JOURNAL OF SOFTWARE ENGINEERING & INTELLIGENT SYSTEMS
ISSN 2518-8739
30th April 2019, Volume 4, Issue 1, JSEIS, CADMEI Copyright © 2016-2019
www.jseis.org

NEW HYBRID AFRICAN BUFFALO OPTIMIZATION AND CUCKOO SEARCH METAHEURISTIC APPLIED ON TWO KNAPSACK PROBLEMS

[1]AMIRA GHERBOUDJ

[1] MISC laboratory, Department of Computer Science and its Applications
Abdelhamid Mehri University, Constantine, Algeria
Email: [1] a.gherboudj@umc.edu.dz

ABSTRACT

African Buffalo Optimization (ABO) and Cuckoo search (CS) are two recent metaheuristics-based swarm intelligence. ABO is inspired by the buffalo's behavior and lifestyle. CS is inspired by the cuckoo's behavior and lévy flights mechanism. The aim of this contribution is threefold. The first aim is to propose a new hybrid metaheuristic between ABO and CS metaheuristics. The second aim is to propose two discrete binary versions of the proposed metaheuristic to cope with binary optimization problems. The third aim is to solve two knapsack problems: Single knapsack problem (KP) and multidimensional knapsack problem (MKP), which are NP-hard optimization problems. Computational results show the effectiveness of our algorithms and their ability to achieve best and promising solutions.

Keywords: hybrid metaheuristic; swarm intelligence; African buffalo optimization, cuckoo search, binary problems, knapsack problems;

Academic Editor: Muhammad Imran Babar, APCOMS, Rawalpindi, Pakistan.
Published: 30th April, 2019
Funding: Author has no funds.
Article Type: Research Article

1. INTRODUCTION

Combinatorial optimization is intrinsically linked to operational research. It uses mathematics and computer science to deal with challenging optimization problems that are often NP-hard problems. Various optimization methods have been proposed to solve combinatorial optimization problems and metaheuristic methods showed their performance in solving these difficult problems within a reasonable running time. Several metaheuristics in the literature are inspired by nature systems like: Simulated Annealing [1], Genetic Algorithms [2], Tabu Search [3] and algorithms based on swarm intelligence such as: Cuckoo Search [4] and African Buffalo Optimization [5].

Swarm intelligence methods are generally inspired by lifestyle of some species evolving in groups and their collective behavior in solving their problems. They are very popular and widely used to solve optimization problems. Swarm intelligence methods are mainly based on a strategy of sharing information among multiple agents. This strategy helps to provide the high efficiency of most swarm intelligence methods [6].

African Buffalo Optimization (ABO) is one of the most recent swarm intelligence methods. Despite its young age, ABO was used to solve several hard optimization problems such as: Traveling Salesman Problem [7], Cannel Allocation Problem [8], Knapsack Problems [9], Budget-Constrained Maximal Covering Problem [10]. The applications of ABO to solve hard optimization problems showed its effectiveness and robustness. It is a simple-to-implement algorithm and it demonstrates a good capacity in exploitation and exploration of the search space [5]. However, the study presented in [9] showed that the algorithm may suffer from the stagnation of research caused by meeting the local optimum. Cuckoo Search (CS) is also one of the recent swarm intelligence methods. The applications of CS metaheuristic on optimization problems showed its performance. It has proved its exceptional effectiveness in establishing a great balance between exploitation and exploration of the search space [11] [12].

In this paper, three contributions are presented. Firstly, new hybrid swarm intelligence metaheuristic is proposed. It is called "Hybrid African Buffalo Optimization and Cuckoo Search": HABOCS. HABOCS is inspired by ABO and CS metaheuristics, the aim of this combination is to improve the performance of ABO in dealing with the stagnation of research. Secondly, two discrete binary versions of the proposed metaheuristic are presented to prove that the suggested algorithm can cope with binary optimization problems. Thirdly, the proposed algorithms are used to solve two knapsack problems: Single knapsack problem (KP) and multidimensional knapsack problem (MKP) as examples of NP-hard binary optimization problems.

Corresponding Author: Amira Gherboudj
Email Address: a.gherboudj@umc.edu.dz

The remainder of this paper is organized as follows: Section 2 and 3 present an overview of African Buffalo Optimization (ABO) and Cuckoo Search (CS) algorithms, respectively. Section 4 describes the proposed hybrid algorithm (HABOCS). The Two proposed discrete binary versions of HABOCS algorithm are accosted in section 5. Section 6 addresses KP and MKP problems. Experimental results are presented and discussed in section 7. Finally, conclusion and perspectives are provided in the eighth section of this paper.

2. AFRICAN BUFFALO OPTIMIZATION

The African Buffalo Optimization was proposed in 2015. It was 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.

- Their extensive memory capacity. It helps the buffaloes to keep track of their routes.
- Their cooperative communicative ability whether in good or bad times.
- Their democratic nature borne out of extreme intelligence. In the cases where there are opposing calls by the herd members, the buffaloes have a way of doing an 'election' and the majority decision determines the next line of action [5].

