International Journal of Computer Applications (0975 - 8887)
Volume 122 - No. 12, July 2015

Micro Bat Algorithm for High Dimensional Optimization Problems

Ali Osman Topal
Epoka University
Tiran, Albania

Oguz Altun
Yildiz Technical University
Istanbul, Turkey

Yunus Emre Yildiz
Epoka University
Tiran, Albania

ABSTRACT

Very recently bat inspired algorithms have gained increasing attention as a powerful technique for solving optimization problems. Bat algorithm (BA) is the first one in this group. It is based on the echolocation behavior of bats. BA is very good at exploitation however it is generally poor at exploration. Dynamic Virtual Bats Algorithm (DVBA) is another bat inspired algorithm, which is proposed lately. Although the algorithm is fundamentally inspired from BA, it is conceptually very different. DVBA employs just two bats and uses role based search mechanism. It is very efficient in exploration but relatively poor in exploitation, when it comes to high dimensional problems. In this paper, a novel micro-bat algorithm (μBA) is proposed which possess the advantages of both algorithms. μBA employs a very small population compared to its classical version. It combines the swarming technique of bats in Bat Algorithm with the role based search in Dynamic Virtual Bats Algorithm. Our empirical results demonstrate that the proposed μBA achieves a good balance between exploration and exploitation. And it exhibits a better overall performance than the standard BA with larger and smaller populations on high dimensional problems.

Keywords

Micro Bat Algorithm, Dynamic Virtual Bat Algorithm, nature-inspired algorithms, metaheuristics, optimization

1. INTRODUCTION

Many nature-inspired algorithms such as Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Genetic Algorithm (GA), and Artificial Bee Colony (ABC) have been intensively studied and applied to various optimization problems. In recent years, a new and rapidly growing nature-inspired algorithm - Bat Algorithm (BA), which is inspired by the behavior of bats searching for food by echolocation, attracts more and more attention due to its simplicity and effectiveness [1]. BA has been applied to many kinds of real world optimization problems, such as combined economic loan and emission [2], design of skeletal structures [3], grey economics dynamic system [4], parameter estimation in nonlinear dynamic biological systems [5], and speed reducer design [6]. Although BA has outstanding and encouraging performance in exploitation, it needs improvements in exploration. The exploration and exploitation trade off problem is a real challenge for researchers. To handle this problem efficiently, many strategies have been developed; one of these strategies is the use of reduced populations and another one is the hybridization of algorithms.

Micro-Genetic algorithms (Micro-GA) [7], Micro- Particle Swarm Optimization (μPSO) [8], Micro-Bacterial Foraging Algorithm (Micro-BFA) [9], and Micro-Differential Evolution (Micro-DE) [10] are examples of reduced population solutions. However, this solution can render algorithms to search stagnation which is the main-drawback of micro-approaches. Small population results in rapid premature convergence on multimodal high dimensional problems. This deficiency is addressed by different particle distribution techniques on search space and combined with multiple restarts that prevent convergence to the same solution.

In 2014, another bat inspired algorithm DVBA [11] is proposed. DVBA is inspired by bat's ability to manipulate frequency and wavelength of sound waves emitted during their hunt, hence a successor of the Bat Algorithm. It differs from the Bat Algorithm in terms of populations. In DVBA, a role based search is developed by using two bats: explorer and exploiter bat. So it can be said that, it is a kind of micro algorithm. Although DVBA is a micro algorithm, it handles the exploration effectively by its dynamic role exchange characteristic between the bats. While the exploiter bat makes intensive search around the best found solution, the explorer bat explores the search space.

In this work we propose a micro Bat Algorithm with a Dynamic Virtual Bats Algorithm named micro-BA. μBA uses different techniques from other micro algorithms to handle the search stagnation problem. In μBA, the exploitation ability of BA and the exploration ability of DVBA are combined to improve the accuracy and the precision of both algorithms. The performance of the proposed μBA is extensively evaluated on a suite of 10 high dimensional optimization problems and compared favorably with DVBA and BA with different population sizes.

The rest of this paper is organized as follows. Section 2 summarizes BA. The DVBA is overviewed in Section 3. The proposed μBA is presented in Section 4. Section 5 presents simulation results on the use of μBA for solving high dimensional problems. Finally, conclusions are drawn in Section 6.

2. BAT ALGORITHM OVERVIEW

Bat algorithm was introduced by Yang in 2010 [1]. It is a population based algorithm which uses bat's echolocation ability to get optimum solution for tough optimization problems. Echolocation is a typical sonar system which bats use to detect prey, avoid obstacles, and locate bats' roosting crevices in the dark. Bat emits

sound pulses and listens to the returning echoes by using the delay time, loudness of the response and the time difference between its ears, it can tell the shape, size, and the velocity of the prey. Bat has also the ability to change the way it emits the sound pulses. If it emits the sound pulses with high frequency, they will not travel longer but give detailed information about its close environment which helps bat to detect the prey position precisely. When bat emits sound pulses with low frequency, they will travel farther, and give rough information about its surroundings. Another feature of bat's echolocation is the loudness; when bat close to prey, it emits sound pulses very quietly but during the exploration it is very loud [12] [13]. Bats hunting strategies can be idealized or approximated as follows:

(1) Bats can detect the distance between the prey and the obstacles by using echolocation.

(2) During the search for prey, bats fly randomly with velocity V_i, with fixed sound pulse frequency f_i, varying wavelength λ, and loudness A_0.

(3) Loudness can change from large value A_0 to minimum constant value A_{min} . Beside these rules bat algorithm also assumes that the frequency f varies in a range $[f_{min}, f_{max}]$.

