# TWO DISCRETE BINARY VERSIONS OF AFRICAN BUFFALO OPTIMIZATION METAHEURISTIC

Amira GHERBOUDJ

MISC laboratory, NTIC faculty, AbdelhamidMehri University Constantine 2, Algeria

## *ABSTRACT*

*African Buffalo Optimization (ABO) is one of the most recent swarms intelligence based metaheuristics. ABO algorithm is inspired by the buffalo's behavior and lifestyle. Unfortunately, the standard ABO algorithm is proposed only for continuous optimization problems. In this paper, the authors propose two discrete binary ABO algorithms to deal with binary optimization problems. In the first version (called SBABO) they use the sigmoid function and probability model to generate binary solutions. In the second version (called LBABO) they use some logical operator to operate the binary solutions. Computational results on two knapsack problems (KP and MKP) instances show the effectiveness of the proposed algorithm and their ability to achieve good and promising solutions.*

## *KEYWORDS*

*Optimization, metaheuristic, swarm intelligence, binary problems, African Buffalo Optimization, knapsack problems*

## 1. INTRODUCTION

Solving optimization problem is finding a solution of sufficient quality (i.e. optimal or near optimal) from a set of solutions taking into account the constraints imposed and objectives to meet. It is to maximize or minimize one or a set of fitness functions respecting constraints of the treated problem. This research line that sought the attention of several research teams is intrinsically linked to operational research and it uses mathematical and computer tricks.

Various Methods have been proposed to solve optimization problems. They are often classified into two classes: exact methods and approximate methods. The prohibitive cost associated with exact methods has excited researchers to use approximate methods. The investigation in the area of approximate methods gave rise to a subclass of approximate methods called "Metaheuristics". They are general and applicable methods on a wide range of optimization problems.

Metaheuristics based on swarm intelligence have built a very active trend over the last decade. They are generally inspired by the collective behavior of some species in evolving in groups and solving their problems. These species gather in swarm to build a collective force that allows them to surpass their very limited individual capacities.

The African Buffalo Optimization (ABO) is one of the most recent swarm intelligence based Metaheuristics. It was proposed in 2015, by Julius BeneoluchiOdili et al [1]. ABO is inspired from the behavior of African buffaloes in the vast African forests and savannahs [1]. The recent applications of ABO Metaheuristic for optimization problems have shown its promising effectiveness as it has been proven in [2], [3] and [4].The original ABO algorithm is a continuous version, which solves only continuous problems. The aim of this paper is to propose two binary versions of ABO algorithm to cope with binary optimization problems. The main difference between the original version of ABO algorithm and the proposed binary versions is that, in the original ABO algorithm, the solution is composed of a set of real numbers. While in the proposed binary versions, the solution is composed of a set of bits.

The remainder of this paper is organized as follows. Section 2 presents an overview of African Buffalo Optimization (ABO) Algorithm. The proposed algorithms (SBABO and LBABO) are presented in section 3. Experimental results are presented and discussed in section 4, and a conclusion and perspectives are provided in the fifth section of this paper.

## 2. AFRICAN BUFFALO OPTIMIZATION

In order to solve complex problems, ideas gleaned from natural mechanisms have been exploited to develop heuristics. Nature inspired optimization algorithms has been extensively investigated during the last decade paving the way for new computing paradigms such as neural networks, evolutionary computing, swarm optimization, etc. The ultimate goal is to develop systems that have the ability to learn incrementally, to be adaptable to their environment and to be tolerant to noise. One of the recent developed bioinspired algorithms is the African Buffalo Optimization algorithm.

The African Buffalo Optimization is inspired from the cooperative and competitive behavior of buffaloes. ABO models the three characteristic behaviors of the African buffaloes that enable their search for pastures. First is their extensive memory capacity. This enables the buffaloes to keep track of their routes. The second attribute of the buffaloes is their cooperative communicative ability whether in good or bad times. The third attribute of the buffaloes is their democratic nature borne out of extreme intelligence. In cases where there are opposing calls by members of the herd, the buffaloes have a way of doing an ‘election’ where the majority decision determines the next line of action [1]. Furthermore, ABO algorithm models the two sounds for communication that buffaloes use to exploit and explore the search space:

- The warning sound “waaa” with which they ask the herd to keep moving because the present location is unfavorable, lacks pasture or is dangerous. This sound encourages the buffaloes to explore the research space.

- The alert sound “maaa” with which they stay on the present location because it is holds promise of good grazing pastures and is safe. This sound encourages the buffaloes to exploit the research space

Algorithm1 presents a pseudo algorithm of the ABO method.