  Furthermore, ABO algorithm models the two sounds for communication that buffaloes use to exploit and explore the search space:
- The warning sound "waaa" is used to 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" is used to ask the herd to stay on the present location because it 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. The generation of new solutions is done using equations 1 and 2.

$$m_{k+1} = m_k + lp_1(bg\max - w_k) + lp_2(bp\max_k - w_k) \quad (1)$$

$$w_{k+1} = \frac{w_k + m_k}{\lambda},\ 0 < \lambda \le 1 \quad (2)$$

Where:

- w_k and m_k present the exploration and exploitation moves respectively of the k^{th} buffalo (k=1,2,……..N) ;
- λ , presents the unit of time interval over the movement of buffalo.
- lp_1 and lp_2 are learning factors;
- bgmax is the herd's best fitness;
- $bpmax_k$ is the individual buffalo's best.

Algorithm 1: ABO Algorithm [5]

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

3. CUCKOO SEARCH ALGORITHM

Cuckoo Search was proposed in 2009 by Yang and Deb [4] [11]. CS algorithm is inspired by the obligate brood parasitism of some Cuckoo species by laying their eggs in the nests of other bird's species. Cuckoos use an aggressive strategy of reproduction that involves the female hack nests of other birds to lay their fertilized eggs. 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 CS proposed by [4] is based on the following three idealized rules:

- Each cuckoo lays one egg at a time, and dumps it in a randomly chosen nest.
- The best nests with high quality of eggs (solutions) will carry over to the next generations.
- The number of available host nests is fixed, and a host can discover an alien egg with a probability $p_a \in [0, 1]$. In this case, the host bird can either throw the egg away or abandon the nest so as to build a completely new nest in a new location.

The last assumption can be approximated by a fraction pa of the n nests being replaced by new nests (with new random solutions at new locations). In CS algorithm, the generation of new solutions is done using Lévy flights [4]. Algorithm 2 presents the CS algorithm.

Algorithm 2: CS algorithm [11]

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

4. THE PROPOSED HYBRID HYBRID ALGORITHM

The algorithm's performance is very sensitive to an adequate exploitation and exploration of the search space. This helps the algorithm to avoid the premature convergence towards semi-optimal (i.e. near optimal) solutions. The pioneers of ABO algorithm proved in [5] that ABO algorithm is a fast algorithm. Otherwise, the study presented in the same paper showed that despite the effectiveness of the ABO algorithm, it cannot find the optimal solution for all instances. Furthermore, according to [9] ABO may suffer from the research stagnation caused by meeting the local optimum. This can be set by a good balance between exploitation and exploration of the search space.

CS has proved its effectiveness in exploitation and exploration of the search space [11] [12]. It uses a phase that replaces a portion of bad solutions with others having a good quality. This phase allows a good exploration of the search space and helps the algorithm to escape from the local optimum problem.

In this contribution, a new hybrid algorithm is presented. It integrates the phase of changing a portion of bad solutions to the ABO procedures in order to improve its performance.

The proposed algorithm is called HABOCS. It architecture (see Figure 1) contains three main modules. The first one contains the ABO dynamics. It models the three characteristic behaviors of the African buffaloes that enable their search for pastures: their extensive memory capacity, their cooperative communicative ability whether in good or bad times and their democratic nature borne out of extreme intelligence. The second module contains CS steps to improve the population by replacing a portion pa of bad solutions with good solutions. The third module contains the objective function and the selection operator. The selection operator is similar to the elitism strategy used in Genetic Algorithms. The HABOCS steps are presented in Algorithm 3.

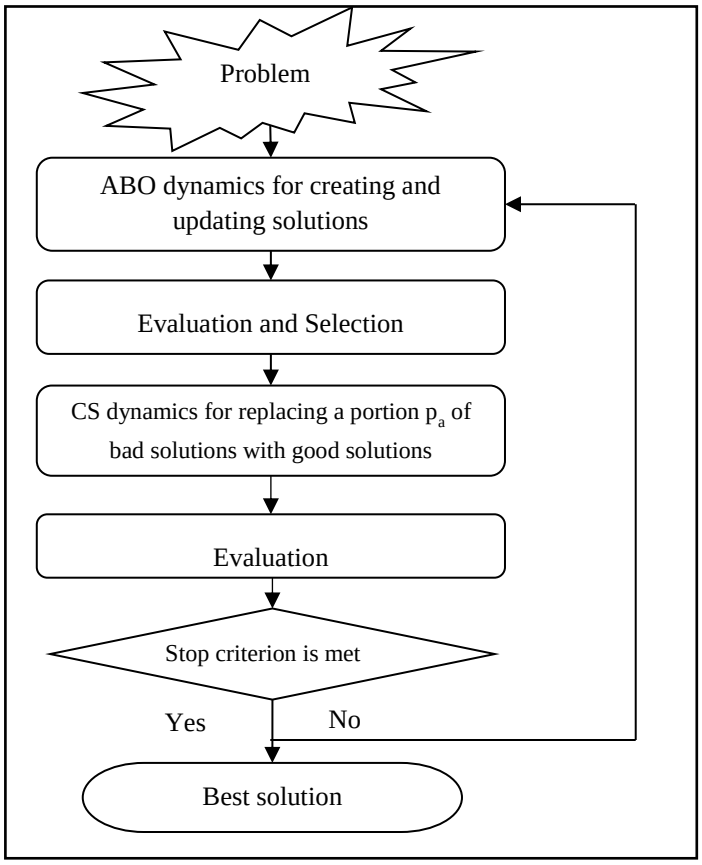

Figure. 1 HABOCS architecture

Algorithm 3: HABOCS 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. Update the location of buffaloes using (2).
5. **Abandon a fraction (p_a) of worse buffaloes.**
6. **Build new ones at new random locations.**
7. **Update bpmax and bgmax.**
8. **If the stopping criteria is not met, go back to algorithm step 3, else go to step 9.**
9. Output the best solution.

