JACET

J. ADV COMP ENG TECHNOL, 6(3) Summer 2020

An Improved Bat Algorithm with Grey Wolf Optimizer for Solving Continuous Optimization Problems

Narges Jafari [1], Farhad Soleimanian Gharehchopogh[2]
1,2- Department of Computer Engineering, Urmia branch, Islamic Azad University, Urmia, IRAN.
(bonab.farhad@gmail.com)

Received (2019-09-25) Accepted (2020-05-04)

Abstract: *Metaheuristic algorithms are used to solve NP-hard optimization problems. These algorithms have two main components, i.e. exploration and exploitation, and try to strike a balance between exploration and exploitation to achieve the best possible near-optimal solution. The bat algorithm is one of the metaheuristic algorithms with poor exploration and exploitation. In this paper, exploration and exploitation processes of Gray Wolf Optimizer (GWO) algorithm are applied to some of the solutions produced by the bat algorithm. Therefore, part of the population of the bat algorithm is changed by two processes (i.e. exploration and exploitation) of GWO; the new population enters the bat algorithm population when its result is better than that of the exploitation and exploration operators of the bat algorithm. Thereby, better new solutions are introduced into the bat algorithm at each step. In this paper, 20 mathematic benchmark functions are used to evaluate and compare the proposed method. The simulation results show that the proposed method outperforms the bat algorithm and other metaheuristic algorithms in most implementations and has a high performance.*

Keywords: *Bat algorithm, Gray wolf optimizer, Continuous Problems, Optimization.*

How to cite this article:
Narges Jafari , Farhad Soleimanian Gharehchopogh. An Improved Bat Algorithm with Grey Wolf Optimizer for Solving Continuous Optimization Problems. J. ADV COMP ENG TECHNOL, 6(3) Summer 2020 : 119-132.

I. INTRODUCTION

Optimization means finding the best possible or desirable solution to the problem that is commonly encountered in all disciplines of engineering and science. Optimization problems include a wide range of problems. Optimization algorithms occurring in nature can be either deterministic or random[1, 2]. Deterministic methods and algorithms for solving optimization problems require huge complex calculations that increase the probability of failure. Therefore, the exact algorithms are able to find the optimal solution accurately; however, they are not efficient in the case of NP-hard problems and their execution time increases exponentially with the increase of the dimensions of the optimization problem. Approximation algorithms are able to find the optimal or appropriate solution at a good execution time with high efficiency in solving optimization problems. The use of nature-inspired random optimization algorithms with efficient computations has been suggested rather than deterministic methods and algorithms [3]. Heuristic and metaheuristic algorithms are approximation methods in solving optimization problems; heuristic algorithms most often seek proper near optimal solutions in acceptable computational time [4-6]. However, heuristic algorithms do not guarantee optimal solutions and the main drawbacks of heuristic algorithms include the production of a limited number of solutions, trapping in local optima and early convergence. Due to the disadvantages of heuristic algorithms, metaheuristic algorithms have been introduced. Each metaheuristic

algorithm uses its own methods to get out of the local optima trap or to avoid getting trapped in the local optima. Metaheuristic algorithms are presented to solve optimization problems and improve the ability to find high quality solutions to all optimization problems without getting trapped in local optima.

Metaheuristic optimization algorithms are becoming more and more popular in engineering applications [4-13]. Because 1) they have relatively simple concepts and are easy to implement, 2) they do not need objective function derivative data, 3) they can avoid getting trapped in local optima, 4) they can be used in a wide variety of problems. Nature-inspired metaheuristic algorithms solve optimization problems through mimicking biological or physical phenomena. These algorithms can be grouped into three main categories: evolution-based methods, physics-based methods, and congestion-based methods. Evolution-based approaches are inspired by the law of natural evolution.

The first and most popular metaheuristic algorithm is called Genetic Algorithm, introduced by Holland in 1992 [14], followed by the popular particle swarm optimization (PSO) algorithm first presented by Kennedy and Abraham [15] and later, their modified and improved versions were also developed. After presenting these two basic and robust metaheuristic algorithms, later many other metaheuristic algorithms were introduced for solving optimization problems, one of which is the differential evolution algorithm [16] which is one of the stochastic optimization methods applying mutation, crossover, and selection operators on every generation of population to achieve the global optimum. The artificial bee algorithm (ABC) is an population-based stochastic global optimization approach inspired by foraging behavior of honey bee colony in nature to solve continuous optimization problems with large spaces[17]. Firefly algorithm, inspired by the flashing behavior of the fireflies, was introduced by Yang in 2008 [18].

The bat algorithm [19] was developed by Young in 2010. This algorithm is inspired by the echolocation behavior of micro-bats. The bats emit a very loud pulse and listen to the reflection. To overcome the early and the late convergence and to avoid getting trapped in local optima, the weakness of a metaheuristic algorithm is first identified and then covered with another metaheuristic algorithm that does not have that problem or weakness. Therefore, in this case, both hybrid algorithms cover each other's weaknesses and improve the performance of the algorithm. This method is mostly used in hybridization of metaheuristic algorithms because the operation of the algorithm is already tested and does not create new complexity and problems. GWO is a new evolutionary algorithm introduced in 2014 [20], which has attracted the interest of many researchers in application related to optimization of various problems because the simulation results show the superiority of the GWO over some other robust metaheuristic algorithms.

The main purpose in hybrid metaheuristic algorithms is based on three principles[1]. The first principle is to hybrid two different algorithmic to cover their weaknesses. The second principle is that we seek to quickly achieve global optimum by hybrid the processes of two algorithms. The third principle, which is the most important one for all metaheuristic algorithms, is to maintain a balance between exploration and exploitation by hybrid the processes of two metaheuristic algorithms. Of course, by keeping the balance between exploration and exploitation the algorithm will perform better and try to get as close to the optimal global solution as possible.

In this paper, the bat algorithm is improved with the use of GWO to cover the weaknesses of the bat algorithm in solving continuous optimization problems. The bat algorithm has poor exploration and exploitation, and to cover this weaknesses of the algorithm, the operators of the GWO algorithm have been used. In the proposed method, the exploration and exploitation process of GWO are applied on some solutions produced by the bat algorithm. In this way, some solutions of solving the bat algorithm can be optimized sooner and increase its convergence. On the other hand, a new exploration operator can escape the local trap of the bat algorithm.

The rest of the paper is organized as follows: Section 2 considers the previous works. Section 3 describes the fundamental research, and Section 4 discusses the proposed method. Section 5 provides the obtained results. Section 6 includes conclusion and future work.

