Solving the Multiple choice Multidimensional Knapsack problem with ABC algorithm

Arij Mkaouar
National Engineering School of Sfax
Sfax,Tunisia
arij.mkaouar@yahoo.fr

Skander Htiouech
CS and AI department
University of jeddah
shtiouech@uj.edu.sa

Habib Chabchoub
Al Ain University of Science and Technology
Abu Dhabi, United Arab Emirates
habib.chabchoub@aau.ac.ae

***Abstract*— The Multidimensional Multiple-choice Knapsack Problem (MMKP) is an NP-hard problem. Many heuristics algorithms have been developed to solve this combinatorial optimization problem. In this work, a new method based on Artificial Bee Colony algorithm (ABC) and surrogate constraint is proposed to solve the MMKP. Experimental results show that this method is competitive with the state-of-the-art approaches.**

Keywords—MMKP, ABC algorithm, Surrogate constraint, Combinatorial optimization problem

I. Introduction

The MMKP is one of the most challenging combinatorial optimization problems. Its high computational complexity arises from combining two hard constraints which are the multidimensional resource constraint and the choice constraint. Given n disjoint groups, with n_i items in each group i $(i=1,.., n)$. Let m the number of the knapsack resources. The capacity of each resource k $(k=1,.., m)$ is given by b^k. Each item j $(j=1,..,n_i)$ of group i has a non-negative profit value c_{ij} and consume a_{ij}^k amount from each resource k. The MMKP consists in picking exactly one item from each group in order to maximize the total profit value without exceeding the resources capacity. The MMKP can be expressed as:

$$Maximize \sum_{i=1}^{n}\sum_{j=1}^{n_i} c_{ij}x_{ij} \tag{1}$$

$$s.c \qquad \sum_{i=1}^{n}\sum_{j=1}^{n_i} a_{ij}^k x_{ij} \le b^k, k=1,\dots,m \tag{2}$$

$$\sum_{j=1}^{n_i} x_{ij} = 1, i=1,\dots,n \tag{3}$$

$$x_{ij} \in \{0,1\}, i=1,..,n, j=1,\dots,n_i \tag{4}$$

A pick represent a configuration of n items, one from each group. Equation (2) ensures that the availability of the resources is not exceeded. If $m=1$, then the MMKP becomes a multi-choice knapsack problem [33] (called MCKP). Equation (3) guarantees that exactly one item from each group must be selected. If this constraint is neglected then the MMKP becomes a multidimensional knapsack [32] (called MKP). In equation (4), the variable x_{ij} is equal to 1 if the item j from the group i is picked, 0 otherwise. To avoid trivial solutions, we assume the following inequality for every $1\le j \le n_i$.

$$\sum_{i=1}^{n} \min_{1\le j\le n_i}\{a_{ij}^k\} \le b^k \le \sum_{i=1}^{n} \max_{1\le j\le n_i}\{a_{ij}^k\}, for\ k=1,\dots,m$$

The MMKP is an NP-hard in the strong sense [2]. It is not always possible to find a feasible solution in a reasonable computing time especially for big instances. Since exact methods are reserved for limited scale problem, heuristic approaches are needed for solving it. In this work, we propose a new algorithm called ABC_MMKP based on ABC algorithm and surrogate constraint. The paper is organized as follows. In Section 2, we describe the relevant related works on MMKP. ABC algorithm is presented in section 3. The details of our contribution are described in section 4. The computational results are reported in Section 5. Section 6 summarizes our contributions and discusses directions for further improvement.

II. Related work

Two general sets of algorithms for combinatorial optimization problem are defined, exact algorithms and heuristics. Exact algorithms guarantee the optimality of the solution but they are only able to handle problems of limited size [6]. Almost exact methods for MMKP are based on branch and bound [3] [4] [5]. In 1998, Khan [5] suggested the first branch and bound algorithm for MMKP. In 2007, Sbihi [4] described a best–first strategy to solve an exact branch and bound algorithm providing a better results than Khan [5]. In 2008, Razzazi and Ghasemi [8] used depth-first strategy to present a more powerful branching and bound algorithm. This algorithm outperforms the approach of Sbihi [4]. Ghasemi and Razzazi [3] described in 2011 an exact algorithm based on an approximate core to solve MMKP. Good quality solutions are obtained for this approach with up to 5 knapsack constraints and 1000 items.

