OSC: solving the multidimensional multi-choice knapsack problem with tight strategic Oscillation using Surrogate Constraints

Skander HTIOUECH
Sys-com, National Engineering School of Tunis
UR-CSE. Tunisia Polytechnic School
University of Carthage-Tunisia B.P. 743. 2078 La Marsa. Tunisia
Skander.Htiouech@enib.rnu.tn

Sadok BOUAMAMA
Hana Lab, National School of Computer Science of Tunisia
Sadok.Bouamama@ensi.rnu.tn

Rabah ATTIA
UR-CSE. Tunisia Polytechnic School.
University of Carthage-Tunisia B.P. 743. 2078 La Marsa. Tunisia
Rabah.Attia@enit.rnu.tn

ABSTRACT

The multidimensional multi-choice knapsack problem (MMKP) is one of the most complex members of the Knapsack Problem (KP) family. It has been used to model large problems such as telecommunications, quality of service (QoS), management problem in computer networks and admission control problem in the adaptive multimedia systems. In this paper, we propose a new approach based on strategic oscillation using surrogate constraint information. We introduce new rules to control oscillation process to solve the MMKP. The main idea is to explore both sides of the feasibility border that consists in alternating both constructive and destructive phases in a strategic oscillating manner. In order to strengthen the surrogate constraint information, we enhance the method with constraints normalization. This may improve the computational results. Numerical results show that the performance of this approach is competitive with previously published results. Performance analysis of the method shows the merits of its using in this problem class.

General Terms:

Metaheuristic, knapsack problem

Keywords:

Combinatorial optimization, multiple choice, knapsack problem, tabu search, surrogate constraints

1. INTRODUCTION

The MMKP is a NP-hard problem [7] as it generalizes the standard knapsack problem (KP), a combinatorial optimization problem, one of the most complex members of the knapsack problem family. Unlike the majority variants of the knapsack problem, the MMKP is very difficult to solve in practice. This is partly due to its choice constraints. Furthermore, even finding a feasible solution for the problem is NP-hard. Consider a set of item groups. Each item has a particular value in the objective function and requires certain amount of resources. The goal of the MMKP is to pick exactly one item from each group such that the resource constraints are not violated and the revenue is maximized, as well. Mathematical formulation of MMKP can be written as follows :

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

$$\textit{Subject to} \quad \sum_{i=o1}^{n}\sum_{j=1}^{n_i} a_{ij}^{k}x_{ij} \leq b^k \ , k=1,\ldots,m \tag{2}$$

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

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

where b=$(b^1, b^2, \ldots, b^m)$is the capacity vector of the multi-constrained knapsack resources, and a set of n disjoint item groups N=$\{N_1, \ldots, N_i, \ldots, N_n\}$ where each group i, i=$1, \ldots, n$ has n_i items. Each item j, j=$1, \ldots n_i$, of the ith group has a non-negative profit value c_{ij}, and requires an amount of resources represented by the weight vector a_{ij}=$(a_{ij}^1, a_{ij}^2, \ldots, a_{ij}^k)$. Note that weight terms a_{ij}^k (with $1 \leq k \leq m, 1 \leq i \leq n, 1 \leq j \leq n_i$) are nonnegative. It is worthy to note that x_{ij} takes either 1 or 0, which means that item j of the ith group is picked or not, respectively. To eliminate trivial solutions, we assume that for all $1 \leq j \leq n_i$ we have :

$$\sum_{i=1}^{n} min\{a_{ij}^k\} \leq b^k \leq \sum_{i=1}^{n} min\{a_{ij}^k\} \quad k=1,\ldots,m \tag{5}$$

Recently, Romain *et al.* [42] interested by an original real-world problem coming from tourism field and describe a modeling of the problem like an MMKP. They give a first approach that mixes knowledge management and operational research. In [31], the

MMKP has been used to model the quality of service (QoS) management problem in computer networks and the admission control problem in the adaptive multimedia systems [7][26][32]. Various other resource allocation problems can also be mapped directly to MMKP [28][33]. Multimedia server is also equivalent to the knapsack with limited resources, e.g. CPU cycles, I/O bandwidth and memory. Applications of the UM are presented in [6][8]. The precision of results provided by previously published heuristics designed for MMKP depends on the quality of the start solution usually provided by another algorithm. This initial condition is harder to find for MMKP than for standard KP. The goal of this paper is to design a new oscillation heuristic to solve the MMKP which convergence is independent of the initial solution. This new heuristic involves tabu search techniques and starts by either feasible or not feasible solution without affecting the result quality at the cost of negligible computing time fluctuation. This property is induced by the oscillation nature of the proposed heuristic between both sides of feasibility border. This new heuristic shows competitive results compared to most recent published results and namely much lower computing time.