5. TWO DISCRETE BINARY VERSIONS OF THE PROPOSED ALGORITHM

Optimization problems can be classified into two main categories: 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 modeled as discrete binary search space such as: flowshop scheduling problem [13], job-shop scheduling problem [14], routing problems [15], KP [16] [17]and its variants like MKP [17], quadratic KP [18], quadratic multiple KP [19] and so on.

The original ABO and CS algorithms operate in continuous search space. They give a set of real numbers as a solution of the handled problem. Therefore, their grouping gives birth to the continuous version of a new algorithm. A continuous method can easily be used to solve continuous optimization problems. However, it needs a special adaptation for solving binary problems. A binary optimization problem needs a binary solution and the real solutions are not acceptable, because they are considered as illegal solutions. In this contribution, two binary versions of HABOCS are proposed. They are called SHABOCS and LHABOCS. The objective of these binary

versions is to extend the HABOCS algorithm towards discrete binary areas and prove that it can cope with binary optimization problems.

5.1 SHABOCS algorithm

In the SHABOCS algorithm (see Algorithm 5), a binarization phase (see Algorithm 4) of solutions is introduced in the core of HABOCS algorithm in order to obtain a binary solution for the treated problem. The objective of this phase (i.e binarization) is to transform a solution xi from real area to binary area. To meet this need, the authors propose to constrain the solution xi in the interval [0, 1] using the Sigmoid Function as follows:

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

Where S(xi) 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, a random number is generated from the interval [0, 1] for each dimension i of the solution x and it is compared with the flipping chance S(xi) 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.

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

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 presents the chance for bit i to be flipping. The flipping chance is then used to compute the binary solution x′i. Algorithm 4 represents the binarisation algorithm and Algorithm 5 shows the SHABOCS algorithm.

Algorithm 4: Binarization Algorithm (BA)

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;
}

Algorithm 5: SHABOCS algorithm

1. Objective function $f(x) = (x1,x2,\ldots xn)^T$.
2. Initialization: randomly place buffaloes to nodes at the solution space.
3. Update the buffalo's fitness values using (1).
4. **Get the binary buffalo using BA Algorithm.**
5. Update the location of buffaloes using (2).
6. Abandon a fraction (p_a) of worse buffaloes.
7. Build new ones at new random locations.
8. Update bpmax and bgmax.
9. If the stopping criteria is not met, go back to algorithm step 3, else go to step 10.
10. Output the best solution.

5.2 LHABOCS algorithm

In the LHABOCS algorithm, the search is started with a binary population (the solutions are binary from the beginning). The arithmetic operators used in the solution update equations (i.e. Eq 1 and 2) are replaced by logical operators as below:

Assuming two values val1 and val2:

1) val_2 - val_1

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

2) $val_2 + val_1$

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

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 6 represents the LHABOCS algorithm.

Algorithm 6: LHABOCS 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 **using binary values**.
3. Update the buffalo's fitness values using (1) **and logical operators**.
4. Update the location of buffaloes using (2) **and logical operators**.
5. Abandon a fraction (p_a) of worse buffaloes.
6. Build new ones at new random locations.
7. Update bpmax and bgmax.
8. If the stopping criterion is not met, go back to algorithm step 3, else go to step 9.
9. Output the best solution.

6. KNAPSACK PROBLEMS

The KP is a NP-hard problem [20]. 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 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 is 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 [20]:

$$Maximize \sum_{i=1}^{N} p_i x_i \quad (5)$$

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

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

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 (Chu and Beasley, 1998). In the MKP, each item xi has a profit pi like in the simple KP. However, instead of having a single knapsack to fill, a number M of knapsack of capacity Cj (j = 1, …, M) is imposed. Each xi has a weight wij that depends on the knapsack j (example: an object can have a weight 3 in knapsack 1, 5 in knapsack 2, etc). The 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 [21]:

$$Maximize \sum_{i=1}^{N} p_i x_i \quad (7)$$

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

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

The MKP can be used to formulate many industrial problems such as capital budgeting problem, allocating processors and databases in a distributed computer system, cutting stock, project selection and cargo loading problems [22]. Clearly, there are 2N potential solutions for these problems. It is obvious that KP and its variants are combinatorial optimization problems. Several techniques have been proposed to deal with KPs [20]. 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. In the next section, experimental results with some KP and MKP benchmarks are shown.

7. EXPERIMENTAL RESULTS

This section is represented in two sub-sections. The first one gives some details about the environment of experiments, the parameter settings and the instances solving. The second one reports, compares and discusses the obtained results.

7.1 Experimental data

The proposed algorithms (SHABOCS and LHABOCS) were implemented in MATLAB R2014a. Our experiments were performed using a laptop computer running Windows 7, Intel(R) Core(TM) i3-3110M CPU@ 2.40 GHz, 2.40GHz, 4GB RAM.