II. PREVIOUS WORKS

Zhang and Wang proposed a version of the bat algorithm for image processing [21]. They applied two changes to the original bat algorithm. First, they used constant frequency and loudness, and then they added a mutation operator to increase population diversity. They tested it on image processing and found that the proposed algorithm performs better than the bat algorithm. In another study, the Bat algorithm was hybridization with different evolutionary techniques. This hybridization improved the local search capability of the original bat algorithm. Saha et al. [22] improved the convergence rate of the bat algorithm by interpreting the numerical concept and tested it against several criteria. The simulation results showed that their method increases the convergence rate and accuracy of the bat algorithm. Yılmaz et al. [23] improved the exploration mechanism of the original bat algorithm. They modified the equation of loudness and pulse rate of emission in bat algorithm. They tested this modified bat algorithm on 15 different benchmark functions and concluded that the modified bat algorithm performs better than the bat algorithm. In addition, Li and Zhou also developed the exploration mechanism of the bat algorithm by introducing a scale of complex coding value into the bat algorithm [24]. They separately updated the real and imaginary parts of the complex coding to increase population diversity. To tackle high-dimension problems, a type of algorithm called the bat algorithm with Gaussian walk was developed by Cai et al. [25]. They improved local search capability by introducing a Gaussian walk instead of a uniform random walk. They also modified the equation of updating the bat algorithm's speed, which led to population diversity. This approach extends the search dimension. The dense bat algorithm developed in[26] by Dao et al. is suitable for hardware resource constrained environments. They replaced the design variable of the search space of the bat algorithm with possible population representation. Their study showed that this method can be used effectively in cases where memory is limited.

Fister et al. [27] developed a self-adapting bat algorithm in which the control parameters, like the case of the differential evolution algorithm, are self-adapting by themselves. They tested it on ten benchmark functions and found that the proposed method could be used effectively in continuous optimization problems. To develop local and global search capability in the Bat algorithm, Jun et al. developed the double-subpopulation variant [28]. They used two subgroups, namely external subgroup and internal subgroup. Global exploitation is improved by external subgroups and local exploitation is improved by internal subgroups. They tested the proposed algorithm on several benchmark functions and concluded that the proposed algorithm outperforms the Bat algorithm. Wang et al. [29] presented an improved version of the bat algorithm; they hybridization of the bat algorithm with differential evolution to select the best solution in the bat population. They used this algorithm in three-dimensional programming problems and concluded that the proposed method outperforms the bat algorithm.

In the most recent research, a new hybrid bat algorithm has been proposed by Liu et al. in 2018 for solving continuous optimization problems [30]. Three modifications are applied to the Bat algorithm to increase local search capability and the ability to avoid getting trapped in local optima. The performance of the proposed method was evaluated with benchmark functions, and the results showed that the modified algorithm performs much better.

In another study, to overcome early convergence and low exploration, directional echolocation was introduced to Bat algorithm to increase its exploration and exploitation capabilities [31]. In addition, three other improvements were incorporated into the Bat algorithm to enhance its performance. The statistical test results showed the superiority of the proposed algorithm. In 2019, an improved bat algorithm was proposed by Purkait et al. [32] for solving optimization problems. The main purpose of this paper is to present and interpret the behavior of the bat algorithm in the form of a modified bat algorithm. This paper also describes the concept, advantages, limitations, and application of the bat algorithm in the different domains.

III. FUNDAMENTAL RESEARCH

1. Bat Algorithm

Yang developed a metaheuristic optimization bat-inspired algorithm in 2010 [19]. This algorithm is based on the echolocation behavior of bats with varying pulse rates of emission and loudness. Search is augmented by the "random walk" algorithm. The bat algorithm was developed to use the key idea of frequency adjustment based on the echolocation behavior of bats. The bat's echolocation features in the bat algorithm can be divided into three ideal rules[33]:

- All bats use echolocation to sense the distance, and they also recognize the difference between food/prey and obstacles along the way in some magical way.
- The bats randomly fly at a speed of vi at a location of xi with a constant frequency f_{min} of varying wavelengths λ and loudness of A_0 to search for prey. They can automatically adjust the wavelength (or frequency) of the emitted pulse and the emitted pulse rate, r ∈[0,1] , depending on the prey's proximity.

Although the loudness may vary, we assume that the loudness varies from a positive A_0 to a minimum value A_{min}.

The basic steps of the bat algorithm are summarized in the pseudo-code presented in Figure (1).

```
Bat Algorithm
Objective function f(x), x = (x1 …XD)
Initialize the bat population Xi (i = 1, 2... n) and VI
Define pulse frequency fi at Xi
Initialize pulse rotes and the loudness A
While (t <Max number of iterations)
Generate new solutions by adjusting frequency,
And updating velocities and locations/solutions [equations (2) to (4)]
 If (rand> r)
Select solution among the best solutions
Generate a local solution around the selected best solution
 End if
Generate a new solution by flying randomly
 If (round < A &f (xi) <f(x*))
Accept the new solutions
 End if
Rank the bats and find the current best x*
End while
Post process results and visualization
```

For each bat (i) a location xi and speed vi are assumed in a d-dimensional search space, which must be updated subsequently during each iteration. The new solutions of xti and speed vti at walk time t can be calculated by the equations (1), (2) and (3):

$$F_i = f_{min} + (f_{max}-f_{min})\,\beta \quad (1)$$

$$V_{ti} = v_{t-1} + (x_{t-1i} - x^*)\,f_i \quad (2)$$

$$X_{ti} = x_{t-1i} + v_{ti} \quad (3)$$

In the above-mentioned equations, β in the range [0, 1] is a random vector derived from a uniform distribution. Here x* is the best global solution to be found so far after comparing all solutions to all n bats in the current interaction. The $\lambda_i f_i$ is speed boost. We can use both f_i (and λ_i) to adjust the speed while keeping other factors λ_i (or f_i) constant, depending on the problem.

2. GWO

GWO algorithm[20] is inspired by the life of gray wolves in the nature; the wolves have a particular hierarchy, i.e. alpha, beta, delta and gamma wolves as shown in Figure (2).

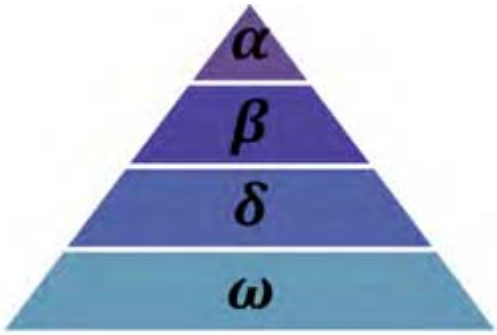

Figure 2: Optimization Algorithm Hierarchy

As shown in Figure (2), the alpha is at the top of the pack and is mainly responsible for deciding on hunting, sleeping place, time to wake, and so on. The alpha's decisions are dictated to the pack; however, some kind of democratic behavior has also been observed, in which an alpha follows the other wolves in the pack. The betas are subordinate wolves that help the alpha in decision-making or other pack activities. In

fact, beta plays the role of an adviser to the alpha and discipliner for the pack. The beta reinforces the alpha's commands throughout the pack and gives feedback to the alpha. The lowest ranking gray wolf is omega. The omega plays the role of scapegoat. The omega always has to submit to other dominant wolves. They are the last wolves that are allowed to eat. If a wolf is not an alpha, beta, or omega, he/she is called a subordinate or delta. Deltas have to submit to alphas and betas, but they dominate omega. Scouts, sentinels, elders, hunters and caretakers belong to this category. Scouts are responsible for watching the boundaries of the territory and warning the pack in case of any danger. The main phases of GWO hunting are as follows:

- Tracking, chasing and approaching the prey
- Pursuing, encircling, and harassing the prey until it stops moving
- Attack towards the prey

In order to mathematically model the social hierarchy of gray wolf, the fittest solution is considered as alpha(α). The second and third best solutions are beta(β) and delta(δ), respectively. The rest of the candidate solutions are assumed to be omega(ω). In GWO, the hunting (optimization) process is guided by α, β and δ. The ω wolves follow these three wolves. Gray wolves encircle the prey during hunting. In order to mathematically model this encircling behavior, we presented equations (4) and (5) [20].

$$\vec{D} = \left|\vec{C}\,.\vec{X}_p(t)\vec{X}(t)\right| \tag{4}$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A}\,.\vec{D} \tag{5}$$

In equations (4) and (5), t is the number of current iterations, $\vec{A}$ and $\vec{C}$ are coefficient vectors, $\vec{X}_p$ is the position vector of the prey and $\vec{X}$ is the position vector of a gray wolf. The vectors $\vec{A}$ and $\vec{C}$ are calculated as shown in equations (6) and (7):

$$\vec{A} = 2\vec{a}.\vec{r}_1 - \vec{a} \tag{6}$$

$$\vec{C} = 2.\vec{r}_2 \tag{7}$$

In Equation (6) and Equation (7), $\vec{a}$ linearly decreases from 2 to 0 over the course of iterations. $\vec{r}_1$ and $\vec{r}_2$ are random vectors in [0, 1]. At the hunt phase, we save the first three best solutions obtained so far and oblige the other search agents (including the omega) to update their position according to the position of the best search agent. equations (8), (9) and (10) are presented to simulate hunting.

$$\vec{D}_\alpha = \left|\vec{C}_1.\vec{X}_\alpha - \vec{\mathrm{X}}\right|, \quad \vec{D}_\beta = \left|\vec{C}_2.\vec{X}_\beta - \vec{\mathrm{X}}\right|, \quad \vec{D}_\delta = \left|\vec{C}_4.\vec{X}_\delta - \vec{\mathrm{X}}\right| \tag{8}$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1.\left(\vec{D}_\alpha\right), \quad \vec{X}_2 = \vec{X}_\beta - \vec{A}_2.\left(\vec{D}_\beta\right), \quad \vec{X}_4 = \vec{X}_\delta - \vec{A}_4.\left(\vec{D}_\delta\right) \tag{9}$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_4}{4} \tag{10}$$

Of course, for exploitation in GWO, in order to mathematically model approaching the prey the value of $\vec{a}$ is reduced; in other words, $\vec{A}$ is a random value in [-2a, 2a], where a is decreased from 2 to zero over the course of iterations. Also, to model exploration or search for prey in this algorithm, the vector $\vec{A}$ with random values greater than 1 or smaller than -1 are utilized to oblige the search agent to diverge from the prey. Another component of GWO that favors exploration is $\vec{C}$, which contains random values in [0, 2].

IV. PROPOSED METHOD

In this section, the proposed method is developed based on the hybridization of the bat algorithm and the GWO algorithm. The purpose of this hybridization is to cover the weaknesses of the bat algorithm with the operators of the GWO algorithm. And, it is also used to balance the exploration and exploitation of the bat algorithm. Therefore, the first stage is to select the initial population for the GWO algorithm operators. And, then the initial population of the GWO algorithm is selecting according to Equation (11) of the total population of the bat algorithm.

$\text{Pop}_{\text{GWO}} = \text{Pop}_{\text{BAT}}(\text{Idx}, 1{:}\,D)$;	(11)
Idx=Randint(1, Npop); numel(Idx) = Npop/2	

In Equation (11), half of the population of the bat algorithm is randomly selected for the GWO algorithm operators. In this equation, Pop_{GWO} shows the population of the GWO algorithm and the Pop_{BAT} relationship represents the population of the GWO algorithm. And, D represents the dimension of each solution, *Idx* represents a random number between 1 and Npop. The number of elements by the *numel* function is equal to half the population of the bat algorithm. Of course, we should not lose the best for GWO algorithm operators.

In the proposed method, the Alpha, Beta and Delta wolves are determined at each stage of the bat algorithm, which can be defined as an Equation (12).

$\vec{X}_a =$ sort(Pop_{BAT}) and select first solution	(12)
$\vec{X}_\beta =$ sort(Pop_{BAT}) and select second solution	
$\vec{X}_\beta =$ sort(Pop_{BAT}) and select third solution	

According to Equation (12), the bat algorithm will be sorted at each stage of the population, and the first best solution will be Alpha $(\vec{X}_a)$, the second best solution will be Beta $(\vec{X}_\beta)$, and the third best solution will be Delta $(\vec{X}_\delta)$. In the next step of the proposed method, the GWO algorithm operators based on the three wolves Alpha, Beta and Delta should be applied to the selected population to balance the exploration and exploitation of the bat algorithm. In the GWO algorithm, the exploration and exploitation process is based on the value of $\vec{A}$ and the value of $\vec{a}$ according to Figure (3).

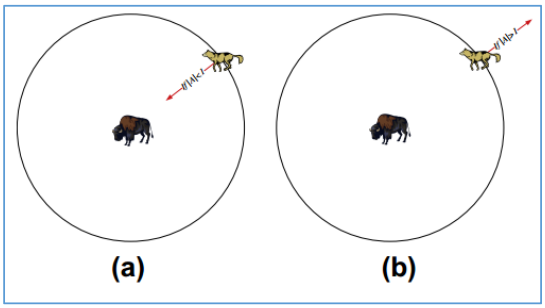

Figure 3: Exploitation and Exploration in GWO

Therefore, in the GWO algorithm as shown in Figure (3), the value of $\vec{a}$ is reduced to model exploitation and also to increase exploration, a value of $\vec{a}$ must increases. This is done automatically in the GWO algorithm.

So that, the value of $|A| < 1$ is somewhat similar to the exploitation of the GWO algorithm, and the value of $|A| > 1$ Do. Wolves are forced to give up prey and find more suitable prey and make explorations. Therefore, The GWO algorithm is a balance of exploration and exploitation, and can be well used in the proposed method to improve the bat algorithm. In this way, the population of the bat algorithm is changed by the GWO algorithm, and a new population is generated, which must be compared again with the population of the bat algorithm and replaced if better. To do this, we used a greedy method according to pseudo-code (1):

pseudo-code (1): Update Population BAT Algorithm

01: POP_{GWO} are new solution
02: N is Size POP_{GWO}
03: Idx is index solution in POP_{BAT}
04: For j=1: N
05: Fit_j=$POP_{GWO}(j, 1:D)$
06: Fit_k=POP_{BAT}(Idx (j),1: D)
07: IF($Fit_j < Fit_k$):
08: $POP_{BAT}(Idx\ (j), 1:D) = POP_{GWO}(j, 1:D)$
09: $Fit_k = Fit_k$
10: End
11: End

According to the pseudo-code of algorithm 1, we used a greedy method to replace and update solutions for bat algorithm in the proposed method. In this algorithm, Solutions that have been modified or improved by the GWO algorithm are stored in POP_{GWO} . And, we also store the GWO algorithm solution numbers from the bat algorithm population stored in the *Idx* variable. Therefore, any solution in the GWO algorithm population is greedily compared to its parent in the bat algorithm population based on the fitness function and if it is better, it will be replaced. The proposed method would have showed as Figure(4) after adding exploration and exploitation processes of GWO algorithms to improve bat algorithm.