**Algorithm 1: ABO Algorithm**

```
1. Objective function $f(x) = (x_1,x_2,\ldots x_n)^T$;
2. Initialization: randomly place buffaloes to nodes at the solution space;
3. Update the buffaloes fitness values using (1);
4. Update the location of buffalo k ($bpmax._k$ and bgmax) using (2);
5. Is bgmax updating. Yes, go to 6. No, go to 2;
6. If the stopping criteria is not met, go back to algorithm step 3, else go to step 7;
7. Output the best solution.
```

The generation of new solutions is done by using equations 1 and 2.

$$m._{k+1} = m._k + lp_1(bg\ \max - w._k) + lp_2(bp\ \max._k - w._k)\ (1)$$

$$w._{k+1} = \frac{w._k + m._k}{- +0.5}\ (2)$$

Where:

- $w._k$ and $m._k$ presents the exploration and exploitation moves respectively of the kth buffalo (k=1,2,………..N) ;
- $lp_1$ and $lp_2$ are learning factors;
- bgmax is the herd's best fitness;
- $bpmax._k$ the individual buffalo's best.

## 3. The Proposed Discrete Binary Versions

Optimization problems can be classed into two main classes: continuous optimization problems and discrete optimization problems. In continuous optimization problems, the solution is presented by a set of real numbers. However, in discrete optimization problems, the solution is presented by a set of integer numbers. Discrete binary optimization problems are a sub-class of the discrete optimization problems class, in which a solution is presented by a set of bits. Many optimization problems can be modelled as a discrete binary search space such as, flowshop scheduling problem [5], job-shop scheduling problem [6], routing problems [7], KP [8] and its variants such as the MKP [9], the quadratic KP [10], the quadratic multiple KP [11] and so one.

The original ABO algorithm operates in continuous search space. It gives a set of real numbers as a solution of the handled problem. However, a binary optimization problem needs a binary solution and the real solutions are not acceptable because they are considered as illegal solutions. In the aim to extend the ABO algorithm to discrete binary areas, we propose in this paper tow binary versions of ABO that we called SBABO and LBABO. The main objective of the SBABO and LBABO algorithms is to deal with the binary optimization problems.

## 3.1 SBABO ALGORITHM

In the SBABO algorithm, we introduce a binarization phase of solutions in the core of the original ABO in order to obtain a binary solution for the treated problem. The objective of this phase (i.ebinarization) is to transform a solution xi from real area to binary area. To meet this need, we propose to constrain the solution xi in the interval [0, 1] using the Sigmoid Function as follows:

$$S(x_i) = \frac{1}{(1+e^{-x_i})} \tag{3}$$

Where $S(x_i)$ is the flipping chance of bit $x'_i$. It presents the probability of bit $x'_i$ takes the value 1. To obtain the binary solution $x'_i$, we have to generate a random number from the interval [0,1] for each dimension i of the solution x and compare it with the flipping chance $S(x_i)$ as mentioned below in equation (4). If the random number is lower than the flipping chance of bit $x'_i$, $x'_i$ takes the value 1. Otherwise, $x'_i$ takes the value 0.

$$x'_i = \begin{cases} 1 & \text{If } r < S(x_i),\ r \in [0,\ 1] \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Consequently, having a solution $x_i$ encoded as a set of real numbers, the sigmoid function is used to transform the solution $x_i$ into a set of probabilities that presents the chance for bit i to be flipping. The flipping chance is then used to compute the binary solution $x'_i$. Algorithm 2 presents the SBABO algorithm an Algorithm 3 presents the binarization algorithm

**Algorithm 2: SBABO Algorithm**

```
1. Objective function $f(x) = (x_1,x_2,\ldots x_n)^T$
2. Initialization: randomly place buffaloes to nodes at the solution space;
3.Update the buffalo's fitness values using (1);
4.Get the binary buffaloes using the Binarization Algorithm;
5. Update the location of buffalo k ($bpmax._k$ and bgmax) using (2);
6. Is bgmax updating. Yes, go to 7. No, go to 2;
7. If the stopping criteria is not met, go back to algorithm step 3, else go to step 8.
8. Output the best solution.
```

**Algorithm 3:Binarization Algorithm**

```
Input: Real solution presentation $x_i$
Output: Binary solution presentation $x_i'$
For (i = 1 to (problem size)) {
    Calculate $S(x_i)$ using (3);
    If (random number $r < S(x_i)$)
$x'_i$=1;
    Otherwise
$x'_i$ =0;
}
```

## 3.2 LBABO ALGORITHM

In the LBABO algorithm, we propose to start the search by a binary population (the solutions are binary from the beginning) and replace the arithmetic operators used in the solution update equations (i.e. 1 and 2) by logical operators as follow:

1) $val_2$-$val_1$

Assuming two values $val_1$ and $val_2$:

$$val_2 - val_1 = \begin{cases} val_2 \text{ if } val_2 \# val_1 \\ \text{not } (val_1) \text{ otherwise (i.e. } val \ _2 = val \end{cases}$$

2) $val_2$+$val_1$

$$val_2 + val_1 = \begin{cases} \text{not } (val_1) \text{ if } val_2 \# val_1 \\ \text{and } (val_2, val_1) \text{ otherwise (i.e.} \end{cases}$$

3) Coefficient * (or /) val

$$\text{Coefficient } *(/) \text{ val} = \begin{cases} \text{val if coefficient} > r,\ r \in [0,1] \\ \text{not (val) otherwise} \end{cases}$$

Coefficient can be $lp_1$ or $lp_2$. Algorithm 4 presents the LBABO algorithm

**Algorithm 4. LBABO Algorithm**

```
1. Objective function f(x) = (x1,x2,…xn)^T;
2. Initialization: randomly place buffaloes to nodes at the solution space **using binary values**;
3. Update the buffalo's fitness values using (1) **and logical operators**;
4. Update the location of buffalo k (bpmax.k and bgmax) using (2) **and logical operators**;
5. Is bgmax updating. Yes, go to 6. No, go to 2;
6. If the stopping criteria is not met, go back to algorithm step 3, else go to step 7.
7. Output the best solution.
```

## 4. EXPERIMENTAL RESULTS

In order to investigate the performance of the proposed algorithms to solve hard binary optimization problems, we used some knapsack problem benchmarks of two knapsack problem versions: Single knapsack problem (KP) and multidimensional knapsack problem (MKP).

### 4.1 KP AND MKP PROBLEMS

The KP is a NP-hard problem [12]. Numerous practical applications of the KP can be found in many areas involving resource distribution, investment decision making, budget controlling, project selection and so one. The KP can be defined as follows: Assuming that we have a knapsack with maximum capacity C and a set of N objects. Each object i has a profit $p_i$ and a weight $w_i$. The problem consists to select a subset of objects that maximize the knapsack profit without exceeding the maximum capacity of the knapsack. The problem can be formulated as [12]:

$$Maximize \sum_{i=1}^{N} p_i x_i \tag{5}$$

$$Subject \sum_{i=1}^{N} w_i x_i \leq C \tag{6}$$

$$x_i \in, \{0,1\}$$

Many variants of the KP were proposed in the literature including the MKP. MKP is an important issue in the class of KP. It is a NP-hard problem [13]. In the MKP, each item $x_i$ has a profit $p_i$ like in the simple KP. However, instead of having a single knapsack to fill, we have a number M of knapsack of capacity $C_j$ (j = 1, …, M). Each $x_i$ has a weight $w_{ij}$ that depends of the knapsack j (example: an object can have a weight 3 in knapsack 1, 5 in knapsack 2, etc.). A selected object must be in all knapsacks. The objective in MKP is to find a subset of objects that maximize the total profit without exceeding the capacity of all dimensions of the knapsack. MKP can be stated as follows [14]:

$$Maximize \sum_{i=1}^{N} p_i x_i \tag{7}$$

$$Subject \sum_{i=1}^{N} w_{ij} x_i \leq C_j \tag{8}$$

$$J=1,\ldots, M \text{ and } \; x_i \in, \{0,1\}$$

The MKP can be used to formulate many industrial problems such as the capital budgeting problem, allocating processors and databases in a distributed computer system, cutting stock, project selection and cargo loading problems [15]. Clearly, there are $2^N$ potential solutions for these problems. It is obviously that KP and its variants are combinatorial optimization problems. Several techniques have been proposed to deal with KPs [12]. However, it appears to be impossible to obtain exact solutions in polynomial time. The main reason is that the required computation grows exponentially with the size of the problem. Therefore, it is often desirable to find near optimal solutions to these problems.

## 4.2 EXPERIMENTAL DATA

The proposed SBABO and LBABO algorithms were implemented in MATLAB R2014a. Using a laptop computer running Windows 7, Intel(R) Core(TM) i3-3110M CPU@ 2.40 GHz, 2.40GHz, 4GB RAM. The used parameters in the experiments are:

- Population size: 40.
- Iterations: 300.
- lp1=0.7.
- lp2=0.5.

