.168E+02	3.621E+02
	Std	3.584E+01	4.095E+01	3.767E+01	1.252E+02	1.644E+02	**2.533E+01**	8.306E+01
	Min	−1.074E+03	**−1.076E+03**	1.131E+02	5.202E+02	3.915E+02	2.564E+02	2.611E+02
	Max	**−9.449E+02**	−9.064E+02	2.686E+02	9.585E+02	1.137E+03	3.596E+02	6.594E+02
F17	Mean	9.511E+20	**1.721E+08**	1.363E+12	1.046E+33	3.533E+41	2.263E+21	1.049E+40
	Std	4.794E+21	**7.099E+08**	4.261E+12	3.737E+33	1.697E+42	5.976E+21	4.671E+40
	Min	3.090E+08	**7.771E+01**	1.011E+06	1.609E+20	2.573E+33	3.229E+17	7.488E+04
	Max	3.323E+22	**4.768E+09**	1.713E+13	1.724E+34	8.664E+42	2.433E+22	2.390E+41
F18	Mean	−1.668E+07	**−1.939E+07**	3.423E+04	6.204E+05	8.815E+05	4.242E+04	3.194E+05
	Std	8.081E+05	8.374E+05	2.590E+04	2.312E+05	1.932E+05	**1.118E+04**	1.916E+05
	Min	−1.821E+07	**−2.108E+07**	1.685E+03	3.078E+05	5.169E+05	2.831E+04	6.326E+03
	Max	−1.492E+07	**−1.723E+07**	9.707E+04	1.223E+06	1.329E+06	8.620E+04	7.001E+05
F19	Mean	4.345E+00	**1.447E−03**	1.515E+02	1.054E+03	4.052E+02	2.214E+02	9.884E+07
	Std	1.661E+00	**6.123E−04**	4.105E+01	2.706E+02	6.898E+01	4.094E+01	2.076E+08
	Min	1.819E+00	**5.219E−04**	7.536E+01	5.754E+02	2.960E+02	1.337E+02	1.122E+01
	Max	8.424E+00	**3.216E−03**	2.506E+02	1.616E+03	5.713E+02	3.009E+02	9.316E+08
F20	Mean	−1.278E+01	**−3.739E+01**	3.053E+01	2.885E+01	2.776E+01	2.718E+01	7.684E+00
	Std	**1.414E+00**	4.219E+00	1.668E+00	5.410E+00	2.174E+00	2.463E+00	3.387E+00
	Min	−1.695E+01	**−4.671E+01**	2.719E+01	8.767E+00	2.230E+01	2.157E+01	2.534E+00
	Max	−1.092E+01	**−2.569E+01**	3.320E+01	3.283E+01	3.144E+01	2.988E+01	1.464E+01

TABLE 10. Pairwise comparison – third experiment.

NeBA *vs.*	dBA	BA	PSO	HS	CS	GA
Wins/Losses	15/05	18/02	19/01	17/03	17/03	14/06
Sign test	**4.14E−02**	**4.02E−04**	**4.01E−05**	**2.58E−03**	**2.58E−03**	1.15E−01
Wilcoxon test	**7.30E−03**	**1.69E−03**	**5.72E−06**	**2.61E−04**	**5.86E−04**	**3.15E−03**

TABLE 11. Friedman, aligned Friedman and Quade tests results.

Algorithm	Friedman	Aligned Friedman	Quade
NeBA	2.00	48.17	1.51
BA	3.45	53.62	2.77
dBA	2.80	56.025	2.8
PSO	5.99	103.14	5.89
HS	5.69	104.75	6.09
CS	4.20	65.72	4.30
GA	3.85	62.05	4.61
Statistic	54.40	16.31	17.62
p-value	6.46E−10	1.21E−02	2.32E−14

algorithms but GA, while the Wilcoxon's shows the superiority of NeBA over all the other algorithms. For multiple comparisons, the Friedman, the aligned Friedman, and the Quade ranks are used, with NeBA as the control method. Results of the test are presented in Table 11 and were computed using the CONTROLTEST Java package proposed by Derrac *et al.* [7].

The first rows of the Table 11 represent the average ranking of the algorithms obtained for each test. Based on these rankings, an algorithm is considered better if it has a low rank. Among all algorithms, NeBA has the lowest rank for the three tests, which means that it performs better than the other algorithms.