This paper is organized as follows; context for our study are given in section 2; Relevant related works on KP and MMKP are described in section 3. Section 4 presents our contribution details. First, choice rules based on surrogate constraint information are defined. Then, a constraint normalization scheme is utilized to strengthen the surrogate constraint used to implement the oscillation based heuristic. The computational results are reported in Section 5. Section 6 summarizes the contributions of this work and discusses directions for further works.

2. CONTEXT AND MOTIVATIONS

Many different classes of knapsack problems are found in the literature [33][11], including multi-objective [11], multidimensional (MDKP)[2][3], multiple-choice (MCKP) [12][13], and bounded problems [14], and others. The classical knapsack problem (KP) aims to choose a subset from an infinite set of items. The latter should satisfy the capacity constraint. As cited in [11], KP has many practical applications such as cargo loading, cutting stock, capital budgeting and project selection applications. The 0-1 knapsack problem (KP 0-1), using binary variables, is a special MDKP case. In this KP class, the capacity vector size m is equal to 1. KP 0-1 could be solved by pseudo-polynomial time function [2][3]. The MDKP extends the classical knapsack problem to m constraints. For example, if m=2, the MDKP becomes a bi-dimensional problem. The MDKP makes up the framework to evaluate new metaheuristics. The MCKP is an other extension of the KP in which the items are divided into several classes. In each class, only one item has to be selected [33]. The MMKP can be defined from the MCKP in the same way as the MDKP is derived from the KP. So, MMKP could be seen as the combination of MDKP and MCKP [7]. The MMKP is a variant of the MDKP where items are divided into classes, and exactly one item per class has to be selected. Because of its choice constraints, MMKP is more difficult to solve. For these reasons, complete methods are thwarted by combinatorial explosion. However, many metaheuristics could be used for its resolution. In this paper, we present a new metaheuristic for solving MMKP. This approach introduces a new level where balancing between intensification and diversification can be realized.

3. LITERATURE REVIEW

The classical KP is known as a special case of the MMKP [10]. In addition it is well known that MMKP is NP-hard [11]. The MMKP is also closely related to some other well known non-standard knapsack problems such as the multiple-choice knapsack problem [12][13] and the multidimensional knapsack problem [2][3][23][24]. Actually, the MMKP is the combination of the MDKP and the MCKP [7][9]. Although very little literature is directly related to the MMKP. In fact, most works deal with the multiple-choice knapsack problem (MCKP), a special case of the MMKP with only one resource constraint $(m = 1)$ and a set of mutually disjoint multiple-choice constraints. Multiple-choice programming can also be traced back as done by Healy Jr [12]. Pertinent details of multiple-choice programming fundamentals are given in [13][14]. A simple dominance concept has been widely used in the MCKP literature to eliminate dominated variables prior to the enumeration stage from each multiple-choice group [14]. Dyer and Walker [15] extended the dominance issue to the MMKP and showed that the expected proportion of nondominated variables in the MMKP is a function of the number of resource constraints and the cardinality of the multiple-choice sets. Another variant of the MMKP is the multidimensional knapsack problem (MDKP) $(n = 1$ and eliminating the constraint $(3))$, which has a sizable literature [7]. The MDKP has a large domain of applications such as project selection [16], cargo loading [17], capital budgeting [18] and cutting stock [19]. A variety of approaches have been applied to the MDKP such as exact algorithms, greedy heuristic methods, bound based heuristics and approximate dynamic programming. The constraint aggregation techniques, such as Lagrangian relaxation [20] and surrogate constraint information [21] and [22], have been extensively applied for solving the MDKP. More detailed information for the MDKP literature can be found in [14][23][24] and [37]. Depending on the nature of the solution, the algorithms for MMKP can be divided into two families: complete methods and incomplete methods. First ones striving for exact solutions are, also, known as exact algorithms. Incomplete methods scarify completeness for efficiency and generate near-optimal solutions. Finding exact solutions is NP-hard [7][40]. Khan [7], Khan *et al.* [26] and Hifi *et al.* [25] designed exact algorithms for solving the MMKP. The latter uses Branch and Bound method with Linear Programming (BBLP) techniques. Sbihi [40], described a branch-and-bound algorithm that starts by computing a feasible solution using the heuristic proposed in [27]. In on hand, the branching scheme of this algorithm consists in fixing one item in the solution. In the other hand, it explores the search tree using a best-first strategy. Razzazi *et al.* [47] proposed a different branch-and-bound method in which the nodes are explored using a depth-first strategy, and upper bounds are obtained by solving a surrogate relaxation of the problem. In [46], Ghasemi *et al.* describes an exact algorithm for the MMKP based on an approximate core. The authors report on promising results for large uncorrelated instances and for correlated instances with up to 5 constraints.

