Abstract—Swarm intelligence algorithms have been successfully applied to intractable optimization problems. Bat algorithm is one of the latest optimization metaheuristics and research about its capabilities and possible improvements is at the early stage. This algorithm has been recently hybridized with differential evolution and improved results were demonstrated on standard benchmark functions for unconstrained optimization. In this paper, in order to further enhance the performance of this hybridized algorithm, a modified bat-inspired differential evolution algorithm is proposed. The modifications include operators for mutation and crossover and modified elitism during selection of the best solution. It also involves the introduction of a new loudness and pulse rate functions in order to establish better balance between exploration and exploitation. We used the same five standard benchmark functions to verify the proposed algorithm. Experimental results show that in almost all cases, our proposed method outperforms the hybrid bat algorithm.

Keywords—bat algorithm; swarm intelligence; metaheuristic optimization; global optimization

I. INTRODUCTION

Optimization is present and applicable in almost all aspects of human civilization. Many optimization problems in business and industry belong to the group of intractable continuous or discrete optimization problems. Different approaches have been developed for solving such optimization problems. Possible classification of these approaches can divide them into the following groups: classical methods, stochastic algorithms, population based algorithms and other approaches. Classical methods for the same origin values follow the same path and always give the same final results. Stochastic algorithms are based on the randomization and they give different final results each time, even when starting from the same initial values [1]. Population based algorithms deal with a set of solutions and try to improve them on each iteration step. Although the difference between them exists, they will find the similar suboptimal solutions each time. These algorithms can be classified into evolutionary algorithms [2], [3] and swarm intelligence based algorithms [4]. One of the most common representatives of evolutionary algorithms is differential evolution (DE). Differential evolution algorithm is a stochastic search algorithm with self-organizing tendency and it does not use the derivatives [5].

The swarm intelligence is a research branch that models the population of interacting agents. Although a centralized component that controls the behaviour of individuals does not exist in swarm intelligence systems, local interactions among individual agents give the intelligent global behaviour. The most prominent classes of the swarm-intelligence systems are: ant colony optimization (ACO), particle swarm optimization (PSO), artificial bee colony (ABC), seeker optimization algorithm (SOA), firefly algorithm (FA), cuckoo search (CS), krill herd (KH) and bat algorithm (BA).

Bat algorithm was introduced by Yang in 2010 [23]. It is a metaheuristic search algorithm inspired by the echolocation characteristic of bats [24], [25]. The primary purpose for bat's echolocation is to serve as a hunting strategy. A comprehensive review about swarm intelligence involving the bat algorithm is performed by Zhang and Wang [26] and Huang and Zhao [27]. Furthermore, Tsai proposed an evolving bat algorithm to improve the performance of standard BA with better efficiency [28]. Basically, the bat algorithm is powerful at intensification, but at times it may get trapped into some local optima, so that it may not perform diversification very well. It was the main reason for research and improvements of this standard bat algorithm. One of the improvements to the standard bat algorithm is the recently developed hybrid bat algorithm (HBA) [29]. However, by analyzing the HBA algorithm it can be seen that further improvement are possible. In this paper we propose additional modifications and tuning of the HBA that almost uniformly improve results.

The rest of the paper is organized as follows. Section 2 briefly describes global numerical optimization problems, the differential evolution (DE) algorithm, and the bat algorithm (BA). Section 3 gives a detailed description of our proposed modified bat-inspired differential evolution (MBDE)
algorithm. The experimental and comparative results of the implemented algorithm are discussed in Section 4. Finally, Section 5 is the conclusion.

II. RELATED WORK

In this section we will present a brief background theory of the optimization problems, differential evolution (DE) algorithm, and bat algorithm (BA).

A. Optimization Problems

In mathematics and computer science, the optimization problems are those in which a set of unknowns \( \{x_i, \ldots, x_n\} \) is required to be determined so that an objective function \( f \) is minimized (maximized) and the number of constraints is satisfied. In an optimization problem the equalities and inequalities which represent the limitations imposed on the decision-making problem that are required to be satisfied are called constraints. A global optimization problem can be defined as follows:

\[
\min_x f(x) \quad (1)
\]

where a point \( x^* = \{x_1^*, \ldots, x_n^*\} \) from the set \( A \) of feasible points and a search space is either a global minimum if \( f(x^*) \leq f(x) \) or maximum if \( f(x^*) \geq f(x) \) for all \( x \) that belong to the set \( A \). Hence, the goal of optimization process is to find the values of decision variables \( x \) that participate in a maximum or minimum of an objective function \( f \).