The algorithm starts by placing n bats randomly in the search space. Velocity V_i, frequency f_i, pulse rate r_i, and loudness A_i are initialized for each bat at the beginning. In each iteration of the main loop, by using Eq. (1, 2, and 3) bat's position and velocity are updated.

$$f_i = f_{min} + (f_{max} - f_{min})\beta, \tag{1}$$

$$V_i^t = V_i^{t-1} + (x_i^t - x_*)f_i, \tag{2}$$

$$x_i^t = x_i^{t-1} + V_i^t, \tag{3}$$

where $\beta \in [0, 1]$.
Then the algorithm evaluates the finesses (solutions) and chooses the current best position x_*. After these updates, if the bat's pulse rate r_i is low (it means the bat is far away from the prey), with a high probability $(rand() > r_i)$ it will fly near to the current best bat and make a random short fly there. If its pulse rate is high then it should be near the prey and it will just make a random fly around its current position. New position x_{new} is obtained by Equation (4).

$$x_{new} = x_{old} + \epsilon A^t, \tag{4}$$

where $\epsilon \in [-1, 1]$ a random number, $A^t = \frac{1}{n}\sum_{i=1}^{n} A_i^t$ is the average loudness of all the bats at this time step, and n is the number of bats. After the random fly, if the bats position is better than the current global best and its loudness is loud enough to be greater than a random number $(rand() < A_i)$, the bat will fly to this position and current global best will be updated with the new one. The bats pulse rate r_i will be increased and loudness A_i will be decreased as shown in Eq. (5). Then again bats will be evaluated and the current best x_* will be updated.

$$A_i^{t+1} = \alpha A_i^t, \quad r_i^{t+1} = r_i^0[1 - exp(-\gamma t)], \tag{5}$$

A pseudo-code of the bat algorithm is given in Algorithm 1.

Algorithm 1 BAT algorithm pseudo code

```
Objective function f(x), x = (x_1, ..., x_d)^T
Initialize the bat population x_i, v_i (i = 1, 2, ..., n)
Define pulse frequency (f_i) at x_i
Initialize pulse rates r_i and the loudness A_i
while (t < Max number of iterations) do
    Generate new solutions by adjusting frequency,
    and updating velocities and locations/solutions
    [equations (1) to (3)]
    if (rand > r_i) then
        Select a solution among the best solutions
        Generated a local solution around the
        selected best solution
    end if
    Generate a new solution by flying randomly
    if (rand < A_i & f(x_i) < f(x_*)) then
        Accept the new solutions
        Increase r_i and reduce A_i
    end if
    Rank the bats and find the current best x_*
end while
Post process results and visualization
```

3. DYNAMIC VIRTUAL BAT ALGORITHM (DVBA) OVERVIEW

Dynamic Virtual Bats Algorithm (DVBA) [11] is a recently introduced optimization algorithm which imitates the bats echolocation behavior in nature. Bat emits sound pulses with long wavelength λ and low frequency f during the search for prey. So bat can scan a large area. When it detects prey, it emits the sound pulses with short wavelength and high frequency that helps bat to locate the prey precisely. In DVBA, two bats are used to imitate this hunting behavior. Each bat has its own role in the algorithm and during the search they exchange these roles according to their positions. These bats are referred as explorer bat and exploiter bat. The bat that is in a better position becomes the exploiter meanwhile the other becomes the explorer. While the exploiter bat increases the intensification of the search around the best solution, the explorer bat will continue to explore other solutions.
In Fig. 1 and Fig. 2, the hunting strategy of a bat is simulated. Here, the black triangle represents the bat and the plus represents the prey. The black dots are the positions on the waves which are going to be scrutinized for a better solution. During the search for prey, the explorer bat's search scope gets in its widest shape; the distance between the search points and the angle between the wave vectors gets larger (Fig. 1), and if it detects prey, search scope gets smaller gradually. On the contrary, if the bat becomes the exploiter bat, its search scope gets in its narrowest shape (Fig. 2) so it can locate the prey precisely.
An overview of the Dynamic Virtual Bats Algorithm is given below.

3.1 Initialization of bat population

The bat population **X** = $(x_{i,j})$ is generated as in Equation (6):

$$x_{i,j} = x_{minj} + rand(0, 1)(x_{maxj} - x_{minj}) \tag{6}$$

where $i = 1, 2$ denotes the bats and, $j = 1, 2, ..., d$ denotes search space dimensions. x_{minj} and x_{maxj} are the lower and upper bounds of the search space for the dimension j, respectively.

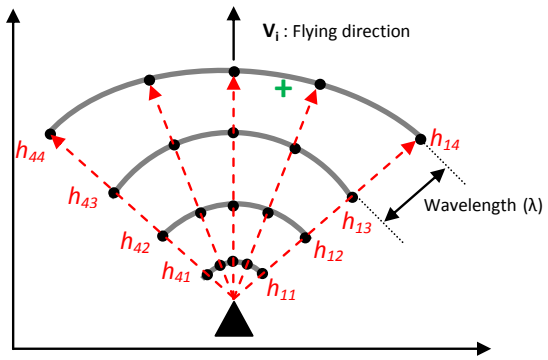

Fig. 1. Exploration: Explorer bat is searching for prey with a wide search scope.

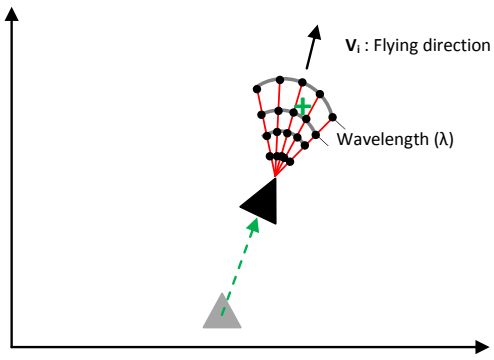

Fig. 2. Exploitation: Exploiter bat is chasing prey with a narrow search space.

DVBA uses only 2 bats, so i takes only values 1 and 2. The random number generator, rand(0,1), returns uniformly distributed random number from within the range [0, 1]. Also for each bat a flying direction $\mathbf{V}_i$ is generated randomly. Initially, each bat starts with a default wavelength λ_i and frequency f_i like in Fig. 1 to start the search as the explorer bats.

3.2 Search scope of the bats

In Fig. 1, the distribution of the search points are shown in details. The search scope is simulated by using wave direction vectors V_{ir} and search points h_{rk} on the vectors. Here, r denotes the number of wave vectors and k is the number of search points on a wave vector. So rk gives us the total number of search points on a search scope. The wideness of the search scope is inversely proportional with the frequency of the sound pulses f_i. The distance between search points are equal to the wavelength λ_i of the sound pulses.

