I.J. Information Engineering and Electronic Business, **2013, 4, 8-15**
Published Online October 2013 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijieeb.2013.04.02

A Multi-objective Binary Cuckoo Search for Bi-criteria Knapsack Problem

Abdesslem Layeb
MISC Lab., Computer science department, Mentouri University of Constantine, Algeria
Layeb@umc.edu.dz

Nesrine Lahouesna
MISC Lab., Computer science department, Mentouri University of Constantine, Algeria
Lahouesna@umc.edu.dz

Bouchra Kireche
MISC Lab., Computer science department, Mentouri University of Constantine, Algeria
Kireche@umc.edu.dz

***Abstract*—** Cuckoo Search (CS) is one of the most recent population-based metaheuristics. CS algorithm is based on the cuckoo's behavior and the mechanism of Lévy flights. The Binary Cuckoo Search algorithm (BCS) is new discrete version used to solve binary optimization problem based on sigmoid function. In this paper, we propose a new cuckoo search for binary multiobjective optimization. Pareto dominance is used to find optimal pareto solutions. Computational results on some bi-criteria knapsack instances show the effectiveness of the proposed algorithm and its ability to achieve good quality solutions.

***Index Terms*—** Combinatorial optimization, Evolutionary computation, Cuckoo Search, Binary Cuckoo Search, knapsack problem, multi-objective optimization.

I. INTRODUCTION

The combinatorial optimization plays a very important role in operational research, discrete mathematics and computer science. The aim of this field is to solve several combinatorial optimization problems that are difficult to solve. Generally there is one objective to optimize, but often several objectives are considered to be optimized, which makes the problem hard to resolve, this kind of problem is called multi-objective optimization problem. Several techniques were developed to solve such problems that can be classified into three classes; aggregate methods, non Pareto methods, and Pareto methods [1]. The Pareto methods are widely used, because they give generally good solutions. The Pareto methods are based on the notion of Pareto dominance, a solution dominates another solution if there is a partial order between the objectives of the two solutions, given a set of objectives, a solution Pareto dominates another if the first is not inferior to the second in all objectives, and, moreover, there is at least one objective where it is strictly better [1].

On the other hand, evolutionary computation has been proven to be well appropriate to solve complex multiobjective problems. It presents many interesting features such as adaptation, emergence, learning and parallelism [2]. Evolutionary algorithms are capable to explore the Pareto optimal front of multi objective optimization problems. They are able to find several Pareto-optimal solutions in a single run. Recently, solving multiobjective optimization problems and its applications using evolutionary algorithms have been investigated by many authors [2-3].

In recent years, optimizing by swarm intelligence has become a research interest to many research scientists of evolutionary computation fields. The main algorithm for swarm intelligence is Particle Swarm Optimization (PSO) [3, 4], which is inspired by the paradigm of birds grouping. PSO was used successfully in various hard optimization problems. One of the most recent variant of PSO algorithm is Cuckoo Search algorithm (CS). CS is an optimization algorithm developed by Xin-She Yang and Suash Deb in 2009 [5]. It was inspired by the obligate brood parasitism of some cuckoo species by laying their eggs in the nests of other host birds (of other species). Some bird's host can involve direct conflicts with the intruding cuckoos. For example, if a bird's host discovers that the eggs are strange eggs, it will either throw these alien eggs away or simply abandon its nest and build a new nest elsewhere [6]. The cuckoo's behavior and the mechanism of Lévy flights [7, 8] have leading to design an efficient inspired algorithm performing optimization search. The recent applications of Cuckoo Search for optimization problems have shown its promising effectiveness.

Binary Cuckoo Search (BCS) is new binary optimization based on the core of the CS algorithm used to cope with binary optimization problems. The main difference between the original version of CS 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 discrete 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 second aim of this paper is to prove that the Cuckoo Search algorithm is effective in dealing with binary combinatorial optimization problems [9].

The aim of this paper is to present an extension of BCS to solve multiobjective problems and formulate a multiobjective binary cuckoo search (MOBCS) algorithm. To validate and prove the performance and the effectiveness of our Binary multi-objective Cuckoo Search algorithm, we have tested it on some bi-objective knapsack problem instances. Experimental results show the effectiveness of the proposed algorithm and its ability to achieve good quality solutions.

