# Comparative Study of Cuckoo Inspired Metaheuristics Applying to Knapsack Problems

Amira Gherboudj
MISC laboratory,
NTIC faculty,
Abdelhamid Mehri University,
Constantine, Algeria

## ABSTRACT
Cuckoo Optimization Algorithm (COA) and Cuckoo Search Algorithm (CS) are two population-based metaheuristics. They are based on the cuckoo's behavior in their lifestyle and their characteristics in egg laying and breeding. Both algorithms are proposed for continuous optimization problems. In this paper, we propose a comparative study of COA and CS. For this we have proposed a binary version of COA (called BCOA) algorithm using the Sigmoid function like we have do in a later work, in which we have proposed a binary version of CS algorithm that we have called BCS. In aim to compare the efficiency of the too algorithms, we have used the proposed BCOA to resolve knapsack problem (KP) and Multidimensional knapsack problem (MKP) problems and we have compared the obtained results with those obtained by BCS.

## Keywords
Combinatorial optimization, Cuckoo Optimization Algorithm, Cuckoo Search, Binary Cuckoo Optimization Algorithm, Binary Cuckoo Search, knapsack problem, Multidimensional knapsack problem.

## 1. INTRODUCTION
Solving optimization problems is a research line that sought the attention of several research teams. It is intrinsically linked to operational research and it uses mathematical and computer tricks. The resolution of an optimization problem is to find a solution of sufficient quality from a set of solutions in terms of a constraints givens and objectives to meet. It is to maximize or minimize one or a set of fitness functions respecting constraints of the treated problem. To solve any optimization problem we need an algorithmic process of spatial and temporal complexity witch can rank optimization problems in different classes (P, NP, NP-complete and NP-Hard).

Methods for solving optimization problems are many. They are often classified into two classes: exact methods and approximate methods. The prohibitive cost associated with the use of exact methods, has excited researchers to use approximate methods. The investigation in the area of approximate methods gave rise to another class of methods called "Metaheuristics". Metaheuristics are general and applicable methods on a wide range of optimization problems. They are often inspired by natural systems in different fields: physics, biology, ethology ... Cuckoo Optimization Algorithm (COA) and Cuckoo Search (CS) algorithm are two examples of bioinspired metaheuristics. They are based on the cuckoo's behavior in their lifestyle and their characteristics in egg laying and breeding.

The aim of this paper is twofold: The first one is to compare the two metaheuristics inspired of Cuckoo's behavior. The second aim of this paper is to propose a binary version of Cuckoo Optimization Algorithm that we have called Binary Cuckoo Optimization Algorithm (BCOA) to cope with binary optimization problems. The main difference between the original version of COA algorithm and the proposed discrete binary version is that, in the original Cuckoo Search, the solution is composed of a set of real numbers, while in the proposed binary version; the solution is composed of a set of bits. The main feature of our approach consists in using a sigmoid function and probability model in order to generate binary values.

The remainder of this paper is organized as follows. Section 2 presents an overview of Cuckoo Search algorithm and Cuckoo Optimization Algorithm. The proposed algorithm (BCOA) is described in section 3. Section 4 presents the knapsack problems formulation. Experimental results are discussed in section 5, and a conclusion is provided in the sixth section of this paper.

## 2. CUCKOO SEARCH AND CUCKOO OPTIMIZATION ALGORITHM
In order to solve complex problems, ideas gleaned from natural mechanisms have been exploited to develop heuristics and metaheuristics. 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… . The ultimate goal is to develop systems that have ability to learn incrementally, to be adaptable to their environment and to be tolerant to noise. Two of the recent developed bioinspired algorithms are the Cuckoo Search (CS) [1] and the Cuckoo Optimization Algorithm (COA) [2] which are based on lifestyle of Cuckoo bird. Cuckoos use an aggressive strategy of reproduction that involves the female hack nests of other birds to lay their eggs fertilized. Sometimes, the egg of cuckoo in the nest is discovered and the hacked birds discard or abandon the nest and start their own brood elsewhere.

The main steps of Cuckoo Search proposed by Yang and Deb in 2009 [1] are presented in figure1 and the main steps of Cuckoo Optimization Algorithm proposed by Rajabioun in 2011 [2] are presented in figure2.