Several experiments were performed to assess the efficiency and performance of our algorithms. In the first experiment, we have tested and compared our algorithms with a harmony search algorithm (NGHS) on some small KP instances taken from [16]. In the second experiment, we have used some big KP instances used in [8] to test and compare the proposed SBABO algorithm with the Binary Particle Swarm Optimization algorithm (BPSO) [17] which has a common point with the proposed SBABO algorithm. In fact, the two algorithms (SBABO and BPSO) used the Sigmoid Function to generate the binary solution. The used instances are six different instances with different problem sizes, in which the weights and profits are selected randomly. The different problem sizes N are 120, 200, 500, 700, 900 and 1000. In these instances, the knapsack capacity is calculated by using equation 9 [8]. The factor 3/4 indicates that about 75% of items are in the optimal solution.

$$C = \frac{3}{4}\sum_{i=0}^{N} w_i \tag{9}$$

In the third experiment, we have evaluated the performance of our algorithms on some MKP benchmarks taken from OR-Library. We have tested the proposed algorithms on some MKP instances taken from seven benchmarks named mknap1. The obtained results are compared with the exact solution (best known).

Finally, statistical tests of Freidman are carried out to test the significance of the difference in the accuracy of each method in this experiment. And the performances of the proposed algorithms (SBABO and LBABO) are also compared in terms of execution CPU time with the two problems (KP and MKP).

## 4.3. RESULTS AND DISCUSSION

Table 1 shows the experimental results of our algorithms (SBABO and LBABO) and the harmony search algorithm (NGHS) on ten KP tests with different sizes. The first column, indicates the instance name, the second column indicates the problem size, i.e. number of objects. The third and fourth columns indicate the obtained results by the SBABO and LBABO algorithms and the last column indicates the obtained results by the NGHS algorithm. Observation of the presented results in Table 1 indicates that the proposed discrete binary algorithms (i.e: SBABO and LBABO) perform well than NGHS algorithm in F6 test. The SBABO perform well than LBABO and NGHS algorithms in F8 test. And the three algorithms have the same results in the other instances.

Table 1.Experimental results with small kp instances

| *Test* | *Size* | *SBABO* | *LBABO* | *NGHS* |
|---|---|---|---|---|
| F1 | 10 | 295 | 295 | 295 |
| F2 | 20 | 1024 | 1024 | 1024 |
| F3 | 4 | 35 | 35 | 35 |
| F4 | 4 | 23 | 23 | 23 |
| F5 | 15 | 481.0694 | 481.0694 | 481.0694 |
| F6 | 10 | 52 | 52 | 50 |
| F7 | 7 | 107 | 107 | 107 |
| F8 | 23 | 9767 | 9767 | 9767 |
| F9 | 5 | 130 | 130 | 130 |
| F10 | 20 | 1025 | 1025 | 1025 |

Table 2 shows the experimental results of SBABO, LBABO and BPSO algorithms on big KP instances. The first column presents the problem size (i.e., instance). The second, third and fourth columns present the obtained results by the BPSO, SBABO and LBABO algorithms respectively. With each instance, the first line presents the best solutions and the second one presents the solution averages. The presented results show that the SBABO and LBABO algorithms outperform the BPSO algorithm, and the LBABO algorithm outperforms the SBABO algorithm. The statistical Friedman test in Figure 1 presents a comparison of the BPSO, SBABO and LBABO results. The LBABO algorithm ranks first in the Friedman test. The SBABO ranks second and BPSO ranks third. This statistical test shows that there is a significant difference between LBABO and BPSO algorithms.

Table 2. Experimental results with big kp instances.

| Instance | BPSO | SBABO | LBABO |
|---|---|---|---|
| | 4296 | 4316 | 4504 |
| **120** | 3840.8 | 4088.09 | 4357 |
| | 7456 | 6778 | 7530 |
| **200** | 5703 | 6480.56 | 7284.22 |
| | 13116 | 14730 | 16853 |
| **500** | 12471.2 | 14396.11 | 16174.25 |
| | 18276 | 20501 | 23278 |
| **700** | 17097.4 | 19348.07 | 22530.4 |
| | 22857 | 24767 | 30196 |
| **900** | 21736.6 | 24270.83 | 28864.5 |
| | 24933 | 27306 | 32948 |
| **1000** | 24050 | 26607.3 | 31936.86 |

Whereas, the difference between LBABO and SBABO results is not statistically significant. Consequently, the obtained results confirm that the proposed algorithms outperform the BPSO algorithm and prove that the proposed algorithms give good results.

Table 3 shows experimental results of the proposed algorithms over 7 instances of MKP problem. The first column indicates the instance index. The second and third column indicates the number of object and knapsack dimension, respectively. The fourth, fifth and sixth columns indicate the best known, the SBABO and the LBABO solutions, respectively. As we can see, the SBABO algorithm is able to find the best solution of the six first instances from the 7 instances. The LBABO algorithm is able to find the best solution of the five first instances from the 7 instances. The SBABO algorithm outperforms the LBABO algorithm on the two last instances (6 and 7). The statistical Friedman test in Figure 2 shows a comparison of the best known, SBABO and LBABO results. The SBABO ranks second after best known and LBABO ranks third. This statistical test shows that the difference between best known, LBABO and SBABO results is not statistically significant. Consequently, the obtained results confirm and prove that the proposed algorithms give good and promising results that can be considerably increased by the introduction of some specified knapsack heuristic operators using problem specific knowledge.