The remainder of this paper is organized as follows. Section 2 presents the knapsack problems formulation. An overview of the cuckoo search algorithm is presented in section 3. In section 4, the proposed algorithm is described. Experimental results are discussed in section 5, and a conclusion is provided in the sixth section of this paper.

II. KNAPSACK PROBLEMS

The knapsack problem (KP) is an NP-hard problem [10]. Numerous practical application of the KP can be found in many areas involving 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 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:

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

$$\text{Subject} \sum_{i=1}^{N} w_i x_i \le C \tag{2}$$

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

Many variants of the knapsack problem were proposed in the literature including the Multiobjective Knapsack Problem (MOKP). MOKP is an important issue in the class of knapsack problem. It is an NP-hard problem [11]. In the MOKP, each item x_i has two or many profits p_i^k where k is the number of profit. In MOKP, we can have a single knapsack to fill or M knapsacks 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 MOKP is to find a subset of objects that maximize the different profit function without exceeding the capacity of all dimensions of the knapsack. In our case, we have not only one solution, but a set of Pareto optimal solutions; bi-criteriaMOKP can be stated as follows:

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

$$\text{Maximize} \sum_{i=1}^{N} p_i^2 x_i \tag{4}$$

$$\text{Subject} \sum_{i=1}^{N} w_{ij} x_i \le C_j \quad j = 1...M \tag{5}$$

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

Clearly, there are 2^N potential solutions for these problems. It is obviously that multi-objective knapsack problem is a hard combinatorial optimization problem. Consequently, several techniques have been proposed to deal with knapsack problems. 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 this paper, we are interested by applying Binary Cuckoo Search algorithm to solve these problems.

III. CUCKOO SEARCH AND BINARY CUCKOO SEARCH ALGORITHMS

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 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 Cuckoo Search (CS) [5] which is based on life style 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 Cuckoo Search proposed by Yang and Deb in 2009 [5] 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 p_a of the n nests being replaced by new nests (with new random solutions at new locations).

The generation of new solutions x_i^{t+1} is done by using Lévy flights (equation.6). Lévy flights essentially provide a random walk while their random steps are drawn from a Lévy distribution for large steps which has an infinite variance with an infinite mean (equation.7). Here the consecutive jumps/steps of a cuckoo essentially form a random walk process which obeys a power-law step-length distribution with a heavy tail [5].

$$x_i^{t+1} = x_i^t + \alpha \oplus Lévy(\lambda) \tag{6}$$

$$Lévy \sim \mathrm{u} = \mathrm{t}^{-\lambda} \quad 1 < \lambda \leq 3 \tag{7}$$

Where x_i^{t+1} and x_i^t represent the solution i at times t+1 and t, respectively.$\alpha > 0$ is the step size which should be related to the scales of the problem of interest, generally we take $\alpha = 1$. The product $\oplus$ means entry-wise multiplications. This entry-wise product is similar to those used in PSO, but here the random walk via Lévy flights is more e fficient in exploring the search space as its step length is much longer in the long run. The generation of Levy step size is often tricky, and a good algorithm is Mantegna's algorithm [5]. The main characteristic of CS algorithm is its simplicity. In fact, comparing with other population or agent-based metaheuristic algorithms such as particle swarm optimization and harmony search, there are few parameters to set. The applications of CS into engineering optimization problems have shown its encouraging efficiency. For example, a promising quantum inspired cuckoo search algorithm is recently proposed to solve knapsack problems [12]. An efficient computation approach based on cuckoo search has been proposed for data fusion in wireless sensor networks [13]. The basic steps of cuckoo search algorithm are presented in a pseudo code shown in Figure 1.

```
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
```

Figure. 1. Cuckoo Search algorithm.

Recently, a binary version of CS is proposed in [9] called BCS. BCS uses the sigmoid function in order to generate binary solutions. 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 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 second module contains the objective function and the selection operator. The selection operator is similar to the elitism strategy used in genetic algorithms.

IV. THE PROPOSED MULTIOBJECTIVE BINARY CUCKOO SEARCH ALGORITHM (MOBCS)

In order to deal with the multiobjective optimization problems with K different objectives function, we have modified the standard BCS algorithm; we have added the notion of the dominance in order to select the best solutions. The main steps of the proposed algorithm are described in figure 2.