Proposed Algorithm
Objective function f(x), x = (x_1 …X_D)
Initialize the bat population X_i (i = 1, 2... n) and V_I
Define pulse frequency f_i at X_i
Initialize pulse rotes and the loudness A
New parameter
Initialize a, A and C
Find X_α , X_β ,X_δ with by Equation (12)

While (t <Max number of iterations)
Generate new solutions by adjusting frequency,
And updating velocities and locations/solutions
If (rand> r)
Select a solution among the best solutions
Generate a local solution around the Select best solution
End if
Generate a new solution by flying randomly
If (rand < A &f (x_i) <f(x^*))
Accept the new solutions
End if
New part
PopGwo=Copy random of Bat algorithm population
For each search agent in PopGwo
update the position of the current search agent by Equation (10)
end for
update a
update A and C by equations (6 and 7)
calculate the fitness of all search agents
update $X\alpha$, $X\beta$ and $X\delta$ with by Equation (12)
update Bat population and GWO population Algorithm 1
Rank the bats and find the current best x*
End while
Post process results and visualization

Figure 4: pseudo-code of the proposed method

According to the pseudo-code of the proposed method and the explanations given, the flowchart of the proposed method can be shown in Figure (5).

As you can see in Figure (5), the new modification method consists of the bat optimization algorithm and the GWO. In fact, in the proposed method, the exploration and exploitation processes of GWO are applied to some of the solutions produced by the bat algorithm. As a result, a part of the population of the bat algorithm is changed by two exploration and exploitation operators of GWO and this new population enters the bat algorithm population when its results are better than that of the exploration and exploitation operators of bat algorithm. By doing so, we will use the results of exploration and exploitation processes of GWO when they obtain better results than the bat algorithm and ensure that modification will lead to improvement.

V. RESULT AND DISCUSSION

In this section, we will evaluate the proposed method and other comparative algorithms. Comparative algorithms include the Harmony Search (HS) algorithm, firefly algorithm, bat algorithm, and GWO. Standard mathematical functions called benchmark functions are used to evaluate and investigate all optimization and metaheuristic algorithms. In this section, we used 20 mathematical benchmark functions, summarized in details in Table(1). The performance and efficiency of the proposed method and other comparative algorithms were then tested and evaluated using this standard function in terms of optimization and the results were recorded and displayed in separate graphs.

