Buzzard Optimization Algorithm: A Nature-Inspired Metaheuristic Algorithm

Ali Arshaghi[1*], Mohsen Ashourian [2], Leila Ghabeli [3]
1- Department of Electrical Engineering, Central Tehran Branch, Islamic Azad University, Tehran, Iran
Email: ali.arshaghi@gmail.com (Corresponding author)
2- Department of Electrical Engineering, Majlesi Branch, Islamic Azad University, Isfahan, Iran.
Email: mohsena@ieee.org
3- Department of Electrical Engineering, Central Tehran Branch, Islamic Azad University, Tehran, Iran.
Email: lghabeli@gmail.com

Received: December 2018 Revised: March 2019 Accepted: May 2019

ABSTRACT:
Various algorithms have proposed during the last decade for solving different complex optimization problems. The meta-heuristic algorithms have been highly noted among researchers. In this paper, a new algorithm, known as the Buzzards Optimization Algorithm (BUZOA), is introduced. Marvelous and special lifestyle of buzzards and their competition characteristics for prey has been the basic motivation for this new optimization algorithm. The algorithm performance has been compared with newest and well-known meta-heuristics on some benchmark problems and test functions. Results have shown the high performance of the proposed BUZOA compared to the other well known algorithms.

KEYWORDS: Buzzard Optimization Algorithm, Global Optimization, Benchmark, Bio Inspired Meta-Heuristic.

1. INTRODUCTION

Duplicate optimization is one of the oldest types of optimization. This type of optimization could be summarized in a few words, improving the previous state to an optimal point with an acceptable error value. Therefore, it is possible to create different designs for these types of methods; where, this sustainability improves the situation for reaching the acceptable error. Therefore, a wide range of issues could be solved based on these methods. Such repetitive methods can be seen in the world around, and especially in nature easily. Therefore, different mathematical methods can be used to optimize different issues, benchmarked the different methods and behaviors, and ideas for continuing a generation to understand these methods. Based on these models, each member of a generation recognizes the similarity of behavior to primary patterns, and resembles and benchmarks the remaining members of the generation based on it and like to that its beavers. In the first aspect, it is necessary to identify the parameters that determine the superiority of each member of the family in the rapid improvement and rapid change of their behavioral patterns than to their past behavioral patterns. If the goal is to reach the most efficient and best path for continuing the generation, all members of the generation will be informed about that path, and all members of the generation will map their path to that optimal path. For example, the shortest path for achieving a goal that can produce a more consistent generation with specific conditions can be as one of the parameters that determines the superiority of a behavior's generation than to another behavior generation.

Solving several engineering optimization problems are very difficult. In these problems, search space and size of population and problem size grows exponentially. Therefore, various meta-heuristic algorithms have been implemented for solving such problems, because the archaic optimization methods are not fit for solving these problems. Hence, over the recent few decades, researchers have described and evidenced that work of meta-heuristic algorithms which is good in abstruse problems such as, timing problems [1], [2], image and video processing [3-10], pattern recognition [11], [12], adjusting of neural networks [13-19] and data clustering[20, 21], optimal control [22-27], etc. In past years, human inspired of the nature to find suitable solution in solving problems, thus, nowadays, there are several algorithms inspired by nature [28, 29]. For example, Water flow-like algorithm [30], Monkey Search [31], Honey Bee Optimization Algorithm (MBO) [32], Cuckoo Search [33] and Cuckoo optimization algorithm [34], Firefly algorithm [35], Biogeography-based optimization algorithm [36], Particle Swarm Optimization [37], The Cat Swarm algorithm [38],

Bacterial Foraging Algorithm [39], Krill herd [40], GA [41], Dolphin Partner Optimization [42], Dolphin echolocation algorithm [43], Bat inspired Algorithm [44], Ant Colony Optimization [45], The Fish School Search [46], Artificial Immune Systems [47], Wolf search [48] and Grey Wolf Optimizer [49], The Social spider optimization [50], Flower pollination algorithm [51], Forest optimization Algorithm [52], Water cycle algorithm [53], The Shuffled Frog Leaping algorithm [54], invasive weed optimization [55].

All algorithms above mentioned are used in many various fields [56-59]. However, for solving all optimization problems and finding the best solution there is no special algorithm. Some algorithms are suitable for specific problems. Therefore, the tendency is to perform research in this field for finding new optimization algorithms in future for solving some problems and finding the best solution [60].

In this study a new optimization algorithm based on Buzzard's behavior and manner's prey in nature, namely the Buzzard Optimization Algorithm (BUZOA), is presented. This Algorithm mostly is similar to those of the PSO and COA Algorithms. Cuckoo optimization Algorithm (COA) was defined by Yang in 2009. This algorithm was designed based on the natural life of the cuckoo bird, according to the features of Cuckoo's life, from lifestyle, this bird is used for optimization. Cuckoo's life is such that it does not have a nest for itself and it places eggs in other nest of birds. PSO is an optimization algorithm that operates on the basis of random population generation. This algorithm, models by Benchmarking and paraphrasing and simulating the behavior of collective flight birds or group moving of fishes.

This paper is consisting of several sections: section 1 presents the introduction, in section 2 the Buzzard Optimization Algorithm (BUZOA) is summarized, and the details of the algorithm explained step by step. Experimental results and comparing this algorithm with another algorithm are expressed in section 3, and in the section 4, conclusions are stated.

2. BUZZARD OPTIMIZATION ALGORITHM (BUZOA)

In this first section, we explained the type of life and features' buzzards, afterward, Buzzard Optimization Algorithm (BUZOA) is dissected.

2.1. Inspiration and Proposed Algorithm

BUZO is the optimized algorithm which is operated based on the random population generation. This algorithm is modeled by paraphrasing and simulated with the behavior of collective flight (group) buzzards (vultures). Apiece member in same group is presented by the ability vector and the position vector in all the search space. The time iteration has presented the new position of a particle by the position vectors and the ability vectors in the search space. for each iteration, the new and next position of a particle, regarding the current ability vector, and the best position founded by that particle and the best particle in the same group in the turkey buzzards position is updated. This algorithm at first is defined for the continuous parameters. In this paper, various sections of this algorithm will be introduced and investigated.

2.1.1. Definition and Profitability of buzzard (vulture)

Vulture is the name of two groups of prey birds. New plume buzzard which is native of the America continent and old plume vultures which are native of Europe, Asia, and Africa. Also, there is no buzzard in Australia and in the South Pole. Vultures are useful animals to prevent putrefy and infection of carrions, especially in the tropical regions. This bird feeds of dead animals and humans occasionally. Corpses are full of infection and illness, diseases such as cholera and charbon which kill any creature. But these diseases do not hurt the vultures, and they help to clean land from illness and infection by eating the corpse. The vulture sometimes eats the corpse of cow and sick sheep.

2.1.2. Strange features of the buzzards (vulture)

- The apparent features of this bird is a bald head. The vulture to separate the dead animal's meat put their head into the dead animal's body.
- The vultures have a long and inverse beak. Beak's vulture is strong but not so much to able break the dead body. They usually wait until stronger birds do it for them.
- The vulture can feed about 20% of its own weight in one meal. The body of the vulture is equipped with a special digestive system.
- The vultures are rarely attacking bouncing animals, but they may kill the injured or sick animal. when war time, a great number of vultures have been seen on the top of the battleground.
- Buzzards are a scavenger and they would like to eat carrion. They eat and Smell dead creatures.
- With laziness flying in the air, if they feel any smell, they will follow it with their sensitive tip for feeding. It was only common sense.
- The vultures appear very quickly and descend from the sky.
- Buzzards normally eat small mice. Furthermore, they eat birds, reptile animals, amphibians, large insects and earthworms.
- Black vultures hunt in a large group in order to have high power for live hunting.
- Scavengers prefer to eat fresh carcasses instead

of the rotten and putrid, and meat of vegetarian rather than predators.

- The buzzards rotate and fly over the head of the animal which is dying and they wait for last breath of them, then they will come down and eat them.

2.1.3. The Defense method

When the vulture is threatened by another animal, it throws its acidic vomit of itself to them. Vomiting has a disgusting smell for many days. This bird also urinates on its legs. The uric acid in the urine annihilate the germs caused by walking on the carrions, which clings to its legs. Hunting of prey by buzzards is shown in Fig. 1.

Fig. 1. Hunting of prey by buzzards.

Turkey vultures (redhead) have a strong sense of smell so that they feel the smell of carcass from intervals of 1000 meters. Contrary, black vultures have weak olfactory power. They fly over Turkey vultures and use the strong smell sense of them in their favor, while they find the location of a corpse, in an exciting challenge they throw away the Turkey vultures and they eat the corpse of animals alone.

Hunts and hunting scenarios and methods of finding carrions by the scavengers are explained below.