Like any other population-based metaheuristics, the first step in the MOBCS algorithm involves setting the parameters for the algorithm. The main advantage of the MOBCS algorithm is that there are fewer parameters to be set in this algorithm than in PSO. In MOBCS, there are essentially two main parameters to initialize, population size N and a fraction p_a of the worst nests to be rejected and replaced. In the second step, a swarm of N host nests is created to represent some possible solutions. However, it should be noted that in order to reduce the convergence time, it is recommended to start with a diverse population containing both good and bad solutions. For this purpose, we can use some heuristics to build good initial solutions. A Pareto archive is created containing non-dominated solutions. The Pareto frontier is the set of points from feasible region F that are not strictly dominated by any other point in F. The dominance relation is defined as follow:

Let $x_1, x_2 \in F$, and F is a feasible region of solutions. We say that x_1 dominates x_2 if $f(x_1)$ is said to be partially greater than $f(x_2)$, i.e.

$$f_i(x_1) \geq f_i(x_2), \forall\, i = 1,2,...,n$$

And $f_i(x_1) > f_i(x_2), \exists\, i = 1,2,...,n$,

If there is no $x \in F$ that dominates x_1 , then x_1 is called Pareto optimal solution.

The algorithm progresses through a number of generations according to the MOBCS dynamics. During each iteration, the following main tasks are performed. A new cuckoo is built using the Lévy flights operator. The Lévy flight provides a random walk that consists of taking successive random steps in which the step lengths are distributed according to a heavy tailed probability distribution. The next step is to evaluate the current cuckoo. For that, we apply the sigmoid function to get a binary solution which represents a potential solution for the binary optimization problems. After this step, we update the Pareto archive containing the non-dominated solutions. In our algorithm, in order to generate the next population, we have used an aggregate function of the objective function in order to rank the solutions of the current population. The process is repeated until the final condition is met.

```
Input: N and $p_a$
Output: the last best solution

Objective function $f(x), x = (x_1, .., x_d)^T$ ;
Initial a population of N host nests $x_i$ (i = 1, 2, ..., N);
Construct the Pareto archive
  while (t < MaxGeneration) or (stop criterion){
    • Get a cuckoo (say i) randomly by Lévy flights;
    • Evaluate its quality/fitness $F_i$;
    • Update the Pareto archive
    • Abandon a fraction ($p_a$) of worse nests according to aggregate function;
    • Build new ones at new locations via Lévy flights;
    • Evaluate the new population;
    • Update the Pareto archive
  }
```

Figure. 2. Multiobjective Binary Cuckoo Search.

V. EXPERIMENTAL RESULTS AND DISCUSSION

The proposed MOBCS algorithm was implemented in Matlab 7. To assess the efficiency and performance of our algorithm (MOBCS), several experiments were performed. The used instances are dived into three classes: small tests, medium tests, and large instances and containing two objectives and one constraint. The instances are created randomly with different problem sizes, in which the weights and profits are selected randomly. Each instance is called *BoknapN_x* where *N* is the number of items taken in the range 50 to 3000, and *x* is the test number. In these instances, the knapsack capacity is calculated by using the following formula:

$$C = \frac{3}{4}\sum_{i=0}^{N} w_i \qquad (8)$$

In all experiments, the parameters of cuckoo search algorithm are set as follows: The size of initial population N= 40; the maximum iteration number is chosen in the range [2000, 50000], 2000 iterations for small instances and 50000 iterations for hard instances; the discovery rate Pa=0.25.

The results are summarized in the following tables (1, 2, and 3), the first column is the test name, and the second is the non-dominated optimal solutions, where the last column is the mean of the non-dominated solutions. As we have noted, the proposed approach is able to find good non-dominated solutions compared to the initial solutions. The figures 3, 4, 5, 6 and 7 show an example of Pareto optimal solutions, the red points are the non-dominated solutions and the green ones are the dominated solutions.