For MMKP real-time applications, it is an illusion to reach the optimality due to the complexity of the problem and the need for rapid system response. For larger scale instances, several heuristic approaches were proposed to find near-optimal solutions in reasonable computing time [6]. In 1997, Moser [7] proposed the first heuristic for resolving the MMKP based on lagrangian relaxation and repetitive permutation. Other works using a reactive local search (RLS) and a modified reactive local search (MRLS) appeared in [10] and showed better result than Moser [7]. In 2009, a hybrid algorithm that combines local branching with column generation was proposed by Cherfi and Hifi [11] and markedly outperformed all previous approaches. In 2013, Htiouech *et al.* [12] proposed an oscillation method and introduce surrogate constraint to solve the MMKP problem. Xia, Gao and Li [13] presented a fast-stochastic local search heuristic for the MMKP that uses an iterative perturbative search based on penalty weights for dimensions, and an additive weighting to diversify the search. Also, a first level tabu search was proposed by Hiremath *et al.* [14] and performed quite well compared to the legacy heuristic approaches. In [15], Crévits *et al.* developed a semi-continuous relaxation heuristic that relaxes partially the integrality constraints to force the variables values close to 0 or 1. Mansi *et al.* [16] presented another hybrid heuristic also based on the iterative relaxation that uses a new family of cuts to define a reduced problem and a reformulation procedure. In 2017, new iterative pseudo-gap enumeration based on a family of additional cuts derived from the reduced costs constraint of the non-basic variables was proposed by Gao *et al.* [17]. Recently, Caserta *et al.* [18] described a robust algorithm based on Lagrangian relaxation and the corridor method CM to solve MMKP. In this model, item's resource consumption values are uncertain. This work severely relied on the mathematical structure of the model and shed lights on prices of robustness. Experiments results showed that the introduction of robustness for the MMKP is very inexpensive in term of additional computing time.

III. ABC ALGORITHM

Swarm intelligence is one of the most modern and interesting research branch in artificial intelligence. It prototypes the collective behavior of biological systems like ant colonies [34] and bird flocking [35]. The intelligent honey bee behavior [36] is another example of swarm intelligence that was modeled in many artificial algorithms. ABC is an optimization algorithm inspired from the food-seeking behavior of honey bees and was proposed by karaboga in 2005 [24].

In ABC algorithm, bees are divided in three different groups: employed bees, onlookers and scouts [25]. The employed bees exploit the food source, carry the information about food source and then share this information with onlooker bees. Onlookers take the responsibility of evaluating the solutions obtained by the employed bees. Scouts are the colony's explorer; they are always searching for new food sources in new areas near the hive. In ABC algorithm, each food source is a possible solution for the problem under consideration and the nectar amount of a food source represents the quality of the solution represented by the fitness value. The number of food sources is the same as the number of employed bees and there is exactly one employed bee for every food source.

Each cycle of the search consists of three steps: moving the employed and onlooker bees onto the food sources and calculating their nectar amounts and determining the scout bees and then moving them randomly onto the possible food sources. A food source represents a possible solution to the problem to be optimized. The nectar amount of a food source corresponds to the quality of the solution represented by that food source. Onlookers are placed on the foods by using "roulette wheel selection" method [23]. Every bee colony has scouts that are the colony's explorers. The explorers do not have any guidance while looking for food. They are primarily concerned with finding any kind of food source. As a result of such behavior, the scouts are characterized by low search costs and a low average in food source quality. Occasionally, the scouts can accidentally discover rich, entirely unknown food sources. In the case of artificial bees, the artificial scouts could have the fast discovery of the group of feasible solutions as a task. One of the employed bees is selected and classified as the scout bee. The classification is controlled by a control parameter called limit. If a solution representing a food source is not improved by a predetermined number of trials, then that food source is abandoned by its employed bee and the employed bee associated with that food source becomes a scout. The number of trials for releasing a food source is equal to the value of limit, which is an important control parameter of ABC algorithm [25]. The ABC algorithm contains four main steps:

a) *The initialization*: each x_m *(m=1,..,SN)* of the solution (food source) population is initialized via the expression :

$$x_m = l_i + rand(0,1) * (u_i - l_i) \tag{5}$$

Where u_i and l_i are the upper and the lower bound of the objective function.