Since MMKP is NP-hard in the strong sense, it is often possible that an optimal solution may not be found within a reasonable computational time. In the case of MMKP, not only exact methods are thwarted by exponential combinatorial time but also incomplete method results stand very close to exact ones [29][39][34][35][41]. For these reasons, incomplete approaches are preferred. Crevits *et al.* [44] uses the similar approach proposed in [46] to explore a heuristic based on a new relaxation. This approach stands on removing the integrality constraints and forcing the variables to be close to 0 or 1. This relaxation is more general than the linear programming (LP) and mixed integer programming relaxations used in [45]. Recently, Cherfi *et al.* [48] describes three new approaches for the MMKP. The patters are, respectively, based on a local branching algorithm, on a hybrid algorithm combining local branching with column generation, and on a truncated branch-and-bound algorithm that embeds the previous hybrid method. Iqbal *et al.* [35] developed also an ant colony optimization approach for the MMKP. The authors improved the convergence of their approach by using a local search routine in their algorithm. More recently, Mansi *et al.* [49] describe a new linear programming relaxations based solution which reduces the problem by fixing some its variables. These solutions are used to update the global lower and upper bounds.

4. OUR APPROACH : OSC

4.1 Foundation and basis

Glover and Kochenberger [2] introduced a critical-event tabu search approach which assumes that the memory structure is arranged around the feasibility border of the MDKP. This heuristic (referred to GK) uses a strategic oscillation that navigates both sides of the border to achieve a balance between intensification and diversification procedures. A parameter *span* is used to indicate the depth of the oscillation about the boundary, measured in terms of the number of variables added after crossing the boundary from the feasible side in a constructive phase and the number of variables dropped after crossing the boundary from the infeasible side in a destructive phase. Starting by a minimum value, the span is gradually increased to a maximum value. A series of constructive and destructive phases is performed for each value of the span parameter. When the span reaches the maximum value, it is gradually decreased to the minimum value. Once the span decreases to the minimum value, it is again gradually increased to the maximum value, and this oscillation process continues [2]. Hanafi and Freville [3] also demonstrated special version of this method that balances the interaction between intensification and diversification strategies for the MDKP. Tabu search fundamentals and strategies are widely discussed in Glover and Laguna [1].

4.2 General oscillation strategy for MMKP

In tabu search methods [2], intensification forces the search to examine attractive regions while diversification drives the search into new unexplored regions. The main goal is then to find an efficient balance between intensification and diversification strategies. Proposed by Glover [21], surrogate relaxation consists in replacing a problem constraints set by only one, called surrogate constraint [4][22]. Crossing and intensive exploration of the promising region is controlled by using information deduced from the surrogate constraints and the memory of the search. We introduce an adaptive and flexible memory subdivided into short and long term by incorporating recency-based and frequency-based memory. Algorithm 1 makes up the framework of our strategic oscillation method.

4.3 Surrogate constraint based choice rules

In order to determine which variables to add, drop or swap, our choice is based on using surrogate constraint information. Since MCKP is much easier to solve than the MMKP [7], we replace all knapsack constraints of equation (2) by surrogate constraints. So equation (2) is replaced by equation (6) as follows :

$$\sum_{k=1}^{m} \mu_k \sum_{i=1}^{n} \sum_{j=1}^{n_i} a_{ij}^k x_{ij} \leq \sum_{k=1}^{m} b^k \tag{6}$$

Algorithm 1 Oscillation for the MMKP

1. Initialization
Initialize tabu search and oscillation parameters
Generate an initial solution x to be feasible or infeasible
2. General Step
repeat
  / * Intensification* /
  if the current solution is feasible **then**
    repeat
      Intensification phase around the current solution to improve it. (swap move)
    until no improvement after a $number_{max}$ of movements
  end if
  / * Diversification * /
  repeat
    switch direction of research
    repeat
      Constructive phase : add Move from the current zone to the infeasible region
      $span = span + 1$
    until $span = span_{max}$
    repeat
      Destructive phase : drop move from the current solution to feasible region
      $span = span - 1$
    until $span = span_{min}$
  until $Stop$
  Updating oscillation parameters and tabu search
  if the current solution is feasible **then**
    go to 2.
  end if
until the stopping criterion is satisfied
return best solution found

Where μ is an m-dimensional non-negative multiplier vector. Equation (6) can, then, be reformulated by equation (7). And so our reasoning on the variable s_{ij} become more easier.

$$\sum_{i=1}^{n} \sum_{j=1}^{n_i} s_{ij} x_{ij} \leq s_0 \tag{7}$$

where $s_{ij} = \sum_{k=1}^{m} \mu_k a_{ij}^k$ and $s_0 = \sum_{k=1}^{m} \mu_k b^k$. To generate a surrogate constraint, we compute first a set of multipliers values λ_k as in [2]. λ_k values are determined not only considering the surrogate constraint multipliers but also depending on the feasibility of the current solution. In addition to that, we consider the state of the search process. In all cases, we compute equation (8) for each constraint $k = 1, \ldots, m$.