**Objective function** $f(x)$, $x = (x_1, .., x_d)^T$ ;

**Initial a population** of n host nests $x_i$ (i = 1, 2, ..., n);
**while** (t < MaxGeneration) or (stop criterion);
- Get a cuckoo (say i) randomly by Lévy flights;
- Evaluate its quality/fitness $F_i$;
- Choose a nest among n (say j) randomly;
- **if** ($F_i > F_j$),
  Replace j by the new solution;
  **end**
- Abandon a fraction ($p_a$) of worse nests
- build new ones at new locations via Lévy flights;
- Keep the best solutions (or nests with quality solutions);
- Rank the solutions and find the current best;

**end while**

**Fig 1: Cuckoo Search [1]**

1. Initialize cuckoo habitat with some random points on the profit function.
2. Dedicate some eggs to each cuckoo.
3. Define ELR for each cuckoo.
4. Let cuckoos to lay eggs inside their corresponding ELR.
5. Kill those eggs that are recognized by host birds.
6. Let eggs hatch and chicks grow.
7. Evaluate the habitat of each newly grown cuckoo.
8. Limit cuckoos maximum number in environment and kill those who live in worst habitats.
9. Cluster cuckoos and find best grop and select goal habitat
10. Let new cuckoo population immigrate toward goal habitat
11. If stop condition is satisfied stop, if not go to 2.

**Fig 2: Cuckoo Optimization Algorithm [2]**

## 2.1 CS VS COA

CS and COA are two metaheuristics based population solutions. They are inspired from Cuckoo's behavior. Each one (CS and COA) begins the search by a population of solutions. During research and optimization, new solutions appear and other disappear. In the CS algorithm, the new solution is created based on the current solution and Levy flight [1]. In contrast, in the COA algorithm new solution is created based on the current solution, the Egg Laying Radius (ERL), $\lambda$ and $\omega$ parameters [2].

In order to select new generation solutions, the two methods (CS and COA) use the elitism strategy (used in genetic algorithms). They choose the best solutions to represent the new generation (population) members and eliminate worst solutions.

**Table 1. CS vs COA**

| | **CS** | **COA** |
|---|---|---|
| **Autors** | Xin-She Yang and Suash Deb | Ramin Rajabioun |
| **Publication year** | 2009 | 2011 |
| **Metaheuristic type** | Based population of solutions | |
| **Inspiration source** | Cuckoo's behaviour and lifestyle | |
| **Similarity to naturel Cuckoo's behavior** | Some notions | Big similarity |
| **Research space** | continuous | |
| **Parameters** | ▪ Population size<br>▪ $P_a$ | ▪ Population size<br>▪ Number of eggs for each cuckoo (Min and Max)<br>▪ Number of cuckoos by group<br>▪ Maximum number of cuckoos that can live at the same time<br>▪ $\lambda, \varphi$ *and* $\omega$ |
| **Population size** | unchangeable | Changeable (increases) |
| **Creation of new solution** | Using:<br>▪ The current solution<br>▪ Lévy Flights | Using:<br>▪ The current solution<br>▪ ELR<br>▪ $\lambda, \varphi$ *and* $\omega$ *parameters* |
| **Selection of cuckoos of the new generation** | According to their qualities (fitness): elitism selection | |
| **Destruction cuckoos** | According to their qualities | ▪ According to their qualities<br>▪ According to their representations |

Furthermore, in CS the solutions can be eliminated in one case. It is the case where the new solution quality (the chick cuckoo) is better than that of its derivative (the cuckoo parent). In contrast, elimination of solutions in COA is performed in two cases. The first one is when a solution exists in the population more than once. In this case, the copy of solution will be eliminated to avoid redundancy solutions of the population. The second case is after classification solutions according to their quality. The bad solutions will be eliminated to avoid exceeding the maximum size of the population and fall in explosion population problem.

In the CS algorithm, the population solutions number is fixed and stable. In fact, the algorithm starts searching with in N solutions and converges with N solutions. However, in the

COA the population size is not stable. The algorithm (COA) starts searching with N cuckoos (solution) and converges with M cuckoos, where M > N. Because in the COA each cuckoo lays several eggs at once (iterative) and they can all survive. While in the CS, a cuckoo lays one egg at a time. According to the egg quality, this one can survive replacing its parent or it will be destroyed because of its poor quality compared to that of its parent. In addition, following the steps of the two algorithms, we conclude that COA has a lot of simulations of the natural cuckoo behavior (eggs number of each cuckoo, habitat, migration…).