b) *Employed bees phase:* Employed bees determine a neighbor food source v_m using the formula given by equation:

$$v_m = x_m + \emptyset_m(x_m - x_k) \tag{6}$$

Where x_k is a randomly selected food source and $\emptyset_m$ is a random number within the range [-1,1].

The fitness value of the food source is calculated via the formula:

$$fit_m(x_m) = \begin{cases} \dfrac{1}{1 + f_m(x_m)} & if \quad f_m(x_m) \geq 0 \\ 1 + abs(f_m(x_m)) & if \quad f_m(x_m) < 0 \end{cases} \tag{7}$$

Where $f_m(x_m)$ is the objective function value of solution x_m.

c) *Onlooker bees:* The probability value p_m with which x_m is chosen by an onlooker bee can be calculated by using the expression given in equation:

$$p_m = \frac{fit_m(x_m)}{\sum_{m=1}^{SN} fit_m(x_m)} \tag{8}$$

d) *Scout bees phase:* The scout bees discover a new food source by using the equation (5).

The general structure of ABC algorithm is as follows:

```
ABC algorithm

 1. Initialization
 Repeat
     1.  Employed Bees Phase
     2.  Onlooker Bees Phase
     3.  Scout Bees Phase
     4.  Memorize the best solution achieved so far.
```

Until (requirements are met)

A. Motivation to choose ABC algorithm to solve MMKP

- Due to its performance in dealing with many optimization problems.
- Lack of ABC usage in MMKP problem optimization
- Simple logic used in the ABC (easy implementation).
- The existence of a strong analogy to apply ABC algorithm for MMKP.

B. Analogy between the intelligent bee behavior and the proposed algorithm (ABC_MMKP)

In this paper, the proposed algorithm is based on the four main actions of the basic ABC: a generator that use the distance concept and returns new solutions from new zone not yet explored (scout bees), a mutator that explore the local area and returns solutions close to the current solution (employed bees), an evaluator, which evaluates the quality of a solution as a floater (onlookers bees) and a memory to record the best global solution found so far.

The food source is symbolized by the candidate solution and the amount of nectar in this source is represented by the quality of a solution denote.

It is important to note that we have conserved the principle of the 4 main steps of the ABC while modification are made to adopt the ABC to the MMKP. In our algorithm the stochastic aspect of the basic ABC which is based on selecting random number value is replaced by determined mathematical expression to be appropriate to the MMKP problem such is described in section 4.

The algorithm sends bees to search (scout bees for new solutions, employed for improvements around a known solution), and triggers events whenever an improved solution is found, until the end of the algorithm.

Scout bees represent diversification process and employed bees represent the intensification process. More the scout bees (diversification) and employed bees (intensification) work together in collaboration, harmony and balance more efficient algorithm we get and better results we obtained.

IV. Methodology

1) *Finding a feasible solution:* Initially, the item consuming the minimum of resources of each group is selected in order to begin with a solution that gives as much flexibility as possible for its following choices. The feasibility factor f_{ij} for each item *j (j=1,..,n$_i$)* belonging to group *i (i=1,..,n)* is computed such that :

$$f_{ij} = \sum_{k=1}^{m} \frac{r_{ij}^{k}}{b_k}, i = 1, \dots, n, j = 1, \dots, n_i \quad (9)$$

Following the choice constraint (3) the *n* groups must participate for each pick of *n* items, therefore, we start by selecting the item having the lowest f_{ij} value from each group.