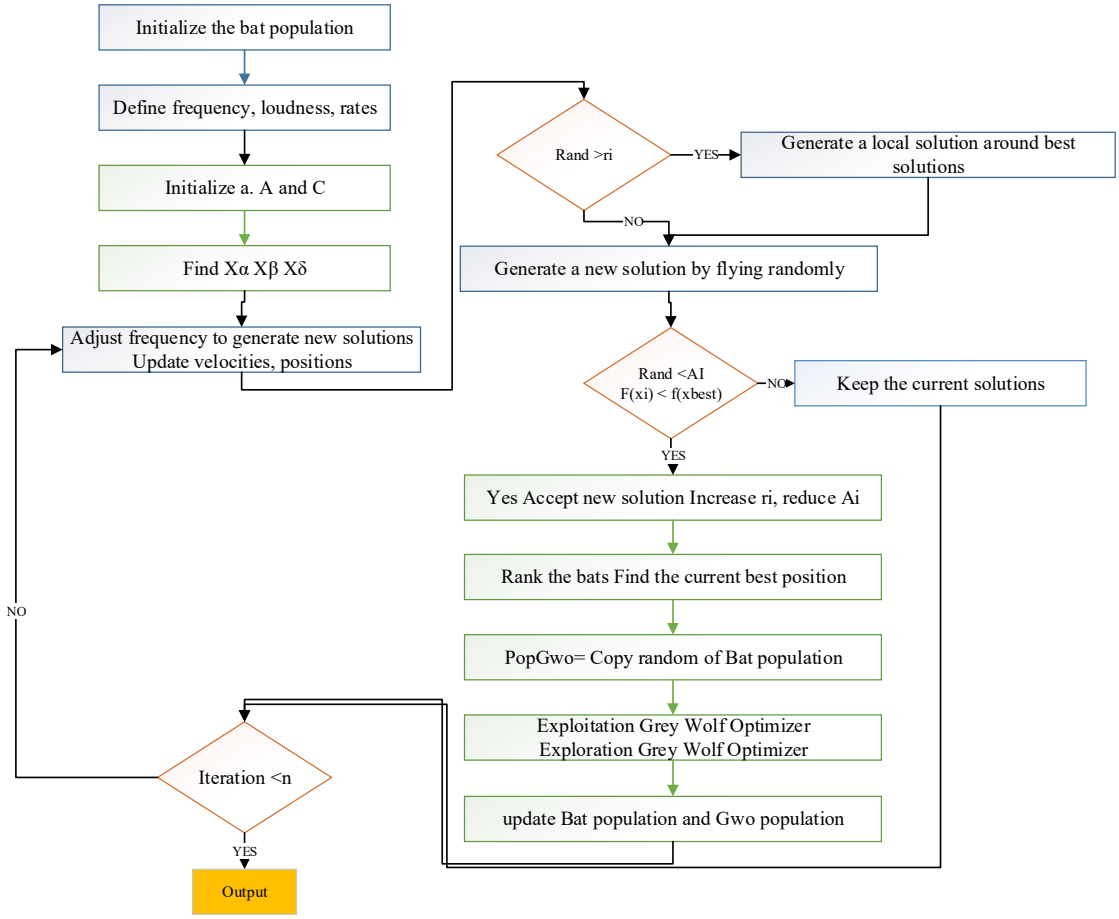

Figure 5: Flowchart of the Proposed Method

TABLE 1: STANDARD BENCHMARK FUNCTIONS TO EVALUATE THE PROPOSED METHOD

Number	Function	Formulation	Range	D	Min
1	Bohachevsky3	$f(x) = x_1^2 - 2x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2 + 0.3$	[-100, 100]	2	0
2	Shubert	$f(x) = (\sum_{i=1}^{5} i\cos(i+1)x_1 + i)(\sum_{i=1}^{5} i\cos(i+1)x_2 + i)$	[-10, 10]	2	-186.73
3	Easom	$f(x) = -\cos(x_1)\cos(x_2)\exp(-(-\pi)^2 - (x_2-\pi)^2)$	[-100, 100]	2	-1
4	Bohachevsky2	$f(x) = x_1^2 - 2x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2 + 0.3$	[-100, 100]	2	0
5	Matyas	$f(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$	[-10, 10]	2	0
6	Six Hump Camel Back	$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 + 4x_2^2 + 4x_2^4$	[-5, 5]	2	-1.03163
7	Michalewicz2	$f(x) = -\sum_{i=1}^{D}\sin(x_i)(\sin(ix_i^2/\pi$	[0, π]	2	-1.8013
8	Booth	$f(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$	[-10, 10]	2	0
9	Michalewicz5	$f(x) = -\sum_{i=1}^{D}\sin(x_i)(\sin(ix_i^2/\pi$	[0, π]	4	-4.6877
10	Rastrigin	$f(x) = \sum_{i=1}^{D}(x_i^2 - \cos(2\pi x_i) + 10)^2$	[-5.12, 5.12]	4	0
11	Zakharov	$f(x) = \sum_{i=1}^{D}x_i^2 + (\sum_{i=1}^{D}0.5ix_i)^2 + (\sum_{i=1}^{D}0.5ix_i)^4$	[-5, 10]	4	0
12	Step	$f(x) = (\sum_{i=1}^{D}x_i + 0.5)^2$	[-5.12, 5.12]	10	0
13	Shere	$f(x) = \sum_{i=1}^{D}x_i^2$	[-100, 100]	10	0
14	Rosenbrock	$f(x) = \sum_{i=1}^{D-1}100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	[-30, 30]	10	0
15	SumSquares	$f(x) = \sum_{i=1}^{D}ix_i^2$	[-10, 10]	10	0
16	Ackley	$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{D}x\right. - \exp\left(\frac{1}{n}\sum_{i=1}^{D-1}\cos(2\pi x_i)\right) + 20 + e$	[-32, 32]	10	0
17	Schwefel 2.22	$f(x) = \sum_{i=1}^{D}\lvert x_i\rvert + \prod_{i=1}^{D}\lvert x_i\rvert$	[-10, 10]	10	0
18	Schwefel 1.2	$f(x) = \sum_{i=1}^{D}(\sum_{j=1}^{i}x_j)^2$	[-100, 100]	30	0
19	Quartic	$f(x) = \sum_{i=1}^{D}ix_i^4 + \text{Rand}$	[-1.28, 1.28]	30	0
20	Dixon-Price	$f(x) = (x_1 - 1)^2 + \sum_{i=1}^{D}i(2x_i^2 - x_i - 1)^2$	[-10, 10]	30	0

To compare metaheuristic algorithms, we first set the basic parameters of the algorithms, as listed in Table (2). We considered the population number and the number of iterations to be the same for all metaheuristic algorithms: iterations=2000 and population=50.

TABLE2: PARAMETERIZATION OF PROPOSED METHOD AND OTHER ALGORITHMS

Algorithm	Values of Parameters
HS [9]	bw=0.2,HMCR=0.5,PAR=0.3, population size is 50
Firefly Algorithm [18]	α=.2,β0=1,γ=1; population size is 50, Nonlooker =50
Bat Algorithm[۱۹]	A and R = = 0.5 0.9 and population size is 50
GWO [20]	population size is 50
Proposed Method	A and R = = 0.5 0.9 and population size is 50

In the following, the proposed method will be implemented on different 2D, 4D, 10D, and 20D benchmark functions. The reason for using different dimensions is to show how the algorithm performs in different dimensions. In addition, the minimum value of the proposed method and other algorithms will be determined over several generations and the extent of algorithm optimization will be shown graphically for comparison. Moreover, the proposed method and other algorithms are compared in terms of the best and the worst values of the objective function and the mean of objective function of the whole population and other statistical parameters. The results of implementation of the proposed method and other basic algorithms on eight 2D benchmark functions are shown in figures (6) to (7), respectively.

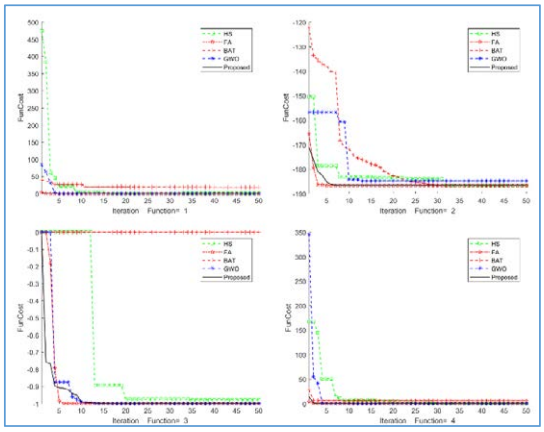

Figure 6: Comparison of the proposed method with other metaheuristic algorithms implemented on functions 1-4 with two dimensions

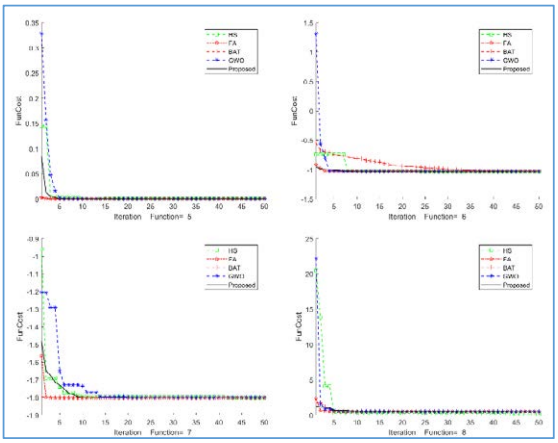

Figure 7: Comparison of the proposed method with other metaheuristic algorithms implemented on functions 5-8 with two dimensions

According to the results, the comparison of the proposed method with other metaheuristic algorithms implemented on 2D functions as shown in figures (6) and (7) shows that the proposed method performs very well on 2D optimization functions and has a better performance than the others. It can also be seen that other metaheuristic algorithms perform well on 2D functions; however, their performance decrease with increasing dimensions. The results of implementing the proposed method and other basic algorithms on the three 4D benchmark functions are shown in Figure (8).

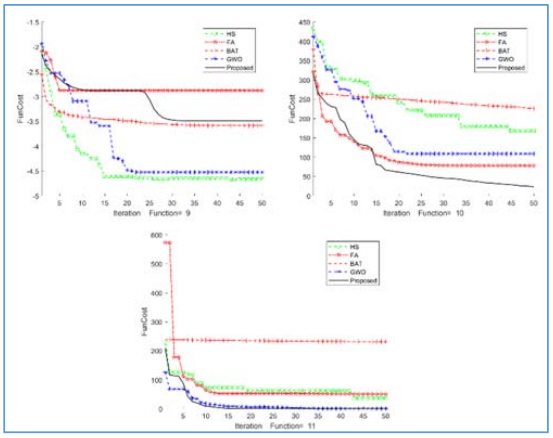

Figure 8: Comparison of the proposed method with other metaheuristic algorithms implemented on functions 9-11 with four dimensions

According to the results obtained by comparing the proposed method with other metaheuristic algorithms implemented on the 4D functions in Figure (8), it is can be seen that the proposed method shows relatively stronger performance compared to the other algorithms when implemented on 4D optimization functions. It can also be seen that some algorithms, such as the bat algorithm, lose their performance by increasing the dimensions of algorithms. The results of implementation of the proposed method and other basic algorithms on six 10D benchmark functions are shown in figures (9) and (10).