$$\Delta^k = b^k - \sum_{i=1}^{n} \sum_{\{j/x_{ij}=1\}}^{n_i} a_{ij}^k \tag{8}$$

Thus Δ^k (initially equal to b^k) indicates the remaining right-hand-side (RHS) value after the current assignment for the k^{th}constraint. Note that this value is negative if the constraint k is violated. Each knapsack constraint is multiplied by its corresponding multiplier λ_k and the weighted sum provides the surrogate constraint. In the first case, when the solution is feasible, the λ_k for each constraint is always set up to $(\Delta^k)^{-1}$. In the second case, when the solution is infeasible and the search is in the constructive phase, then for all $k = 1, \ldots, m$

$\lambda_k = \begin{cases} (\Delta^k)^{-1} & if\ \Delta^k > 0 \\ 2+ \mid \Delta^k \mid & if\ \Delta^k \leq 0 \end{cases}$

In the last case, when the solution is infeasible and the search is in the destructive phase

$\lambda_k = \begin{cases} 0 & if\ \Delta^k \geq 0 \\ (\mid \Delta^k \mid + \sum_{i=1}^{n} \sum_{\{j/x_{ij}=0\}}^{n_i} a_{ij}^k)^{-1} & if\ \Delta^k < 0 \end{cases}$

for all $k = 1, \ldots, m$. When $\Delta^k < 0$, λ_k can be expressed as : $(\sum_{i=1}^{n} \sum_{j=1}^{n_i} a_{ij}^k - b^k)^{-1}$.

The setting rules above focus on the influence of the most tight or violated constraints and, therefore, encourage the search around critical solutions. Recall that critical solutions are those obtained immediately before and after crossing the feasibility boundary. In the same way, critical events correspond to solutions generated by the constructive phase at the final moment prior to becoming infeasible and by the destructive phase at the first moment after regaining feasibility.

Since $\mu^k \geq 0$ and $a_{ij}^k \geq 0$, it follows that $s_{ij} \geq 0$. The choice rule for the constructive phase selects the variable x_{ij} to switch from 0 to 1 in order to

$$\textit{Maximise} \left\{ r_{ij} = \frac{c_{ij}}{s_{ij}} | x_{ij} = 0 \right\} \quad (9)$$

The choice rule for the destructive phase of our approach selects the variable x_{ij} to switch from 1 to 0 in order to

$$\textit{Minimise} \left\{ r_{ij} = \frac{c_{ij}}{s_{ij}} | x_{ij} = 1 \right\} \quad (10)$$

When the solution is feasible, swap moves are chosen to improve the quality of the current solution. We do this by selecting the variable x_{ij} to switch from 0 to 1. Otherwise we select the variable x_{ih} to switch from 1 to 0. The feature considered in our choice of x_{ij} and x_{ih} have to

$$\textit{Maximise} \left\{ r_{ijh} = \frac{c_{ij}/s_{ij}}{c_{ih}/s_{ih}} | x_{ij} = 0, x_{ih} = 1 \right\} \quad (11)$$

Where items j and h are in the same group i.

4.4 Normalization for surrogate constraint

To strengthen the surrogate constraint we implement the normalization scheme developed by Glover [4]. The normalization of each constraint depends on the state of the search and also the current solution (More detailed information about the normalization for surrogate constraint can be found in [2], [21] and [22].

—Case 1: in the constructive phase, firstly, we divide every k constraint in (2) by the updated (RHS) value to create a first-level normalization. Secondly, we multiply by the sum of the normalized left-hand-side (LHS) coefficients. This is done for every current unassigned variable, in order to create the second level normalization. Last of all, we sum up the second level normalization constraints to form a surrogate constraint. After consecutive levels of normalization, s_{ij} can be expressed as :
$s_{ij} = \sum_{k=1}^{m} \frac{A_k.e}{(\Delta^k)^2} a_{ij}^k$
and s_0 can be expressed as :
$s_0 = \sum_{k=1}^{m} \frac{A_k.e}{\Delta^k}$, where e is a unit vector and
$A_k = (\sum_{i=1}^{n} \sum_{j=1}^{n_i} a_{ij}^k | x_{ij} = 0)$.

—Case 2: in the destructive phase, firstly, we divide every k constraint in (2) by the updated (RHS) value to create a first-level normalization. Secondly, we divide by the sum of the normalized left-hand-side (LHS) coefficients. This is done for every current unassigned variable, in order to create the second level normalization. Last of all, we add a second level normalization constraints to form a surrogate constraint. After consecutive levels of normalization, s_{ij} and s_0 can be expressed as :
$s_{ij} = \sum_{k=1}^{m} \frac{a_{ij}^k}{A_k.e}$ and $s_0 = \sum_{k=1}^{m} \frac{\Delta^k}{A_k.e}$,
where $A_k = (\sum_{i=1}^{n} \sum_{j=1}^{n_i} a_{ij}^k | x_{ij} = 1)$.