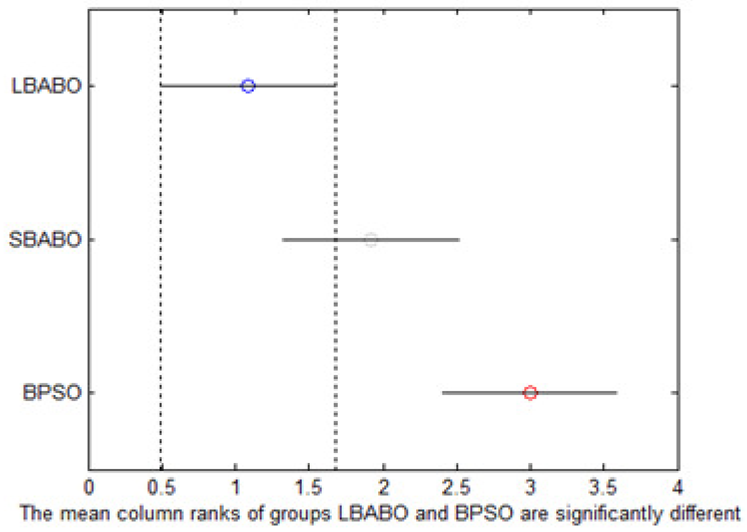

Figure 1. Friedman test compares SBABO, LBABO and BPSO on big KP instances.

Table 3.Experimental results of MKP with mknap1 instances

| N° | N | M | Best Known | SBABO | LBABO |
|---|---|---|---|---|---|
| 1 | 6 | 10 | 3800 | 3800 | 3800 |
| 2 | 10 | 10 | 8706,1 | 8706,1 | 8706,1 |
| 3 | 15 | 10 | 4015 | 4015 | 4015 |
| 4 | 20 | 10 | 6120 | 6120 | 6120 |
| 5 | 28 | 10 | 12400 | 12400 | 12400 |
| 6 | 39 | 5 | 10618 | 10618 | 10554 |
| 7 | 50 | 5 | 16537 | 16442 | 16371 |

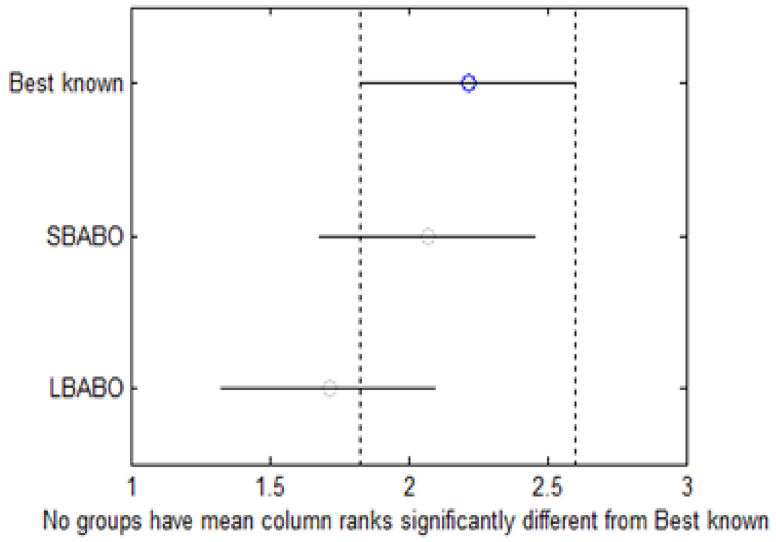

Figure 2. Friedman test compares SBABO, LBABO and best known on MKP instances.

Table 4 and 5 show a comparison of average computation time with KP and MKP instances, estimated by seconds, using a population of 40 solutions, 300 iterations with 5 executions of the programs. The obtained results are schematized in Figures 3 and 4. In terms of computing time, the obtained results do not show a big difference in execution time. In fact, in some instances SBABO is faster and in others LBABO is faster. In general, the two algorithms converge in the same interval time. This comes back to the fact that the two algorithms have the same body, only the phase of binarization of the solution that differs.