3.3 Deciding the roles of the bats

Both bats are looking for a better position (h^*) in their search scopes as explorer bats at the beginning. If there is a better position than its current position, the bat will fly to it. After that, the positions of the bats will be compared to decide the roles. The bat which has a better solution will become the exploiter bat (2) and the other one will be the explorer bat (1). The roles will be exchanged according to this rule till termination condition is reached.

3.4 Changing the size of the bat's search scope

If a bat becomes the explorer bat, its search scope will be expanded by increasing the wavelength and decreasing the frequency Eq.78. This will increase the space between the search points h_{rk} and expand the search scope of the bat as in Fig. 1. For the exploiter bat, the search scope size will be shrunk to increase the exploitation by decreasing wavelength and increasing frequency as shown in Fig. 2.

$$\lambda_i^{t+1} = \lambda_i^t \pm \rho \tag{7}$$

$$f_i^{t+1} = f_i^t \pm \rho \tag{8}$$

$$\rho = mean(\frac{x_{max} - x_{min}}{\beta}), \qquad \{\beta \in \Re : \beta > 0\}, \tag{9}$$

where β is positive real constant which is used as increment rate divisor in [11].

3.5 Position and direction update

As long as, a bat has better solution in its search scope, it will keep updating its position with the better one. Unless there is no better solution than its current position in its search scope, the bat will turn around randomly and keep scanning its nearby surrounding space. It will keep spinning in this position and expanding its search scope until it finds a better solution.

3.6 Pseudo code of the Dynamic Virtual Bats Algorithm

Based on the virtual bat's behavior, the basic steps of the algorithm for minimizing an objective function $f(x)$ are shown in Algorithm 2.

4. MICRO BAT ALGORITHM

μBA is developed using the ideas from the Bat Algorithm and Dynamic Virtual Bats Algorithm. In this combination, the weaknesses of the algorithms are avoided and the advantages are used. The weaknesses and the advantages of the BA and the DVBA can be summarized as follows.
In Bat Algorithm, the bats repeat three main steps as the iterations proceed.

(1) Bats move towards the best found position.
(2) Bats, with a probability of $rand() < r$, fly near to the best position.
(3) Bats fly randomly either near to the best bat or any position in the search space.

Here, r is the sound impulse rate and it increases exponentially as the iterations proceeds (Eq 5). It is clear that, the possibility of flying near to the best position will increase rapidly for each bat after some iterations [14]. In another word, Bat Algorithm loses its exploration capability rapidly and increases its exploitation capability at the following iterations. That can cause Bat Algorithm to converge prematurely [15].
In DVBA, there are two roles which are exchanged between the bats according to their positions during the search. This dynamic

Algorithm 2 DVBA pseudo code. f_{gbest} is the global best solution and d is the number of dimensions.[11]

```
Objective function f(x), x = (x_1, ..., x_d)^T
Initialize the bat population x_i (i = 1, 2) and v_i
Initialize wavelength λ_i and frequency f_i
Initialize the number of the waves
while (t < Max number of iterations) do
    for each bat do
        Create a sound waves scope
        Evaluate the solutions on the waves
        Choose the best solution on the waves, h_*
    if (f(h_*) < f(x_i)) then
        Move to new solution
        Decrease λ_i and increase f_i78
    else if (f(x_i) > f_gbest) then
        Change the direction randomly
        Increase λ_i and decrease f_i
    else if (f(x_i) = f_gbest) then
        Minimize λ_i and maximize f_i
        Change the direction randomly
    end if
    Rank the bats and find the current best x_gbest
end while
```

role exchange gives DVBA higher diversity capability but slower convergence [11] [16]. The size of the bats' search scope has a major affect on DVBA's performance. The search scope size is limited by the wavelength which might not be long enough to detect better solutions near its surrounding space. Thus, it is very likely that the explorer bat will be trapped in local optima. Additionally, the exploiter bats search scope can become very small during the exploitation process and it will move very slowly. Therefore it might not reach the global optima within the bounded computation time.
In the μBA, the position of the prey represents a possible solution to the optimization problems. To discover the prey, three bats are employed which are referred as explorer bat, exploiter bat, and scout bat. The explorer and the exploiter bats show the same characteristics as in DVBA. However, they don't exchange the roles during the search and the explorer bat helps the exploiter bat to speed up the exploitation. The scout bat was added to increase search diversity. The μBA combined the BA's fast convergence characteristic with DVBA's exploration and dynamic search capabilities. The behavior of the virtual bats and the outline of the μBA are given below.

4.1 The explorer bat

The explorer bat emits the sound pulses with low frequency and long wavelength so it can scan a large area (Fig. 1). The explorer bat checks the solutions on its search scope and flies to the best solution. Unless there is no better solution than its current position, the explorer bat will turn around randomly and keep scanning its nearby surrounding space until it finds a better solution. So it is clear that it can be trapped easily in local optima like in DVBA. To avoid this local optima trap, we give the explorer bat a chance to make a random fly in the vicinity of the exploiter bat (Eq 11). That will not only help the explorer bat to escape from the trap but also increase the speed of the exploiter bat through the global optima. The similar probabilistic approach $(rand() > ri)$ from the Bat Algorithm is used to give this chance to the explorer bat. In μBA, r_i is switched by P and called the random flight probability. The probability of random flight P and the new position of the bat are calculated as follows:

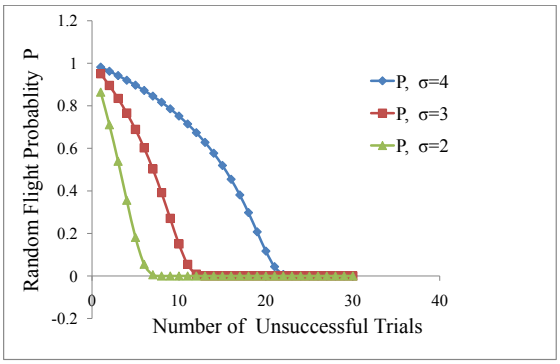

Fig. 3. The effect of σ on P as the unsuccessful attempts increases.

$$P^{t+1} = P^t[1 - exp(-trial\sigma)], \tag{10}$$

$$x_i^{t+1} = x_{gbest} + \rho, \tag{11}$$

where σ constant. *trial* denotes the number of unsuccessful attempts to escape from the local optima trap. ρ is calculated as in Eq. 9. As the explorer bat spins around the local optima trap, the P will decrease exponentially and the probability of flying near to the exploiter bat $(rand() > P)$ will increase. In Eq.10, if σ is chosen very small, the explorer bat will fly near to the exploiter bat too soon and won't able to explore its nearby surrounding space. Thus, σ should be chosen carefully. As the unsuccessful attempts (trial) increases, the effect of σ on P is shown in Fig.3.