4.5 Add, drop and swap moves

4.5.1 Add move. The add move is the principal move of the constructive phase of our approach. Adding a variable to the current solution S (initially equal to 0) is equivalent to set it up to 1. In this current step, the object added maximizes the quantity r_{ij} as described in algorithm 2.

Algorithm 2 procedure addMove(S)

$G^* = Argmin \left\{ \sum_{j=1}^{n_i} x_{ij} | i \in G \right\}$
$r_{i^*j^*} = max \left\{ r_{ij} | x_{ij} = 0, i \in G^*, j = 1, \ldots, n_i \right\}$
$x_{i^*j^*} = 1$
$S^* = S^* + x_{i^*j^*}$

4.5.2 Drop move. The heuristic gradually chooses which variables to drop during the destructive phase. Drop an object from the current solution S (initially equal to 1) is equivalent to reset it to 0. In this step, the dropped object minimizes the quantity r_{ij} as described in algorithm 3.

Algorithm 3 procedure DropMove(S)

$G^* = Argmax \left\{ \sum_{j=1}^{n_i} x_{ij} | i \in G \right\}$
$r_{i^*j^*} = min \left\{ r_{ij} | x_{ij} = 0, i \in G^*, j = 1, \ldots, n_i \right\}$
$x_{i^*j^*} = 0$
$S^* = S^* - x_{i^*j^*}$

4.5.3 Swap move. When the solution is feasible, our approach improves, step by step, its quality. The improvement should respect the feasibility of the solution and it is done by swap moves. Algorithm 4 shows how the objects to be swapped from the same group maximizes the quantity r_{ijh}.

Algorithm 4 procedure SwapMove(S)

$r_{i^*j^*h^*} = max \left\{ r_{ijh} | i \in G, j = 1, \ldots, n_i, h = 1, \ldots, n_i, \right\}$
such as $x_{ij} = 0, x_{ih} = 1$ and
$\sum_{i=1}^{n} \sum_{j=1}^{n_i} a_{ij}^k x_{ij} - a_{i^*h^*}^k + a_{i^*j^*}^k \leq b^k, k = 1, \ldots, m,$
and $c_{i^*j^*} < c_{i^*j^*}$
$x_{i^*j^*} = 1$
$x_{i^*h^*} = 0$
$S^* = S^* + x_{i^*j^*} - x_{i^*h^*}$

4.6 Span control

The depth of oscillations on both sides of the boundary can be managed in several ways [30]. In Glover and Kochenberger [2], the depth of the oscillations is predetermined. It consequently confers periodic and symmetrical movements on both sides of the boundary of the feasible space. The oscillation about the feasibility boundary is controlled by the span parameter δ counting the add/drop moves and varying between 1 and an upper bound. The periodicity of the oscillations is controlled by the parameters *p1* and *p2*. δ depends primarily on the number of iterations and changes as follows :
Increasing span : if the span value is in [1, *p1*] then allow $p2\times\delta$ iterations *i.e* we increase δ by 1.
If the span value is in [*p1+1*, *p2*] allow $p2$ iterations *i.e* we increase the span by 1. When δ exceeds $p2$, we set it up back to $p2$ and start to decreasing the span parameter.
Decreasing span : if the span value is in [*p2*, *p1+1*] allow $p2$ iterations *i.e* we decrease the span by 1.
If the span value is in [*p1*, *1*] allow $p2\times\delta$ iterations *i.e* we decrease the δ by 1.
When the δ reaches 0, we set up it back to 1 and start increasing the span parameter.

4.7 Memory management

In order to avoid local optima looping and neighborhoods, we determine the tabu state of a potential move by adding the recency-based and frequency-based memory information. To slightly scramble the search using the recency information, we record last t solutions in a tabu circular list. The latter is updated for each oscillation iteration. Hence the recency tabu vector RT is computed by summing up the last t solutions. It is tempting to note that if $RT_{ij} > 0$ then the variable x_{ij} is currently on the tabu list. Therefore, this variable should no longer change. Besides, the long term frequency information is captured by summing up all the collected solutions. The resulting vector is called FT.
The flexibility is provided by introducing a penalty term when evaluating the ratio r_{ij} for each variable. Let r_{max} be the maximum ratios value of all eligible variables and $PenR$ (equal to $r_{max}\times RT_{ij}$) the penalty coefficient associated with the recency information subtracted from the original r_{ij}. In addition, we associate another penalty term with the frequency information. This penalty term is subtracted from the original r_{ij} to penalize the frequently appeared variables in current solutions. We calculate this value denoted $PenF$ as indicated in equation (12):