The last two rows of the Table 11 show the test statistics and p-values. For Friedman Test and aligned Friedman Test, the statistic is distributed according to the chi-square distribution with 6 degrees of freedom, while the Quade test statistic is distributed according to the F-distribution with 6 and 114 degrees of freedom. The computed p-values of each test are low, thus meaning that the null hypothesis is rejected at a high level of significance, in other words, the tests show that there is a notable difference between the algorithms.

To explore the differences between algorithms, a set of *post-hoc* procedures were used to obtain p-values which determine the degree of rejection of the hypotheses comparing each of the chosen algorithms with the control method. In our case, the control method is our proposed algorithm NeBA.

Table 12 presents the z-values, unadjusted p-values, and the adjusted p-values computed using a set of *post-hoc* procedures of the Friedman, the aligned Friedman, and Quade tests.

The unadjusted p-values presented in the Table 12 show that for the Friedman test, NeBA is significantly more efficient than five algorithms PSO, HS, CS, GA, and BA, while the aligned Friedman test considers that meaningful differences only exist with HS and PSO. Quade test on the other hand shows that the higher differences are between NeBA and HS, PSO, GA, and CS. In that sense, the Quade test is supporting that NeBA performs better in hard problems compared with the four mentioned algorithms. However, as reported in [7], these p-values are not suitable for multiple comparisons because of the accumulation of the family error problem. To deal with this, several procedures of p-value adjustment were proposed in their work. In our work, we choose the most powerful ones, for instance, the Holland procedure, the Rom procedure, the Finner procedure, and the Li procedure. As we can see in the Table 12, the Friedman test shows that there is a meaningful improvement of NeBA over PSO, HS, CS, GA, and BA for the four *post-hoc* procedures, while the aligned Friedman test only finds the differences with HS and PSO.

The Quade test, on the other hand, confirms the improvement of NeBA over HS, PSO, and GA. This result access that although NeBA obtains better results than the other algorithms, these behave similarly or better if the problem difficulties are taken into consideration.

Another interesting measure presented in Table 13, is the contrast estimation based on medians. This measure is used for computing the real differences between two algorithms [11], assuming that the expected differences between performances of algorithms are the same across problems. Thus, the performance of algorithms is measured by the importance of the differences between them [7].

TABLE 12. Result of *post-hoc* procedures at $\alpha = 0.05$.

Test	i	Algorithm	z-value	Unadjusted p	p_{Holl}	p_{Rom}	p_{Finn}	p_{Li}
Friedman	1	PSO	5.855400	4.758E−9	2.855E−8	2.714E−8	2.855E−8	6.274E−9
	2	HS	5.416245	6.086E−8	3.043E−7	2.894E−7	1.825E−7	8.024E−8
	3	CS	3.220470	0.001279	0.005109	0.004881	0.002557	0.001684
	4	GA	2.708122	0.006766	0.020162	0.020299	0.010132	0.008842
	5	BA	2.122582	0.033788	0.066435	0.067577	0.040408	0.042650
	6	dBA	1.171080	0.241566	0.241566	0.241566	0.241566	0.241566
Aligned Friedman	1	HS	4.411046	1.028E−5	6.172E−5	5.868E−5	6.172E−5	3.125E−5
	2	PSO	4.286297	1.816E−5	9.083E−5	8.638E−5	6.172E−5	5.519E−5
	3	CS	1.368340	0.171205	0.528168	0.652988	0.313099	0.342195
	4	GA	1.081807	0.279338	0.625721	0.670890	0.388216	0.459099
	5	dBA	0.612049	0.540504	0.788864	0.670890	0.606688	0.621545
	6	BA	0.424926	0.670890	0.788864	0.670890	0.670890	0.670890
Quade	1	HS	3.386501	7.078E−4	0.004239	0.004038	0.004239	0.001093
	2	PSO	3.241869	0.001187	0.005923	0.005646	0.004239	0.001833
	3	GA	2.292943	0.021851	0.084581	0.083342	0.043225	0.032695
	4	CS	2.067176	0.038717	0.111713	0.116152	0.057510	0.056506
	5	dBA	0.948925	0.342658	0.567901	0.353531	0.395565	0.346425
	6	BA	0.927760	0.353531	0.567901	0.353531	0.395565	0.353531

TABLE 13. Contrast estimation.