First State: If there is a bunch of vultures in the sky, at least one red-head Turkey vulture is below of them in the sky, which glide to find smell from a dead animal. This Turkey vulture will cover some areas; they hunt by using their sense of smell sharp. In the meantime, the black vulture will be several hundred or more, flying above the Turkey vulture, they often spend time slowly, flying on one place and looking by one's eyes to Turkey vulture that is bottom in the sky which is doing whole real seek. If you see a bunch of vultures rotating in the sky, they do not cover any space. They only look at Turkey vulture. At once, Turkey vulture smells a carcass and goes down for eating. The more aggressive of the black vultures are coming down; they chase them so much in order to eat at first. Turkey vulture gets the remnants. A large group of Turkey vultures is called "kettles".

Second State: If the Turkey vulture is not around, the black vultures themselves hunting by viewing, which is very hard for them. In the case of them, they will be flying in a vast space where a dead animal is easily discovered. But, suddenly they see a potential food, they do not lose any time for rotating in the sky, they come down and scan quickly. Firstly, the landed scavengers are found in dead animals. But that's good because the black vultures detect the accumulation of terrestrial scavengers.

Third State: Occasionally the black vultures rotate over a bunch of wolves or dogs which eat the carrions with greed. Some vultures are lucky and try to eat at the same time, but it is dangerous. Therefore, most of them flying and waiting, while others landing near the earth, wait for their luck toward the food at the fastest pace, which it is safe.

For the Buzzards, there are three main scenarios for hunting techniques that often do this.

• While you see the vultures are flying, each of them is waiting for a Turkey vulture to smell food, and only they will waste time.

• Or Buzzards searching with their eye for finding carrions.

• Or they are waiting for a bigger predator, perhaps more dangerous animal or terrestrial scavengers to end up eating, then buzzards land on the ground and eat prey. They do not rotate for good food, sometimes they put it to rotten and vultures eat them.

2.2. Initial Definitions of the BUZO Algorithm

Suppose we have a d -dimensional buzzard search space. i -th particle is described by the vector of position l_i as follows in this d -dimensional space:

$$L = \left(l_{i_1}, l_{i_2}, l_{i_3}, ...l_{i_d} \right) \tag{1}$$

C -vector is the ability of smell, and the ability of taste for the i -th particle is defined by the vector C_i as follows:

$$C_i = \left(c_{i_1}, c_{i_2}, c_{i_3}, ...c_{i_d} \right) \tag{2}$$

The best position which i -th particle is found is the vector $C^*_{i,best}$ and are shown as follows:

$$C^*_{i,best} = \left(c^*_{i_1}, c^*_{i_2}, c^*_{i_3}, ...c^*_{i_d} \right) \tag{3}$$

The best position that has found the best particle in the whole particle is $C^*_{g,best}$ and defines as follows:

$$C^*_{g,best} = \left(c^*_{g_1}, c^*_{g_2}, c^*_{g_3}, ...c^*_{gd}\right) \quad (4)$$

To update the location of each particle, the following equations are used:

$$L_1(t) = \alpha_1 L(t-1) + \alpha_2 * rand * \left(C^*_{i,best} - C_i(t-1)\right) \quad (5)$$

$$L_2(t) = L_1(t) + \beta * rand_1 * \left(C_s(t) - C_i(t-1)\right) + (1-\beta) * rand_2 * \left(C_v(t) - C_i(t-1)\right) + \gamma * rand_1 * \left(C^*_{g,best} - C_i(t-1)\right) + (1-\gamma) * rand_2 * \left(C^*_{g,best} - C_i(t-1)\right) \quad (6)$$

$$C_i(t) = C_i(t-1) + L_i(t) \quad (7)$$

$$C_1(t) = \alpha_1 C(t-1) + \alpha_2 * rand * \left(C^*_{i,best} - l_i(t-1)\right) \quad (8)$$

$$C_2(t) = C_1(t) + \beta * rand * \left(C_s(t) - C_i(t-1)\right) + (1-\beta) * rand * \left(C_v(t) - C_i(t-1)\right) + \gamma * rand * \left(C^*_{g,best} - l_i(t-1)\right) \quad (9)$$

$$L_i(t) = L_i(t-1) + C_i(t) \quad (10)$$

α_1: the inertia weighting factor (motion in the insider path) which indicates the effect of the ability vector before repetition $C_i(t)$ on the ability vector in the current repetition ($C_i(t+1)$).

α_2, β, indicate the training constant coefficient (motion in the path of the best value of the particle examined)

γ: is training constant coefficient (motion on the path of the best particle found in the total population)

$rand_1, rand_2$ are two random numbers with uniform distribution in interval 0 to 1

$C_i(t-1)$ is the ability vectors in repetition $(t-1)$ -th

$L_i(t-1)$ is the Position vector in repetition $(t-1)$ -th

To prevent the excessive increase in the ability and speed of a particle, in the movement from one location to another location, we limit the variation of the ability to the range $C_{\min}$ to $C_{\max}$; meaning $C_{\min} \le C \le C_{\max}$. The upper and lower limitation of ability will be determined regarding to the type of problem.

$C_s = smell\ capability$

$C_v = vision\ capability$

smell = 1-taste

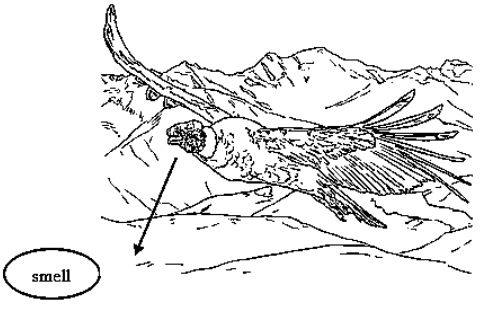

Fig. 2. the smells that buzzards are found from the prey.

In Fig. 2, the smells that buzzards are found from the prey are plotted

2.3. The Benefits of This Algorithm

This optimization algorithm is defined in two ways: one way is limited and buzzards themselves do the search and the search is limited, which is a kind of exploitation algorithm. The other way, there are two types of turkey vultures and predators that find prey and they have a wide range of searches that are unlimited. This type of search is the exploration algorithm. The advantage of this algorithm is that we have two types of limited and infinite searches. If the prey was found in a finite method, it is found. Otherwise, it will be found in the unlimited search method that has a wide range of searches.

There are three modes in this algorithm

(001) There are not the predator and the turkey vultures, but there are buzzards (vultures) themselves.

(010) There are not predators and buzzards, but the turkey vultures themselves are on the ground.

(100) There are not the turkey vultures and buzzards but predators are on the ground.

- The first mode (001) speed is higher
- The second and third mode (010 and 100) speed is lower, but the search range is greater than the first mode.

The first mode (001) [1]there are not any turkey vultures and predator, and buzzards (vultures) are themselves and flying in the sky, as shown in Fig. 3. If the turkey vultures are not around, the buzzards will hunt

[1] Binary first mode

themselves with their glancing and searching, which is very hard for them. They are looking for searching and finding food, and most of them flying in the sky and landed on land as soon as they find food, and they have limited search. This is an exploitation optimization algorithm.

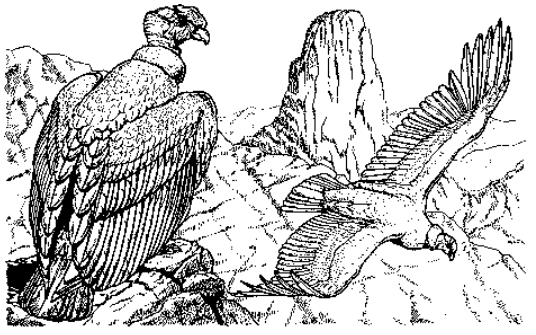

Fig. 3. The first mode of hunting.

The second mode (010)[2] there is no predator and buzzards, and the red-headed turkey vulture on the ground is looking for carcasses and prey, as shown in figure 4. They look for food on the ground and search and smell, and landed on land while they find food. It is a kind of exploration optimization algorithm.

Fig. 4. The second mode of hunting.

The third Mode (100)[3] there are not any turkey vultures and buzzards but there are some predators, as shown in Fig. 5. In this case, some predatory animals, such as wolves and dog will find prey and the black vultures come to them and eat the prey. In this case, the search range is widespread; it is one of the exploration optimization algorithms.

Fig. 5. The third mode of hunting.

2.4. Limitation of Space

Some issues have a certain definition domain for their parameters, and they have only in this domain the limited value, logical and defined. In other words, if there is an encumbrance or constriction in the problem; this constraint should be considered by the mechanism to prevent entrance of particles to the non-virtual space. This mechanism is called limitation space. If these mechanisms are not used, the answer founded from the algorithm is wrong or unreliable by the algorithm. For example, the following function for the negative value of x in most programming is an error.

$$f\left(x\right)=\sum_{d=1}^{D}\sqrt{x} \tag{11}$$

The mechanism which uses this constraint is as follows:

$$x=\max\left(0,x\right) \tag{12}$$

In the above function, the permissible values of x ($x\geq 0$), are mapped without any change, but the non-virtual values ($x<0$) are mapped to the permissible value of $x=0$. More generally, if want the location of the particle to be $\left[\alpha_{\min},\alpha_{\max}\right]^{D}$, then the following equation can be used to restrict:

$$\alpha_d=\min\left\{\max\left(L_d,C_d,\alpha_{\min}\right),\alpha_{\max}\right\} \tag{13}$$

[2] Binary second mode

[3] Binary third mode

By using the above equation, the location of particles that are outside the defined range are mapped within the allowable range. Where, the location of other particles that are in the allowable range are not altered.