4.2 The exploiter bat

The exploiter bat is used to increase the intensification of the search on the best found solution. It has very narrow search scope (Fig. 2) so it can make intense exploitation. If the explorer bat or the scout bat finds a better solution than the exploiter bat's current position, it flies to this solution and starts exploiting there.

4.3 The scout bat

The scout bat has a large search scope like the explorer bat (Fig. 1). However, unlike the explorer bat, the scout bat doesn't consider its current position to make the next move. It simply chooses the best position from its search scope and flies there. Same as in Simulated Annealing (SA) [17], it will even fly to worse solution than its current solution. The scout bat keeps flying all around the search space without having any local optima trap problem. That increases the diversification capability of the μBA.
In a robust search process, exploration and exploitation processes must be carried out together. In the μBA, while explorer and exploiter bat carry out the exploitation process in the search space, the scout bat control the exploration process with the explorer bat. According to all these approximations and improvements μBA can be given as in Algorithm 3.
In the next section, experimental results on high-dimensional instances of widely used optimization problems are reported.

Table 1. Description of the benchmark functions. Here D: dimensionality of the functions, C: function characteristics with values - U: unimodal, M: Multimodal, S: Separable, N: Non-Separable.

No	Name	Formula	D	C	f_{min}	Search Space
f_1	Ackley	$f_1(x) = 20 + e - 20exp(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}) - exp(\frac{1}{d}\sum_{i=1}^{d} cos(2\pi x_i))$	10, 30, 50	MN	0	$(-32, 32)^d$
f_2	Bohachevsky	$f_2(x) = \sum_{i=1}^{d-1}[x_i^2 + 2x_{t+1}^2 - 0.3cos(3\pi x_i) - 0.4cos(4\pi x_{i+1}) + 0.7]$	10, 30, 50	MN	0	$(-15, 15)^d$
f_3	Griewangk	$f_3(x) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} cos(\frac{x_i}{\sqrt{i}}) + 1$	10, 30, 50	MN	0	$(-600, 600)^d$
f_4	Rastrigin	$f_4(x) = 10d + \sum_{i=1}^{d}[x_i^2 - 10cos(2\pi x_i)]$	10, 30, 50	MS	0	$(-5.12, 5.12)^d$
f_5	Powell	$f_{16}(x) = \sum_{i=1}^{d-2}(x_{i-1}+10x_i)^2+5(x_{i+1}-x_{i+2})^2+(x_i-2x_{i+1})^4 + 10(x_{i-1} - x_{i+2})^4$	10, 30, 50	UN	0	$(-4, 5)^d$
f_6	Rosenbrock	$f_5(x) = \sum_{i=1}^{d-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	10, 30, 50	MN	0	$(-15, 15)^d$
f_7	Sphere	$f_{13}(x) = \sum_{i=1}^{d} x_i^2$	10, 30, 50	US	0	$(-5.12, 5.12)^d$
f_8	Shifted Rastrigin	$f_8(x) = f_4(z), z = x-o, o = [o_1, o_2, ..., o_d]$: shifted global optimum.	10, 30, 50	US	0	$(-5.12, 5.12)^d$
f_9	Shifted Rotated Ackley	$f_9(x) = f_1(z), z = M(x - o), o = [o_1, o_2, ..., o_d]$: shifted global optimum	10, 30, 50	UN	0	$(-32, 32)^d$
f_{10}	Shifted Rotated Griewangs	$f_{10}(x) = f_3(z), z = M(x - o), o = [o_1, o_2, ..., o_d]$: shifted global optimum	10, 30, 50	US	0	$(-600, 600)^d$

5. NUMERICAL EXPERIMENTS AND RESULTS

5.1 Parameter settings for the algorithms

In order to demonstrate the effectiveness of the μBA, a suite of 10 well-known numerical functions were tested with DVBA, BA with 30 bats, and BA with 3 bats. Each test function was considered three different dimensions, namely, $d = 10, 30$, and 50. Maximum number of function evaluation (FEs) is set to 100.000 for $10 - D$ problems, 300.000 for $30 - D$ problems, and 500.000 for $50 - D$ problems. Algorithms were tested with 30 independent runs for each test functions in order to compile comprehensive data. All the algorithms are developed in the Python environment and run on a PC with a 3.20 GHz CPU and 6.00 GB of RAM. The python codes of BA, DVBA and BA are available at "http://aotopal.epoka.edu.al/micro-BA".
The other specific parameters of algorithms are given below:

5.1.1 BA Settings. Parameters are set as follows: $\alpha = \gamma = 0.5$, frequency is in the range $[0, 2]$, the rate of pulse emission $r \in [0, 1]$ and, the loudness $A_0 = 0.5$ [1].

5.1.2 DVBA Settings. Maximum step size divisor is set to $\beta = 250$ since the search space exponentially increased in our tests. The number of search points and the wave vectors are set to $r = 6$ and $k = 5$, respectively. The range of the wavelength and the frequency are set as follows: $[\lambda_{min}, \lambda_{max}] = [\rho, 10\rho]$ and $[f_{min}, f_{max}] = [\rho, 10\rho]$, where ρ is calculated in Eq. 9. Population size is 2.