The parameters used in the experiments are: Population Size (PS), Maximum Number of Iterations (MNI), lp1, lp2 and pa. To decide the values of PS and MNI, some experiments with the proposed algorithms are performed. During these experiments, it was observed that increasing PS and MNI have a positive effect on the quality of solutions. However, increasing PS and MNI values increases the execution time. Therefore, 40 solutions for Population Size and 300 iterations as Maximum Number of Iterations were decided. The other parameters (i.e. lp1, lp2 and pa) are set as proposed in the original ABO and CS: lp1 and lp2 between 0.1 to 0.6 and pa= 0.25.

Several experiments and comparisons are performed to assess the efficiency and the performance of our algorithms. In the first experiment, the proposed algorithms are tested on some small KP instances taken from [23]. The obtained results are compared with:

- The binary Cuckoo Search algorithm [12] named BCS.
- The binary African Buffalo Optimization algorithms named SBABO and LBABO [9].
- The Harmony search algorithm named NGHS [23] .

In the second experiment, some big KP instances used in [16] are used to test and compare the proposed algorithms with:

- Two binary Particle Swarm Optimization algorithms named BPSO [24] and BP1 [16].
- The binary Cuckoo Search algorithm: BCS.
- The binary African Buffalo Optimization algorithms: SBABO and LBABO.

BPSO, BCS and SBABO algorithms have a common point with the proposed SHABOCS algorithm. In fact, the four algorithms (SHABOCS, BPSO, BCS and SBABO) use the Sigmoid Function to generate the binary solution. BP1 and SBABO are binary versions of Particle Swarm Optimization (PSO) and ABO algorithms, respectively. They have a common point with the proposed LHABOCS algorithm. Indeed, the three algorithms use logical operators to operate the binary solutions. The instances used in the second experiment 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 using equation 9 [16]. The factor 3/4 indicates that about 75% of items are in the optimal solution.

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

In the third experiment, the performance of the proposed algorithms is evaluated on some small size MKP benchmarks taken from OR-Library. The used instances are taken from seven benchmarks named mknap1. The obtained results are compared with:

- The exact solution (best known).
- The binary Cuckoo Search algorithm named BCS.
- The binary African Buffalo Optimization algorithms named SBABO and LBABO.

In the fourth experiment, the proposed algorithms are tested on some big size MKP instances taken from benchmarks named mknapcb1 and mknapcb4. Five tests of the benchmarks mknapcb1 (5.100) are used. The benchmarks mknapcb1 have five constraints and 100 items. Moreover, five tests of the benchmarks mknapcb4 (10.100) were used mknapcb4 have ten constraints and 100 items. The obtained results are compared with:

- The exact solution (best known).
- The obtained solution by the standard binary PSO (denoted as PSO-P) which uses the Penalty Function Technique [17].
- The binary Cuckoo Search algorithm (BCS) and the quantum inspired Cuckoo Search denoted as QICSA [25].
- The binary African Buffalo Optimization algorithms: SBABO and LBABO.

Finally, statistical tests of Freidman are carried out to test the significance of difference in the accuracy of each method in these experiments.

7.2 Results and discussion

Table 1 shows the experimental results of our algorithms, SHABOCS and LHABOCS, compared with BCS, SBABO, LBABO and NGHS algorithms on ten KP tests with different sizes. The first column indicates the instance name, the second column points out the problem size, i.e. number of objects. Columns from three to eight show the obtained results with the BCS, SBABO, LBABO, SHABOCS, LHABOCS and NGHS algorithms, respectively. The observed results in Table 1 revealed that:

- The proposed discrete binary algorithms (SHABOCS and LHABOCS) perform well than NGHS algorithm in F6 test.
- The proposed algorithms perform well than NGHS and LBABO algorithms in F8 test.
- The proposed algorithms obtain the same results as the other algorithms with the rest of the instances.

Table1. 1 Experimental results with small KP instances

Test	*Size*	*BCS*	*SBABO*	*LBABO*	*SHABOCS*	LHABOCS	*NGHS*
F1	10	295	295	295	295	295	295
F2	20	1024	1024	1024	1024	1024	1024
F3	4	35	35	35	35	35	35
F4	4	23	23	23	23	23	23
F5	15	481.0694	481.0694	481.0694	481.0694	481.0694	481.0694
F6	10	**52**	**52**	**52**	**52**	**52**	50
F7	7	107	107	107	107	107	107
F8	23	**9767**	**9767**	9761	**9767**	**9767**	9761
F9	5	130	130	130	130	130	130
F10	20	1025	1025	1025	1025	1025	1025

Table 2 shows the experimental results of SHABOCS and LHABOCS, compared with the BPSO, BP1, BCS, SBABO and LBABO algorithms on big KP instances. The first column represents the problem size (i.e., instance). The second, third, fourth, fifth, sixth, seventh and eighth columns correspond to the obtained results with the BPSO, BP1, BCS, SBABO, LBABO, SHABOCS and LHABOCS algorithms, respectively. With each instance, the best and the average solution are shown in the first line and the second one, successively.