Essentially, in addition to the population size, we distinguish in the CS algorithm only one parameter (Pa). This provides more flexibility for users and allows him to save time by dispensing with the study phase and the appropriate choice of the algorithm parameters. In addition, the pioneers of the CS algorithm showed that the convergence rate of their algorithm is insensitive to the parameters used (population size and Pa). This means that the fine adjustment of algorithm-dependent parameters is not needed for any given problems [1]. However, observing ACO parameters, we distinguish essentially six parameters involved in addition to the population size.

Table 1 presents a comparison of COA and CS algorithms.

# 3. THE PROPOSED ALGORITHM (BCOA)

Optimization problems can be classed into two main classes: continuous optimization problems and discrete optimization problems. In continuous optimization problems, the solution is represented by a set of real numbers. However, in discrete optimization problems, the solution is represented 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 represented by a set of bits. Many optimization problems can be modeled as a discrete binary search space such as, flowshop scheduling problem [5], job-shop scheduling problem [6], routing problems [7], knapsack problem [3] and its variants such as the multidimensional knapsack problem [10], the quadratic knapsack problem [8], the quadratic multiple knapsack problem [9] and so one.

Cuckoo Search and Cuckoo Optimization Algorithm operate in continuous search space. Consequently, they give 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. Therefore, the solutions must be converted from real values to binary values [4]. In the aim to extend the tow algorithms (CS and COA) to discrete binary areas, we proposed in [4] a discrete binary version of CS that we called

BCS (Binary Cuckoo Search). In this paper we propose a discrete binary version of COA algorithm that we called BCOA (Binary Cuckoo Optimization Algorithm). Like with the BCS algorithm, in the BCOA algorithm the problem solution must be represented by a set of bits.

The BCS architecture contains two essential modules. The first module contains the main binary cuckoo dynamics. This module is composed of two main operations: Lévy flights and binary solution representation operations. These two operations combine the Cuckoo Search algorithm steps and the Sigmoid function to obtain a Binary Cuckoo Search. In the first operation, Lévy flight is used to get a new cuckoo. In the second operation, the Sigmoid function is used to calculate the flipping chances of each cuckoo. Then, the binary value of each cuckoo is computed using their flipping chances.

The BCOA architecture is not very different of the BCS one. In fact, The BCOA architecture contains also two essential modules. The first one contains the main binary cuckoo dynamics. It is composed of two main operations: COA operations and binary solution representation operations. These two operations combine the Cuckoo Optimization Algorithm steps and the Sigmoid function to obtain a Binary version of Cuckoo Optimization Algorithm. In the first operation, COA steps are used to generate a new cuckoo. In the second operation, the Sigmoid function is used to calculate the flipping chances of each cuckoo. Then, the binary value of each cuckoo is computed using their flipping chances.

The second modules of the tow algorithms (BCS and BCOA) contain the objective function and the selection operator. The selection operator is similar to the elitism strategy used in genetic algorithms. Figure 3 and 4 show the flowchart of the proposed architectures.

## 3.1 Binary solution representation

The main objective of the BCS and BCOA algorithms is to deal with the binary optimization problems. Therefore, the main feature of the two algorithms (BCS and BCOA) is to transform a solution xi from real area to binary area, and consequently obtain a binary solution representation x'i. To meet this need, we propose to constrain the solution xi in the interval [0, 1] using the Sigmoid function as follows:

$$S(x_i) = 1 / (1 + e^{-x_i}) \quad (1)$$

Where S (xi) is the flipping chance of bit x'i. It represents 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(xi) as mentioned below in (equation.2). If the random number is lower than the flipping chance of bit x'i, than x'i takes the value 1. Otherwise, x'i takes the value 0.