$$\frac{r_{max}}{\textit{index of the current iteration}\times C} \tag{12}$$

Where C is a frequency penalty scalar.
Immediately after a "turn around", we apply these penalty terms in the first k adds or drops. This manipulation aims to more foster the diversity in the search process. The parameter k starts from 1, after $2t$ iterations (here t corresponds to the tabu list size). k is incremented to $k+1$. We continue the same process until k reaches k_{max}, the maximum value of k. Then k is decreased by 1 every $2t$ iterations until it becomes equal to 1 again and so on. Here $countvar$ is a number of adds or drops after a last "turn around". Let i^*j^* the index of the next object to choose. In the constructive (respectively destructive) phase ($x_{ij} = 0$) (resp. $x_{ij} = 1$), we choose the object i^*j^* with max (respectively min) $Evaluation(i,j)$ calculated as follow :
If $countvar > k$, then $Evaluation(i,j) = r_{ij}$
If $countvar < k$, then
$Evaluation(i,j) = r_{ij} - PenR\times RT_{ij} - PenF\times FT_{ij}$

Table 1. Details of Khan's instances

$Inst$	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

5. EXPERIMENTATIONS AND TESTS

5.1 Experimental design

We compare the performance of our new appraoch (Osc) to other heuristic algorithms from the literature. The benchmarks data set on the MMKP are given in [30]. For each instance, we present in Table 1 the number of groups n, the number of items n_i in each group i, the number of constraints m and the total number N of variables $N = \sum_{i=1}^{n} n_i$. We set up our approach parameters as follow:

—Span control : p1 = 3, p2 = 7
—Tabu list size t = 4
—kmax = 5
—frequency penalty scalar: 1000
—Initial solution: every variable is 0 ($x_{ij} = 0$ for all $i = 1, ..., n$ and $j = 1, ..., n_i$)
—*NumIter*= $n \times n_i$: number of iterations. Iteration corresponds to a pass of both a constructive phase and a destructive phase.
—*IterAutorized*=50 : number of iterations authorized without amelioration

5.2 Experimental results

Table 2 shows the performance results of different approaches including ours. For each instance, it reports the obtained results by CPLEX, Khan [7], CPC [27], FanTabu [34], HIFI [29] and Ant [35].
We can clearly see that our approach gives the optimal solution for instances I01, I02, I05 and I06. In one hand, we come to this success thanks to the strategic oscillation which betters not only the diversification but also the intensification of the search process. Especially it gives its proofs at the feasibility boundary regions. In the other hand, this good result could be explained by using surrogate constraints informations. This use guides the search process to worthy neighborhoods. Note that the normalization of surrogate constraint informations strengthens the optimization process.
For the same reasons cited before, our approach improves the best literature results, although that it cannot attain CPLEX ones. In fact, for the instances I08, I09, I10, I11, I12 and I13 our approach outperforms all the other well known approaches *i.e* Heu, Cpc, $F.Tabu$, $Hifi$ and Ant.
Unfortunately, for instances I03, I04 and I07, OSC results are very close to the best result but the approach is unable to attain them. It

Table 2. Solution Quality Comparison

$Inst$	$Cplex$	Heu	Cpc	$F.Tabu$	$Hifi$	Ant	OSC
1	173	154	159	169	173	173	**173**
I02	364	354	312	354	364	364	**364**
I03	1602	1518	1407	1557	1602	1598	1594
I04	3597	3297	3322	3473	3569	3562	3514
I05	3905.7	3894.5	3889.9	3905.7	3905.7	3905.7	**3905.7**
I06	4799.3	4788.2	4723.1	4799.3	4799.3	4799.3	**4799.3**
I07	24587	-	23237	23691	24159	24170	24162
I08	36877	-	35403	35684	36401	36211	**36405**
I09	49167	-	47154	47202	48367	48204	**48567**
I10	61437	-	58990	58964	60475	60258	**60858**
I11	73773	-	70685	70555	72558	72240	**73022**
I12	86071	-	82754	81833	84707	84282	**85284**
I13	98429	-	94465	94168	96834	96343	**97545**

is tempting to note that the given result for these instances is obtained by using the same parameters values indicated in our experimental design. Moreover, if we change these parameters values, our approach will easily raise to the best results. Indeed, in these instances the oscillations should be minimized due to the relatively easiness of these instances.

OSC provides solutions with total value on average equal to 1.03% close to the optimum and better performance than those obtained by Khan[7], Hifi *et al.*[29] and Iqbal *et al.*[35]. The surrogate constraint approach achieves better solutions over 10 out of the 13 instances. The exceptions being for instances I03, I04 and I07 that achieve an objective function value $1,73\%$ close to optimum are due to the parameters choice. The insight here is that where exploration area is comparatively small, OSC explores better areas and can give better solutions. But for large instances (I08, I09, I10, I11, I12 and I13) the exploration area is much larger. It should be noted that if we increase the number of iterations allowed to $n^2 \times n_i$, OSC algorithm finds the optimal solution for I03 and I04. In the first six instances, our method is $0,63\%$ close to optima values given in [7]. However, the quality of the seven instances are improved (6 instances of 7) in average term by $0,96\%$ compared with the results published in [35] which are the best known solutions found after 100 runs. For instance I13 (which is a very large instance of MMKP; 4000 variables), note that our approach takes more lower computational time to reach its result which is $0,89\%$ of the optimal solution . So our approach is very significant for large-scale real-time problems. Remember that OSC starts execution with the trivial initial solution (every variable $x_{ij} = 0$), which can be not feasible, which increases the effort to find a good first feasible solutions. This is why we assume that a feasible solution starting provided by a fast algorithm like the one used in [43] can significantly improve the quality of solution or the run time.