Table. 2 Experimental results with big KP instances. (NF: Not Found)

Instance	BPSO	BP1	BCS	SBABO	LBABO	SHABOCS	LHABOCS
120	4296	4469	4210	4316	4504	**4786**	***4786***
	3840.8	4130.4	*NF*	*4088.09*	*4357*	***4783***	*4782.36*
200	7456	7522	6989	6778	7530	**8269**	*8264*
	5703	6989.9	*NF*	*6480.56*	*7284.22*	***8223.21***	*7893.21*
500	13116	17647	16364	14730	16853	**19906**	*19322*
	12471.2	16848.2	*NF*	*14396.11*	16174.25	***19557***	*18904.64*

700	18276	24207	22734	20501	23278	**27114**	*26074*
	17097.4	23335.6	*NF*	19348.07	22530.4	**25736.1**	*25322.3*
900	22857	30898	29322	24767	30196	**34288**	*32763*
	21736.6	29833.8	*NF*	*24270.83*	28864.5	***33257.17***	*32406.5*
1000	24933	34319	31679	27306	32948	**37767**	*35836*
	24050	33065	*NF*	26607.3	31936.86	**36271.33**	*35408.67*

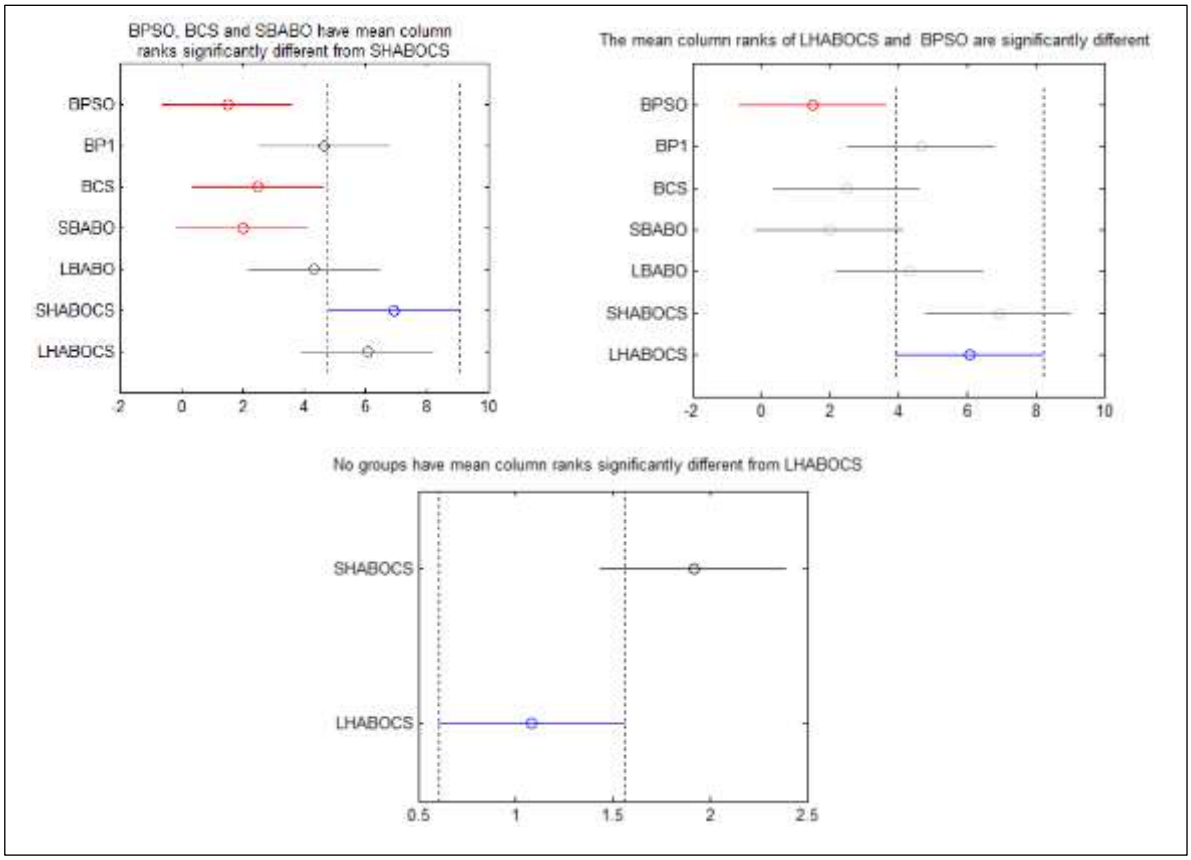

Figure. 2 Friedman test with big KP instances

The obtained results show that:

- SHABOCS and LHABOCS outperform the other algorithms with all instances.
- SHABOCS outperforms LHABOCS with the majority of the instances in terms of best known and average (except best known of the first instance that is identical).
- SHABOCS outperforms BPSO, BCS and SBABO algorithms, despite that the fourth algorithms use the sigmoid function to generate binary solutions.
- LHABOCS algorithms outperforms BP1 and LBABO algorithms, despite that the three algorithms use the logical operators to operate and generate binary solutions.

The statistical test of Friedman (Figure 2) presents a comparison of the SHABOCS, LHABOCS, BPSO, BCS, BP1, SBABO and LBABO algorithms on big KP instances. The SHABOCS and LHABOCS algorithms rank first in the Friedman test. This statistical test shows that:

- A significant difference between SHABOCS and SBABO, BCS, BPSO algorithms is observed.
- A significant difference between LHABOCS and BPSO algorithm is also detected.
- The difference between SHABOCS and LHABOCS results is not statistically significant.

Consequently, the obtained results confirm that the proposed algorithms outperform the other algorithms and prove that the proposed algorithms give very good results.