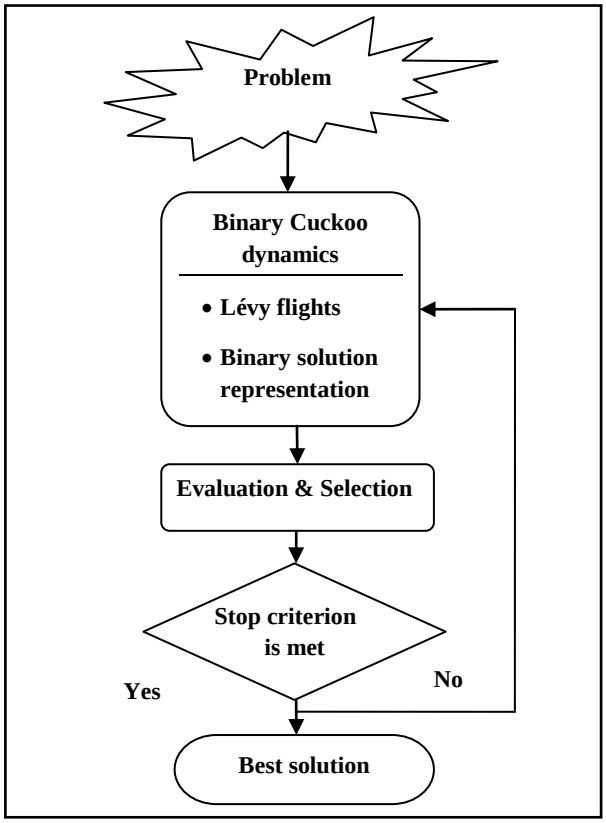


**Fig 3: Flowchart of the BCS Architecture [4]**

$$x'_i = \begin{cases} 1 \text{ if } r < S(x_i) & \text{with r} \in [0, 1] \\ 0 \text{ Otherwise} \end{cases} \quad (2)$$

Consequently, having a solution xi encoded as a set of real numbers, the sigmoid function is used to transform the solution xi into a set of probabilities that represents the chance for bit i to be flipping. The flipping chance is then used to compute the binary solution x'i.

For example: assuming that we have a problem with N= 6 objects i.e. problem size is 6. xi = [2.314, -3.4510, 1.9412, 0.3498, -1.5634, 3.8461] is the obtained solution with original COA algorithm ; S (xi) = [0.9100, 0.0307, 0.8745, 0.5866, 0.1732, 0.9791] is the set of flipping chances (probabilities) of each bit x'i calculated by the Sigmoid function; Then the chance for each bit to be flipping is: chancei = [91.00%, 3.07%, 87.45%, 58.66%, 17.32%, 97.91%]; In order to obtain a binary solution, we must generate 6 random numbers r from the interval [0, 1], for example r = [0.8147, 0.1270, 0.9134, 0.6324, 0.0975, 0.5469].

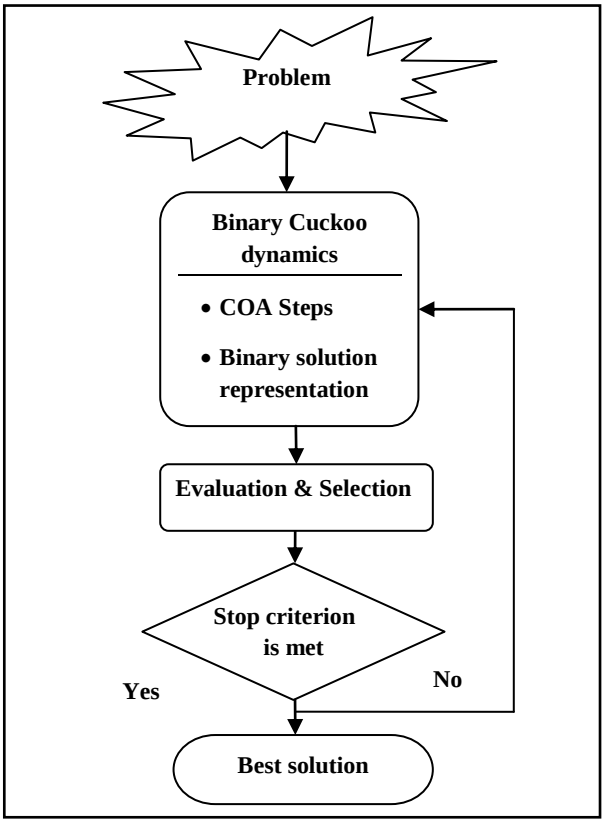


**Fig 4: Flowchart of the BCOA Architecture**.

Following the defined instructions in (equation 2), the first, fifth and sixth bits of the binary solution take the value 1 because there flipping chances (0.9100, 0.1732, 0.9791, respectively) are higher than the generated random numbers (0.8147, 0.0975, 0.5469, respectively). However, the second, third and fourth bits take the value 0, because there flipping chances (0.0307, 0.8745, 0.5866, respectively) are lower than the generated random numbers (0.1270, 0.9134, 0.6324, respectively). Thus, x'i = [1, 0, 0, 0, 1, 1] is the binary solution representation. Which mean that the selected objects are 1, 5 and 6. A pseudo code of the Binary Solution Representation (BSR) algorithm is shown in Figure 5.