If the initial solution is created, swap moves between items of the same groups are processed to generate a population *SN* of candidate solutions. The best solutions found are memorized and then placed in a different mechanism for intensive research in their neighborhood (intensification process). The surrogate constraint is used to get better quality of solutions. The process is repeated until the stop criterion is satisfied. Subsequently, the algorithm updated its memories by keeping only the best-local solutions found so far and the best global solution obtained from the start until the search is stopped.

2) *Diversification:* Scout bees are always searching for new food sources in new area not yet exploited of different distances from the hive. By imitating the intelligent bee behavior, the algorithm looks for new population of feasible candidate solutions in the neighborhood domain of the current solution. To ensure that the new candidate solutions are scattered throughout the search space and to not fail into local optima, in each iteration, the algorithm select a different set $\Theta_{i'}$ of *d* items *(i'=1,..,d)* from the current solution to be exchanged with *d* new items from the same group. The number *d* of items to be exchange diverges from d_{min} (the smallest values allowed by the algorithm) to d_{max} (number of group). The choice of items to be exchange is not stochastic, however, it depends on the lowest values of pseudo-utility ratio U_{ij} for items having $x_{ij} =1$.

$$u_{ij} = \frac{v_{ij}}{\sum_{k=1}^{m} r_{ij}^{k}/C_k} | x_{ij} = 1, i = 1, \dots, n, j = 1, \dots, n_i \quad (10)$$

The *d* new items are selected instead the current items in order to maximize:

$$\Delta_{\alpha} = \sum_{i'=1}^{d} \sum_{m=1}^{k} \frac{r_{i'k}^{k}}{(\Delta_{\beta}^{k})^2} * A_k, k = 1, \dots, m \quad (11)$$

$$\Delta_{\beta} = b_k - \sum_{i=1}^{n} \sum_{\{j | x_{i'j}=1\}}^{n_i} r_{ij}^{k}, k = 1, \dots, m \quad (12)$$

This step is repeated *SN* times with different number *d* of items to be exchange each time. *NS* is the number of the food sources (candidate solutions). After the diversification process, *NS* feasible candidate solutions are obtained. Equations (11) and (12) ensure that the candidate solutions found save as much as possible of the resources consumed which gives more litheness for the intensification step.

3) *Intensification (Surrogate constraint):* When the solution is feasible, the approach improves, little by little, its quality. The improvement must respect the feasibility of the solution by swap moves of items in the same group. Surrogate constraint is used to determine which items to exchange. Proposed by Glover [20] [21], the aggregated relaxation surrogate consists in replacing the constraints of a problem by a single, called surrogate constraint [22]. The application of the relaxation surrogate on the resource constraints of the MMKP gives us a new formulation for equation (2):

$$s.c \sum_{k=1}^{m} u_k \sum_{i=1}^{n} \sum_{j=1}^{n_i} a_{ij}^{k} x_{ij} \leq \sum_{k=1}^{m} b^k, k = 1, .., m \quad (13)$$

Where u is a vector of positives multipliers of dimension *m*. To simplify the reasoning, the constraint (13) is formulated by Htiouech *et al.* [12] as follow:

$$\sum_{i=1}^{n}\sum_{j=1}^{n_i} s_{ij}x_{ij} \leq s_0 \tag{14}$$

$$s_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{n_i} u_k a_{ij}^k \, , and \; s_0 = u_k b^k, k = 1,..,m \tag{15}$$

To generate a surrogate constraint, we compute the equation (16) for each constraint k = 1,.., m.

$$\Delta^k = b^k - \sum_{i=1}^{n}\sum_{\{j/x_{ij}=1\}}^{n_i} a_{ij}^k, k = 1,..,m \tag{16}$$

The value of Δ^k is negative if the constraint k is violated. Since in our algorithm surrogate constraint is only used when the solution is feasible then Δ^k always has a positive value and multipliers value for each constraint is always set up to $(\Delta^k)^{-1}$.