Table 4.Comparative CPU time with KP instances.

| ***Test*** | ***Size*** | ***SBABO*** | ***LBABO*** |
|---|---|---|---|
| **F1** | 10 | 2.50 | 2.75 |
| **F2** | 20 | 2.16 | 2.49 |
| **F3** | 4 | 1.67 | 1.48 |
| **F4** | 4 | 1.85 | 1.63 |
| **F5** | 15 | 2.03 | 2.73 |
| **F6** | 10 | 1.59 | 1.94 |
| **F7** | 7 | 2.08 | 2.30 |
| **F8** | 23 | 2.01 | 3.54 |
| **F9** | 5 | 2.02 | 2.14 |
| **F10** | 20 | 2.21 | 2.96 |

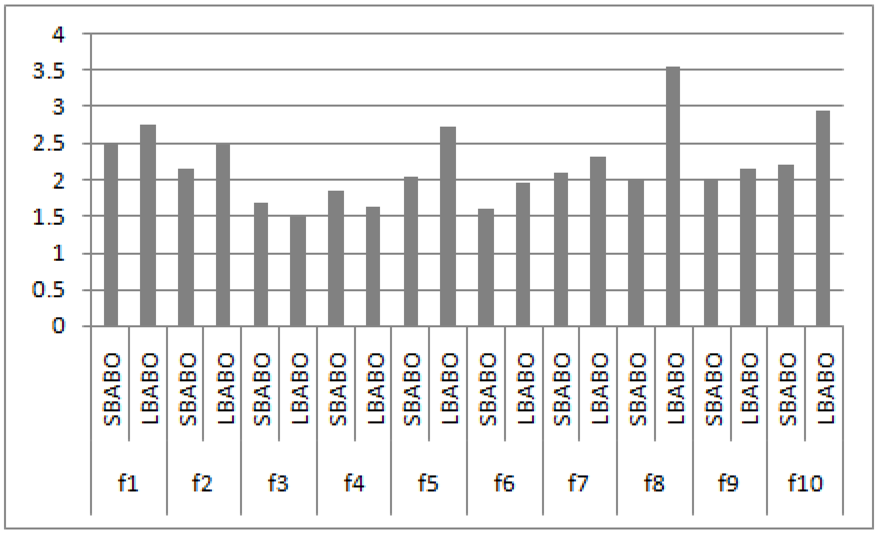

Figure 3.CPU time with KP instances

Table 5.Comparative CPU time with MKP instances

| N° | n | M | SBABO | LBABO |
|---|---|---|---|---|
| **1** | 6 | 10 | 3.09 | 2.99 |
| **2** | 10 | 10 | 3.38 | 2.72 |
| **3** | 15 | 10 | 3.49 | 3.30 |
| **4** | 20 | 10 | 3.55 | 4.28 |
| **5** | 28 | 10 | 3.75 | 5.11 |
| **6** | 39 | 5 | 4.97 | 5.45 |
| **7** | 50 | 5 | 5.95 | 5.83 |

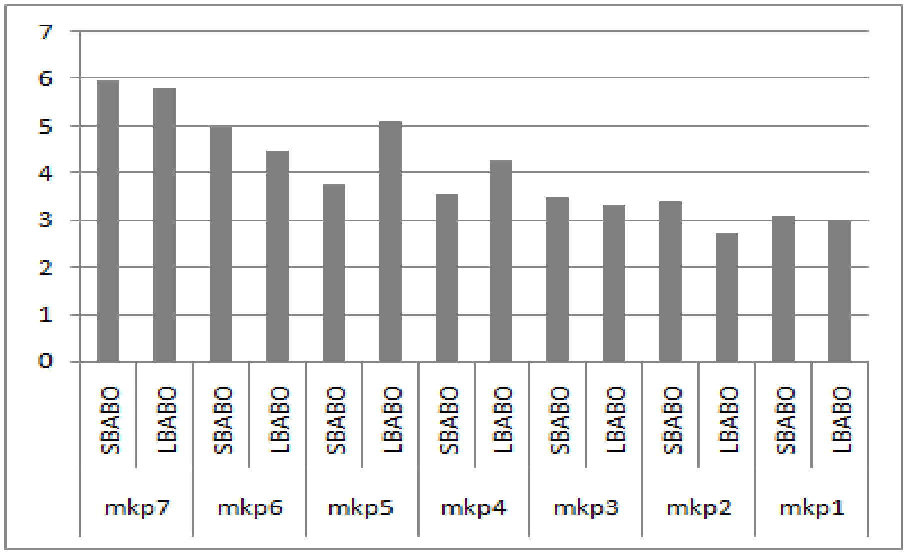

Figure 4.CPU time with MKP instances.

The ABO algorithm is a new swarm optimization algorithm. Considering its young age, there are few applications in optimization problems based on ABO algorithm. The main purpose of this paper is to validate that the ABO method is also effective for binary combinatorial optimization problems. That is why we proposed two discrete binary versions of ABO algorithm (called SBABO and LBABO) which led to two efficient ABO algorithm versions to deal with binary optimization problems.