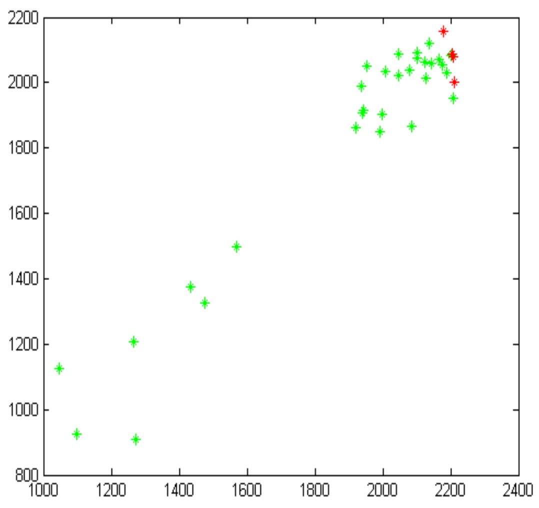

Figure 3.Optimal front pareto for instance Boknap100_01.

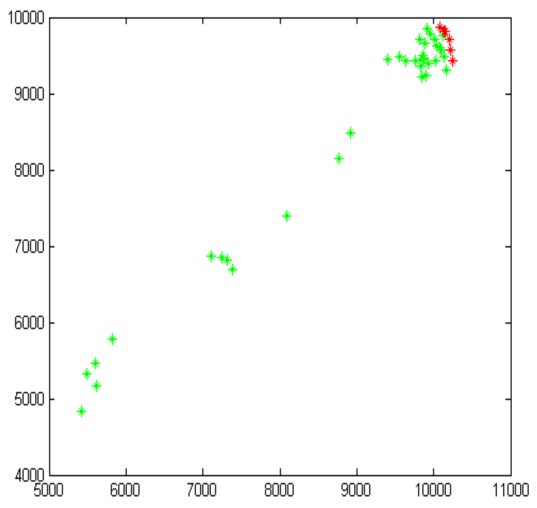

Figure 4. Optimal front pareto for instance Boknap500_01

TABLE 1. Results for small instances

Instance	Pareto final	Pareto Front mean
Boknap50_1	{(1060,1121) (1116,1092) (1097,1099) (1087,1100)}	(1090,1103)
Boknap50_2	{(1233,1058) (1256,1011)}	(1244.5, 1034.5)
Boknap50_3	{(1042,1022) (974,1056) (994,1040) (989,1042)}	(999.7, 1040)
Boknap50_4	{(1083,1141) (1025,1142) (995,1155)}	(1034.3, 1146)
Boknap100_1	{(2178,2158)(2205,2087)(2209,2079) (2210,2003)}	(2081.8, 2200.5)
Boknap100_2	{(2081,2000) (1981,2004) (2132,1888) (2119,1962)}	(2078.3, 1963,5)
Boknap100_3	{(2227,1986) (2234,1960) (2126,1987) (2038,1991)}	(21563, 1981)
Boknap100_4	{(2133,2306) (2182,2145) (2178,2165) (2148,2235)}	(2160.3, 2212.8)
Boknap200_1	{(3994,4165) (4074,4011) (3995,4050) (4056,4024)}	(4029.8, 4062.5)
Boknap200_2	{(4158,4191) (4041,4199) (4017,4211) (4167,4130) (4168,4092)}	(4110.2, 4164.6)
Boknap200_3	{(4109,4209) (4125,4183) (3971,4219) (4126,4129) (3919,4231)}	(4050,4194.2)
Boknap200_4	{(4112,3936) (4176,3715) (4144,3799) (4034,3965) (3991,3982)}	(4091.4, 3879.4)