Surrogate constraint enforces the algorithm to focuses on the tightest resources to encourage the search around promising solutions. Swap moves between items belonging to the same group are performed to improve the quality of the solution. Items to exchange are selected in order to:

$$Maximize \left\{ r_{ijh} = \frac{c_{ih}/s_{ih}}{c_{ij}/s_{ij}} | x_{ih} = 1, x_{ih} = 0 \right\} \tag{17}$$

Where the item *h* is picked from the group *i* instead of the item *j*. Incorporating surrogate constraints within the intensification process greatly improve the quality of the solutions. The algorithms repeat the intensification and diversification process until one of the stop condition is satisfied. But this also makes the number of candidate solutions grows in each iteration witch increase the computational complexity of the algorithms. An evaluator process is recommended to select the most contributed candidate solution.

4) *Fitness process:* After diversification and intensification process, candidate solutions so far found are placed into an evaluation process. Evaluation process chooses a solution based on an accurate search, which depends on the quality of the solution (fitness). The selection probability of a solution s is calculated as follows:

$$p_s = \sum_{i=1}^{n}\sum_{j=1}^{n_i} \frac{v_{ij}}{\sum_{k=1}^{m} a_{ij}^k / b^k} | x_{ij} = 1 \tag{18}$$

Figure 1 describes the algorithm flowchart. The main idea of the algorithm can be simplified in these 4 main steps:

1. Research of candidate solution in new zone area (scout bees)
2. Improvement of the candidate solutions using the intensification process (employed bees)
3. Evaluation of the solutions fitness (onlookers bees)
4. Memorize the best solution reached so far

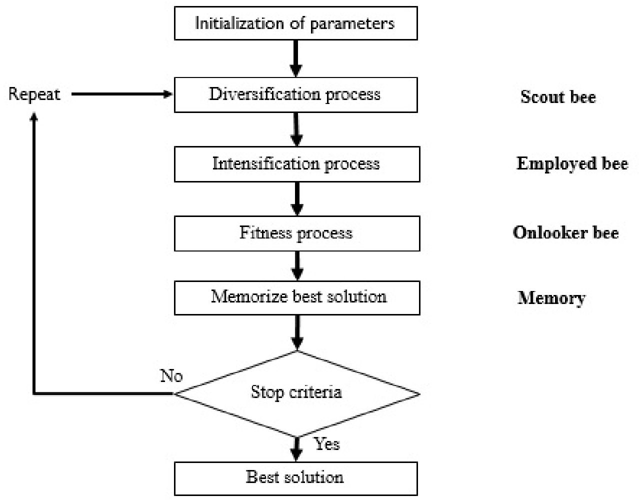

Fig. 1. Flowchart of the proposed the algorithm

The pseudo code of the proposed algorithm is presented in Fig. 2.

```
ABC_MMKP Algorithm
   1. Initialization of the parameters
   2. Compute the feasibility factor $f_{ij}$ of each item
   3. Find a feasible solution s
   Repeat
   /*Diversification*/
   4.    d=$d_{min}$ and NS=0
      While (d≤$d_{max}$)
         5. Calculate the pseudo ratio utility $U_{ij}$
         6. select d items having the lowest $U_{ij}$
            Repeat
               7. exchange items in order to maximize $\Delta_\alpha$
            Until the stop condition is satisfied
         8. memorize the obtained candidate solution
         9. Update d and NS
      Until the stop condition is satisfied
   /*intensification*/
      Repeat (for each candidate solution repeat)
         10. Apply surrogate constraint
         11. Compute $\Delta^k$
         12. Exchange items belonging to the same group
             in order to maximize $r_{ijh}$
         13. Update the global best solution
      Until the stop condition is satisfied
   Until one of the stop criteria is satisfied
```

Fig. 2. The high-level pseudo code for the proposed ABC_MMKP algorithm

V. EXPERIMENTS.

A. Benchmark instances

We test the proposed algorithm (ABC_MMKP) on a set of instances proposed by Khan *et al.* [9] in 2005 (E1) which known in the literature that his instances are very difficult to solve (strongly correlated). E1 contains 13 instances with a variety of data of all sizes, small, medium and large. The number of variables varies between 25 for the smallest and 4000 variables for the largest. The total number of feasible and unfeasible solutions of a MMKP is equal to $\prod_{i=1}^{n} n_i$. The 13th instance, for example, contains 400 groups each with 10 items, which generates 10^{400} combinatorial combinations!