**Input:** Real solution representation $x_i$
**Output:** Binary solution representation $x'_i$

For ( i=1 to (problem size)) {
$S(x_i) = 1/(1+e^{-x_i})$ ;
If (random number $r < S(x_i)$ )
$x'_i = 1$;
Otherwise
$x'_i = 0$;
}

**Fig 5: Binary Solution Representation (BSR) algorithm [4]**

## 3.2 Outlines of the proposed algorithm

Now, we describe how Cuckoo Optimization Algorithm scheme including the Sigmoid function has been embedded

within a new population-based metaheuristics, and how it can find a binary solution to a binary optimization problem.

The BCOA algorithm starts by initializing cuckoo habitats. For that, the initial population is created. This one contains both good and bad solutions. During each iteration, some eggs are dedicated to each cuckoo and the algorithm progresses through a number of generations according to the COA dynamics as explained in [2]. In order to evaluate the different cuckoos, we apply the BSR algorithm (Figure 5) to get a binary solution which represents a potential solution for the binary optimization problems. After this step, the algorithm limits cuckoos maximum number, it eliminates the worst ones and finds the best group to select the goal habitat. The new population immigrates toward goal habitat and the algorithm restart if the stop condition is not satisfied. A pseudo code of BCOA algorithm is shown in Figure 6.

1. Initialize cuckoo habitat with some random points on the profit function.
2. Dedicate some eggs to each cuckoo.
3. Define ELR for each cuckoo.
4. Let cuckoos to lay eggs inside their corresponding ELR.
5. Kill those eggs that are recognized by host birds.
6. Let eggs hatch and chicks grow.
7. **Get the binary representation of cuckoos by the BSR algorithm.**
8. Evaluate the habitat of each newly grown cuckoo.
9. Limit cuckoos maximum number in environment and kill those who live in worst habitats.
10. Cluster cuckoos and find best grop and select goal habitat.
11. Let new cuckoo population immigrate toward goal habitat.
12. If stop condition is satisfied stop, if not go to 2.

**Fig 6: Binary Cuckoo Optimization Algorithm.**

## 4. KNAPSACK PROBLEMS

The knapsack problem (KP) is a NP-hard problem [11]. Numerous practical application of the KP can be found in many areas involving resource distribution, investment decision making, budget controlling, project selection and so on. The knapsack problem 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 pi and a weight wi. 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:

$$\text{Maximize} \sum_{i=1}^{N} p_i x_i \tag{3}$$

$$\text{Subject} \sum_{i=1}^{N} w_i x_i \le C \tag{4}$$

$$\text{With} \quad x_i \in \{0,1\}$$

Many variants of the knapsack problem were proposed in the literature including the Multidimensional Knapsack Problem (MKP). MKP is an important issue in the class of knapsack problem. It is a NP-hard problem [12]. In the MKP, each item xi has a profit pi like in the simple knapsack problem. However, instead of having a single knapsack to fill, we have a number M of knapsack of capacity Cj (j = 1 ... M). Each xi has a weight wij 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:

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

$$\sum_{i=1}^{N} w_{ij} x_i \le C_j \quad j = 1...M$$

$$\text{With} \quad x_i \in \{0,1\} \tag{6}$$

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 [13].

Clearly, there are 2N potential solutions for these problems. It is obviously that knapsack problem and its variants are combinatorial optimization problems. Several techniques have been proposed to deal with knapsack problems [11]. 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.

## 5. EXPERIMENTAL RESULTS

Several experiments were performed to compare the efficiency and performance of BCS and BCOA algorithms, which has a common point. In fact, the two algorithms (BCS and the BCOA) used the Sigmoid function to generate the binary solution and they present a binary versions of the tow metaheuristics inspired of cuckoo's behavior. The algorithms were used to resolve KP and MKP problems. They were implemented in Matlab 7.3. In the first experiment, we have tested and compared the proposed BCOA algorithm with the BCS algorithm that we have proposed in [4] on some Knapsack Problem instances taken from [3]. The used instances are 7 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 the following formula:

$$C = \frac{3}{4} \sum_{i=0}^{N} w_i \tag{7}$$

In the second experiment, we have compared the performance of BCOA and BCS algorithm on some Multidimensional Knapsack Problem benchmarks taken from OR-Library. The obtained results are compared with the exact solution (best known). Finally, statistical tests of Freidman were carried out to test the significance of the difference in the accuracy of each algorithm in the experiments.