TABLE 2. Results for medium instances

Instance	Pareto final	Pareto Front mean
Boknap500_1	{(10152,9810)(10210,9719)(10246,9431) (10133,9818)(10215,9570)(10081,9872)}	(10173,9703.3)
Boknap500_2	{(10054,10064)(10147,9829)(9808,10114)}	(10003,10002)
Boknap500_3	{(10048,9665)(9936,9668)(10145,9423)}	(10043,9585.3)
Boknap500_4	{(101399,10240)}	(10139.9 ,10240)
Boknap700_1	{(14400,13740)(13795,13834) (13891,13788)}	(14029,13787)
Boknap700_2	{(14049,13857)}	(14049,13857)
Boknap700_2	{(13878,14190)(13560,14198)(14028,13486) (13960,13697)(13915,13973)}	(13868,13909)
Boknap700_4	{(14231,13867)(13284,13870)}	(13758,13869)
Boknap1000_1	{(19878,19497)(19672,19779)(19718,19626) (20052,19307)(19418,19783)(19692,19659) (19775,19520)(19954,19330)}	(19770,19563)
Boknap1000_2	{(19239,19336)(19283,18929)(19318,18923) (18762,19354)(18669,19606)}	(19054,19230)
Boknap1000_3	{(19905,19247) (19920,18955)(19092,19340) (19355,19323)}	(19568,19216)
Boknap1000_4	{(19364,19907)(18902,10018)(18914,19933)}	(19060,19953)
Boknap1200_1	{(24302,23963)(23914,23980) (24303,23488)}	(24173,23810)
Boknap1200_2	{(23044,24015)(22703,24398)}	(22874,24207)
Boknap1200_3	{(23293,23405)(23085,23500) (23051,23644) (23470,23235)}	(23225,23446)
Boknap1200_4	{(23807,23400)(23843,22950) (22950,22912)(23559,23465)}	(23790,23182)

Table 3. Results for large instances

Instance	Pareto final	Pareto Front mean
Boknap1500_1	{(29285,29761)}	(29285,29761)
Boknap1500_2	{(28748,29769) (29089,29506) (29086,29515)}	(28974,29597)
Boknap1500_3	{(30548,28853)(30006,28890)}	(30277,28872)
Boknap1500_4	{(28792,19344) (28902,28091) (28878,29213) (28216,29454) (28177,29649) (28477,29452)}	(28574.4, 29367)
Boknap2000_1	{(39618,38126)(39455,38351)}	(39537,38239)
Boknap2000_2	{(39317,39330)}	(39317,39330)
Boknap2000_3	{(38961,39615) (38912,39636) (39050,39005) (39120,38876) (38967,39207)}	(39002,39268)
Boknap2000_4	{(38964,39322) (38069,39650) (39165,38753) (38683,39411)}	(38720,39284)
Boknap3000_1	{(59147,58171)(58136,58497)}	(58462,58334)
Boknap3000_2	{(58938,57794) (58462,57838) (58009,58167) (58039,57849)}	(58362,57912)
Boknap3000_3	{(57881,58839)}	(57881,58839)
Boknap3000_4	{(58581,59576) (58314,60255)}	(58448,59916)

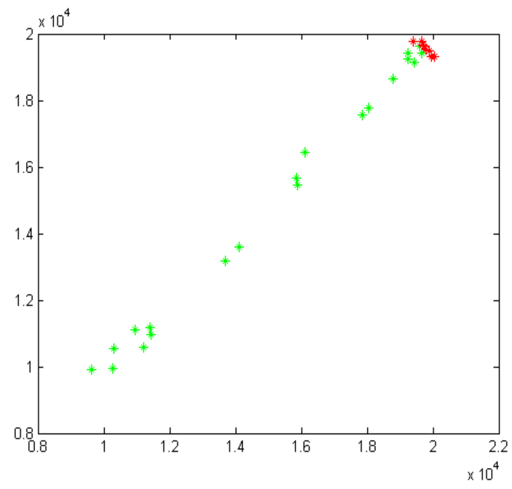
Figure 5. Optimal front pareto for instance Boknap1000_01

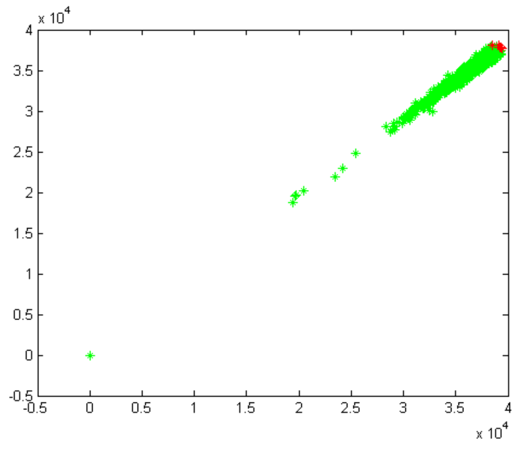
Figure 6. Optimal front pareto for instance Boknap2000_01