5.1.3 micro-BA Settings. Same parameters are used from DVBA to create the search scope of the bats. P is started from 1 and $\sigma = 3$.

5.2 Benchmark Functions

To evaluate the performance of the algorithms, a set of 10 standard benchmark functions is used. The benchmark set include unimodal, multimodal, separable, non-separable, shifted, and rotated optimization functions. In shifted and rotated test functions, the global optimum is shifted to a random position and the functions are rotated. By using shifted and rotated functions, we would be able to test the algorithms on more challenging, real world like problems [18]. Specifically, functions $f_1 - f_4$ are multimodal functions, $f_4 - f_7$ are unimodal functions, and $f_8 - f_{10}$ are shifted and rotated functions. We rotated the functions $f(z) = f(Mx)$, where $f(z)$ is the new function and M is an orthogonal rotation matrix. The global optimum is shifted to a random position by $f(z) = f(x - o_{new} - o_{old})$, where o_{old} is the old global optimum and o_{new} is the new global optimum which is not lying at the center of the search range [19]. The description of the benchmark functions are shown in Table 1.

5.3 Experimental results and discussion

In order to test the efficiency of the proposed algorithm, our experiment's results were compared with the standard BA with 3 bats, the standard BA with 30 bats, and the standard DVBA. The test results are shown in Table 2 and 3 in terms of the best fitness values (BFV), the worst fitness values (WFV), the mean and the standard

Table 2. Performance comparison of DVBA, BA with 3 bats, BA with 30 bats, and μBA over 10 test functions of 10, 30, and 50 dimensions.

	Function	Dim	Algorithms	BFV	WFV	Mean	STDEV	Significant
f_1	Ackley	10	DVBA	0.0216	4.5465	2.6865	0.9056	+
			BA-3bats	2.3247	4.6683	3.5937	0.6870	+
			BA-30bats	2.0193	4.0331	2.8221	**0.4931**	+
			μBA	**0.0146**	**3.0281**	**1.7313**	1.0036	-
		30	DVBA	**3.0604**	20.0798	16.6419	6.7907	+
			BA-3bats	19.9420	19.9667	19.9568	0.0113	+
			BA-30bats	19.9434	19.9566	19.9497	**0.0042**	+
			μBA	3.2660	**5.0264**	**3.8358**	0.5119	-
		50	DVBA	20.0848	20.2278	20.1788	0.0514	+
			BA-3bats	19.9160	19.9660	19.9483	**0.0171**	+
			BA-30bats	3.5297	19.9636	18.3037	4.9246	+
			μBA	**2.8373**	**5.4899**	**4.4476**	0.7227	-
f_2	Bohachevsky	10	DVBA	1.5428	4.4134	2.9063	0.7375	+
			BA-3bats	1.9253	2.9166	**2.2136**	**0.2793**	N
			BA-30bats	1.7723	**2.8279**	2.3783	0.3067	N
			μBA	**0.2317**	4.4773	2.3240	1.1738	-
		30	DVBA	14.7338	26.1569	19.7721	3.9088	+
			BA-3bats	17.4313	20.3219	18.4629	**1.0059**	N
			BA-30bats	17.5538	22.1077	19.3455	1.6834	+
			μBA	**13.3856**	**20.0175**	**17.9844**	2.4474	-
		50	DVBA	37.6436	44.4501	42.1866	2.5494	+
			BA-3bats	38.0999	44.1018	41.5410	**2.1463**	+
			BA-30bats	39.2839	44.8488	42.8278	1.9773	+
			μBA	**24.4439**	**38.2343**	**31.9908**	4.8434	-
f_3	Griewangk	10	DVBA	1.3576	2.4926	2.1272	0.3303	+
			BA-3bats	0.5478	0.9346	**0.7585**	**0.0941**	N
			BA-30bats	0.5217	**0.8826**	0.7510	0.1073	N
			μBA	**0.7075**	0.8509	0.7980	0.0434	-
		30	DVBA	10.4302	13.7109	12.9437	1.2662	+
			BA-3bats	1.3904	1.4195	1.4067	**0.0095**	+
			BA-30bats	1.3501	1.4370	1.4050	0.0296	+
			μBA	**1.0804**	**1.1049**	**1.0928**	0.0104	-
		50	DVBA	33.3351	38.2587	35.6369	1.7511	+
			BA-3bats	2.1307	2.2624	2.2165	0.0471	+
			BA-30bats	1.7346	2.1055	1.9677	0.1245	+
			μBA	**1.2277**	**1.2935**	**1.2605**	**0.0215**	-
f_4	Rastrigin	10	DVBA	17.2326	**41.9676**	30.0917	**8.6982**	N
			BA-3bats	56.1000	110.5451	85.3065	19.8476	+
			BA-30bats	18.3934	69.8718	50.2790	18.2945	+
			μBA	**15.2605**	58.9553	**27.1348**	16.3371	-
		30	DVBA	**141.0519**	243.0097	**195.9515**	33.3322	+
			BA-3bats	176.5749	293.3492	235.6283	37.8489	+
			BA-30bats	183.9583	245.1114	221.4598	20.2705	+
			μBA	183.6387	**210.9989**	198.0123	**9.9362**	-
		50	DVBA	394.2856	484.7279	444.4586	29.2374	+
			BA-3bats	370.7712	497.3180	440.2515	57.1338	+
			BA-30bats	**349.3720**	483.0212	407.0945	53.4919	+
			μBA	376.2547	**431.6168**	**406.1995**	**21.9270**	-
f_5	Powell	10	DVBA	0.0953	0.1916	0.1400	0.0253	+
			BA-3bats	0.0660	0.1715	0.1153	0.0283	+
			BA-30bats	0.0682	0.1743	0.1001	0.0301	+
			μBA	**0.0075**	**0.0162**	**0.0114**	**0.0027**	-
		30	DVBA	0.9749	1.1569	1.0879	0.0629	+
			BA-3bats	1.4853	2.1225	1.8663	0.2397	+
			BA-30bats	1.6184	2.0971	1.9913	0.1867	+
			μBA	**0.0908**	**0.1183**	**0.1032**	**0.0107**	-
		50	DVBA	2.6277	3.6723	3.1850	0.4118	+
			BA-3bats	4.9632	6.3985	5.6586	0.4875	+
			BA-30bats	4.8066	6.3320	5.5817	0.4941	+
			μBA	**0.2807**	**0.3442**	**0.3176**	**0.0239**	-

Table 3. Performance comparison of DVBA, BA with 3 bats, BA with 30 bats, and μBA over 10 test functions of 10, 30, and 50 dimensions.