B. Differential Evolution Algorithm (DE)

Differential evolution (DE) was introduced by Storm and Price [5]. It is a stochastic search algorithm with self-organizing tendency. Thus, it is a population based, derivative-free algorithm. It uses a special recombination operator that performs a linear combination of a number on individuals (usually three) and one parent (which is subject to be replaced) to create one child. The algorithm consists of three main parts: mutation, crossover and selection.

Differential mutation randomly chooses three distinct vectors \( x_a, x_b, x_c \) at some cycle \( t \), and then generate a donor vector \( u' \) by the following scheme:

\[
u'_{i} = x'_i + F(x'_a - x'_b), \quad i = 1, \ldots, n \quad (2)
\]

where \( F \) belongs to the closed interval \([0,1]\) and denotes the differential weight that scales the rate of modification.

Differential crossover is controlled by a crossover probability \( C_r \) (\( C_r \) belongs to the interval \([0,1]\)) and an uniformly distributed random number \( r \) from the interval \([0,1]\). Crossover can be expressed as:

\[
v_{r_{i}}^{j} = \begin{cases} u'_{i}, & \text{if } (r_{i} \leq C_r) \\ x'_{i}, & \text{otherwise} \end{cases} \quad j = 1, \ldots, d \quad (3)
\]

Differential selection can be expressed as follows:

\[
x_{r_{i}}^{j+1} = \begin{cases} v_{r_{i}}^{j}, & \text{if } (f(v_{r_{i}}^{j}) \leq f(x_{i}^{j})) \\ x_{i}^{j}, & \text{otherwise} \end{cases} \quad (4)
\]

It can be seen that the selection selects the solution with the best fitness. Also, the overall search efficiency is controlled by the differential weight \( F \) and the crossover probability \( C_r \). In this paper, we use the binomial “DE/rand/1/bin” scheme. The outline of the basic DE algorithm can be described as shown in Fig 1.

![Fig. 1. Differential evolution algorithm (DE)](image)

C. Bat Algorithm (BA)

Bat algorithm is a new population based metaheuristic approach proposed by Xin-She Yang [23]. The algorithm exploits the so-called echolocation of the bats. Echolocation is typical sonar which bats use to detect prey and to avoid obstacles. It is generally known that sound pulses are transformed into a frequency which reflects from obstacles. The bats navigate by using the time delay from emission to reflection. The pulse rate can be simply determined in the range from 0 to 1, where 0 means that there is no emission and 1 means that the bat’s emitting is at their maximum. In order to transform these behaviors of bats to algorithm, Yang used three generalized rules:

- All bats use echolocation to sense distance, and they also ‘know’ the surroundings in some magical way;
- Bats fly randomly with velocity \( v \) at position \( x \) with a fixed frequency \( f_{\min} \), varying wavelength \( \lambda \) and loudness \( A_0 \) to search for prey. They can automatically adjust the wavelength of their emitted pulses and adjust the rate of pulse emission \( r \) from \([0,1]\), depending on the proximity of their target;
- Although the loudness can vary in many ways, it is assumed that the loudness varies from a positive large value \( A_0 \) to a minimum constant value \( A_{\min} \).
Based on these approximations and idealization, the basic steps of the bat algorithm (BA) can be described as shown Fig.2.

In BA algorithm, initialization of the bat population is performed randomly and each bat is defined by its locations $x^t$, velocity $v^t$, frequency $f^t$, loudness $A^t$, and the emission pulse rate $r_t$ in a $D$-dimensional search space. The new solutions $x^t$ are performed by moving virtual bats according to the following equations:

$$f^t = f_{\text{min}} + (f_{\text{max}} - f_{\text{min}})\beta$$  \hspace{1cm} (5)  
$$v^t = v^{t-1} + (x^t - x_{\text{best}})A^t$$  \hspace{1cm} (6)  
$$x^t = x^{t-1} + v^t$$  \hspace{1cm} (7)$$