2.5. The Steps of Implementation BUZO Algorithm

Sometimes in the differences reference can be seen differently in steps of implementation algorithm. In other words, the steps are separated in some cases, and in some cases, combine two or more steps together into one step. However, this issue does not cause any problem in programming, the reason is; the most important issue is the steps executive of the program, which is explained in the subsequent, and the method of separating these steps. For example, in some references, steps 5 and 6 are combined. Meaning that, the steps of updating the particle capability and transfer particles to the new locations are considered as one step. This change, will not create any problem in the implementation of the algorithm.

2.6. Stage 1, Random Generation of the Primitive Population Particle

Random generation of the initial population simply is the random determine of the initial location of particles with uniform distribution in solving space (search space). Random generation of the primary population stage mostly is existing in all probabilities optimization algorithms. But in this algorithm, in addition to the initial random location of particles, a value for the initial ability of particles is dedicated. The initially proposed range for the ability of particles can be extracted from the following equation.

$$\frac{L_{\min} - L_{\max}}{2} \le C \le \frac{L_{\max} - L_{\min}}{2} \tag{14}$$

2.7. Selecting the Number of Initial Particles

We know that increasing the number of primary particles reduces the number of repetitions needed to converge the algorithm. But sometimes it can be seen that users of optimization algorithms assume that this reduction in the number of repetitions means reducing time implementation of a program to achieve convergence, while it is completely incorrect. However, increasing the number of primary particles will reduce the number of repetitions, but an increase in the number of particles causes the algorithm to spend more time in the particle evaluation stage. This increased time of evaluation causes that despite a decrease in the number of repetitions, there is no reducing in the runtime of the algorithm until convergence is achieved. Therefore, increasing the number of particles cannot be used to reduce the implementation time of the algorithm. There is another misconception that the number of particles can be reduced to deduct the runtime of the algorithm. The idea is also tricky to reduce the needed time for evaluating particles; where, the algorithm wants to get the optimal answer, and it increases the number of repetitions. If we consider the convergence condition to be unchanged at the expense of the best member in several consecutive repetitions, it ultimately does not reduce the execution time of the program for achieving the optimal response. It should also be noted that a decrease in the number of particles may be stymied in the local minima, the algorithm cannot reach the main minimum. If we consider the number of repetitions as the convergence condition, although runtime of the algorithm is reduced by decreasing the number of initial particles, the obtained solution is not optimal answers for the problem, because the algorithm is executed incompletely.

In summary, the number of the primary population is determined by the issue. In general, the number of primary particles is a compromise between the involved parameters in the problem. Experimentally selecting a primitive population particle of 20 to 30 particles is a good choice that responds truly for almost all test issues. We can consider the number of particles slightly more than the necessary limit; to have a little safety margin when faced with local minimal.

2.8. The Objective Function Evaluation (Cost or Expense Calculation)

At this stage, we must evaluate each particle which represents a solution for the issue under case study. Depending on the issue under consideration, the evaluation method will be different. For example, if it is possible to define a mathematical function for the target, with substituting the input parameters (which are extracted from the position vector particle) in this mathematical function, will calculate the cost value of this particle easily. It should be noted that, each particle contains the complete information of the input parameters issue, where, this information is extracted and replaced in target function.

Sometimes there is no possibility to define a mathematical function for evaluation particle. This case occurs when we link the algorithm to software or use the algorithm for experimental data. In this case, we need to know the information about the input parameters of the software, or the test of the particle position vector which is extracted and replaced in the software linked or the algorithm or applied on the relevant test. By running the software or performing the experiment, and observing and measuring results, the cost of each particle could be determined.

2.9. Recording the Best Position of Each Particle ($C^*_{i,best}$) and Best Position among all Particles (

$C^*_{g,best}$ **)**

At this stage, regarding the number repetitions, two states can be investigated:

If we are at the first repetition $(t=1)$. The current position of each particle could be considered as the best location found for that particle.

$$C^*_{i,best} = L_i(t) \quad , i = 1,2,3,...d \tag{15}$$

$$\cos t(C^*_{i,best}) = \cos t(L_j(t)) \tag{16}$$

In other repetitions, the amount of the achieved cost for particles in step 2 could be compared with the best cost achieved for each particle. If this cost is less than of best cost recorded for this particle, then the location and cost of this particle will be replaced with the previous value. Otherwise, no change will be made in the place and cost recorded for this particle, ie:

$$\begin{cases} if\ \cos t(L_i(t)) < \cos t(C^*_{i,best}) \\ \quad else \quad Not \quad change \end{cases} \Rightarrow \tag{17}$$

$$\begin{cases} \cos t(C^*_{i,best}) = \cos t(L_j(t)) \\ C^*_{i,best} = L_i(t) \end{cases} \quad i = 1,2,3,...d$$

Updating the ability vector of all particles

$$C_1(t) = \alpha_1 C(t-1) + \alpha_2 * rand * \left(C^*_{i,best} - l_i(t-1)\right) \tag{18}$$

$$\begin{aligned} C_2(t) &= C_1(t) + \beta * rand * \left(C_s(t) - C_i(t-1)\right) + \\ &(1-\beta) * rand * \left(C_v(t) - C_i(t-1)\right) + \\ &\gamma * rand * \left(C^*_{g,best} - l_i(t-1)\right) \end{aligned} \tag{19}$$

$$L_i(t) = L_i(t-1) + C_i(t) \tag{20}$$

Where, $\alpha_1, \alpha_2, \beta, \gamma$ are determined experimentally by considering the problem. However, generally α_1 should be less than one, because if it chooses larger than one, then $C(t)$ will increase constantly insofar as it diverges. Also, note that in the theory, the coefficient α_1 could be negative, but in the practical use of this algorithm, never consider these coefficients as negative value. The reason isl the negative value of α_1 causes fluctuation in $C(t)$. Selecting the small value for this coefficient (α_1) will cause problems. Often in this buzzard algorithm, this coefficient is considered positive and its ranges from 0.7 to 0.8. The coefficients of β and γ should not be considered very large, because choosing the large values for these two coefficients cause a sharp deviation of the particle from its own path. Often, in this algorithm, the coefficients of these particles are considered positive and its range from 1.5 to 1.7.
It should be noted that the above values are not necessarily only possible choices for $\alpha_1, \alpha_2, \beta, \gamma$, coefficients, but regarding the issue under the study, they may be better choices than the above values.

2.10. Convergence Test

The convergence test in this algorithm is similar to other optimization algorithms. There are various methods for investigating the algorithm. For example, it is possible to specify the exact number of repetitions from the beginning, and at each step investigate whether reached the number of repetitions to the specified value or not? If the number of algorithm iterations is less than the initial determined value, then must go back to step 2 of BUZO algorithm. Otherwise, the algorithm ends. Another method that often is used in the convergence test of the algorithm is that, if in the successive repetitions (for example 15 or 20 repetitions), any changes created in the cost value of best particle, then the algorithm ends; otherwise, it needs to return to step 2.

Flowchart BUZO algorithm is shown in Fig. 6.

3. SIMULATION RESULTS

To appraisal, the performance and efficiency of BUZO Algorithm, 30 test functions from [61] is selected. the latest benchmark functions and their formulas are shown in Tables 1, 2. These functions are called CEC2014 benchmark functions. All details of these functions are in [61]. The proposed BUZO algorithm is compared with some algorithms, Such as IWO [55], BBO [36], GSA [62], HuS [63], BA [44], WWO [64] algorithms. The results are explained in section 3 and tables. The parameters settings of six first algorithms are shown in [64].