For each instance, we denote *n* the number of groups, n_i the number items in each group and *m* the number of resource constraints. The table I describes the details of the instances.

TABLE I. INSTANCE DETAILS

Instance	n	n_i	m	N
I01	5	5	5	25
I02	5	10	5	50
I03	15	10	10	150
I04	20	10	10	200
I05	25	10	10	250
I06	30	10	10	300
I07	100	10	10	1000
I08	150	10	10	1500
I09	200	10	10	2000
I10	250	10	10	2500
I11	300	10	10	3000
I12	350	10	10	3500
I13	400	10	10	4000

B. Experimental results and discussion

Our algorithm is coded in JAVA NetBeans 8.2 and all experiments were done on a PC with a 2.53 GHz Intel i5 CPU and 3GB of memory.

We present the obtained results by our algorithms on the set of instances E1. To evaluate our approaches, we compare our results with existing literature approaches for the MMKP, namely respectively, Hifi *et al.* [10] (MRLS in 2006), Iqbal *et al.* [28] (Ant in 2010), Htiouech *et al.* [12](Osc in 2013), Xia [13] (SLS_ MMKP en 2015).

TABLE II. COMPARISON BETWEEN ALGORITHMS'S RESULTS

INST	Cplex	MRLS	Ant	Osc	SLS_ MMKP	ABC_ MMKP	CPU
I01	173	173	173	173	173	156	0.008
I02	364	364	364	364	364	354	0.025
I03	1602	1595	1602	1594	1602	1577	0.12
I04	3597	3564	3569	3514	3587.7	3493	0.083
I05	3905.7	3905.7	3905.7	3905.7	3905.7	3905.7	0.264
I06	4799.3	4799.3	4799.3	4799.3	4799.3	4799.3	0.082
I07	24587	24121	24159	24162	24270.06	**24274**	1.023
I08	36877	36110	36240	36405	36397.86	**36492**	2.666
I09	49167	48291	48367	48567	48512.09	**48657**	4.524
I10	61437	60291	60475	60858	60609.18	**60944**	7.709
I11	73773	72283	72558	73022	72718.31	**73163**	10.986
I12	86071	84446	84707	85284	84839.34	**85420**	14.808
I13	98429	96580	96834	97545	97000.42	**97723**	19.386

From the table II, the proposed approach reached very good quality of results compared with other approaches from literature. ABC_MMKP algorithm outperforms 9 out of the 13 instances (from instance 5 to instance 13) and gives 98.88% solution quality compared with the optimal solution provided by Cplex. Also, the comparison with the state-of-art approaches proves that ABC_MMKP algorithm provides better quality of results for medium and large scale instances. The computational results show that ABC_MMKP algorithm, if compared with approaches from the literature, improves the objective function values, on average of 1.06% from MRLS, 0.72% from Ant, 0.22% from Oss and 0.44% from SLS_ MMKP. Therefore, the computational time of the proposed algorithm is also highly interesting. Note that for the 13th instance that contains 4000 variables and 10 constraints (large scale problem) takes less than 20 seconds to be solved. More computational time detail are giving in table III.

TABLE III. COMPUTATIONAL TIME COMPARISION BETWEEN ABC_MMKP AND SLS_MMKP

INST	SLS_ MMKP	CPU	ABC_ MMKP	CPU
I07	24270.06	16.48	**24274**	1.023
I08	36397.86	13.28	**36492**	2.666
I09	48512.09	39.17	**48657**	4.524
I10	60609.18	23.71	**60944**	7.709
I11	72718.31	35.42	**73163**	10.986
I12	84839.34	55.99	**85420**	14.808
I13	97000.42	72.79	**97723**	19.386

Table III gives the computational time comparisons between SLS_MMKP and our heuristic ABC_MMKP. In the three other algorithms MRLS, Ant and Osc, the computational time were not reported. For SLS_MMKP, it is mentioned their algorithm was running on an Intel(R) Core(TM) i3-3220 3.30 GHz CPU 4GB RAM machine with 32bit Linux. Also, for each problem instance, they performed 100 independent trials with different random seeds system, the average solution value is presented Table III.