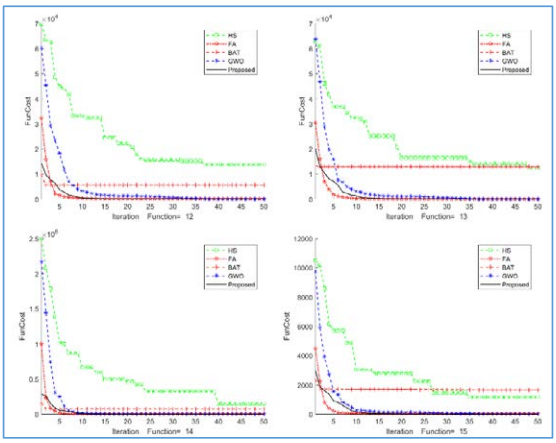

Figure 9: Comparison of the proposed method with other metaheuristic algorithms implemented on functions 12-15 with ten dimensions

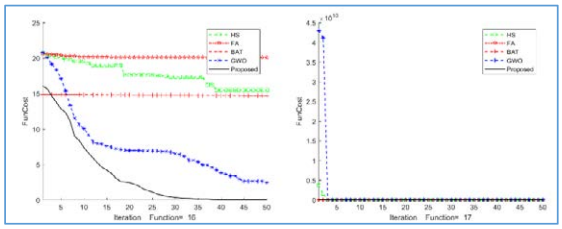

Figure10: Comparison of the proposed method with other metaheuristic algorithms implemented on functions 16-17 with ten dimensions

According to the results obtained by comparing the proposed method with other metaheuristic algorithms implemented on 10D functions in figures (9) to (10), it can be seen that the proposed method has a stronger performance compared to other metaheuristic algorithms implemented on 10D optimization functions. These results show that the performance of the proposed method does not decrease by increasing the number of iterations; however, performance of some algorithms, such as the HS algorithm, decrease with increasing the dimensions of algorithms. The results of implementing the proposed method and other basic algorithms on four 20D benchmark functions are shown in Figure (11).

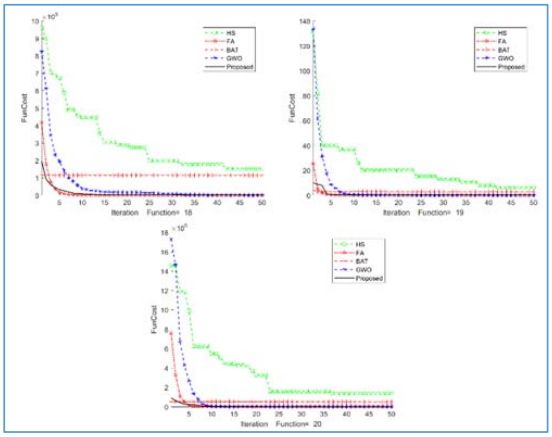

Figure 11: Comparison of the proposed method with other metaheuristic algorithms implemented on functions 18-20 with 20 dimensions

According to the results obtained from comparing the proposed method with other metaheuristic algorithms implemented on 20D functions in Figure (11), it is shown that the proposed method outperforms other algorithms and better makes the optimization functions with high dimensions converge towards the objective compared to the other algorithms. Considering the results of figures (6) to (11), it can be claimed that the proposed method solves the optimization problems in low, medium, and high dimensions and performs better than other basic algorithms, such as the bat algorithm itself. In the following, a statistical evaluation of the proposed method, HS algorithm [9], Firefly Algorithm [18], bat algorithm[19], and GWO [20] are presented. Each of the statistical criteria is defined in Table (3).

TABLE3: CALCULATION OF STATISTICAL CRITERIA

Criteria	Formula
Best	Best= Min(All Fitness Population)
Worst	Worst= Max (All Fitness Population)
Mean	$Mean = \frac{\sum_{i=1}^{Npop} Fitness(i)}{Npop}$
STD	$STD = \sqrt{\frac{1}{N}\sum_{i=1}^{Npop}(Fitness_i - mean(All\ fitness))}$

Therefore, to validate the proposed method, we performed further tests to compare the proposed method with other algorithms in terms of statistical criteria such as mean, best and worst values of the objective function and standard deviation. This test further demonstrates the effectiveness of the algorithms on the whole population; the results are shown in Table 4.