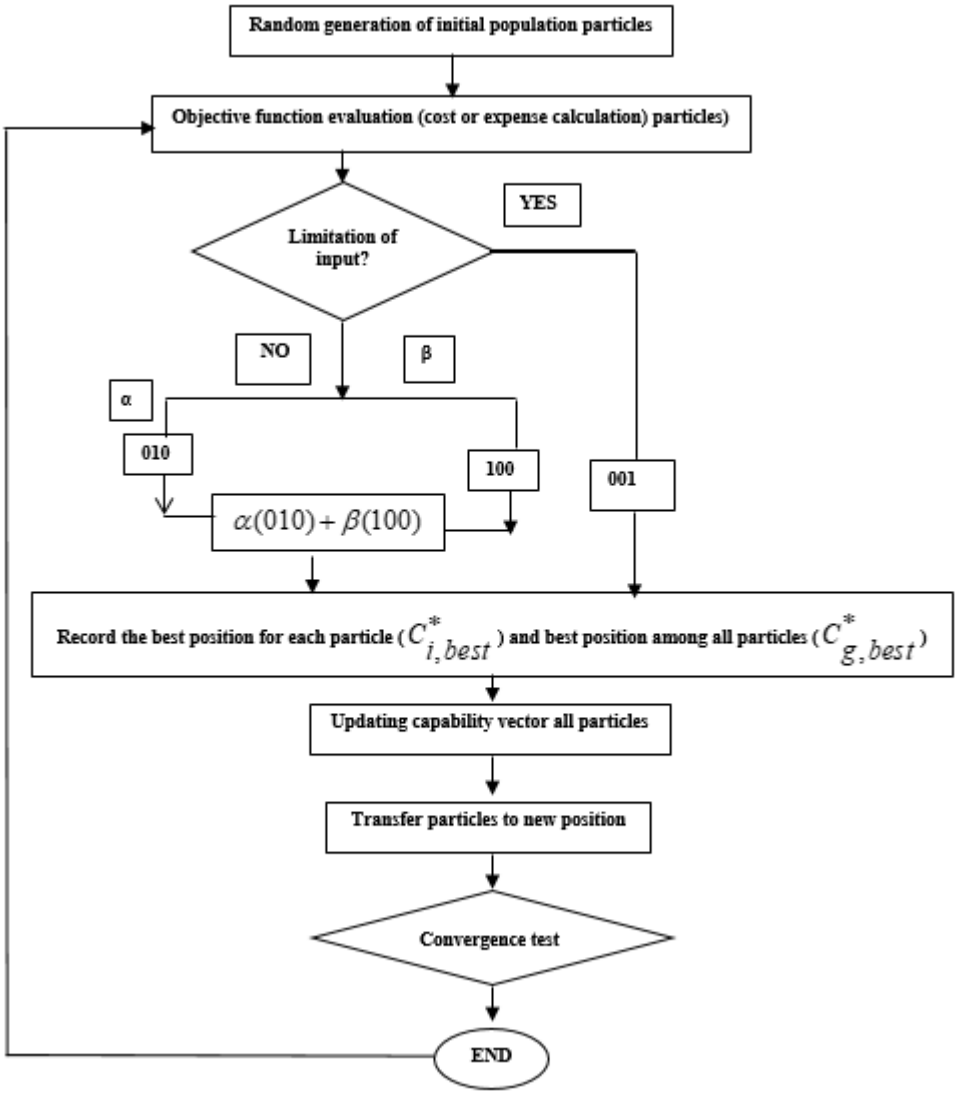

Fig. 6. The flowchart of BUZO algorithm.

Table 1. CEC2014 basic function.

Type	Function	Equation	Optimal
Unimodal	Rotated High Conditioned Elliptic	$F_1(x) = f_1(M(x-o_1)) + F_1^*$	100
	Rotated Bent Cigar	$F_2(x) = f_2(M(x-o_2)) + F_2^*$	200
	Rotated Discuss	$F_3(x) = f_3(M(x-o_3)) + F_3^*$	300
Multimodal	Shifted and rotated Rosenbrock	$F_4(x) = f_4\left(M\left(\frac{2.048(x-o_4)}{100}\right)+1\right) + F_4^*$	400
	Shifted and rotated Ackley	$F_5(x) = f_5(M(x-o_5)) + F_8^*$	500
	Shifted and rotated Weiestrass	$F_6(x) = f_6\left(M\left(\frac{0.5(x-o_6)}{100}\right)\right) + F_6^*$	600

	Shifted and rotated Griewank	$F_7(x) = f_7\left(M\left(\frac{600(x-o_7)}{100}\right)\right) + F_7^*$	700
	Shifted Rastrigin	$F_8(x) = f_8\left(M\left(\frac{5.12(x-o_8)}{100}\right)\right) + F_7^*$	800
	Shifted and rotated Rastrigin	$F_9(x) = f_8\left(M\left(\frac{5.12(x-o_9)}{100}\right)\right) + F_9^*$	900
	Shifted Schwefel	$F_{10}(x) = f_9\left(M\left(\frac{1000(x-o_{10})}{100}\right)\right) + F_{10}^*$	1000
	Shifted and rotated Schwefel	$F_{11}(x) = f_9\left(M\left(\frac{1000(x-o_{11})}{100}\right)\right) + F_{11}^*$	1100
	Shifted and rotated Katsuura	$F_{12}(x) = f_{10}\left(M\left(\frac{5(x-o_{12})}{100}\right)\right) + F_{12}^*$	1200
	Shifted and rotated Happycat	$F_{13}(x) = f_{11}\left(M\left(\frac{5(x-o_{13})}{100}\right)\right) + F_{13}^*$	1300
	Shifted and rotated HgBat	$F_{14}(x) = f_{12}\left(M\left(\frac{5(x-o_{14})}{100}\right)\right) + F_{14}^*$	1400
	Shifted and rotated Expanded Griewank-Rosenbrock	$F_{15}(x) = f_{13}\left(M\left(\frac{5(x-o_{15})}{100}\right)+1\right) + F_{15}^*$	1500
	Shifted and rotated Expanded Scaffer F6	$F_{16}(x) = f_{14}\left(M(x-o_{16})+1\right) + F_{16}^*$	1600
Hybrid	Hybrid 1 (N=3)	P=[0.3,0.3,0.4] g_1: Modifide Schwefel's function f_9 g_1: Rastrigin function f_8 g_3: High Conditioned Elliptic function f_1	1700
	Hybrid 2 (N=3)	P=[0.3,0.3,0.4] g_1: Bent Cigar function f_2 g_1: HGBat function f_{12} g_3: Rastrigin function f_8	1800
	Hybrid 3 (N=4)	P=[0.2,0.2,0.3,0.3] g_1: Griewank function f_7 g_1: Weiestrass function f_6 g_3: Rosenbrock function f_4 g_4: Expanded Scaffer's Weiestrass function f_{14}	1900
	Hybrid 4 (N=4)	P=[0.2,0.2,0.3,0.3] g_1: HGBat function f_{12} g_1: Discuss function f_3 g_3: Expanded Griewank plus Ronsenbrockt function f_{13} g_4: Rastrigin function f_8	2000
	Hybrid 5 (N=5)	P=[0.1,0.2,0.2,0.2,0.3] g_1: Expanded Scaffer's Weiestrass function f_{14} g_2: HGBat function f_{12} g_3: Rosenbrock function f_4 g_4: Modifide Schwefel's function f_9 g_5: High Conditioned Elliptic function f_1	2100
	Hybrid 6 (N=5)	P=[0.1,0.2,0.2,0.2,0.3] g_1: Katsuura function f_{10} g_2: HappyCat function f_{11} g_3: Expanded Griewank plus Ronsenbrockt function f_{13}	2200

		g_4: Modifide Schwefel's function f_9 g_5: Ackley's function f_5	
Composition	Composition 1(N=5)	P=[0.1,0.2,0.2,0.2,0.3] g_1: Rosenbrock function f_4 g_2: High Conditioned Elliptic function f_1 g_3: Bent Cigar function f_2 g_4: Discuss function f_3 g_5: High Conditioned Elliptic function f_1	2300
	Composition 2(N=3)	P=[0.3,0.3,0.4] g_1: Katsuura function f_{10} g_2: Schwefel function f_9 g_3: Expanded Scaffer's Weiestrass function f_{14}	2400
	Composition 3 (N=3)	P=[0.3,0.3,0.4] g_1: Happy Cat function f_{11} g_2: Schwefel function f_9 g_3: High Conditioned Elliptic function f_1	2500
	Composition 4 (N=5)	P=[0.1,0.2,0.2,0.2,0.3] g_1: HappyCat function f_{11} g_2: Expanded Griewank plus Ronsenbrockt function f_{13} g_3: High Conditioned Elliptic function f_1 g_4: Weiestrass function f_6 g_5: Griewank function f_7	2600
	Composition 5 (N=5)	P=[0.1,0.2,0.2,0.2,0.3] g_1: Expanded Scaffer's Weiestrass function f_{14} g_2: Schwefelfunction f_9 g_3: HappyCat function f_{11} g_4: Weiestrass function f_6 g_5: High Conditioned Elliptic function f_1	2700
	Composition 6 (N=10)	P=[0.1,0.2,0.2,0.2,0.3] g_1Shifted and rotated Expanded Griewank- Rosenbrockfunction f_{15} g_2: Expanded Griewank plus Ronsenbrockt function f_{13} g_3: HappyCat function f_{11} g_4: Shifted and rotated Expanded Scaffer Weiestrass function f_{16} g_5: High Conditioned Elliptic function f_1	2800
	Composition 7 (N=10)	Composition function7 (f 17, f 18, f 19)	2900
	Composition 8 (N=14)	Composition function8 (f 20, f 21, f 22)	3000

Table 2. Test functions applied in the simulation test.