where $\beta$ from the closed interval $[0,1]$ is a random vector drawn generated by a uniform distribution. Here $x^t$ is the current global best location (solution) which is located after comparing all the solutions among all the bats. Initially, each bat is randomly assigned a frequency which is drawn uniformly from the interval $[f_{\text{min}}, f_{\text{max}}]$. In our implementation, we will use $f_{\text{min}} = 0$ and $f_{\text{max}} = 2$.

```
Begin  
Step 1: Initialization. Set the generation counter $t=1$; Initialize the population of $n$ bats randomly and each bat corresponding to a potential solution to the given problem; Define loudness $A_t$, pulse frequency $Q_t$ and the initial velocities $v_i$, ($i=1,2,\ldots,N$); Set pulse rate $r_t$.
Step 2: Repeat. Generate new solutions by adjusting frequency, and updating velocities and locations/solutions Eq. [(5)–(7)]; if (rand > $r_t$) then Select a solution among the best solutions; Generate a local solution around the selected best solution; end if Generate a new solution by flying randomly; if (rand<$A_t$ & & $f(x_{\text{best}}) < f(x_{\text{current}})$) then Accept the new solutions; Increase $r_t$ and reduce $A_t$; Eq. [(9)–(10)]; end if Rank the bats and find the current best $x^*$; $t = t + 1$; Step 3: Until the termination criteria is not satisfied or $t$ < MaxGeneration  
Step 4: Post-processing the results and visualizat.  
End  
```

Fig. 2. Bat algorithm (BA)

A random walk with direct exploitation is used for the local search that modifies the current best solution according the equation:

$$x_{\text{new}} = x^* + \varepsilon^* A_i$$  \hspace{1cm} (8)$$

where $\varepsilon$ from the interval $[-1,1]$ is a scaling factor which is a random number, while $A_i=\langle A^t \rangle$ is the average loudness of all bats at this time step $t$. We can see that a scaling factor $\varepsilon$ also represents direction and intensity of random-walk. Note that the procedure for updating the velocity and position of the bats is similar to that of the PSO algorithm. Thus, the BA algorithm can be seen as a well-established balance between the standard PSO algorithm and the intensive local search controlled by the loudness and pulse rate. The local search is launched with the proximity depending on the rate $r_t$ of pulse emission for the $i$-th bat. As the loudness usually decreases once a bat has found its pray, while the rate of pulse emission increases, the loudness can be chosen as any value of convenience. Mathematically, these characteristics are captured with the following equations:

$$A_{t+1} = \alpha A_i$$  \hspace{1cm} (9)  
$$r_{t+1} = r_t^0 (1 - e^{-\gamma})$$  \hspace{1cm} (10)$$

where $\alpha$ and $\gamma$ are constants. Actually, $\alpha$ parameter plays a similar role as the cooling factor of a cooling schedule in the simulated annealing.

Hybridization of the bat algorithm (HBA) [29] improved results, however, the fine-tuning of parameters can be further improved, as well as the functions for rate pulse and loudness. The HBA algorithm makes no promises that the best solutions will not be corrupted by mutation and crossover operators. Since the improvements obtained by this algorithm are not completely satisfactory for all the benchmark test problems, we propose a new method that will give better results in terms of accuracy, convergence speed and establishing a good balance between diversification and intensification. Therefore, in this paper, we merge two approaches to produce a new modified bat-inspired differential evolution (MBDE) algorithm according to the principle of bat algorithm and differential evolution. The experimental results indicate that the proposed MBDE algorithm performs more efficiently and accurately than the hybrid bat algorithm.

III. MODIFIED BAT ALGORITHM

Based on the introduction of the DE and BA algorithms in previous section, we will describe how we merge them to get the modified bat algorithm based on differential evolution, which modifies the solutions with poor fitness in order to achieve a good balance between intensification and diversification. The mainframe of the MBA is presented in Fig. 3.

In order to increase the diversification of the BA, so as to avoid trapping into local optima, a main modification of the basic BA includes adding a differential operators as mutation and crossover in DE, with the aim of speeding up convergence of the BA and preserving the attractive characteristics of that algorithm.

The difference between the MBDE and BA is that the mutation and crossover operators are used to improve the original BA generating a new solution for each bat. Therefore, the MBDE can efficiently explore the new search space with the DE and exploit the population information with the BA. In this way, the MBDE algorithm can avoid trapping into local optima. This algorithm includes three modifications.

The first modification is that we use logarithmically equally spaced numbers from 1 to 0 for loudness, and to take instead of Eq. (10) formulas $r_{t+1}^\alpha = r_t^0 (1 - \text{pow}(-\gamma))$ for pulse rates. It will
be shown that the best results are obtained for initial pulse rates $r^i_0=0.9$, initial loudness $A_0=0.95$ and $\alpha=0.9$.