TABLE (4): EVALUATION OF PROPOSED METHOD USING STATISTICAL CRITERIA

F	Measure	HS	FA	BAT	GWO	Proposed Method
F1	Best	7.75E-01	0.00E+00	1.81E+01	0.00E+00	0.00E+00
	Worst	7.75E-01	0.00E+00	1.94E+01	1.36E-12	1.30E-05
	Mean	7.75E-01	0.00E+00	1.84E+01	6.34E-14	2.26E-06
	Std	2.24E-16	0.00E+00	3.57E-01	2.27E-13	2.88E-06
F2	Best	-1.87E+02	-1.87E+02	-1.87E+02	-1.85E+02	-1.87E+02
	Worst	-1.87E+02	-1.87E+02	-1.82E+02	1.27E+02	-8.46E+01
	Mean	-1.87E+02	-1.87E+02	-1.86E+02	9.92E+00	-1.79E+02
	Std	1.44E-13	2.88E-14	1.01E+00	4.02E+01	2.19E+01
F3	Best	-9.78E-01	-1.00E+00	-7.59E-05	-1.00E+00	-1.00E+00
	Worst	-9.78E-01	-1.00E+00	-4.21E-05	8.70E-03	-1.00E+00
	Mean	-9.78E-01	-1.00E+00	-6.97E-05	-2.02E-02	-1.00E+00
	Std	6.73E-16	0.00E+00	7.78E-06	7.94E-02	1.12E-06
F4	Best	4.54E-04	0.00E+00	6.06E+00	0.00E+00	0.00E+00
	Worst	4.54E-04	0.00E+00	6.34E+00	1.33E-15	1.03E-06
	Mean	4.54E-04	0.00E+00	6.13E+00	2.58E-16	2.98E-07
	Std	0.00E+00	0.00E+00	5.69E-02	3.00E-16	2.67E-07
F5	Best	1.19E-04	5.80E-34	4.14E-10	1.26E-23	4.42E-24
	Worst	3.40E-04	3.51E-32	2.49E-05	2.87E-18	2.91E-07
	Mean	2.43E-04	1.24E-32	1.06E-06	1.55E-19	2.61E-08
	Std	4.62E-05	1.04E-32	3.79E-06	5.21E-19	4.22E-08
F6	Best	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00
	Worst	-1.03E+00	-1.03E+00	-6.88E-01	2.74E+03	-1.03E+00
	Mean	-1.03E+00	-1.03E+00	-1.01E+00	9.95E+01	-1.03E+00
	Std	8.97E-16	2.25E-16	5.91E-02	4.16E+02	2.10E-06
F7	Best	-1.80E+00	-1.80E+00	-1.80E+00	-1.80E+00	-1.80E+00
	Worst	-1.80E+00	-1.80E+00	-3.59E-01	7.37E-01	-1.80E+00
	Mean	-1.80E+00	-1.80E+00	-1.77E+00	-8.30E-02	-1.80E+00
	Std	3.65E-05	1.04E-15	2.04E-01	3.89E-01	1.28E-05
F8	Best	9.29E-02	5.07E-01	5.12E-01	5.07E-01	5.07E-01
	Worst	2.09E-01	5.07E-01	5.41E-01	2.38E+04	5.07E-01
	Mean	1.46E-01	5.07E-01	5.16E-01	1.39E+03	5.07E-01
	Std	5.82E-02	2.04E-16	5.13E-03	4.03E+03	1.79E-05
F9	Best	-4.67E+00	-2.88E+00	-3.59E+00	-4.53E+00	-3.49E+00
	Worst	-4.66E+00	-2.88E+00	-7.27E-01	3.42E-01	-3.48E+00
	Mean	-4.66E+00	-2.88E+00	-3.52E+00	-4.80E-01	-3.49E+00
	Std	2.74E-03	9.28E-05	4.04E-01	5.58E-01	8.01E-04
F10	Best	1.69E+02	7.81E+01	2.26E+02	1.09E+02	2.34E+01
	Worst	2.22E+02	7.81E+01	2.61E+02	5.66E+02	4.55E+01
	Mean	2.08E+02	7.81E+01	2.33E+02	3.48E+02	2.71E+01
	Std	1.33E+01	2.44E-04	7.23E+00	6.66E+01	4.53E+00
F11	Best	3.64E+01	5.02E+01	2.31E+02	1.73E-02	1.72E-06
	Worst	1.22E+02	5.03E+01	2.34E+02	8.84E+01	1.27E-03
	Mean	9.27E+01	5.03E+01	2.32E+02	5.04E+00	6.99E-05
	Std	1.97E+01	1.95E-02	7.65E-01	1.43E+01	1.70E-04
F12	Best	1.37E+04	1.38E+02	5.56E+03	2.39E+01	3.05E+00
	Worst	2.06E+04	1.38E+02	5.59E+03	2.29E+03	3.70E+00
	Mean	1.79E+04	1.38E+02	5.57E+03	1.90E+02	3.22E+00
	Std	1.92E+03	4.57E-04	6.19E+00	3.54E+02	1.50E-01
F13	Best	1.22E+04	9.72E+00	1.28E+04	6.82E+00	7.00E-04
	Worst	2.09E+04	9.85E+00	1.28E+04	2.46E+02	1.85E-01
	Mean	1.83E+04	9.81E+00	1.28E+04	5.40E+01	8.48E-03
	Std	2.17E+03	2.82E-02	7.28E+00	3.99E+01	2.90E-02
F14	Best	1.53E+07	1.30E+02	7.06E+06	1.33E+03	2.90E+01
	Worst	4.09E+07	1.31E+02	4.77E+07	9.91E+07	5.02E+01
	Mean	3.48E+07	1.30E+02	7.91E+06	2.85E+06	3.02E+01
	Std	5.34E+06	1.97E-01	5.75E+06	1.49E+07	3.12E+00
F15	Best	1.15E+03	1.69E+01	1.63E+03	2.91E+00	2.37E-04
	Worst	2.32E+03	1.69E+01	1.68E+03	3.36E+01	1.72E-02
	Mean	1.94E+03	1.69E+01	1.65E+03	1.34E+01	2.12E-03
	Std	2.82E+02	6.55E-06	1.42E+01	8.23E+00	3.03E-03
F16	Best	1.55E+01	2.01E+01	1.47E+01	2.43E+00	4.74E-03
	Worst	1.77E+01	2.01E+01	1.48E+01	9.70E+00	2.27E-02
	Mean	1.72E+01	2.01E+01	1.47E+01	4.75E+00	9.95E-03
	Std	4.57E-01	4.85E-04	2.80E-02	1.29E+00	4.78E-03
F17	Best	3.59E+01	1.11E+00	5.23E+01	7.92E-01	7.90E-03
	Worst	5.09E+01	1.12E+00	1.24E+03	2.36E+00	4.24E-02
	Mean	4.60E+01	1.12E+00	1.84E+02	1.44E+00	2.64E-02
	Std	3.70E+00	8.12E-05	1.99E+02	3.42E-01	4.27E-03
F18	Best	1.53E+05	1.31E+02	1.14E+05	3.11E+02	4.19E-03
	Worst	2.71E+05	1.31E+02	1.14E+05	2.51E+03	2.78E+00
	Mean	2.33E+05	1.31E+02	1.14E+05	1.10E+03	9.88E-02
	Std	2.86E+04	5.98E-03	8.13E+01	5.77E+02	3.68E-01
F19	Best	6.34E+00	1.03E-02	2.81E+00	6.42E-02	7.09E-03
	Worst	1.66E+01	1.67E-02	2.74E+02	7.97E+01	4.66E-01
	Mean	1.31E+01	1.41E-02	8.50E+00	5.95E+00	1.71E-01
	Std	2.65E+00	1.86E-03	3.83E+01	1.34E+01	9.90E-02
F20	Best	1.31E+05	6.40E+01	4.75E+04	1.16E+01	6.68E-01
	Worst	3.02E+05	6.40E+01	5.10E+04	4.49E+03	7.40E-01
	Mean	2.39E+05	6.40E+01	4.83E+04	3.25E+02	6.75E-01
	Std	4.75E+04	2.14E-03	8.11E+02	8.16E+02	1.44E-02

Based on the results of the implementation of the proposed method, the HS algorithm [9], Firefly Algorithm [18], bat algorithm[19], and GWO [20] with low, medium, and high dimensions shown in Table (4), it can be stated that the proposed method yielded good results in terms of statistical criteria such as worst value of the objective function, mean and standard deviation. These criteria also show how the proposed method changes all the solutions available in the population and makes them converge towards the objective. The standard deviation of the proposed method shows a good value in most functions, indicating that the proposed method with the hybridization operators of two Bat algorithms and GWO was well able to influence the whole population. Therefore, the better the statistical criteria are, the better the proposed method improves all the solutions available in the population and can show better performance and efficiency.