	Function	Dim	Algorithms	BFV	WFV	Mean	STDEV	Significant
f_6	Rosenbrock	10	DVBA	11.3205	166.3953	29.0556	45.8787	+
			BA-3bats	10.6727	196.4292	46.6597	68.0033	+
			BA-30bats	10.5964	17.4026	13.4332	2.3198	+
			μBA	**4.1786**	**10.2598**	**8.9103**	**1.6942**	-
		30	DVBA	85.7328	1072.1944	299.3281	386.9613	+
			BA-3bats	96.5783	143.2552	126.3990	**16.9769**	+
			BA-30bats	97.8295	427.0891	238.8529	139.9443	+
			μBA	**31.3629**	**87.8815**	**43.5059**	22.2007	-
		50	DVBA	240.6517	1269.5626	579.9789	327.6391	+
			BA-3bats	298.4399	861.5234	572.4530	176.6554	+
			BA-30bats	323.3869	972.9987	450.1929	186.3294	+
			μBA	**61.4320**	**314.5447**	**165.5107**	**79.8678**	-
f_7	Sphere	10	DVBA	0.2612	**0.5413**	0.3739	0.0911	+
			BA-3bats	1.3693	3.6073	2.8133	0.6197	+
			BA-30bats	1.3021	3.0094	2.3567	0.5229	+
			μBA	**5.65e-07**	0.6860	**0.1326**	**0.2366**	-
		30	DVBA	0.0553	0.0759	0.0643	0.0077	+
			BA-3bats	0.1010	0.1320	0.1133	0.0102	+
			BA-30bats	0.1025	0.1232	0.1116	0.0085	+
			μBA	**0.0111**	**0.0135**	**0.0121**	**0.0009**	-
		50	DVBA	0.1497	0.2200	0.1832	0.0230	+
			BA-3bats	0.2781	0.3822	0.3373	0.0281	+
			BA-30bats	0.3020	0.3799	0.3408	0.0216	+
			μBA	**0.0132**	**0.0206**	**0.0175**	**0.0019**	-
f_8	Shifted Rastrigin	10	DVBA	28.9949	127.0091	72.8769	22.4583	+
			BA-3bats	46.2250	188.3732	103.7364	40.2252	+
			BA-30bats	25.0820	92.8895	57.0023	18.4316	+
			μBA	**11.0428**	**76.7472**	**39.8817**	**17.9924**	-
		30	DVBA	211.0952	498.3511	368.8925	83.9036	+
			BA-3bats	297.7370	653.7204	460.0270	90.4837	+
			BA-30bats	250.8744	469.1721	337.1287	65.2766	+
			μBA	**148.7219**	**314.5511**	**232.5197**	**55.4156**	-
		50	DVBA	606.4386	821.6168	708.6816	**62.4324**	+
			BA-3bats	692.8852	1009.7283	847.5367	110.5501	+
			BA-30bats	644.1346	837.0152	720.2708	68.0795	+
			μBA	**321.5644**	**610.9722**	**462.8984**	106.1891	-
f_9	Shifted Rotated Ackley	10	DVBA	1.7521	20.3361	7.4764	8.2766	+
			BA-3bats	2.2921	20.4352	19.3113	**3.2061**	+
			BA-30bats	1.9483	20.4179	14.8852	8.3534	+
			μBA	**0.4599**	**20.2919**	**5.3013**	7.5357	-
		30	DVBA	20.8117	21.0153	20.9279	0.0548	N
			BA-3bats	20.8161	21.0337	20.9658	0.0595	N
			BA-30bats	20.8892	21.0349	20.9639	**0.0421**	N
			μBA	**2.8299**	**20.7155**	**17.0460**	7.1056	-
		50	DVBA	21.0451	21.1607	21.1241	0.0308	+
			BA-3bats	21.1297	21.1911	21.1597	**0.0195**	+
			BA-30bats	21.0353	21.1986	21.1363	0.0448	+
			μBA	**20.7314**	**20.8755**	**20.7972**	0.0491	-
f_{10}	Shifted Rotated Griewangs	10	DVBA	2.2979	3.7364	3.0175	0.4656	+
			BA-3bats	0.8388	1.0362	0.9506	**0.0630**	+
			BA-30bats	0.8025	1.0262	0.9003	0.0850	+
			μBA	**0.4069**	**0.9148**	**0.7429**	0.1519	-
		30	DVBA	7.2322	8.7454	7.3364	1.9442	+
			BA-3bats	1.6303	1.9125	1.7939	0.0934	+
			BA-30bats	1.6556	1.8832	1.7739	0.0766	+
			μBA	**1.1605**	**1.2358**	**1.2008**	**0.0221**	-
		50	DVBA	10.8454	14.9527	11.6678	2.5123	+
			BA-3bats	3.2262	3.7766	3.4695	**0.1737**	+
			BA-30bats	3.0935	3.9139	3.4128	0.2344	+
			μBA	**1.4815**	**1.6367**	**1.5713**	0.0453	-