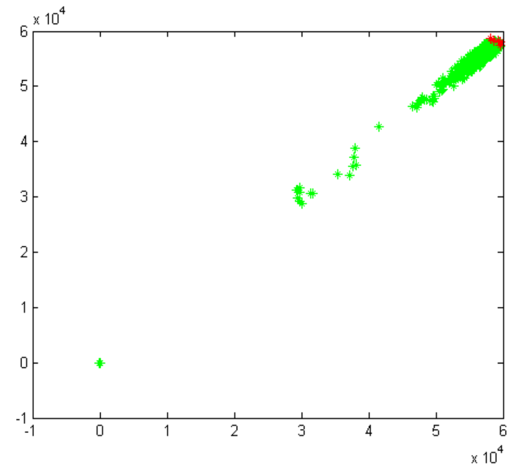
Figure 7. Optimal front pareto for instance Boknap3000_01.

VI. CONCLUSION

In this work, we have presented a new Multi-objective binary Cuckoo Search algorithm called (MOBCS). The main feature of our approach consists in using of sigmoid function and probability model in order to generate binary solutions. Moreover, the Pareto dominance is integrated in the standard algorithm BCS in order to find the non-dominated optimal solutions. To evaluate and prove the performance and the effectiveness of the proposed algorithm MOBCS, we have applied and tested it on some MOKP instances. The experimental studies prove the feasibility and the effectiveness of our approach. Indeed, in the all the cases MOBCS is able to find the Pareto optimal solutions. However, there are several issues to improve our algorithm. For example, in order to improve the performance of our algorithm, it is better to integrate other selection based Pareto operations.

REFERENCES

[1] A. Warburton, Approximation of Pareto optima in multiple-objective, shortest-path problems, Operations Research, Vol.35, No.1, pp.70–79, 1987.

[2] K. Deb et al., A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II, Proceedings of the Parallel Problem Solving from Nature. Springer Lecture Notes in Computer Science No. 1917, pp. 849-858 Paris, France, 2000.

[3] M.R. Sierra and C.A.C. Coello, Multi-objective particle swarm optimizers: A survey of the state-of-the-art, International Journal of Computational Intelligence Research , Vol. 2, No. 3, pp.287–308, 2006

[4] R. C. Eberhart, Y.Shi, and J. Kennedy, Swarm Intelligence. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann, San Francisco, CA, USA, 2001.

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

[6] P. Barthelemy, J. Bertolotti, D. S. Wiersma, A Lévy flight for light. Nature, Vol. 453, pp. 495-498, 2008.

[7] R. B. Payne, M. D. Sorenson, and K. Klitz, The Cuckoos, Oxford University Press, 2005.

[8] I. Pavlyukevich, Lévy flights, non-local search and simulated annealing, J. Computational Physics, Vol. 226, pp. 1830-1844, 2007.

[9] A., Gherboudj, A. Layeb, and S. Chikhi, Solving 0-1 knapsack problems by a discrete binary version of cuckoo search algorithm, in the International Journal of Bio-Inspired Computation, Vol. 4, No. 4, pp. 229-236, 2012.

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

[11] C. Bazgan, H. Hugot, and D.Vanderpooten, "Solving efficiently the 0 – 1 multiobjective knapsack problem," Computers and Operations Research,Vol. 36, No.1, pp. 260–279, 2009.

[12] A. Layeb, A novel quantum inspired cuckoo search for knapsack problems, Int. J. Bio-Inspired Computation, Vol. 3, No. 5, pp 297-305, 2011.

[13] M. Dhivya, Energy Efficient Computation of Data Fusion in Wireless Sensor Networks Using Cuckoo Based Particle Approach, Int. J. of Communications, Network and System Sciences, Vol. 4, No. 4, pp. 249-255, 2011.

Authors' Profiles

Abdesslem Layeb: is an Associate Professor in the Department of Computer Science at the University of Constantine, Algeria. He is a member of MISC Laboratory. He received his PhD in Computer Science from the University of Constantine, Algeria. He is interested in combinatorial optimisation methods and their applications in solving several problems.

Nesrine Lahouesna: has a Master degree in Computer science from the University of Constantine

Bouchra Kireche: has a Master degree in Computer science from the University of Constantine