Type	ID	Function	f*
Unimodal	f1	Rotated high conditioned elliptic function	100
	f2	Rotated bent cigar function	200
	f3	Rotated discus function	300
Multimodal	f4	Shifted and rotated Rosenbrock function	400
	f5	Shifted and rotated Ackley's function	500
	f6	Shifted and rotated Weierstrass function	600
	f7	Shifted and rotated Griewank's function	700
	f8	Shifted Rastrigin function	800
	f9	Shifted and rotated Rastrigin's function	900
	f10	Shifted Schwefel function	1000
	f11	Shifted and rotated Schwefel's function	1100
	f12	Shifted and rotated Katsuura function	1200
	f13	Shifted and rotated HappyCat function	1300
	f14	Shifted and rotated HGBat function	1400
	f15	Shifted and rotated Expanded Griewank's plus Rosenbrock's function	1500
	f16	Shifted and rotated Expanded Scaffer's F6 function	1600
Hybrid	f17	Hybrid function1 (f 9, f 8, f 1)	1700
	f18	Hybrid function2 (f 2, f 12, f 8)	1800
	f19	Hybrid function3 (f 7, f 6, f 4, f 14)	1900
	f20	Hybrid function4 (f 12, f 3, f 13, f 8)	2000
	f21	Hybrid function5 (f 14, f 12, f 4, f 9, f 1)	2100
	f22	Hybrid function6 (f 10, f 11, f 13, f 9, f 5)	2200
Composition	f23	Composition function1 (f 4, f 1, f 2, f 3, f 1)	2300
	f24	Composition function2 (f 10, f 9, f 14)	2400
	f25	Composition function3 (f 11, f 9, f 1)	2500
	f26	Composition function4 (f 11, f 13, f 1, f 6, f 7)	2600
	f27	Composition function5 (f 14, f 9, f 11, f 6, f 1)	2700

f28	Composition function6 (f 15, f 13, f 11, f 16, f 1)	2800
f29	Composition function7 (f 17, f 18, f 19)	2900
f30	Composition function8 (f 20, f 21, f 22)	3000

In all functions, number of population is set to 100, function evaluations is set in 150,000 for all functions tested in simulation, and a stopping condition is state; the dimension is 30 (n = 30). Parameters of the BUZ algorithm are wounded by RSM [2] that are explained in Table 3.

Table 3. Parameters values of compared algorithms.

Parameter	Value	Parameter	Value
Population size	100	w, w_{damp}	1, 0.99
Number of variables	10	α_1, α_2	0.7 to 0.8
Lower bound, upper bound	10*ones(1,10)	β	1.5 to 1.7
Max of iteration	100	γ	1.5 to 1.7

The best performance is reported over 60 runs in Table 4 for all functions, “std” denotes standard deviation, “Median" introduce the mean of the result fitness values, “maximum” and “minimum” are the maximum and minimum fitness values of the BUZO algorithm.

Table 4. Results of unimodal benchmark functions for over seven algorithms.

		GSA	BBO	IWO	WWO	HuS	BA	BUZO
F1	Maximum	5.31E+07	8.09E+07	2.77E+06	1.17E+06	1.26E+07	5.51E+08	3.42E+05
	Minimum	4.56E+06	5.75E+06	3.45E+05	1.45E+05	1.61E+06	1.18E+08	2.02E+04
	Median	8.37E+06	2.14E+07	1.43E+06	6.26E+05	5.10E+06	3.10E+08	1.34E+05
	std	1.32E+07	1.67E+07	5.72E+05	2.45E+05	2.62E+06	1.05E+08	1.02E+05
F2	Maximum	1.61E+04	8.05E+06	4.07E+04	1.48E+03	2.41E+04	6.36E+09	1.55E+03
	Minimum	3.47E+03	1.16E+06	6.09E+03	2.00E+02	3.08E+02	1.14E+09	2.10E+02
	Median	8.38E+03	3.95E+06	1.52E+04	2.69E+02	9.08E+03	2.48E+09	6.43E+02
	std	2.90E+03	1.55E+06	8.68E+03	2.03E+02	6.01E+03	7.55E+08	3.96E+02
F3	Maximum	7.58E+04	5.07E+04	1.50E+04	1.32E+03	3.36E+03	1.11E+05	1.20E+03
	Minimum	2.05E+04	5.93E+02	3.51E+03	3.15E+02	3.02E+02	3.44E+04	3.10E+02
	Median	4.51E+04	7.65E+03	7.39E+03	4.88E+02	3.02E+02	7.19E+04	4.09E+02
	std	1.05E+04	1.29E+04	2.68E+03	1.86E+02	5.41E+02	1.76E+04	3.40E+02

According to Table 4, BUZO have best performance on f1 in all criteria than all algorithms and the fit minimum value is on f2, and on f3 has BUZO prepare the best and suitable maximum and minimum values.

Table 5. Results of multimodal benchmark functions over seven algorithms.

		GSA	BBO	IWO	WWO	HuS	BA	BUZO
F4	Maximum	8.49E+02	6.54E+02	5.45E+02	5.43E+02	5.74E+02	1.26E+04	5.04E+02
	Minimum	5.73E+02	4.24E+02	4.02E+02	4.01E+02	4.05E+02	2.01E+03	3.00E+02
	Median	6.83E+02	5.42E+02	5.11E+02	4.02E+02	5.03E+02	3.06E+03	3.16E+02
	std	5.15E+01	3.84E+01	2.89E+01	3.64E+01	3.66E+01	1.98E+03	3.71E+01
F5	Maximum	5.22E+02	5.22E+02	5.22E+02	5.20E+02	5.31E+02	5.22E+02	5.10E+02
	Minimum	5.22E+02	5.20E+02	5.20E+02	5.21E+02	5.31E+02	5.22E+02	5.10E+02
	Median	5.22E+02	5.21E+02	5.20E+02	5.21E+02	5.21E+02	5.22E+02	5.03E+02
	std	6.47E-04	4.22E-02	3.77E-03	6.98E-04	7.83E-02	4.81E-02	3.33E+00
F6	Maximum	6.25E+02	6.19E+02	6.05E+02	6.13E+02	6.29E+02	6.39E+02	5.95E+02
	Minimum	6.17E+02	6.08E+02	6.00E+02	6.01E+02	6.19E+02	6.32E+02	5.90E+02
	Median	6.21E+02	6.14E+02	6.02E+02	6.06E+02	6.23E+02	6.37E+02	5.91E+02
	std	1.83E+00	2.36E+00	1.12E+00	2.62E+00	2.18E+00	1.56E+00	2.07E+00
F7	Maximum	7.01E+02	7.02E+02	7.01E+02	7.01E+02	7.01E+02	9.63E+02	6.80E+02
	Minimum	7.01E+02	7.02E+02	7.01E+02	7.01E+02	7.01E+02	8.19E+02	6.80E+02
	Median	7.01E+02	7.02E+02	7.01E+02	7.01E+02	7.01E+02	9.12E+02	6.67E+02
	std	9.55E-04	2.64E-02	1.21E-02	6.26E-03	5.56E-02	3.23E-01	8.35E-04
F8	Maximum	8.02E+02	9.38E+02	8.75E+02	8.15E+02	9.76E+02	1.12E+03	7.11E+02
	Minimum	8.01E+02	8.38E+02	8.27E+02	8.00E+02	9.11E+02	9.76E+02	7.00E+02
	Median	8.01E+02	8.79E+02	8.43E+02	8.00E+02	9.40E+02	1.08E+03	7.02E+02
	std	2.06E-01	2.07E+01	1.01E+01	2.34E+00	1.27E+01	2.56E+01	3.71E+00