It is noteworthy that by varying the maximum number of iterations allowed by the algorithm, high quality solution could be obtained. The results provided therefore represent a compromise between the quality of the obtained solutions and the computational time required for execution. This experiment results confirms that the proposed algorithm ABC_MMKP performs well on large scale MMKP problem.

VI. CONCLUSION

In this paper, we have developed a new algorithm called ABC_MMKP based on the ABC algorithm to solve MMKP. This algorithm is founded on four main steps. The initialization, the diversification (scout bees), the intensification (employed bees) and the fitness process (onlooker bee). The surrogate constraint is also used to refine the intensification process and improve the solutions quality. The algorithm is tested on 13 benchmarks instances. Results show that ABC_MMKP algorithm performs better than the mentioned algorithms for large scale problems and can be efficiently employed to solve the MMKP for large scale instances. Our algorithm could be enhanced by a combination with other techniques and heuristics in order to increase the efficiency and effectiveness of the search and obtain faster solution with higher solution quality.

REFERENCES

[1] D. Pisinger, "A minimal algorithm for the multiple-choice knapsack problem", European Journal of Operational Research 83 (1995) 394–410.

[2] J. Puchinger, G.R. Raidl, U. Pferschy, "The Multidimensional Knapsack Problem: Structure and Algorithms," INFORMS Journal on Computing, Aug 2009.

[3] T. Ghasemi and M. Razzazi, "Development of core to solve the multidimen- sional multiple-choice Knapsack problem", Computers & Industrial Engineering, 60 (2), 349–360, 2011

[4] A. Sbihi, "A best first search exact algorithm for the multiple-choice multidimensional knapsack problem", Journal of Combinatorial Optimization, 13(4), 337–351, 2007

[5] S. Khan, "Quality adaptation in a multisession multimedia system : model, algorithms andarchitecture". PhDthesis, University of Victoria, 1998.

[6] Y. Chen, J.-K. Hao," A ''reduce and solve'' approach for the multiple-choice multidimensional knapsack problem", European Journal of Operational Research, 239, 313–322, 2014

[7] M. Moser, D. Jokanovic, and N. Shiratori, " An algorithm for the multidimensional multiple-choice knapsack problem", IEICE Transactions in Fundamentals, E80-A(3),1997.

[8] Razzazi, M. and Ghasemi, T., 2008. An exact algorithm for the multiple-choice multidimensional knapsack based on the core. Advances in Computer Science and Engineering, Communications in Computer and Information Science, 6 (1), 275–282.

[9] S. Khan, K. F. Li , E. G. Manning and M. M. Akbar, "Solving the Knapsack problem for adaptive multimedia systems", Studia Informatica An International Journal Special Issue on Cutting Packing and Knapsacking Problems, 2 (4), 157–178, 2002.

[10] M. Hifi, M. Michrafy and A. Sbihi "A reactive local search-based algorithm for the multiple-choice multi-dimensional knapsack problem", Computational Optimization and Applications, 33, 271–285, 2006.

[11] N. Cherfi and M. Hifi, "Hybrid algorithms for the multiple-choice multidimensional knapsack problem", International Journal of Operational Research, 5(1), 89–109,2009.

[12] Htiouech, S.. S. Bouamama, and R. Attia. 2013. "OSC: Solving the Multidimensional Multi-Choice Knapsack Problem with Tight Strategic Oscillation Using Surrogate Constraints." International Journal of Computer Applications 73 (13): 1–22.

[13] Y. Xia, c. Gao and J. Li, "A stochastic local search heuristic for the multidimen- sional multiple-choice Knapsack problem", in Bio-inspired computing-theories and applications (pp. 513–522). Springer, 2015.

[14] C. S. Hiremath and R. R. Hill, "First-level tabu search approach for solving the multiple-choice multidimensional knapsack problem", international Journal of Metaheuristics, 2(2), 174–199,2013.