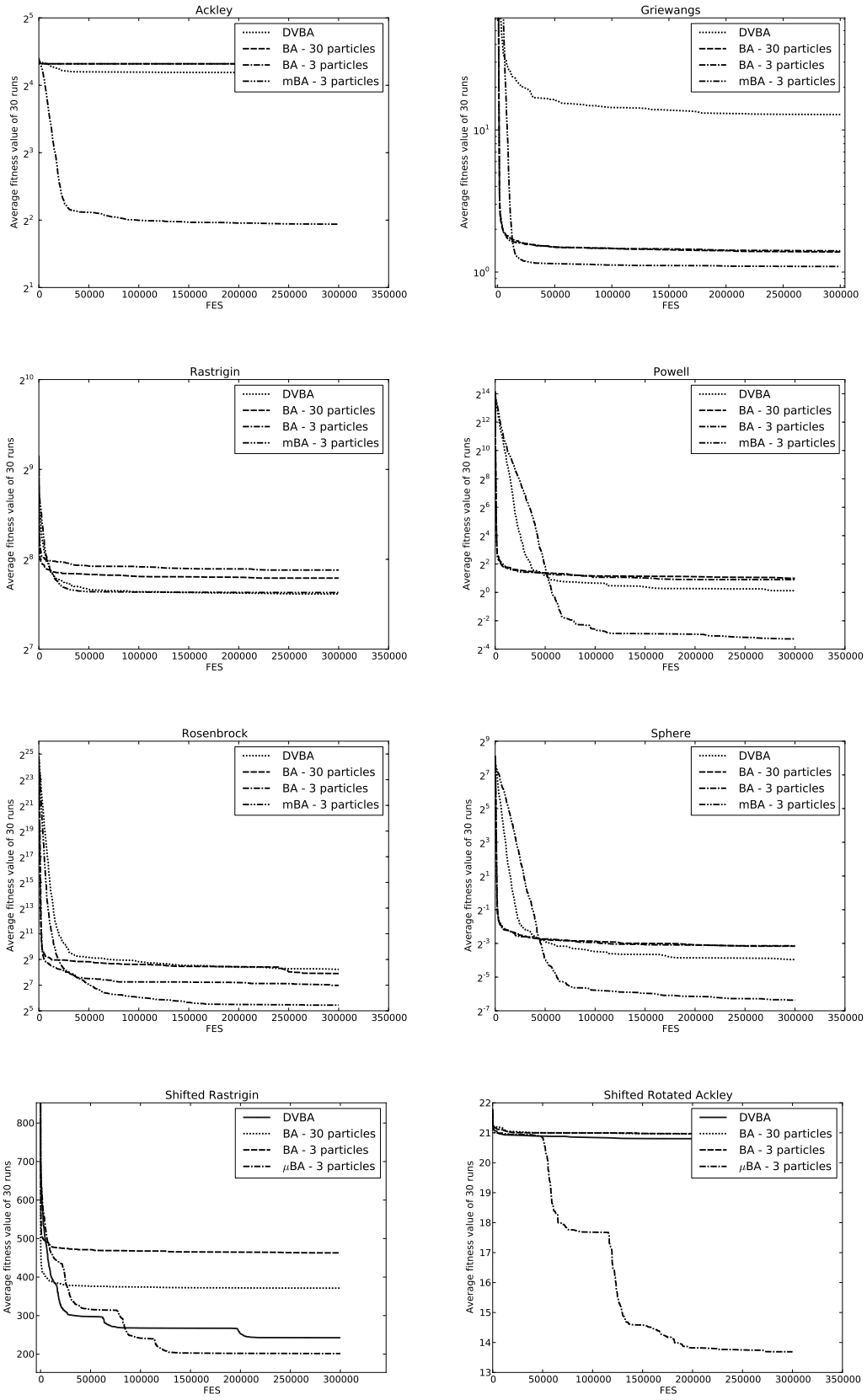

Fig. 4. Convergence characteristics of μBA, BA, and DVBA for the 30-dimensional test functions.

Algorithm 3 μBA pseudo code where x_{gbest} is the global best position and d is the number of dimensions.

```
Objective function f(x), x = (x_1, ..., x_d)^T
Initialize the bat population x_i (i = 1, 2, 3) and velocity v_i
Initialize wavelength λ_i and frequency f_i
Initialize the number of the wave vectors and search points (r, k)
while (t < Max number of iterations) do
    for each bat do
        Create a sound waves scope [11]
        Evaluate the solutions on the waves
        Choose the best solution on the waves, h_*
    if ( i = 1 ) then //Scout Bat
        Move to h_*
        Maximize λ_i and minimize f_i
    end if
    if ( i = 2 ) then //Explorer Bat
        if f(h_*) < f(x_i) then
            Fly to h_*
            Maximize λ_i and minimize f_i
            trial=0
        else
            trial=trial+1
            Change the direction randomly
            Update P by using Eq. 10
        end if
        if ( rand() > P ) then
            Produce a new solution around the exploiter bat
            by Eq. 11
            trial = 0
            P = 1
        end if
    end if
    if ( i = 3 ) then //Exploiter Bat
        Fly to x_gbest
        Minimize λ_i and maximize f_i
        Change the direction randomly
    end if
    Rank the bats and find the current best x_gbest
end while
```

deviation (STDEV) of the results found over the 30 independent runs by each algorithm. The best results are marked in bold. Furthermore, we used t-tests [17] to compare the means of the results produced by the μBA and the other algorithms at the 0.05 level of significance. In the last column of Table 2 and 3 we report the statistical significance level of the results. There $'+'$ indicates that μBA is significantly more successful than selected one at a 0.05 level of significance by two-tailed test, $'N'$ means the difference of means is not statistically significant and, $'-'$ stands for Not Applicable, covering cases for which the two algorithms achieve the same accuracy results.

Fig.4 illustrates the convergence characteristics in terms of the best fitness value of the median run of each algorithm for the test functions with $D = 30$. The convergence graphs of the $10 - D$ and $50 - D$ problems are similar to their $30 - D$ counterparts, so they are omitted here to save space.

From Table 2 and 3 it can be said that μBA gave the best results for multimodal test functions. It is known that the complexity of the problems increases as the dimensionality of the search increases and local traps become harder to escape for the algorithms. However, the μBA performed successfully for the multimodal functions f_1, f_2, and f_3 for all dimensions. For the functions f_4, the algorithms didn't show a significant success but μBA has demonstrated a better ability of global searching. For the unimodal functions, the algorithms got the best results but they performed poorly on f_8 (Rosenbrock). Rosenbrock function is grouped as a unimodal function but it may have multiple minima when the dimension increases and converging to the minimum is difficult [20]. Furthermore, the t-test values show that the performance of the μBA is significantly more efficient than other compared algorithms in terms of the mean. Similar observations can be made for the shifted and rotated functions. Overall, μBA obtains a smaller mean value than the other algorithms for all problems.