**Begin**

**Step 1: Initialization.** Set the generation counter $t=1$;
- Initialize the population of $n$ bats randomly and each bat corresponding to a potential solution to the given problem; Define loudnesses $A_i$, pulse rates $r_i$, pulse frequency $Q_i$, the initial velocities $v_i$ and $r^0_i$ ($i=1$, 2, . . . , $N$);
- Set the differential weight $F$ and crossover probability $C_r$.

**Step 2: Evaluate the fitness value for each bat determined by the objective function $f$.

**Step 3: Repeat.**
- for $i=1$:$N$ do
  - Generate new solutions $x^i(t)$ by adjusting freq. $f^i(t)$, and updating velocities $v^i(t)$ and solutions $x^i(t)$ by Eq. (5)-(7).
  - if ($\text{rand} > r^i(t)$) then
    - Select a solution among the best solutions $x^i$,
    - Generate a local solution $x^i$ around the best selected solution Eq. (8);
  - else $x^i(t)=x^i(t)_{\text{previous}}$;
  - end if
- Generate randomly integers $a$, $b$, $c$ and $j$, such that $a$, $b$, $c$ are from the interval $[0,n*\text{rand}]$, $j$, $\text{rand}$ values from $[0, d*\text{rand}]$, and $a \neq b \neq c \neq i$;
- for $j=1$:$d$ do
  - if ($\text{rand} \leq C_r$ or $j=j_0$) then //crossover
    - $x^i(j) = x^i(j) + F(x^a - x^b)$; // mutation
  - else $x^i(j) = x^i(j)$; // take components by Eq. (7)
  - end if
- end for
- Evaluate the fitness for the offsprings $x^i$, $x^i_{\text{best}}$, $x^i_{\text{best}}$;
  - Select the offspring $x^i_{\text{best}}$ with the best fitness value between the above mentioned offsprings;
  - if ($\text{rand} < A_t$ && $f(x^i_{\text{best}}) < f(x^i_{\text{best}})$)
    - Accept the new solution $x^i_{\text{best}}$;
    - Increase $r$ and reduce $A_t$;
  - end if
- Sort the bats and find the current global best $x^*$;
- end for

**Step 4: Until** the termination criteria is not satisfied or $t$-MaxGeneration.

**Step 4: Post-processing the results and visualization.**

**End**

![Fig. 3. Modified bat algorithm](image)

The second modification is to add the mutation and crossover operators in an attempt to increase diversity of the bats to improve the search efficiency and speed up the convergence to optimum. As we know, for the local search part, once a solution is selected among the current best solutions, a new solution for each bat is generated locally using random walk by Eq. (8), when a random real number $\text{rand}$ drawn from a uniform distribution is greater than pulse rate $r_i$.

If $\text{rand} \leq r_i$, we use mutation and crossover operators in the DE updating the new solution to increase diversity of the bats to improve the search efficiency, as shown in Eq. (3). Through testing benchmarks in Section IV-B, it was found that setting the parameter of differential weight $F$ to 0.75 and crossover probability $C_r$ to 0.95 produced the best results.

The third modification is the introduction of some elitism into the MBDE in order to select the best solution. In this way we prevent the best solutions from being corrupted by differential operators. Hence, the best solutions will be saved in an auxiliary vector, and even if differential operators ruins it, we can revert back to it, if needed.

IV. BENCHMARK FUNCTIONS AND EXPERIMENTAL RESULTS

**A. Benchmark Functions**

In this paper, our test suite consists of five standard benchmark test functions, as used in [29]. The test suite includes many different types of problems such as unimodal, multimodal, separable, non-separable, convex and continuous. The unimodal functions contain a single global minimum with no or single local minimum, while multimodal functions contain more than one local optimum. The separable functions of $n$ variables can be written as a sum on $n$ function of just one variable, while nonseparable functions of $n$ variables cannot be written, because their variables are not independent.