F9	Maximum	1.11E+03	9.84E+02	9.88E+02	9.84E+02	1.09E+03	1.34E+03	8.10E+02
	Minimum	1.03E+03	9.35E+02	9.40E+02	9.36E+02	9.57E+02	1.15E+03	8.00E+02
	Median	1.07E+03	9.49E+02	9.46E+02	9.63E+02	1.03E+03	1.25E+03	8.03E+02
	std	1.74E+01	1.14E+01	1.14E+01	1.11E+01	2.60E+01	4.43E+01	3.28E+00
F10	Maximum	5.25E+03	1.00E+03	3.58E+03	2.72E+03	3.22E+03	7.45E+03	1.11E+03
	Minimum	3.45E+03	1.01E+03	1.59E+03	1.02E+03	1.36E+03	5.26E+03	1.11E+03
	Median	4.37E+03	1.01E+03	2.58E+03	1.49E+03	2.17E+03	6.47E+03	1.00E+03
	std	3.61E+02	6.80E-01	3.80E+02	3.62E+02	4.34E+02	5.19E+02	8.16E-02
F11	Maximum	6.35E+03	4.52E+03	3.80E+03	3.88E+03	4.23E+03	8.75E+03	1.01E+03
	Minimum	3.70E+03	2.12E+03	1.47E+03	2.49E+03	2.20E+03	7.21E+03	1.10E+03
	Median	4.99E+03	3.32E+03	2.92E+03	3.38E+03	3.24E+03	8.25E+03	1.01E+03
	std	5.67E+02	5.12E+02	4.49E+02	2.88E+02	4.66E+02	3.63E+02	3.64E+00
F12	Maximum	1.21E+03	1.21E+03	1.21E+03	1.21E+03	1.21E+03	1.20E+03	1.10E+03
	Minimum	1.21E+03	1.21E+03	1.21E+03	1.21E+03	1.21E+03	1.20E+03	1.20E+03
	Median	1.21E+03	1.21E+03	1.21E+03	1.21E+03	1.22E+03	1.20E+03	1.10E+03
	std	1.00E-03	5.62E-02	1.48E-02	5.61E-02	7.77E-02	3.34E-01	2.01E-02
F13	Maximum	1.31E+03	1.31E+03	1.31E+03	1.31E+03	1.31E+03	1.31E+03	1.10E+03
	Minimum	1.31E+03	1.31E+03	1.31E+03	1.31E+03	1.31E+03	1.31E+03	1.10E+03
	Median	1.31E+03	1.31E+03	1.31E+03	1.31E+03	1.31E+03	1.31E+03	1.10E+03
	std	6.65E-02	1.06E-01	6.50E-02	6.41E-02	6.50E-02	5.48E-01	0.80E+00
F14	Maximum	1.30E+03	1.30E+03	1.30E+03	1.40E+02	1.40E+02	1.50E+03	1.20E+02
	Minimum	1.30E+03	1.30E+03	1.30E+03	1.40E+02	1.40E+02	1.44E+03	1.20E+02
	Median	1.30E+03	1.30E+03	1.30E+03	1.40E+02	1.40E+02	1.44E+03	1.20E+01
	std	4.23E-02	1.99E-01	1.19E-01	4.41E-02	4.74E-02	1.39E-01	0.90E+00
F15	Maximum	1.61E+03	1.63E+02	1.61E+03	1.60E+03	1.62E+03	5.92E+05	1.41E+03
	Minimum	1.60E+03	1.61E+02	1.61E+03	1.60E+03	1.61E+03	1.58E+04	1.30E+03
	Median	1.60E+03	1.61E+02	1.61E+03	1.60E+03	1.62E+03	1.55E+04	1.40E+03
	std	7.30E-01	4.30E+00	8.48E-01	7.75E-01	3.27E+00	1.40E+04	3.31E+00
F16	Maximum	1.71E+03	1.71E+03	1.71E+03	1.71E+03	1.71E+03	1.71E+03	1.52E+03
	Minimum	1.71E+03	1.71E+03	1.71E+03	1.71E+03	1.62E+03	1.63E+03	1.51E+03
	Median	1.62E+03	1.71E+03	1.71E+03	1.71E+03	1.62E+03	1.63E+03	1.51E+03
	std	3.43E-01	5.92E-01	6.14E-01	4.67E-01	7.25E-01	1.90E-01	1.69E+00

Because a large number of local optima scape from local optima, finding good solutions is very hard. Based on Table 5, BUZO represents main performance than all algorithms on these functions

.

Table 6. Results of hybrid benchmark functions for seven algorithms.

		GSA	BBO	IWO	WWO	HuS	BA	BUZO
F17	Maximum	1.14E+06	2.31E+07	3.51E+05	6.16E+04	1.10E+06	9.90E+06	1.60E+03
	Minimum	1.85E+05	1.26E+06	5.38E+03	6.71E+03	1.43E+04	1.46E+06	1.60E+03
	Median	5.63E+05	3.14E+06	6.76E+04	2.62E+04	1.51E+05	4.24E+06	1.63E+03
	std	2.20E+05	4.20E+06	6.85E+04	1.25E+04	1.61E+05	1.78E+06	2.91E+01
F18	Maximum	4.20E+03	1.03E+05	1.80E+04	2.73E+03	1.09E+04	3.64E+08	1.44E+03
	Minimum	2.02E+03	6.75E+03	2.27E+03	1.85E+03	2.02E+03	1.33E+07	1.41E+03
	Median	2.13E+03	2.29E+04	4.36E+03	2.02E+03	2.73E+03	8.54E+07	1.42E+03
	std	3.78E+02	1.97E+04	3.69E+03	1.26E+02	2.25E+03	1.00E+08	1.34E+01
F19	Maximum	2.01E+03	1.99E+03	1.91E+03	1.91E+03	2.04E+03	2.07E+06	1.72E+03
	Minimum	1.91E+03	1.91E+03	1.90E+03	1.90E+03	1.91E+03	1.95E+03	1.70E+03
	Median	2.01E+03	1.91E+03	1.91E+03	1.91E+03	1.92E+03	2.02E+03	1.70E+03
	Std	3.43E+01	2.77E+01	1.66E+00	1.39E+00	3.31E+01	2.03E+01	6.91E+00
F20	Maximum	6.83E+04	8.62E+04	5.34E+03	1.58E+04	6.04E+04	4.44E+04	2.10E+03
	Minimum	2.32E+03	8.66E+03	2.31E+03	2.14E+03	2.23E+04	5.41E+03	2.11E+03
	Median	1.77E+04	2.73E+04	2.74E+03	4.26E+03	3.68E+04	1.63E+04	2.11E+03
	Std	1.38E+04	1.76E+04	7.00E+02	3.18E+03	8.49E+03	1.03E+04	4.32E-01
F21	Maximum	3.09E+05	1.68E+06	9.03E+04	1.76E+05	1.67E+05	3.35E+06	2.22E+03
	Minimum	5.88E+04	6.70E+04	6.74E+03	3.70E+03	1.08E+04	1.44E+05	2.21E+03
	Median	1.71E+05	4.22E+05	3.35E+04	2.92E+04	4.70E+04	9.17E+05	2.21E+03
	Std	6.53E+04	3.35E+05	2.30E+04	3.50E+04	4.24E+04	7.51E+05	2.03E+00
F22	Maximum	3.63E+03	3.29E+03	2.54E+03	2.85E+03	3.67E+03	3.56E+03	2.10E+03
	Minimum	2.63E+03	2.25E+03	2.23E+03	2.22E+03	2.38E+03	2.72E+03	2.10E+03
	Median	3.15E+03	2.71E+03	2.36E+03	2.49E+03	3.08E+03	3.14E+03	2.10E+03
	std	2.50E+02	2.34E+02	7.34E+01	1.43E+02	2.67E+02	2.05E+02	4.36E-01

On the above table of these functions, some functions are hybrid, the variables are used in subcomponents which cause performance reduction of algorithms, and are displayed in table 6, the performance of BUZO algorithm is away on these functions; it has good results compared to the other algorithms.

Table 7. Results of composition benchmark functions for seven algorithms.