	NeBA	BA	dBA	PSO	HS	CS	GA
NeBA	0.000	5.293	194.2	1075	722.5	371.8	616.7
BA	−5.293	0.000	188.9	1070	717.2	366.5	611.4
dBA	−194.2	−188.9	0.000	880.8	528.3	177.6	422.5
PSO	−1075	−1070	−880.8	0.000	−352.5	−703.2	−458.3
HS	−722.5	−717.2	−528.3	352.5	0.000	−350.7	−105.8
CS	−371.8	−366.5	−177.6	703.2	350.7	0.000	244.9
GA	−616.7	−611.4	−422.5	458.3	105.8	−244.9	0.000

Focusing our attention on the rows of the Table 13, an algorithm is considered better than another, if it obtains a positive value of contrast estimation. Besides, the larger the estimated value, the higher the differences between the algorithms. We can see from the row of NeBA, that all the values are positive, therefore it surpasses all the other algorithms.

Following these experiments, we can say that introducing this set of modifications and mainly the non-monotone mechanism into the bat algorithm enhanced its performance. As we can see through the different tests, NeBA surpasses some of the well-known algorithms in terms of solution quality and it is competitive with dBA that we consider one of the best algorithms in the field. Thus, we can conclude that in general, the proposed approach produces better and more competitive results.

5.4. Fourth experiment: CEC2005 benchmark

In the fourth experiment, the proposed algorithm was tested on the CEC2005 Benchmark suit. The functions consist of a total of 25 functions with different characteristics such as scalability, separability, and multimodality.

TABLE 14. CEC2005 benchmark.

Functions		Bound	Optimum
Unimodal functions			
SF01: Shifted Sphere Function	Separable, scalable	$[-100, 100]$	-450
SF02: Shifted Schwefel's Problem 1.2	Non-separable, scalable	$[-100, 100]$	-450
SF03: Shifted Rotated High Conditioned Elliptic Function	Rotated, non-separable, scalable	$[-100, 100]$	-450
SF04: Shifted Schwefel's Problem 1.2 with Noise in Fitness	Non-separable, scalable, noise in fitness	$[-100, 100]$	-450
SF05: Schwefel's Problem 2.6 with Global Optimum on Bounds	Non-separable, scalable, optimum on boundary	$[-100, 100]$	-310
Multimodal functions			
Basic functions			
SF06: Shifted Rosenbrock's Function	Non-separable, scalable, narrow valley from local to global optimum	$[-100, 100]$	390
SF07: Shifted Rotated Griewank's Function without Bounds	Non-separable, scalable, global optimum is outside of the initialization range	$[0, 600]$	-180
SF08: Shifted Rotated Ackley's Function with Global Optimum on Bounds	Non-separable, scalable, optimum on boundary	$[-32, 32]$	-140
SF09: Shifted Rastrigin's Function	Non-separable, scalable, numerous local optima local optima	$[-5, 5]$	-330
SF10: Shifted Rotated Rastrigin's Function	Non-separable, scalable, numerous local optima	$[-5, 5]$	-330
SF11: Shifted Rotated Weierstrass Function	Non-separable, scalable, continuous but differentiable only on a set of points	$[-0.5, 0.5]$	90
SF12: Schwefel's Problem 2.13	Non-separable, scalable	$[-\pi, \pi]$	-460
Expanded functions			
SF13: Expanded Extended Griewank's plus Rosenbrock's Function (F8F2)	Non-separable, scalable	$[-3, 1]$	-130
SF14: Expanded Rotated Extended Scaffe's F6	Non-separable, scalable	$[-100, 100]$	-300
Hybrid composition functions			
SF15: Hybrid Composition Function 1	Separable near the global optimum, scalable, numerous local optima, flat areas	$[-5, 5]$	120
SF16: Rotated Hybrid Composition Function 1	Non-separable, scalable, numerous local optima, flat areas	$[-5, 5]$	120
SF17: Rotated Hybrid Composition Function 1 with Noise in Fitness	Non-separable, scalable, numerous local optima, flat areas Gaussian noise in fitness	$[-5, 5]$	120
SF18: Rotated Hybrid Composition Function 2	Non-separable, scalable, numerous local optima, flat areas	$[-5, 5]$	10
SF19: Rotated Hybrid Composition Function 2 with a Narrow Basin for the Global Optimum	Non-separable, scalable, numerous local optima, flat areas	$[-5, 5]$	10
SF20: Rotated Hybrid Composition Function 2 with the Global Optimum on the Bounds	Non-separable, scalable, numerous local optima, flat areas, optimum on boundary	$[-5, 5]$	10
SF21: Rotated Hybrid Composition Function 3	Non-separable, scalable, numerous local optima	$[-5, 5]$	360
SF22: Rotated Hybrid Composition Function 3 with High Condition Number Matrix	Non-separable, scalable, numerous local optima, optimum on boundary	$[-5, 5]$	360
SF23: Non-Continuous Rotated Hybrid Composition Function 3	Non-separable, scalable, numerous local optima, optimum on boundary	$[-5, 5]$	360
SF24: Rotated Hybrid Composition Function 4	Non-separable, scalable, numerous local optima, flat areas	$[-5, 5]$	260
SF25: Rotated Hybrid Composition Function 4 without Bounds	Non-separable, scalable, numerous local optima, flat areas	$[2, 5]$	260