During the different experiments, we noticed that SBABO algorithm explored the search space better than LBABO (see Figure 5). This comes down to the use of the Sigmoid Function and probability model to generate binary solutions. As shown in Figure 5, LBABO converges faster than SBABO which explains its results with MKP instances. It is notable that the performance of the algorithm is insensitive to their parameters such as $lp_1$ and $lp_2$. These two parameters influence the good balance between exploration and exploitation of the search space. The diversity of the proposed algorithms is assured by the use of the elitism selection which guarantees that the best solutions are kept in each generation. The proposed algorithms can be implemented easily for other binary optimization problems.

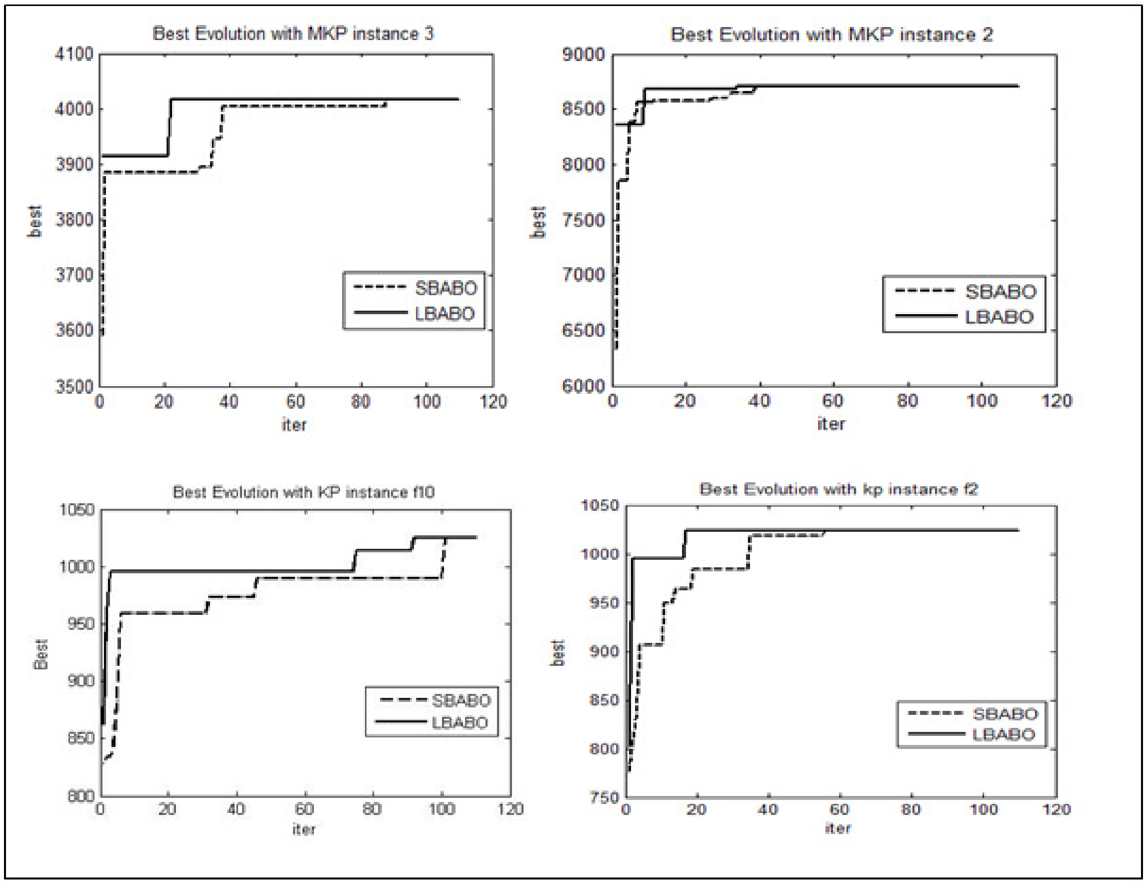

Figure 5. Evolution of best solution with KP and MKP using SBABO and LBABO

## 5. CONCLUSION AND PERSPECTIVES

In this paper, two discrete binary versions of African Buffalo Optimization algorithm are proposed. This contribution has two-fold aims: the first aim is to propose a binary version of ABO algorithm to deal with binary optimization problems. The second aim is to prove the effectiveness of the ABO algorithm in solving NP-hard combinatorial optimization problems. In the first version called SBABO we used the sigmoid function and probability model to generate binary solutions. In the second version called LBABO we used some logical operator to operate the binary solution. The proposed algorithms are used to solving two NP-hard binary combinatorial optimization problems: KP and MKP problems. The obtained results are compared with the harmony search algorithm (NGHS), the best known solution and the Binary Particle Swarm Optimization algorithm (BPSO) which has a common point with the proposed SBABO algorithm (the two algorithms used the sigmoid function). The experimental studies prove the feasibility and the effectiveness of the proposed algorithms. They proved that the proposed algorithms give good and promising results. However, there are several issues to improve the proposed algorithms. Firstly, in order to improve the performance of the proposed algorithms, we recommend integrating of a local search method in the algorithms core. In addition, hybridization with other operations inspired by other popular algorithms such as Genetic algorithm, Particle Swarm Optimization or Cuckoo Search will also be potentially fruitful. The proposed algorithms can be also applied to solve many other binary optimization problems and real industrial problems.