6. CONCLUSION

In this paper, we have introduced a new oscillation approach which explores both sides of the feasibility border to solve MMKP. Surrogate constraint information is used to build the choice rules. A constraint normalization scheme was implemented to strengthen the surrogate constraint. OSC would be a very good candidate for time-critical applications such as adaptive multimedia systems where a near-optimal solution is acceptable, and fast computation is more important than guaranteeing the truly optimal value. To further improve our approach, a more sophisticated use of intelligent algorithms like in [50][51] can be investigated.

7. REFERENCES

[1] Glover.F, Laguna M. Tabu search. Boston: Kluwer Academic Publishers; (1998).

[2] 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)

[3] Hanafi.S, Freville.A, An efficient tabu search approach for the 0-1 multidimensional knapsack problem. European Journal of Operational Research ;106(23):659 75. (1998).

[4] Glover F. Surrogate constraints: tutorial notes. October (1998).

[5] Moser M, Declarative scheduling for optimally graceful QoS degradation, in Proc. IEEE Int. Conf. Multimedia Computing Systems (ICMCS), pp. 86 93(1996).

[6] L. Chen, The utility model applied to layer-coded sources, Dept. of Computer Science, University of Victoria, Victoria, BC, Canada, (1998).

[7] Khan.S, Quality adaptation in a multisession multimedia system : Model, algorithms and architecture, Ph.D. dissertation, Department of Electrical and Computer Engineering, University of Victoria, Canada (1998).

[8] Watson.R, The optimal admission and adaptation of service level agreements in packet networks: Applying the utility model, Masters thesis, Dept. of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada, (2001).

[9] Khan.S, K. F. Li and E.G. Manning, The Utility Model for Adaptive Multimedia System. In International Workshop on Multimedia Modeling, pages 111-126 (1997).

[10] Garey.MR, Johnson.DS, Computers and intractability: a guide to the theory of NP completeness. San Francisco: Freeman; (1979).

[11] Martello.S, Toth.P, Knapsack problems. New York: Wiley; (1990).

[12] Healy Jr. WC, Multiple choice programming. Operations Research;12:122 38. (1964)

[13] Lin EYH. Multiple choice programming: a state-of-the-art review. International Transactions in Operational Research;1:409-421. (1994)

[14] Lin EYH. A bibliographical survey on some well-known nonstandard knapsack problems.;36(4): 274 317 INFOR (1998)

[15] Dyer.ME, Walker.J. Dominance in multi-dimensional multiple-choice knapsack problems. Asia Pacific Journal of Operational Research;15(2):15968. (1998)

[16] Weingatner.HM, Capital budgeting of interrelated projects: survey and synthesis. Operations Research;12: 485-516. (1966)

[17] Peterson.CC, Computational experience with variants of the Balas algorithm applied to the selection of research and development projects. Management Science 13:736 50.(1967)

[18] Gilmore.PC, Gomory.RE. The theory and computation of knapsack functions. Operations Research;14: 1045 (1966)

[19] Shih.W, A branch and bound method for the multi constraint zero-one knapsack problem. Journal of Operations Research Society; 36978. (1979)

[20] Geo-rion.AM, Lagrangian relaxation for integer programming. Mathematical Programming Study 2:82114. (1974)

[21] Glover F. Surrogate constraints. Operations Research;16:741. (1968)

[22] Glover F. Heuristics for integer programming using surrogate constraints. Decision Sciences; 8:15666. (1977)

[23] Chu PC, Beasley JE. A genetic algorithm for the multidimensional knapsack problem. Journal of Heuristics;4: 6386. (1998)

[24] Bertsimas D, Demir R. An approximate dynamic programming approach to multidimensional knapsack problems. Management Science;48(4):550-565 (2002)

[25] Mhand Hifi, Slim Sadfi, Abdelkader Sbihi, An exact algorithm for the multiple-choice multidimensional knapsack problem. Cahiers de la MSE b04024, Maison des Sciences Economiques, University paris pantheon-Sorbonne. (2004)

[26] Khan.S, Li KF, Manning.EG, Akbar.MM, Solving the knapsack problem for adaptive multimedia system. Studia informatica 2:161-82. (2002)