		GSA	BBO	IWO	WWO	HuS	BA	BUZO
F23	Maximum	2.65E+03	2.62E+03	2.62E+03	2.62E+03	2.62E+03	2.88E+03	2.24E+03
	Minimum	2.50E+03	2.62E+03	2.62E+03	2.62E+03	2.62E+03	2.51E+03	2.27E+03
	Median	2.57E+03	2.62E+03	2.62E+03	2.62E+03	2.63E+03	2.51E+03	2.25E+03
	std	6.45E+01	1.32E+00	7.95E+02	1.45E+01	8.45E+01	1.28E+02	8.53E+01
F24	Maximum	2.60E+03	2.65E+03	2.63E+03	2.63E+03	2.71E+03	2.60E+03	2.37E+03
	Minimum	2.61E+03	2.63E+03	2.62E+03	2.62E+03	2.63E+03	2.60E+03	2.30E+03
	Median	2.60E+03	2.63E+03	2.62E+03	2.63E+03	2.66E+03	2.60E+03	2.32E+03
	std	1.71E-02	5.97E+00	1.08E+01	6.89E+00	1.25E+01	1.21E+00	2.13E+01
F25	Maximum	2.71E+03	2.72E+03	2.71E+03	2.72E+03	2.75E+03	2.76E+03	2.60E+03
	Minimum	2.71E+03	2.71E+03	2.71E+03	2.71E+03	2.71E+03	2.70E+03	2.42E+03
	Median	2.71E+03	2.71E+03	2.71E+03	2.71E+03	2.72E+03	2.70E+03	2.36E+03
	std	1.32E+00	3.01E+00	8.08E-01	2.00E+00	6.27E+00	1.50E+01	6.73E+01
F26	Maximum	2.80E+03	2.80E+03	2.72E+03	2.71E+03	2.80E+03	2.70E+03	2.41E+03
	Minimum	2.81E+03	2.70E+03	2.72E+03	2.70E+03	2.70E+03	2.70E+03	2.40E+03
	Median	2.81E+03	2.70E+03	2.71E+03	2.70E+03	2.80E+03	2.70E+03	2.41E+03
	std	5.43E+03	2.20E+01	5.43E+02	6.50E+02	3.53E+01	5.37E+01	3.03E+00
F27	Maximum	4.43E+03	3.51E+03	3.10E+03	3.50E+03	6.47E+03	3.54E+03	2.32E+03
	Minimum	3.11E+03	3.24E+03	3.01E+03	3.10E+03	3.57E+03	3.21E+03	2.30E+03
	Median	3.82E+03	3.41E+03	3.10E+03	3.10E+03	4.84E+03	3.31E+03	2.31E+03
	std	3.51E+02	6.35E+01	3.38E+01	5.90E+01	6.83E+02	6.46E+01	5.69E+00
F28	Maximum	6.92E+03	4.27E+03	3.86E+03	5.40E+03	6.65E+03	6.10E+03	7.06E+03
	Minimum	3.76E+03	3.61E+03	3.56E+03	3.12E+03	4.70E+03	3.01E+03	3.12E+03
	Median	5.43E+03	3.80E+03	3.69E+03	3.79E+03	5.36E+03	4.52E+03	4.15E+03
	std	7.15E+02	9.34E+01	4.12E+01	3.62E+02	4.61E+02	5.93E+02	1.16E+03
F29	Maximum	2.93E+06	8.64E+06	2.79E+04	5.07E+03	4.12E+07	1.36E+07	6.40E+04
	Minimum	3.11E+03	4.27E+03	5.37E+03	3.56E+03	4.81E+03	6.16E+05	3.11E+03
	Median	3.10E+03	5.27E+03	1.58E+04	4.03E+03	1.54E+04	4.21E+06	1.70E+04
	std	3.78E+05	1.11E+06	5.14E+03	3.60E+02	7.70E+06	2.83E+06	2.04E+04
F30	Maximum	1.14E+05	3.75E+04	1.69E+04	7.67E+03	3.74E+04	5.08E+05	3.13E+03
	Minimum	1.22E+04	7.78E+03	6.06E+03	4.26E+03	8.27E+03	6.26E+04	3.01E+03
	Median	1.47E+04	1.56E+04	8.85E+03	5.64E+03	1.51E+04	1.77E+05	3.03E+03
	std	1.84E+04	6.08E+03	2.08E+03	7.39E+02	6.58E+03	9.11E+04	5.21E+01

On the above table of functions, BUZO has ranked for f23-f30 functions in above table (see Table 7). The performance of BUZO is a tiny weak in two functions f28, f29. Therefore, the work of BUZO is the best than the other used algorithms in the paper on the test suite, such as multimodal, unimodal, composition, and hybrid functions, the behaver of BUZO is very suitable and effective on these 30 functions.

4. CONCLUSION

In recent decades, researchers have offered various algorithms that are derived by natural; they have proposed various meta-heuristic optimization algorithms. In this paper, a new optimization algorithm has been introduced, called Buzzard Optimization Algorithm (BUOA). BUOA is defined with simulation life behaviors of buzzards like prey capturing, finding carrions, smell and visual sense, defense method, eccentric features of the buzzards, the role of the ecosystem, black vultures opportunism, finding hunt, the smell of death, and the other behaviors. We have used a set of diverse standard benchmark functions for evaluating the performance of the presented algorithm. The simulations and the results which are obtained by BUZO algorithm have shown superior results in global optima achievement and fast convergence. The proposed algorithm has been compared with other meta-heuristics algorithms, as shown in above tables.

REFERENCES

[1] K. Suresh, N. Kumarappan, **"Hybrid Improved Binary Particle Swarm Optimization Approach for Generation Maintenance Scheduling Problem"**. *Swarm and Evolutionary Computation*. 9, pp. 69-89, 2013.

[2] S. M. Goldansaz, F. Jolai, and A.H.Z. Anaraki, **"A Hybrid Imperialist Competitive Algorithm for Minimizing Makespan in a Multiprocessor Open Shop".** *Applied Mathematical Modelling.* 37, pp. 9603-9616, (2013.

[3] G. a. A. G. Fornarelli, **"An Unsupervised Multi-Swarm Clustering Technique for Image Segmentation".** *Swarm and Evolutionary Computation.* 11, pp. 31-45, 2013.

[4] A. a. A. B. Draa, **"An Artificial Bee Colony Algorithm for Image Contrast Enhancement".** *Swarm and Evolutionary Computation*. 16, pp. 69-84, (2014.

[5] P. Moallem and N. Razmjooy, **"Optimal Threshold Computing in Automatic Image Thresholding using Adaptive Particle Swarm Optimization"**. *Journal of applied research and technology.* 10, pp. 703-712, 2012.

[6] P. Moallem, N. Razmjooy, and B. Mousavi, **"Robust Potato Color Image Segmentation using Adaptive Fuzzy Inference System".** *Iranian Journal of Fuzzy Systems.* 11, pp. 47-65, 2014.

[7] B. S. Mousavi, P. Sargolzaei, N. Razmjooy, V. Hosseinabadi, and F. Soleymani, **"Digital Image Segmentation Using Rule-Base Classifier"**. *American Journal of Scientific Researc*h ISSN. 2011.

[8] B. S. Mousavi and F. Soleymani, "**Semantic Image Classification by Genetic Algorithm Using Optimised Fuzzy System based on Zernike Moments".** S*ignal, Image and Video Processing.* 8, pp.831-842, 2014.

[9] N. Razmjooy, B. S. Mousavi, B. Sadeghi, and M. Khalilpour, **"Image Thresholding Optimization Based on Imperialist Competitive Algorithm,"** in *3rd Iranian Conference on Electrical and Electronics Engineering (ICEEE2011)*, 2011.

[10] N. Razmjooy, B. S. Mousavi, P. Sargolzaei, and F. Soleymani, **"Image Thresholding based on evolutionary Algorithms".** *International Journal of Physical Sciences. 6*, pp. 7203-7211, 2011.

[11] P. N. Suganthan, **"Structural Pattern Recognition Using Genetic Al*gorithms"***. *Pattern Recognition Letters. 35*, pp. 1883-1893, 2002.

[12] G. a. B. B. C. Garai, **"A Novel Hybrid Genetic Algorithm with Tabu Search for Optimizing Multi-Dimensional Functions and point Pattern Recognition".** *Information Sciences.* 221, pp. 28-48, 2013.

[13] R. a. D. K. P. Malviya, **"Tuning of Neural Networks Using Particle Swarm Optimization to Model MIG Welding Process".** *Swarm and Evolutionary Computation. 1*, pp. 223-235, 2011.

[14] S. J. Azadeh A, Sheikhalishahi M, Yazdani M, **"An Integrated Support Vector Regression–Imperialist Competitive Algorithm for Reliability Estimation of a Shearing Machine".** *International Journal of Computer Integrated Manufacturing*. 2015.

[15] S. Meysam Mousavi, et al., **"A New Support Vector Model-Based Imperialist Competitive Algorithm for Time Estimation in New Product Development Projects".** *Robotics and Computer-Integrated Manufacturing. 29*, pp. 157-168, 2013.

[16] P. Moallem and N. Razmjooy, **"A Multi Layer Perceptron Neural Network Trained by invasive Weed Optimization for Potato Color Image Segmentation".** *Trends in Applied Sciences Research. 7*, pp. 445, 2012.

[17] N. Razmjooy, B. S. Mousavi, and F. Soleymani, **"A hybrid Neural Network Imperialist Competitive Algorithm for Skin Color Segmentation".** *Mathematical and Computer Modelling. 57*, pp. 848-856, 2013.

[18] N. Razmjooy and M. Ramezani, **"Training Wavelet Neural Networks Using Hybrid Particle"** *Swarm Optimization and Gravitational Search Algorithm for System Identification*.

[19] N. Razmjooy, F. R. Sheykhahmad, and N. Ghadimi, **"A Hybrid Neural Network–World Cup Optimization Algorithm for Melanoma Detection".** *Open Medicine. 13*, pp. 9-16, 2018.

[20] J. Senthilnath, S.N. Omkar, and V. Mani, **"Clustering Using Firefly Algorithm: Performance Study"**. *Swarm and Evolutionary Computation. 1*, pp. 164-171, 2011.

[21] S. J. Nanda, G. Panda: **"A Survey on Nature Inspired Metaheuristic Algorithms for Partitional Clustering".** *Swarm and Evolutionary Computation.* 16, pp. 1-18, 2014.

[22] H. Hosseini, M. Farsadi, M. Khalilpour, and N. Razmjooy: **"Hybrid Energy Production System with PV Array and Wind Turbine and Pitch Angle"** *Optimal Control by Genetic Algorithm* (GA). 2011.