## REFERENCES

[1] Odili J.B, Kahar M.N.M, Anwar S. African Buffalo Optimization: A Swarm-Intelligence Technique. Procedia Computer Science 76. Elsevier, 2015. 443 – 448.

[2] Odili J.B, Kahar M.N.M. Solving the Traveling Salesman's Problem Using the African Buffalo Optimization. Computational Intelligence and Neuroscience.Volume 2016, Article ID 1510256. Hindawi Publishing Corporation, 2015.

[3] Odili J, Kahar M. N. M, Noraziah A and Kamarulzaman. S. F. A comparative evaluation of swarm intelligence techniques for solving combinatorial optimization problems. International Journal of Advanced Robotic Systems. DOI: 10.1177/1729881417705969. 2017.

[4] Padmapriya R. Maheswari D. Channel Allocation Optimization using African Buffalo Optimization-Super Vector Machine for Networks. Asian Journal of Information Technology, 2017. DOI: 10.3923/ajit.2017.783.788.16: 783-788.

[5] Liao C.J, Tseng C.T. and Luarn P. A discrete version of particle swarm optimization for flowshop scheduling problems. Computers & Operations Research. Elsevier, 2007. Vol. 34, No. 10, pp.3099–3111.

[6] Huang S. H, Tian N, Wang.Y and Ji Z. Multi-objective flexible job-shop scheduling problem using modified discrete particle swarm optimization.Springer Plus, 2016. 5:1432.

[7] Ammi M, Chikhi S. Cooperative Parallel Metaheuristics based Penguin Optimization Search for Solving the Vehicle Routing Problem. International Journal of Applied Metaheuristic Computing, 2016. Vol 7. Issue 1.

[8] Gherboudj, A. and Chikhi, S. BPSO algorithms for knapsack problem. In Özcan, A., Zizka, J. and Nagamalai, D. (Eds.): WiMo/CoNeCo, CCIS, Springer. 2011. Vol. 162, pp.217–227.

[9] Kong M. and Tian P. Apply the particle swarm optimization to the multidimensional knapsack problem. inRutkowski, L. et al. (Eds.): Proc. ICAISC 2006, LNAI, Springer.2006. Vol. 4029, pp.1140–1149.

[10] Julstrom B.A. Greedy. Genetic, and greedy genetic algorithms for the quadratic knapsack problem. In Proc. GECCO '05 Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, Publisher, ACM, 2005. pp.607–614.

[11] Singh A. and Baghel A.S. A new grouping genetic algorithm for the quadratic multiple knapsack problem. In Cotta, C. and van Hemert, J. (Eds.): Proc. EvoCOP 2007, LNCS, Springer. 2007. Vol. 4446, pp.210–218.

[12] Pisinger D. Where are the hard knapsack problems? Computers and Operations Research, 2005. Vol. 32, No. 9, pp.2271–2284.

[13] Chu P.C, Beasley J.E. A genetic algorithm for the multidimensional knapsack problem. Journal of Heuristics,1998. Vol. 4, No. 1, pp.63–86.

[14] Angelelli E, Mansini R. and Speranza M.G. Kernel search: a general heuristic for the multi-dimensional knapsack problem. Computers & Operations Research, Elsevier, 2010. Vol. 37, No. 13, pp.2017–2026.

[15] Vasquez M, Vimont Y. Improved results on the 0-1 multidimensional knapsack problem. European Journal of Operational Research,2005. Vol. 165, No. 1, pp.70–81.

[16] Zou D, Gao L, Li S. and Wu J. Solving 0-1 knapsack problem by a novel global harmony search algorithm. Applied Soft Computing, The Impact of Soft Computing for the Progress of Artificial Intelligence, March, 2011. Vol. 11, No. 2, pp.1556–1564.

[17] Kennedy J, Eberhart R C. A discrete binary version of the particle swarm algorithm. In Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, Piscataway, NJ,1997. pp.4104–4109.

## AUTHOR

**Amira Gherboudj** is Senior Lecturer at the Algerian University of "Frères Mentouri, Constantine 1". Dr. Gherboudj received her PhD degree in computer science in 2013 from the University of "AbdelhamidMehri, Constantine 2".Her research interests include combinatorial optimization methods and their applications to solve several problems from different domains.