TABLE 15. Mean function error values – CEC2005 benchmark BA *vs.* NeBA.

Function	NeBA	BA
SF01	**1.69E−08**	2.93E−02
SF02	**8.56E−08**	5.13E−02
SF03	**4.77E+00**	1.06E+05
SF04	**2.13E+00**	2.20E+03
SF05	**1.03E+00**	3.96E+02
SF06	**8.11E+00**	1.93E+02
SF07	**1.34E+03**	1.46E+03
SF08	**2.01E+01**	2.04E+01
SF09	**2.03E+01**	4.11E+01
SF10	**2.33E+01**	3.33E+01
SF11	**2.35E+00**	8.93E+00
SF12	**2.86E+02**	1.78E+03
SF13	**1.15E+00**	3.38E+00
SF14	**3.47E+00**	3.72E+00
SF15	**2.86E+02**	4.27E+02
SF16	**1.50E+02**	1.68E+02
SF17	**1.54E+02**	1.82E+02
SF18	9.18E+02	**8.42E+02**
SF19	9.13E+02	**8.92E+02**
SF20	9.11E+02	**8.94E+02**
SF21	8.26E+02	**7.31E+02**
SF22	**7.92E+02**	8.05E+02
SF23	9.66E+02	**8.43E+02**
SF24	**3.08E+02**	3.34E+02
SF25	**1.78E+03**	1.83E+03

These functions were modified by applying shifts, rotation, and hybridization operations to make them more complex [29]. The functions are described in Table 14.

Tests were run 50 times, and under CEC2005 criteria, the maximum evolution number (FES) was fixed at 10 000 * Dimension. The dimension was set to 10. Please note that we used the predefined CEC2005 functions from the optproblems python package to compute the results.

We compare NeBA with the standard BA. Table 15 shows the mean of the error values of NeBA and BA on the 25 CEC2005 Benchmarks.

NeBA outperforms standard BA in the vast majority of the functions. Results verify that NeBA defeats BA on all the Unimodal functions, the basic multimodal functions, the expanded multimodal functions, and six out of the eleven hybrid composition multimodal functions. Using the Wins and Losses test as reported in Table 17, NeBA wins in 20 out of 25 functions, thus, with a level of significance of 0.05, it is considered better than BA (an algorithm is considered better if it has at least 20 wins in case of 25 test problems as mentioned in [7]).

Next, Table 16 represents the mean function error values of NeBA, and other algorithms, namely, BA, PSO algorithm, restart covariant matrix evolutionary strategy with an increasing population (IPOP-CMA-ES), the CHC Algorithm, and the directional bat algorithm [4, 7]. The IPOP-CMA-ES is the winner of the CEC2005 competition on real parameter optimization and dBA is the best performing among the other comparison algorithms. Results presented in Table 16 are the ones provided in the paper [4].

Compared with the presented algorithms, the new method performs better than PSO in 15 out of 25 functions and CHC in 10 out of 25 functions, see Table 17 of Wins and Losses. It also wins against the IPOP-CMA-ES on F4, F9, F14, F17, F24, and F25 and dBA on F3, F6, F8, and F11.