Table 3 shows experimental results of the proposed algorithms over 7 small instances of MKP problem. The first column indicates the instance index. The second and third columns indicate the number of object and

knapsack dimension, respectively. The other columns represent the best known, BCS, SBABO, LBABO, SHABOCS and LHABOCS solutions, in that order. These experimental results demonstrate that:

- The proposed algorithms and the BSC algorithms are able to find the best solution of all mknap1 instances.
- The proposed algorithms outperform the SBABO algorithm with the instance 7.
- The proposed algorithms outperform the LBABO algorithm with the instances 6 and 7.

Table. 3 Experimental Results with small MKP instances

N°	N	M	Best known	BCS	SBABO	LBABO	SHABOCSA	LHABOCSA
1	6	10	3800	3800	3800	3800	3800	3800
2	10	10	8706,1	8706,1	8706,1	8706,1	8706,1	8706,1
3	15	10	4015	4015	4015	4015	4015	4015
4	20	10	6120	6120	6120	6120	6120	6120
5	28	10	12400	12400	12400	12400	12400	12400
6	39	5	10618	10618	10618	**10554**	10618	10618
7	50	5	16537	16537	**16442**	**16371**	16537	16537

Table. 4 Experimental Results with big MKP instances

benchmark name	Problem size	Best Known	SHABOCS	LHABOCS	SBABO	LBABO	BCS	QICSA	PSO-P
mknapcb1	5.100.00	24381	**24026**	*23840*	20905	20604	23510	23416	22525
	5.100.01	24274	*23761*	**23859**	20188	20449	22938	22880	22244
	5.100.02	23551	*23208*	**23409**	20149	19272	22518	22525	21822
	5.100.03	23534	**23188**	*23104*	20092	20162	22677	22727	22057
	5.100.04	23991	*23464*	**23724**	20421	20797	23232	22854	22167
mknapcb4	10.100.00	23064	*22556*	**22585**	20075	19175	21841	21796	20895
	10.100.01	22801	*22217*	**22342**	20046	18712	21708	21348	20663
	10.100.02	22131	*21581*	**21617**	19397	18566	20945	20961	20058
	10.100.03	22772	*22111*	**22254**	20063	19655	21395	21377	20908
	10.100.04	22751	**22254**	*22244*	20040	18673	21453	21251	20488

Table 4 shows the experimental results of SHABOCS, LHABOCS, SBABO, LBABO, BCS, QICSA and PSO-P algorithms on some hard instances of mknapcb1 and mknapcb4. Benchmark name, problem size and the best-known solutions are mentioned in columns 1, 2 and 3 in that order. Columns 4, 5, 6, 7, 8, 9 and 10 represent SHABOCS, LHABOCS, SBABO, LBABO, BCS, QICSA and PSO-P solutions, respectively. The obtained results show that:

- The proposed algorithms' outcomes are very near to the best-Known solutions.
- LHABOCS outperforms SHABOCS with the majority of instances (except with 5.100.00, 5.100.02, 10.100.04).

Compared with SBABO, LBABO, BCS, QICSA and PSO-P, the proposed algorithms give very well and promising results.

The statistical Friedman test in Figure 3 presents a comparison of the SHABOCS, LHABOCS, SBABO, LBABO, BCS, QICSA and PSO-P algorithms on hard MKP instances. The SHABOCS and LHABOCS algorithms rank first in the Friedman test. This test proves that:

- The significant difference between LHABOCS and SBABO, LBABO, PSO-P algorithms is confirmed.
- The significant difference between SHABOCS and SBABO, LBABO algorithms is also appeared.
- The difference between SHABOCS and LHABOCS results is not statistically significant.
- The difference between Best known and LHABOCS is not statistically significant.

Consequently, the obtained results confirm that the proposed algorithms outperform the other algorithms and prove their effectiveness.