Additionally, a close inspection of Table 2 and 3 reveals that, BA doesn't sensitive to the population size. BA with 3 bats and BA with 30 bats didn't show significant difference on most of the test functions.

The convergence map of algorithms in Fig.4 shows that the μBA always converges faster than other algorithms on seven problems (f_1, f_5, f_6, f_7, f_8, f_9, and f_{10}). In Fig.4, it can be seen that, while other algorithms suffer from premature convergence problem on the function f_1, f_5, f_7, f_8, and f_9, the μBA escaped from the local optima traps successfully. For the functions f_1 and f_9, μBA outperformed the other algorithms significantly. f_1 and f_9 is characterized by a nearly flat outer region, and a large hole at the center. The function poses a risk for the algorithms to be trapped in one of its many local minima and difficulty to reach the global optima in predefined time. The explorer bat helped the exploiter bat to reach the global optima faster in these functions.

6. CONCLUSION

In this paper, we proposed a micro Bat Algorithm (μBA) to solve single objective high dimensional optimization problems. In μBA, the proposed new search mechanism combines the BA's fast convergence characteristic with DVBA's dynamic search capability to increase the search diversity while pursuing a balance between exploration and exploitation.

To prove the effectiveness and robustness of the proposed algorithms, the μBA was compared with DVBA and BA on variety of optimization problems with different complexities. The results demonstrated that the solution values achieved by μBA are several orders of magnitude better than those of BA and DVBA in many cases. Also, μBA appears to be less effected when dimension of the problem increases significantly. Therefore, it can be said that μBA achieves a good balance between exploration and exploitation and has the best universality on different type of problems.

7. REFERENCES

[1] Xin-She Yang. A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization (NICSO 2010)*, pages 65–74. Springer, 2010.

[2] Bandi Ramesh, V Chandra Jagan Mohan, and VC Veera Reddy. Application of bat algorithm for combined economic load and emission dispatch. *Int. J. of Electricl Engineering and Telecommunications*, 2(1):1–9, 2013.

[3] A Kaveh and P Zakian. Enhanced bat algorithm for optimal design of skeletal structures. *Asian J Civial Eng*, 15(2):179–212, 2014.

[4] Mo Yuanbin, Zhao Xinquan, and Xiang Shujian. Local memory search bat algorithm for grey economic dynamic system. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, 11(9):4925–4934, 2013.

[5] Jiann-Horng Lin, Chao-Wei Chou, Chorng-Horng Yang, Hsien-Leing Tsai, et al. A chaotic levy flight bat algorithm for parameter estimation in nonlinear dynamic biological systems. *Computer and Information Technology*, 2(2):56–63, 2012.

[6] Xin-She Yang and Amir Hossein Gandomi. Bat algorithm: a novel approach for global engineering optimization. *Engineering Computations*, 29(5):464–483, 2012.

[7] Mario Köppen, Katrin Franke, and Raul Vicente-Garcia. Tiny gas for image processing applications. *Computational Intelligence Magazine, IEEE*, 1(2):17–26, 2006.

[8] Konstantinos E Parsopoulos. Cooperative micro-particle swarm optimization. In *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, pages 467–474. ACM, 2009.

[9] Sambarta Dasgupta, Arijit Biswas, Swagatam Das, Bijaya Ketan Panigrahi, and Ajith Abraham. A micro-bacterial foraging algorithm for high-dimensional optimization. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 785–792. IEEE, 2009.

[10] Mauricio Olguin-Carbajal, Enrique Alba, and Javier Arellano-Verdejo. Micro-differential evolution with local search for high dimensional problems. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 48–54. IEEE, 2013.

[11] Ali Osman Topal and Oguz Altun. Dynamic virtual bats algorithm (dvba) for global numerical optimization. In *Intelligent Networking and Collaborative Systems (INCoS), 2014 International Conference on*, pages 320–327. IEEE, 2014.

[12] C Chandrasekar et al. An optimized approach of modified bat algorithm to record deduplication. *International Journal of Computer Applications*, 62(1), 2013.

[13] Matti Airas. Echolocation in bats. In *Proceedings of spatial sound perception and reproduction. The postgrad seminar course of HUT Acoustics Laboratory*, pages 1–25, 2003.

[14] Md Wasi Ul Kabir, Nazmus Sakib, Syed Mustafizur Rahman Chowdhury, and Mohammad Shafiul Alam. A novel adaptive bat algorithm to control explorations and exploitations for continuous optimization problems. *International Journal of Computer Applications*, 94(13):15–20, 2014.

[15] S Yılmaz, E Ugur Kucuksille, and Y Cengiz. Modified bat algorithm. *Elektronika ir Elektrotechnika*, 20(2):71–78, 2014.

[16] Ali Osman Topal, Oguz Altun, and Erisa Terolli. Dynamic virtual bats algorithm (dvba) for minimization of supply chain cost with embedded risk. In *Proceedings of the 2014 European Modelling Symposium*, pages 58–64. IEEE Computer Society, 2014.

[17] Swagatam Das, Ajith Abraham, Uday K Chakraborty, and Amit Konar. Differential evolution using a neighborhood-based mutation operator. *Evolutionary Computation, IEEE Transactions on*, 13(3):526–553, 2009.

[18] JJ Liang, BY Qu, PN Suganthan, and Q Chen. Problem definitions and evaluation criteria for the cec 2015 competition on learning-based real-parameter single objective optimization. *Technical Report201411A, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore*, 2014.

[19] A Kai Qin, Vicky Ling Huang, and Ponnuthurai N Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. *Evolutionary Computation, IEEE Transactions on*, 13(2):398–417, 2009.

[20] Yun-Wei Shang and Yu-Huang Qiu. A note on the extended rosenbrock function. *Evolutionary Computation*, 14(1):119–126, 2006.