VI. CONCLUSION AND FUTURE WORK

In this paper, we improved the bat algorithm, which has poor exploration and exploitation, using GWO, which has a robust exploitation and exploration process. In the proposed method, exploration and exploitation processes of GWO are applied to some of the solutions produced by the bat algorithm. As a result, a part of the population of the bat algorithm is changed by two exploration and exploitation operators of GWO and this new population enters the bat algorithm population when its results are better than that of the exploration and exploitation operators of bat algorithm. By doing so, we will use the results of exploration and exploitation processes of GWO when they obtain better results than the bat algorithm and ensure that modification will lead to improvement. The proposed method, HS algorithm, firefly algorithm, bat algorithm, and GWO were evaluated on benchmark functions in different dimensions. The results showed that the proposed method outperformed other algorithms and also exhibited better convergence in most benchmark functions. In addition, the results of the proposed method in terms of statistical criteria showed that the proposed method provided good results in terms of statistical criteria such as worst value of objective function, mean, and standard deviation. These criteria also showed how the proposed method changes all the solutions available in the population and make them converge towards the objective. The proposed method had a good standard deviation value in most functions, indicating that the proposed method with the hybridization of operators of two bat algorithm and GWO was well able to influence the whole population.

REFERENCES

1. Abedi, M. and F.S. Gharehchopogh, An improved opposition based learning firefly algorithm with dragonfly algorithm for solving continuous optimization problems. Intelligent Data Analysis, 2020. 24(2): p. 309-338.

2. Gharehchopogh, F.S. and H. Gholizadeh, A comprehensive survey: Whale Optimization Algorithm and its applications. Swarm and Evolutionary Computation, 2019. 48: p. 1-24.

3. Shayanfar, H. and F.S. Gharehchopogh, Farmland fertility: A new metaheuristic algorithm for solving continuous optimization problems. Applied Soft Computing, 2018. 71: p. 728-746.

4. Amjad, S.S.G., Farhad, A Novel Hybrid Approach for Email Spam Detection based on Scatter Search Algorithm and K-Nearest Neighbors. Journal of Advances in Computer Engineering and Technology, 2019. 5(3): p. 169-178.

5. Khanalni, S. and F.S. Gharehchopogh, A New Approach for Text Documents Classification with Invasive Weed Optimization and Naive Bayes Classifier. Journal of Advances in Computer Engineering and Technology, 2018. 4(3): p. 31-40.

6. Allahverdipour, A. and F. Soleimanian Gharehchopogh, An Improved K-Nearest Neighbor with Crow Search Algorithm for Feature Selection in Text Documents Classification. Journal of Advances in Computer Research, 2018. 9(2): p. 37-48.

7. Majidpour, H. and F. Soleimanian Gharehchopogh, An Improved Flower Pollination Algorithm with AdaBoost Algorithm for Feature Selection in Text Documents Classification. Journal of Advances in Computer Research, 2018. 9(1): p. 29-40.

8. Gharehchopogh, F.S., S.R. Khaze, and I. Maleki, A new approach in bloggers classification with hybrid of k-nearest neighbor and artificial neural network algorithms. Indian Journal of Science and technology, 2015. 8(3): p. 237.

9. Geem, Z.W., J.H. Kim, and G.V. Loganathan, A new heuristic optimization algorithm: harmony search. simulation, 2001. 76(2): p. 60-68.

10. Rabani, H. and F. Soleimanian Gharehchopogh, An Optimized Firefly Algorithm based on Cellular Learning Automata for Community Detection in Social Networks. Journal of Advances in Computer Research, 2019. 10(3): p. 13-30.

11. Gharehchopogh, F.S., H. Shayanfar, and H. Gholizadeh, A comprehensive survey on symbiotic organisms search algorithms. Artificial Intelligence Review, 2019: p. 1-48.

12. Osmani, A., J.B. Mohasefi, and F.S. Gharehchopogh, Sentiment Classification Using Two Effective Optimization Methods Derived From The Artificial Bee Colony Optimization And Imperialist Competitive Algorithm. The Computer Journal, 2020.

13. Hasanluo, M. and F. Soleimanian Gharehchopogh, Software cost estimation by a new hybrid model of particle swarm optimization and k-nearest neighbor algorithms. Journal of Electrical and Computer Engineering Innovations, 2016. 4(1): p. 49-55.

14. Holland, J.H., Genetic algorithms. Scientific american, 1992. 267(1): p. 66-73.

15. Kennedy, J., Particle swarm optimization, in Encyclopedia of machine learning. 2011, Springer. p. 760-766.

16. Storn, R. and K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization, 1997. 11(4): p. 341-359.

17. Karaboga, D., An idea based on honey bee swarm for numerical optimization. 2005, Technical report-tr06, Erciyes university, engineering faculty, computer engineering department.

18. Yang, X.-S., Firefly algorithm. Nature-inspired metaheuristic algorithms, 2008. 20: p. 79-90.

19. Yang, X.-S., A new metaheuristic bat-inspired algorithm, in Nature inspired cooperative strategies for optimization (NICSO 2010). 2010, Springer. p. 65-74.

20. Mirjalili, S., S.M. Mirjalili, and A. Lewis, Grey wolf optimizer. Advances in engineering software, 2014. 69: p. 46-61.

21. Zhang, J.W. and G.G. Wang. Image matching using a bat algorithm with mutation. in Applied Mechanics and Materials. 2012. Trans Tech Publ.

22. Saha, S.K., et al., A new design method using opposition-based BAT algorithm for IIR system identification problem. International Journal of Bio-Inspired Computation, 2013. 5(2): p. 99-132.

23. Yılmaz, S., E.U. Kucuksille, and Y. Cengiz, Modified bat algorithm. Elektronika ir Elektrotechnika, 2014. 20(2): p. 71-78.

24. Li, L. and Y. Zhou, A novel complex-valued bat algorithm. Neural Computing and Applications, 2014. 25(6): p. 1369-1381.

25. Cai, X., et al., Bat algorithm with Gaussian walk. International Journal of Bio-Inspired Computation, 2014. 6(3): p. 166-174.

26. Dao, T.-K., et al., Compact bat algorithm, in Intelligent Data analysis and its Applications, Volume II. 2014, Springer. p. 57-68.

27. Fister, I., S. Fong, and J. Brest, A novel hybrid self-adaptive bat algorithm. The Scientific World Journal, 2014. 2014.

28. Jun, L., L. Liheng, and W. Xianyi, A double-subpopulation variant of the bat algorithm. Applied Mathematics and Computation, 2015. 263: p. 361-377.

29. Wang, G.-G., H.E. Chu, and S. Mirjalili, Three-dimensional path planning for UCAV using an improved bat algorithm. Aerospace Science and Technology, 2016. 49: p. 231-238.

30. Liu, Q., et al., A novel hybrid bat algorithm for solving continuous optimization problems. Applied Soft Computing, 2018. 73: p. 67-82.

31. Chakri, A., et al., New directional bat algorithm for continuous optimization problems. Expert Systems with Applications, 2017. 69: p. 159-175.

32. Purkait, G., et al., An Improved Bio-inspired BAT Algorithm for Optimization, in Progress in Advanced Computing and Intelligent Engineering. 2019, Springer. p. 241-248.

33. Asghari Agcheh Dizaj, S. and F. Soleimanian Gharehchopogh, A New Approach to Software Cost Estimation by Improving Genetic Algorithm with Bat Algorithm. Journal of Computer & Robotics, 2018. 11(2): p. 17-30.