Table 2 shows the experimental results of BCS and BCOA algorithms on KP random instances. The first column represents the problem size (i.e. Instance). The second and third columns represent the obtained results by the BCS and

the BCOA algorithms respectively. The presented results show that the performance of the BCS algorithm outperforms the BCOA algorithm performance. Indeed, the statistical Freidman test (Figure 7) indicates that the BCS algorithm is better than the BCOA algorithm in this experiment. However, this different is not very significant.

**Table 2. Experimental results of BCS and BCOA with random KP instances**

| Instance | BCS | BCOA |
|---|---|---|
| 120 | 4210 | 4091 |
| 200 | 6989 | 6721 |
| 500 | 16364 | 15772 |
| 700 | 22734 | 21779 |
| 900 | 29322 | 28218 |
| 1000 | 31679 | 30667 |

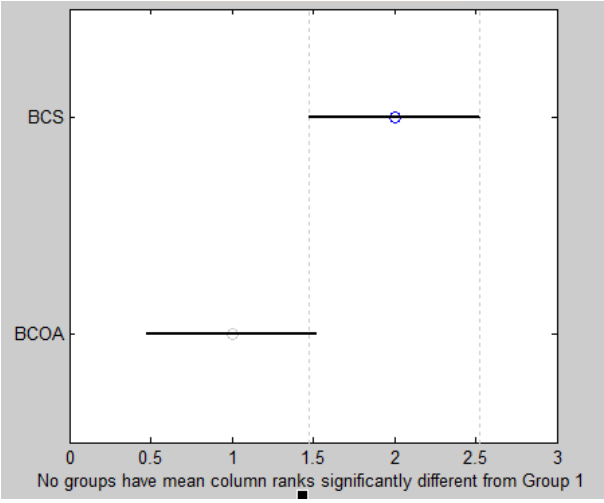


**Fig 7: Friedman test compares BCS and BPSO on Knapsack tests**.

Table 3 show experimental results of BCS and BCOA on some MKP instances (mknap1 instances). The first column indicates the instance index. The second and third column indicates the number of object and knapsack dimension respectively and the fourth, fifth and sixth columns indicate the best known, the BCS solution and the BCOA solution respectively. As we can see, the BCS algorithm is able to find the best solution of all the mknap1 instances. However, BCOA algorithm is able to find the best knowns of the first four instances only.

The statistical Friedman test in Figure 8 represents a comparison of the best known, the BCS and the BCOA results. This statistical test confirms that the difference between BCS and the BCOA results is not very statistically significant.

**Table 3. Experimental Results of MKP with mknap1 instances**

| N° | n | m | Best known | BCS | BCOA |
|---|---|---|---|---|---|
| 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 | 11990 |
| 6 | 39 | 5 | 10618 | 10618 | 10275 |
| 7 | 50 | 5 | 16537 | 16537 | 15631 |

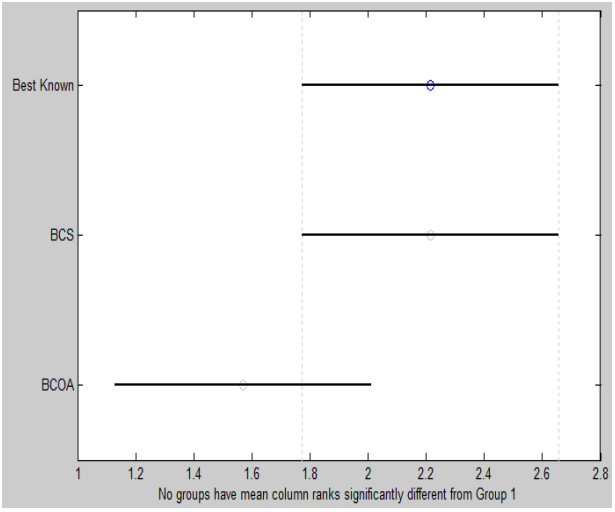


**Fig 8: Friedman test compares BCS, BCOA and best known solutions**.

The main purpose of this paper is to compare the two metaheuristics inspired of cuckoo's behavior (CS and COA) and their binary versions. For that we have proposed the binary version of COA algorithm (BCOA) using the Sigmoid function which is used in BCS algorithm to generate a binary solutions. In the second step, we have used the proposed version (BCOA) for solving two NP-hard combinatorial optimization problems: KP and MKP. These problems are very important for modeling many industrial problems. Moreover, KP and MKP have received the attention of many researchers due to their importance and their NP-Hardness.