TABLE 16. Mean function error values – CEC2005 benchmark.

Function	NeBA	dBA	PSO	IPOP-CMA-ES	CHC	BA
SF01	1.69E−08	**0.00E+00**	1.23E−04	**0.00E+00**	2.46E+00	2.93E−02
SF02	8.56E−08	**0.00E+00**	2.60E−02	**0.00E+00**	1.18E+02	5.13E−02
SF03	4.77E+00	2.36E+05	5.17E+04	**0.00E+00**	2.70E+05	1.06E+05
SF04	2.13E+00	**1.22E−03**	2.49E+00	2.93E+03	9.19E+01	2.20E+03
SF05	1.03E+00	**0.00E+00**	4.10E+02	8.10E−10	2.64E+02	3.96E+02
SF06	8.11E+00	3.54E+01	7.31E+02	**0.00E+00**	1.42E+06	1.93E+02
SF07	1.34E+03	**4.31E−01**	2.68E+01	1.27E+03	1.27E+03	1.46E+03
SF08	2.01E+01	2.04E+01	2.04E+01	**2.00E+01**	2.03E+01	2.04E+01
SF09	2.03E+01	8.22E+00	1.44E+01	2.84E+01	**5.89E+00**	4.11E+01
SF10	2.33E+01	1.05E+01	1.40E+01	2.33E+01	**7.12E+00**	3.33E+01
SF11	2.35E+00	3.76E+00	5.59E+00	**1.34E+00**	1.60E+00	8.93E+00
SF12	2.86E+02	**1.89E+02**	6.36E+02	2.13E+02	7.06E+02	1.78E+03
SF13	1.15E+00	**1.05E+00**	1.50E+00	1.13E+00	8.30E+01	3.38E+00
SF14	3.47E+00	2.97E+00	3.30E+00	3.78E+00	**2.07E+00**	3.72E+00
SF15	2.86E+02	2.17E+02	3.40E+02	**1.93E+02**	2.75E+02	4.27E+02
SF16	1.50E+02	1.18E+02	1.33E+02	1.17E+02	**9.73E+01**	1.68E+02
SF17	1.54E+02	1.27E+02	1.50E+02	3.39E+02	**1.05E+02**	1.82E+02
SF18	9.18E+02	**4.47E+02**	8.51E+02	5.57E+02	8.80E+02	8.42E+02
SF19	9.13E+02	**4.50E+02**	8.50E+02	5.29E+02	8.80E+02	8.92E+02
SF20	9.11E+02	**3.95E+02**	8.51E+02	5.26E+02	8.96E+02	8.94E+02
SF21	8.26E+02	**4.14E+02**	9.14E+02	4.42E+02	8.16E+02	7.31E+02
SF22	7.92E+02	**5.89E+02**	8.07E+02	7.65E+02	7.74E+02	8.05E+02
SF23	9.66E+02	**5.60E+02**	1.03E+03	8.54E+02	1.08E+03	8.43E+02
SF24	3.08E+02	**2.00E+02**	4.12E+02	6.10E+02	2.96E+02	3.34E+02
SF25	1.78E+03	**3.18E+02**	5.10E+02	1.82E+03	1.76E+03	1.83E+03

TABLE 17. Number of wins/losses – CEC2005 benchmark.

NeBA *vs.*	dBA	PSO	IPOP-CMA-ES	CHC	BA
Win/loss	4/21	**15/10**	6/19	10/15	**20/05**

From the outcomes of this last test conducted on the CEC2005 benchmark, it is evident that an algorithm's performance can excel on one subset of test functions while falling short on another. Notably, our NeBA demonstrated impressive results in comparison with the original BA. However, finding a clear pattern that explains the results on the CEC2005 benchmark is still challenging. This brings into question, whether these results are attributed to the no free lunch theorem, which asserts that no optimization algorithm can universally outperform all others on every problem, or whether, we should explore more about other characteristics of the functions to comprehensively analyse how BA and similar algorithms behave.

It's important to stress that our main goal in making these modifications was to improve the BA algorithm's performance. By enhancing its ability to explore and exploit, we observed significant enhancements in NeBA's performance. This demonstrates the effectiveness of our efforts in enhancing the original BA algorithm.