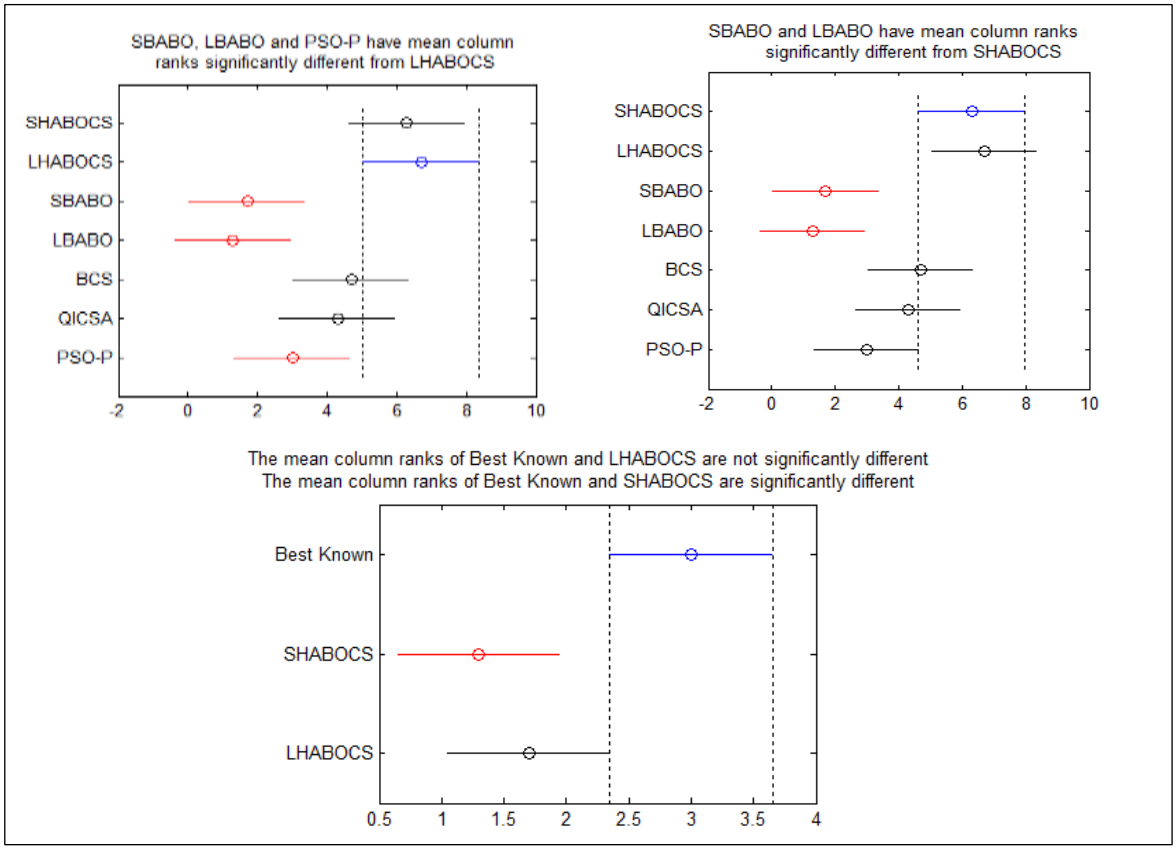

Figure. 3 Friedman test with big MKP instances

Table 5 shows a comparison of average computation time with the mknap1 instances. The computation time is estimated by seconds. In terms of computing time, the obtained results show that the calculation time of the proposed algorithms is higher than that of the binary versions of ABO. The differences in calculation times are explained by the fact that the SHABOCS and LHABOCS require more steps than SBABO and LBABO. However, the difference is not very vast and it can be exceeded since the performance of the proposed algorithm exceeds significantly that of the binary versions of the classical ABO.

According to our knowledge, our contribution represents the first hybrid ABO and CS algorithm. To prove the effectiveness of a new algorithm, it is recommended to compare it with existing algorithms in the literature. In this regard, comparisons with Particle Swarm Optimization (BPSO, BP1 and PSO-P), harmony search (NGHS), Cuckoo Search (BCS and QICSA) and African Buffalo Optimization (SBABO and LBABO) algorithms are performed. Experimental results showed that the proposed algorithm with its two versions is obviously more efficient in finding optimal solutions. The effectiveness of our approaches is explained by the good exploitation and exploration of the search space. This is assured by:

- The ABO technique in updating the location of each buffalo (solution).
- The CS technique in replacing a fraction of worse buffaloes with good buffaloes. This leads the algorithm to effectively explore the search space and locate potential solutions.

Table. 5 Comparative CPU time

N°	SBABO	LBABO	SHABOCS	LHABOCS
1	3.09	2.99	4.81	4.63
2	3.38	2.72	4.72	4.79
3	3.49	3.30	4.54	4.84
4	3.55	4.28	3.9	5.14
5	3.75	5.11	5.64	5.92
6	4.97	5.45	5.5	6.35
7	5.95	5.83	7.17	8.01

It is notable that the algorithms performance is insensitive to their parameters such as lp1 and lp2 and pa. These three 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 powerful of the proposed algorithms is lies essentially to three components: selection of the best solution, local exploitation, and global exploration of the search space. The proposed algorithms are more generic and can be implemented easily for other binary or continuous optimization problems.

8. CONCLUSION AND PERSPECTIVES

In this paper a new hybrid ABO and CS metaheuristic called HABOCS is proposed. In the classical ABO, the solution is updated according to both its historic and the best solution meeting by the group. This allows to good exploration and exploitation of research space. However, the algorithm may suffer from the local optimum problem, especially when the best solution of both buffaloes and group are approaching. CS uses a phase that replaces a portion of bad solutions by others with good quality. This phase allows a good exploration of the search space and helps the algorithm to escape from the local optimum problem. Based on these observations, we thought about integrating the phase of changing a portion of bad solutions to ABO algorithm in order to improve its performance.

Furthermore, two binary versions of the new hybrid metaheuristic are proposed to prove the effectiveness of HABOCS method in solving NP-hard binary combinatorial optimization problems. In the first version called SHABOCS, the Sigmoid Function and probability model are used to generate binary solutions. In the second version named LHABOCS, some logical operators are used to operate the binary solution. The proposed algorithms are used to solve two NP-hard binary combinatorial optimization problems: KP and MKP problems, which have many practical applications.

The obtained results are compared with Particle Swarm Optimization (BPSO, BP1 and PSO-P), harmony search (NGHS), Cuckoo Search (BCS and QICSA), binary versions of ABO (SBABO and LBABO) algorithm and the best-known solution. Experimental studies proved the feasibility and the effectiveness of the proposed algorithms. They confirmed that excellent and promising results were given by our proposed algorithms. However, there are several issues to improve the performance of the proposed algorithms with hard MKP instances like integrating a local search method to the algorithms. Moreover, the introduction of some specified knapsack heuristic operators using problem specific knowledge will have a positive influence on the solutions' quality. The proposed algorithms can also be applied to solve many other binary or continuous optimization problems and real industrial ones.