[15] I. Crévits, S. Hanafi, R. Mansi and C. Wilbaut, "Iterative semi-continuous re- laxation heuristics for the multiple-choice multidimensional Knapsack problem", Computers and Operations Research, 39 (1), 32–41, 2012.

[16] R. Mansi, C. Alves, J. Valério de Carvalho and S. Hanafi, "A hybrid heuristic for the multiple choice multidimensional Knapsack problem", Engineering Optimiza- tion, 45 (8), 983–1004, 2013.

[17] C. Gao, G. Lu , X. Yao and J. Li, " An iterative pseudo-gap enumeration approach for the multidimensional multiple-choice knapsack problem", Eur J Oper Res 2017;260(1):1–11

[18] M. Caserta, S. Vob, "The robust multiple-choice multidimensional knapsack problem", Omega 0 0 0 (2018) 1–12, Elsevier

[19] P.C. Chu, J.E. Beasley, "A genetic algorithm for the multidimensional knapsack problem", Journal of Heuristics 4 (1998) 63–86.

[20] F. Glover, "A multiphase-dual algorithm for the zero-one integer programming problem", Operations Research, 13 :879-919, 1965.

[21] F. Glover, "Surrogate constraints", Operations Research, 16 :741-749, 1968.

[22] F. Glover, "Surrogate constraints : tutorial notes". October (1998).

[23] D.E. Goldberg, "Genetic Algorithms in Search, in: Optimization and Machine Learning" , Addison-Wesley Pub. Co., 1989, , ISBN: 020115767

[24] D. Karaboga, "an idea based on honey bee swarm for numerical optimization", technical report-TR06,Erciyes University, Engineering Faculty, Computer Engineering Department 2005

[25] D. Karaboga, B. Basturk, On The Performance Of Artificial Bee Colony (ABC) Algorithm, Applied Soft Computing,Volume 8, Issue 1, January 2008, Pages 687-697.

[26] M. Moser, D. P. Jokanovic et N. Shiratori. An algorithm for the multidimensional multiple-choice knapsack problem. IEICE Transactions A, E80 :582-589, 1997.

[27] M. Hifi, M. Michrafy, and A. Sbihi. Heuristic algorithms for the multiplechoice multidimensional knapsack problem. Journal of the Operational Research Society, 55(12) :1323-1332, 2004.

[28] S. Iqbal, M. Bari, and M. Rahman. Solving the multi-dimensional multichoice knapsack problem with the help of ants. In Proceedings of the 7th international conference on Swarm intelligence (ANTS 2010), 2010.

[29] Glover.F, GA.Kochenberger, Critical event tabu search for multidimensional knapsack problems. In: Osman IH, Kelly JP, editors. Metaheuristics: theory and applications. p. 407-27. (1996)

[30] MMKP benchmarks, " MMKP benchmarks ". http://www.es.ele.tue.nl/pareto/mmkp/, 2012.

[31] Sbihi A., "Hybrid methods in combinatorial optimization: Exact algorithms and heuristics". Ph.D. thesis, Univ. of Paris I, France, 2003.

[32] Kellerer, H., Pferschy, U., & Pisinger, D. (2004). Multidimensional Knapsack Problems. Knapsack Problems, 235–283. doi:10.1007/978-3-540-24777-7_9

[33] Kellerer, H., Pferschy, U., & Pisinger, D. (2004). The Multiple-Choice Knapsack Problem. Knapsack Problems, 317–347. doi:10.1007/978-3-540-24777-7_11

[34] Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. IEEE Computational Intelligence Magazine, 1(4), 28–39. doi:10.1109/mci.2006.329691

[35] Chazelle, B. (2009). Natural Algorithms. Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, 422–431. doi:10.1137/1.9781611973068.47

[36] Karaboga, D., & Akay, B. (2009). A survey: algorithms simulating bee swarm intelligence. Artificial Intelligence Review, 31(1-4), 61–85. doi:10.1007/s10462-009-9127-4