6. Conclusion and perspectives

In this paper, we introduced a new variant of the Bat Algorithm (BA), termed NeBA. Our modifications aimed to enhance BA's performance by increasing its diversification through the incorporation of a non-monotone

mechanism based on the Metropolis criterion. Additionally, we implemented a roulette wheel selection method for updating bat positions and refined the local search procedure to improve intensification, thereby preventing premature convergence. To assess the efficacy of our approach, we conducted four distinct tests. Firstly, we compared BA and NeBA on 20 classical benchmark functions. Secondly, we conducted an analysis to evaluate the impact of each modification. Thirdly, we compared NeBA with other algorithms from the literature using the same benchmark functions. Lastly, we evaluated NeBA's performance on the CEC2005 benchmark functions. NeBA consistently demonstrated high efficiency, stability, and the ability to produce quality solutions on classical benchmarks. Notably, it outperformed the standard BA and Particle Swarm Optimization (PSO) algorithms on the CEC2005 benchmarks.

In conclusion, the results from our latest experiments prompt consideration of the No Free Lunch (NFL) theorem, which states that no optimization algorithm can outperform all others on all problems. Future research will delve deeper into understanding the influence of function characteristics on algorithm performance and explore the implications of NFL theorems in this context. Additionally, we will explore systematic algorithm tuning methods, including adaptive parameter tuning, and investigate the potential for hybridizing our algorithm with other metaheuristics.

Data Availability Statement. No new data were created or analyzed in this study.

Acknowledgements. The authors like to thank the reviewers for their valuable feedback and suggestions for improving the clarity of the work.

References

[1] N. Adil and H. Lakhbab, A discrete bat algorithm for the multi-compartment vehicle routing problem, in 2020 IEEE 6th International Conference on Optimization and Applications (ICOA). IEEE (2020) 1–5.

[2] M. Alinaghian and N. Shokouhi, Multi-depot multi-compartment vehicle routing problem, solved by a hybrid adaptive large neighborhood search. *Omega* **76** (2018) 85–99.

[3] W.H. Bangyal, A. Hameed, J. Ahmad, Nisar, K., M.R. Haque, A.A.A. Ibrahim, J.J. Rodrigues, A. Khan, D.B. Rawat and R. Etengu, New modified controlled bat algorithm for numerical optimization problem. *Comput. Mater. Continua* **70** (2022) 2241–2259.

[4] A. Chakri, R. Khelif, M. Benouaret and X.S. Yang, New directional bat algorithm for continuous optimization problems. *Expert Syst. App.* **69** (2017) 159–175.

[5] A. Chakri, X.S. Yang, R. Khelif and M. Benouaret, Reliability-based design optimization using the directional bat algorithm. *Neural Comput. App.* **30** (2018) 2381–2402.

[6] L. Davis, Handbook of Genetic Algorithms. CumInCAD (1991).

[7] J. Derrac, S. García, D. Molina and F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.* **1** (2011) 3–18

[8] R. Eberhart and J. Kennedy, A new optimizer using particle swarm theory, in MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science. IEEE (1995) 39–43.

[9] C. Gan, W. Cao, M. Wu and X. Chen, A new bat algorithm based on iterative local search and stochastic inertia weight. *Expert Syst. App.* **104** (2018) 202–212.

[10] A.H. Gandomi and X.S. Yang, Chaotic bat algorithm. *J. Comput. Sci.* **5** (2014) 224–232.

[11] S. García, A. Fernández, J. Luengo and F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power. *Inf. Sci.* **180** (2010) 2044–2064.

[12] Z.W. Geem, J.H. Kim and G. Loganathan, A new heuristic optimization algorithm: harmony search. *Simulation* **76** (2001) 60–68.

[13] F. Glover, Tabu search – part I. *ORSA J. Comput.* **1** (1989) 190–206.

[14] D. Gupta, J. Arora, U. Agrawal, A. Khanna and V.H.C. de Albuquerque, Optimized binary bat algorithm for classification of white blood cells. *Measurement* **143** (2019) 180–190.

[15] X.S. He, W.J. Ding and X.S. Yang, Bat algorithm based on simulated annealing and Gaussian perturbations. *Neural Comput. App.* **25** (2014) 459–468.