## 6. CONCLUSION

In this work, we have proposed a discrete binary version of Cuckoo Optimization Algorithm that we have called Binary Cuckoo Optimization Algorithm (BCOA). Our contribution has twofold aim: the first aim is to compare the two metaheuristics based on cuckoo's behavior (CS and COA). The second aim is to propose a binary version of Cuckoo Optimization Algorithm to deal with binary NP-hard optimization problems. The main feature of our approaches consists in using of sigmoid function and probability model in order to generate binary solutions. To evaluate the performance and the effectiveness of the proposed algorithm (BCOA algorithm), we have applied and tested it on some KP and MKP instances. And we have compared BCOA with the binary version of Cuckoo Search algorithm (BCS). The experimental studies prove the feasibility and the effectiveness of the binary versions of the too metaheuristics inspired from Cuckoo's behavior (BCS and BCOA algorithms). Indeed, in the most cases BCS is able to give better results than BCOA. However, there are several issues to improve BCOA algorithm. In fact, a local search method or other operations inspired from other popular algorithms can be integrated in the core of the BCOA algorithm in order to improve its performance.

## 7. REFERENCES

[1] Yang, X.-S., and Deb, S., Engineering Optimisation by Cuckoo Search, Int. J. Mathematical Modelling and Numerical Optimisation, Vol. 1, No. 4, pp. 330–343, 2010.

[2] R. Rajabioun. Cuckoo Optimization Algorithm. Applied Soft Computing. Vol 11, N° 8, pp 5508-5518, 2011.

[3] Gherboudj. A, Chikhi. S. BPSO Algorithms for Knapsack Problem. A. Özcan, J. Zizka, and D. Nagamalai (Eds.): WiMo/CoNeCo 2011, CCIS 162, pp. 217–227, 2011. Springer (2011).

[4] A. Gherboudj, A. Layeb, S. Chikhi. Solving 0-1 knapsack problems by a discrete binary version of cuckoo search algorithm. International Journal of Bio-Inspired Computation. Vol. 4, N°.4, pp 229-236. Inderscience Publishers ISSN (Print): 1758-0366, ISSN (online): 1758-0374. DOI: 10.1504/IJBIC.2012.048063. 2012.

[5] Liao, C-J., Tseng, C-T. and Luarn, P. (2007) 'A discrete version of particle swarm optimization for flowshop scheduling problems', Computers & Operations Research, Vol. 34, No. 10, pp.3099–3111, Elsevier.

[6] Pongchairerks, P. (2009) 'Particle swarm optimization algorithm applied to scheduling problems', *ScienceAsia*, Vol. 35, No. 1, pp.89–94.

[7] Zhan, Z-h. and Zhang, J. (2009) 'Discrete particle swarm optimization for multiple destination routing problems', in Giacobini, M. et al. (Eds.): *Proc. EvoWorkshops 2009, LNCS*, Vol. 5484, pp.117–122, Springer.

[8] 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, pp. 607-614. ISBN: 1595930108.

[9] Singh. A and Baghel. A-S. A New Grouping Genetic Algorithm for the Quadratic Multiple Knapsack Problem. In proc. C. Cotta and J. van Hemert (Eds.): EvoCOP 2007, LNCS 4446, pp. 210 – 218, Springer (2007).

[10] Kong. M and Tian. P. Apply the Particle Swarm Optimization to the Multidimensional Knapsack Problem. In proc. L. Rutkowski et al. (Eds.): ICAISC 2006, LNAI 4029, pp. 1140–1149, Springer (2006).

[11] Pisinger, D.: Where are the hard knapsack problems? Computers and Operations Research, Vol.32, N°. 9, pp. 2271-2284, 2005.

[12] Chu .P.C, Beasley. J.E. A Genetic Algorithm for the Multidimensional Knapsack Problem. Journal of Heuristics, 4: 63–86 (1998).

[13] Vasquez. M and Vimont. Y., Improved results on the 0–1 multidimensional knapsack problem, European Journal of Operational Research 165 (1), pp. 70–81, 2005.