[27] Mhand Hifi, Michrafy.M, Sbihi.A, Heuristic algorithms for the multiple-choice multidimensional knapsack problem. Journal of the Operational Research Society 55, 13231332 (2004)

[28] Parra-Hemendez R, N.Dimopoulos, A New Heuristic for Solving the Multi-choice Multidimensional knapsack problem. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions. Volume: 35 , Issue: 5 Page(s): 708- 717 (2005)

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

[30] Glover.F, Future paths for integer programming and links to artificial intelligence, Computers and Operations Research I9 (1986)

[31] Lee.C, Lehoczky.J, Rajkumar.R, Siewiorek.D. On quality of service optimization with discrete QoS options. In: RTAS'99: Proceedings of the fifth IEEE real-time technology and applications symposium. Washington, DC, USA: IEEE Computer Society; 1999. p. 276.

[32] Sadid MWH, Islam MR, Hasan SMK, A new strategy for solving multiple- choice multiple-dimension knapsack problem in pram model. In: Asian Applied Computing Conference. 2005.

[33] Kellerer H, Pferschy.U, Pisinger.D, Knapsack problems. Berlin: Springer; 2004.

[34] Chaitr S. Hire.M, Raymond. Hill2, New greedy heuristics for the Multiple-choice Multi-dimensional Knapsack Problem. International Journal of Operational Research Volume 2, 2007 Pages 495-512.

[35] Shahrear Iqbal, Md. Faizul Bari, M. Sohel Rahman, Solving the Multi-dimensional Multi-choice Knapsack Problem with the Help of Ants. ANTS Conference 2010: 312-323

[36] Beasley.J, OR-Library: Distributing test problems by electronic mail. The Journal of the Operational Research Society 41(11), 10691072 (1990)

[37] Alaya.I, C.Solnon et K. Ghedira, Des fourmis pour le sac a dos multidimensionnel. Dans : 4mes Journes Francophones de Recherche Oprationnelle (Francoro 2004), Fribourg, Suisse, pp. 159-160, 2004

[38] Han I, Jimmy Leblet, Gwendal Simon, Hard multidimensional multiple choice knapsack problems, an empirical study, Computers Operations Research, (2010), 37, 172-181.

[39] Cherfi.M, Hifi.M, A column generation method for the multiple-choice multi-dimensional knapsack problem, Comput. Optim. App. online first, Springer Netherlands (2008).

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

[41] Rad Mansia, Cludio Alvesa, J. M. Valrio de Carvalhoa and Said Hanafi. A hybrid heuristic for the multiple choice multidimensional knapsack problem. Engineering Optimization. iFirst, 1-22 (2011)

[42] Romain Picot-Clmente, Florence Mendes, Christophe Cruz, Christophe Nicolle. TOURISM-KM, A variant of MMKP applied to the tourism domain. ICORES 2012, Portugal (2012)

[43] Ykman-Couvreur, Cliaiztal MEC, Leuven Nollet, Vincent, Fast Multi-Dimension Multi-Choice Knapsack Heuristic for MP-SoC Run-Time Management. International Symposium on. Henk System-on-Chip, 2006 pages 1-4.

[44] I.Crevits, S.Hanafi, R.Mansi, and C.Wilbaut, Iterative semi-continuous relaxation heuristics for the multiple-choice multidimensional knapsack problem. Computers and Operations Research, 2011.

[45] S.Hanafi, R.Mansi, and C.Wilbaut, Iterative relaxation-based heuristics for the multiple-choice multidimensional knapsack problem. volume 5818 of Lecture Notes in Computer Science, pages 7383. Springer Verlag, 2009.

[46] T.Ghasemi and M.Razzazi, Development of core to solve the multidimensional multiple-choice knapsack problem. Computers and Operations Research, 60:349-360, 2010.

[47] M. Razzazi and T. Ghasemi, 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, 6:275-282, 2008.

[48] N. Cherfi and M. Hifi, Hybrid algorithms for the multiple-choice multi-dimensional knapsack problem. International Journal of Operational Research, 5(1):89-109, 2009.

[49] Rad Mansi, Cludio Alves, J. M.Valrio de Carvalho and Said Hanafi, A hybrid heuristic for the multiple choice multidimensional knapsack problem. Engineering Optimization, 2012. DOI:10.1080/0305215X.2012.717072 .

[50] Gaige.W, Lihong.G, A Novel Hybrid Bat Algorithm with Harmony Search for Global Numerical Optimization, Journal of Applied Mathematics, vol. 2013, 21pages, 2013. doi 10.1155/2013/696491

[51] Gaige.W, Lihong.G, Hong.Duan, Heqi.W, Luo.L, and Mingzhen.S. A Hybrid Metaheuristic DE/CS Algorithm for UCAV Three-Dimension Path Planning, The Scientific World Journal, vol. 2012, 11 pages, 2012. doi:10.1100/2012/583973