REFERENCE

1 S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "*Optimization by Simulated Annealing*", Science, vol. 220, no 4598, p. 671‑680, mai 1983.

2 J. H. Holland, "*Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*", MIT Press Cambridge, MA, USA. 1992.

3 F. Glover, "*Future paths for integer programming and links to artificial intelligence*", Computers & Operations Research, vol. 13, no 5, p. 533‑549, janv. 1986.

4 X.S. Yang and S. Deb, "*Cuckoo Search via Lévy flights*", in 2009 World Congress on Nature Biologically Inspired Computing (NaBIC), 2009, p. 210‑214.

5 J.B. Odili, M.N.M. Kahar., S. Anwar. "*African Buffalo Optimization: A Swarm-Intelligence Technique*". Procedia Computer Science 76, p. 443- 448. Elsevier. 2015.

6 I. Fister Jr., X.-S. Yang, I. Fister, J. Brest, and D. Fister, "A Brief Review of Nature-Inspired Algorithms for Optimization", Elektrotehniski vestnik. vol 80, n°3,p. 1- 7. juill. 2013.

7 J.B. Odili and M.N.M. Kahar. "*Solving the Traveling Salesman's Problem Using the African Buffalo Optimization*". Computational Intelligence and Neuroscience. vol 2016, Article ID 1510256. Hindawi Publishing Corporation. 2015.

8 R. Padmapriya and D. Maheswari. "*Channel Allocation Optimization using Africain Buffalo Optimization Super Vector Machine for Networks*". Asian Journal of Information Technology. vol 16, n°10, p. 783 - 788. 2017.

9 A. Gherboudj, "*Two discrete binary versions of African Buffalo Optimization metaheuristic*" Dhinaharan Nagamalai et al. (Eds): ACSIT, ICITE, SIPM – 2018, p. 33-46. DOI : 10.5121/csit.2018.80803. 2018.

10 B. Almonacid, J. Reyes-Hagemann, J. Campos-Nazer, and J. Ramos-Aguilar, "*Selecting a biodiversity conservation area with a limited budget using the binary African buffalo optimisation algorithm*", IET Software, vol. 12, no 2, p. 96‑111, juill. 2017.

11 X.-S. Yang and S. Deb, "*Engineering Optimisation by Cuckoo Search*", Int. J. Mathematical Modelling and Numerical Optimisation. vol. 1, n°4, p. 330 - 343. 2010.

12 A. Gherboudj, A. Layeb, and S. Chikhi, "*Solving 0-1 knapsack problems by a discrete binary version of cuckoo search algorithm*", International Journal of Bio-Inspired Computation, vol. 4, no 4, p. 229, 2012.
13 C.-J. Liao, C.-T. Tseng, and P. Luarn, "*A Discrete Version of Particle Swarm Optimization for Flowshop Scheduling Problems*", Comput. Oper. Res., vol. 34, no 10, p. 3099–3111, oct. 2007.
14 S. Huang, N. Tian, Y. Wang, and Z. Ji, "*Multi-objective flexible job-shop scheduling problem using modified discrete particle swarm optimization*", Springerplus, vol. 5, no 1, p. 1432, 2016.
15 M. Ammi and S. Chikhi, "*Cooperative Parallel Metaheuristics based Penguin Optimization Search for Solving the Vehicle Routing Problem*", IJAMC, vol. 7, no 1, p. 1‑18, janv. 2016.
16 A. Gherboudj and S. Chikhi, "*BPSO Algorithms for Knapsack Problem », in Recent Trends in Wireless and Mobile Networks*", A. Özcan, J. Zizka, et D. Nagamalai, Éd. Springer Berlin Heidelberg, 2011, p. 217‑227.
17 M. Kong and P. Tian, "*Apply the Particle Swarm Optimization to the Multidimensional Knapsack Problem*", in Artificial Intelligence and Soft Computing – ICAISC 2006, 2006, p. 1140‑1149.
18 B. A. Julstrom, "*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, p.607- 614. 2005.
19 A. Singh and A. S. Baghel, "*A New Grouping Genetic Algorithm for the Quadratic Multiple Knapsack Problem*", in Evolutionary Computation in Combinatorial Optimization. p. 210‑218. 2007
20 D. Pisinger. "*Where are the hard knapsack problems?*". Computers and Operations Research, vol. 32, n°. 9, p.2271- 2284. 2005.
21 E. Angelelli, R. Mansini, and M. Grazia Speranza, "*Kernel Search: A General Heuristic for the Multi-dimensional Knapsack Problem*", Comput. Oper. Res., vol. 37, no 11, p. 2017–2026, nov. 2010.
22 M. Vasquez and Y. Vimont, "*Improved results on the 0–1 multidimensional knapsack problem*", European Journal of Operational Research, vol. 165, no 1, p. 70‑81, août 2005.
23 D. Zou, L. Gao, S. Li, and J. Wu, "*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. vol 11, n°2, p. 1556 -1564. March. 2011.
24 J. Kennedy and R. C. Eberhart, "*A discrete binary version of the particle swarm algorithm*", in IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, Orlando, FL, USA, 1997, vol. 5, p. 4104‑4108. 1997.
25 A. Layeb, "*A novel quantum inspired cuckoo search for knapsack problems*", International Journal of Bio-Inspired Computation, vol. 3, no 5, p. 297‑305, janv. 2011.

AUTHOR PROFILE