The Griewank's function is the multimodal and nonseparable function. It has the global minimum $f(x^*)=0$ located at the point $x^*=(0,0,\ldots,0)$. It can be defined as:

$$f_i(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

subject to $-600 \leq x_i \leq 600$

The Rosenbrock’s function is the unimodal and nonseparable function. It has value $f(x^*)=0$ at its global minimum which is located at $x^*=(1,1,\ldots,1)$. It can be defined as:

$$f_i(x) = \sum_{i=1}^{n} \left[100(x_{i+1}^2 - x_i^2)^2 + (x_i - 1)^2\right]$$

subject to $-15 \leq x_i \leq 15$

The Sphere function is the unimodal, continuous, convex and separable function. It has value $f(x^*)=0$ at its global minimum which is located at $x^*=(0,0,\ldots,0)$. It can be defined as:

$$f_i(x) = \sum_{i=1}^{n} x_i^2$$

subject to $-15 \leq x_i \leq 15$

The Rastrigin’s function is the multimodal and additively separable function with the addition of cosine modulation to produce many local minima. It has several local optima arranged in a regular lattice, but it only has one global
subject to $-15 \leq x_i \leq 15$

The Ackley’s function is the multimodal, continuous and nonseparable function. It is obtained by modulating an exponential function with a cosine wave of moderate amplitude. Originally, it was formulated by Ackley only for the two-dimensional case. It is presented here in a generalized, scalable version. Its topology is characterized by an almost flat (due to the dominating exponential) outer region and a central hole or peak where the modulations by the cosine wave become more and more influential. It has the global minimum $f(x^*) = 0$ located at the point $x = (0,0,...,0)$. It can be defined as:

$$f_3(x) = 20 + e^{-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2}} - e^{\frac{1}{D} \sum_{i=1}^{D} \cos(2\pi x_i)}$$

subject to $-32 \leq x_i \leq 32$

B. Experimental Results

The proposed method has been implemented in C# programming language. All tests were done on an Intel Core i7 3770 K, 3.5 GHz with 16GB of RAM. The experimental results, both for the proposed MBDE method and hybrid bat algorithm HBA, are summarized in Table I. In this article, we used the same format of table as in paper [27], in order to make the comparisons of obtained results easier.

Table I consists of seven columns in order to facilitate side by side comparison. In the first column, we show the dimensions of the benchmark test functions: 10, 20 or 30 as in [29]. In the second column, we use statistical measures: best, worst, mean and standard deviation values for comparing the obtained results between the MBDE and HBA. In the remaining five columns the results of the MBDE and HBA algorithms are represented for five benchmark functions denoted as $f_1, f_2, f_3, f_4,$ and $f_5$.

In all experiments for each algorithm the same size of population of 20 bats was used and the algorithms were repeated for 25 runs.

From Table I it can be seen that the obtained results are evidently much better for our proposed method compared to the HBA in terms of search precision and robustness. We can note that our proposed method gave the best results for multimodal test functions such as $f_1$ and $f_2$. In the case when the dimensionality of the test functions is equal 10, the MBDE is much superior to the HBA. The reason for obtaining such good results lies in the fine-tuning the control parameters, both in the DE and BA algorithm. Furthermore, these results are obtained as a result of modified functions of pulse rate and loudness. One of the main reasons for achieving high precision, especially for the mentioned functions $f_1$ and $f_2$, is adjusted elitism that keeps the best solutions from being corrupted by the local search or by the operator of mutation and crossover. Introducing such elitism established a good balance between exploration and exploitation.

Table I shows that the mean of the HBA algorithm is only once better than the mean of the MBDE algorithm. In all other cases MBDE outperformed the HBA. It can be seen in Table I, that the best results for the MBDE are obtained by optimizing the functions $f_1$ and $f_2$ with the dimension $D=10$, while the worst results are obtained by optimizing the function $f_3$ with the dimension $D=30$. If we compare the results of the MBDE and HBA algorithms, we can conclude that both algorithms performance is very similar for problem $f_3$ considering the dimensions $D=20$ and $D=30$.

V. Conclusion

In this paper we presented improved hybridization of the basic bat algorithm with the well-known DE algorithm. Bat algorithm has been previously successfully hybridized with differential evolution algorithm (HBA) [29]. Our proposed modified bat-inspired differential evolution (MBDE) algorithm has been implemented and tested on the set of five standard benchmark functions. From the analysis of the experimental results we concluded that our proposed MBDE algorithm generates better quality solutions on most multimodal and unimodal benchmark problems compared to the HBA algorithm and it also improves convergence rate without losing the robustness of the basic bat algorithm. Therefore, the proposed MBDE algorithm modifications, namely operators for mutation and crossover, modified elitism during selection of the best solution and new loudness and pulse rate functions improve the performance of the HBA algorithm and show that bat algorithm as a recently developed algorithm has potential for further improvements.

Acknowledgment

This research was supported by Ministry of Education, Science and Technological Development of the Republic of Serbia, Grant No. III-44006.
REFERENCES