[23] H. Hosseini, M. Farsadi, A. Lak, H. Ghahramani, and N. Razmjooy: **"A Novel Method Using Imperialist Competitive Algorithm (ICA) for Controlling Pitch Angle in Hybrid Wind and PV Array Energy Production System".** I*nternational Journal on Technical and Physical Problems of Engineering (IJTPE).* pp. 145-152.

[24] H. Hosseini, B. Tousi, N. Razmjooy, and M. Khalilpour: **"Design robust Controller for automatic Generation Control In Restructured Power System by Imperialist Competitive Algorithm"**. *IETE Journal of Research.* 59, pp. 745-752, 2013.

[25] N. Razmjooy and M. Khalilpour: **"A New Design for PID Controller by Considering the Operating Points Changes in Hydro-Turbine Connected to the Equivalent Network by using Invasive Weed Optimization (IWO) Algorithm".** *International Journal of Information, Security and Systems Management. 4*, pp. 468-475, 2015.

[26] N. Razmjooy and M. Khalilpour: **"A Robust Controller For Power System Stabilizer By Using Artificial Bee Colony Algorithm".** *Tech J Engin & App Sci. 5*, pp. 106-113, 2015.

[27] N. Razmjooy, M. Khalilpour, and M. Ramezani: **"A New Meta-Heuristic Optimization Algorithm Inspired by FIFA World Cup Competitions: Theory and Its Application in PID Designing for AVR System".** *Journal of Control, Automation and Electrical Systems. 27*, pp. 419-440, 2016.

[28] V. Bhargava, S.E.K. Fateen, A. Bonilla-Petriciolet: Cuckoo Search: **"A New Nature-Inspired Optimization Method for Phase Equilibrium Calculations".** *Fluid Phase Equilibria.* 337, pp. 191-200, 2013.

[29] Y.-J. Zheng, Water wave optimization: **"A New Nature-Inspired Metaheuristic".** *Computers & Operations Research. 55*, pp. 1-11, 2015.

[30] F.-C. Yang, Y.-P. Wang: **"Water Flow-Like Algorithm for object Grouping Problems".** *Journal of the Chinese Institute of Industrial Engineers. 24*, pp. 475-488, 2007.

[31] A. Mucherino, O. Seref. Monkey search: **"A Novel Metaheuristic Search for Global Optimization".** in *Data Mining. Systems Analysis and Optimization in Biomedicine*. 2007.

[32] H. A. M. Abbass: **"Marriage in Honey Bees Optimization-A Haplometrosis Polygynous Swarming Approach"** *in Evolutionary Computation. Proceedings of the 2001 Congress on. 2001. IEE*E. 2001.

[33] X.-S. a. S. D. Yang: "**Cuckoo search via Lévy flights. in Nature & Biologically Inspired Computing".** *NaBIC 2009. World Congress on*. 2009. IEEE. 2009.

[34] R. Rajabioun: **"Cuckoo Optimization Algorithm".** *Applied Soft Computing. 11. 8*, pp. 5508-5518, 2011.

[35] X.-S. Yang: **"Firefly Algorithms for Multimodal Optimization",** i*n Stochastic algorithms: foundations and applications. Springer*. pp. 169-178, 2009.

[36] D. Simon: **"Biogeography-based Optimization. Evolutionary Computation"**. *IEEE Transactions on*. 12, pp. 702-713, 2008.

[37] R. C. a. J. K. Eberhart: **"A New Optimizer using Particle Swarm Theory. in Proceedings of the Sixth International Symposium on Micro Machine And Human Science".** *New York*, NY. 1995.

[38] S.-C. Chu, P.-W. Tsai, J.-S. Pan,: **"Cat swarm optimization, in PRICAI 2006: Trends in Artificial Intelligence".** *Springer*. pp. 854-858, 2006.

[39] K. M. Passino: **"Biomimicry of Bacterial Foraging for Distributed Optimization and Control".** *Control Systems. IEEE. 22*, pp. 52-67, 2002.

[40] A. H. a. A. H. A. Gandomi, Krill herd: **"A New Bio-Inspired Optimization Algorithm".** *Communications in Nonlinear Science and Numerical Simulation*. 17, pp. 4831-4845, 2012.

[41] J. H. Holland: **"Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence".** *Ann Arbor, University of Michigan Press.*, USA. 1975.

[42] Y. Shiqin, J. Jianjun, and Y. Guangxing: **"A dolphin partner optimization. in Intelligent Systems".** *GCIS'09. WRI Global Congress on. 2009. IEEE*. 2009.

[43] A. a. N. F. Kaveh: **"A New Optimization Method: Dolphin Echolocation"**. *Advances in Engineering Software*. 59, pp. 53-70, 2013.

[44] X.-S. Yang: **"A New Metaheuristic Bat-Inspired Algorithm",** *in Nature inspired cooperative strategies for optimization (NICSO 2010). Springer*. pp. 65-74, 2010.

[45] M. Dorigo: Optimization, learning and natural algorithms. 1992.

[46] F. de Lima Neto, et al., **"A novel search algorithm based on fish school behavior. in Systems, Man and Cybernetics".** *SMC 2008. IEEE International Conference on. 2008. IEEE*. 2008.

[47] J. D. Farmer, N.H. Packard, and A.S. Perelson: **"The Immune System, Adaptation, and Machine Learning. Physica D: Nonlinear Phenomena".** *Physica D: Nonlinear Phenomena. 22*, pp. 187-204, 1986.

[48] R. Tang, et al., **"Wolf Search Algorithm With Ephemeral Memory".** *in Digital Information Management (ICDIM). 2012 Seventh International Conference on. 2012. IEEE*. 2012.

[49] S. Mirjalili, S.M. Mirjalili, and A. Lewis: **"Grey Wolf Optimizer".** *Advances in Engineering Software. Advances in Engineering Software*. 69, pp. 46-61, 2014.

[50] E. Cuevas, et al., **"A Swarm Optimization Algorithm Inspired in the Behavior of the Social-Spider".** *Expert Systems with Applications. 40*, pp. 6374-6384, 2013.

[51] X.-S. Yang: **"Flower Pollination Algorithm for Global Optimization, in Unconventional Computation and Natural Computation"**. *Springer*. pp. 240-249, 2012.

[52] M. a. M.-R. F.-D. Ghaemi: **"Forest Optimization Algorithm".** *Expert Systems with Applications. 41*, pp. 6676-6687, 2014.

[53] H. Eskandar, et al., **"Water Cycle Algorithm – A Novel Metaheuristic Optimization Method for Solving Constrained Engineering Optimization Problems".** *Computers & Structures*. 110–111, pp. 151-166, 2012.

[54] M. M. a. K. E. L. Eusuff: **"Optimization of Water Distribution Network Design using the Shuffled Frog Leaping Algorithm".** *Journal of Water Resources Planning and Management. 129*, pp. 210-225, 2003.

[55] A. R. a. C. L. Mehrabian: **"A Novel Numerical Optimization Algorithm Inspired from Weed Colonization".** 1, pp. 355-366, 2006.

[56] D. a. D. R. Arivudainambi: **"Memetic Algorithm for Minimum Energy Broadcast Problem in Wireless Ad Hoc Networks".** *Swarm and Evolutionary Computation. 12*, pp. 57-64, 2013.

[57] J. Hofmann, S. Limmer, and D. Fey: **"Performance Investigations of Genetic Algorithms on Graphics Cards".** *Swarm and Evolutionary Computation. 12*, pp. 33-47, 2013.

[58] S. A. Ludwig: **"Memetic Algorithms Applied to the Optimization of Workflow Compositions"**. *Swarm and Evolutionary Computation. 10*, pp. 31-40, 2013.

[59] C. Changdar, G.S. Mahapatra, and R. Kumar Pal: **"An Efficient Genetic Algorithm for Multi-Objective Solid Travelling Salesman Problem Under Fuzziness".** *Swarm and Evolutionary Computation,* . 15, pp. 27-37, 2014.

[60] D. H. a. W. G. M. Wolpert: **"No Free Lunch Theorems for Optimization".** *Evolutionary Computation,. IEEE Transactions on. 1*, pp. 67-82, 1997.

[61] J. Liang, B. Qu, and P. Suganthan: **"Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-parameter Numerical optimization".** *Computational Intelligence Laboratory.* 2013.

[62] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi: **"GSA: A Gravitational Search Algorithm".** *Information sciences. 179,* pp. 2232-2248, 2009.

[63] R. Oftadeh, M.J. Mahjoob, and M. Shariatpanahi: **"A Novel Meta-Heuristic Optimization Algorithm Inspired by Group Hunting of Animals: Hunting Search".** *Computers & Mathematics with Applications. 60,* pp. 2087-2098, 2010.

[64] Y.-J. Zheng: Water wave optimization: **"A New Nature-Inspired Metaheuristic".** *Computers & Operations Research.* 2014.