[16] J. Kennedy and R. Eberhart, Particle swarm optimization, in Proceedings of ICNN'95 – International Conference on Neural Networks. Vol. 4. IEEE (1995) 1942–1948.

[17] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi, Optimization by simulated annealing. *Science* **220** (1983) 671–680.

[18] Y. Li, X. Li, J. Liu and X. Ruan, An improved bat algorithm based on Lévy flights and adjustment factors. *Symmetry* **11** (2019) 925.

[19] M. Melanie, An Introduction to Genetic Algorithms. MIT Press (1996).

[20] X.B. Meng, X.Z. Gao, Y. Liu and H. Zhang, A novel bat algorithm with habitat selection and Doppler effect in echoes for optimization. *Expert Syst. App.* **42** (2015) 6350–6364.
[21] P. Musikapun and P. Pongcharoen, Solving multi-stage multi-machine multi-product scheduling problem using bat algorithm, in 2nd International Conference on Management and Artificial Intelligence, IPEDR. IACSIT Press, Singapore (2012) 35.
[22] E. Osaba, X.S. Yang, F. Diaz, P. Lopez-Garcia and R. Carballedo, An improved discrete bat algorithm for symmetric and asymmetric traveling salesman problems. *Eng. App. Artif. Intell.* **48** (2016) 59–71.
[23] H.T. Rauf, S. Malik, U. Shoaib, M.N. Irfan and M.I. Lali, Adaptive inertia weight bat algorithm with Sugeno-Function fuzzy search. *Appl. Soft Comput.* **90** (2020) 106159.
[24] A. Rezaee Jordehi, Chaotic bat swarm optimisation (CBSO). *Appl. Soft Comput.* **26** (2015) 523–530.
[25] S. Ropke and D. Pisinger, An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transp. Sci.* **40** (2006) 455–472.
[26] D.K. Sambariya and R. Prasad, Robust tuning of power system stabilizer for small signal stability enhancement using metaheuristic bat algorithm. *Int. J. Electr. Power Energy Syst.* **61** (2014) 229–238.
[27] S.C. Satapathy, N. Sri Madhava Raja, V. Rajinikanth, A.S. Ashour and N. Dey, Multi-level image thresholding using Otsu and chaotic bat algorithm. *Neural Comput. App.* **29** (2018) 1285–1307.
[28] Y. Shi and R. Eberhart, A modified particle swarm optimizer, in 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360). IEEE (1998) 69–73.
[29] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.P. Chen, A. Auger and S. Tiwari, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. KanGAL report 2005005 (2005).
[30] T.K. Tharakeshwar, K.N. Seetharamu and B. Durga Prasad, Multi-objective optimization using bat algorithm for shell and tube heat exchangers. *Appl. Thermal Eng.* **110** (2017) 1029–1038.
[31] A.O. Topal and O. Altun, A novel meta-heuristic algorithm: dynamic virtual bats algorithm. *Inf. Sci.* **354** (2016) 222–235.
[32] C. Wang, W. Song and P. Shen, A new bat algorithm based on a novel topology and its convergence. *J. Comput. Sci.* **66** (2023) 101931.
[33] X.S. Yang, A new metaheuristic bat-inspired algorithm, in Nature Inspired Cooperative Strategies for Optimization (NICSO 2010). Studies in Computational Intelligence, edited by J.R. González, D.A. Pelta, C. Cruz, G. Terrazas and N. Krasnogor. Springer (2010) 65–74.
[34] X.S. Yang, Bat algorithm for multi-objective optimisation. *Int. J. Bio-Inspired Comput.* **3** (2011) 267.
[35] X.S. Yang and S. Deb, Cuckoo search via Lévy flights, in 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC). IEEE (2009) 210–214.
[36] G. Yildizdan and M.K. Baykan, A novel modified bat algorithm hybridizing by differential evolution algorithm. *Expert Syst. App.* **141** (2020) 112949
[37] S. Yılmaz and E.U. Küçüksille, A new modification approach on bat algorithm for solving optimization problems. *Appl. Soft Comput.* **28** (2015) 259–275.
[38] M. Zhang, Z. Cui, Y. Chang, Y. Ren, X. Cai and H. Wang, Bat algorithm with individual local search, in Intelligence Science II. Springer International Publishing, Cham (2018) 442–451.