European Journal of Operational Research 212 (2011) 455–463

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

The constrained compartmentalized knapsack problem: mathematical models and solution methods

Aline A.S. Leão [a,*], Maristela O. Santos [a], Robinson Hoto [b], Marcos N. Arenales [a]

[a] Universidade de São Paulo, Instituto de Ciências Matemáticas e de Computação, Caixa Postal 668, 13560-970 São Carlos, SP, Brazil
[b] Universidade Estadual de Londrina, Caixa Postal 6001, 86051-990 Londrina, Pr, Brazil

ARTICLE INFO

Article history:
Received 1 May 2010
Accepted 14 February 2011
Available online 19 February 2011

Keywords:
Cutting
Knapsack problem
Column generation
Heuristics

ABSTRACT

The constrained compartmentalized knapsack problem can be seen as an extension of the constrained knapsack problem. However, the items are grouped into different classes so that the overall knapsack has to be divided into compartments, and each compartment is loaded with items from the same class. Moreover, building a compartment incurs a fixed cost and a fixed loss of the capacity in the original knapsack, and the compartments are lower and upper bounded. The objective is to maximize the total value of the items loaded in the overall knapsack minus the cost of the compartments. This problem has been formulated as an integer non-linear program, and in this paper, we reformulate the non-linear model as an integer linear master problem with a large number of variables. Some heuristics based on the solution of the restricted master problem are investigated. A new and more compact integer linear model is also presented, which can be solved by a branch-and-bound commercial solver that found most of the optimal solutions for the constrained compartmentalized knapsack problem. On the other hand, heuristics provide good solutions with low computational effort.

1. Introduction

The one-dimensional cutting stock problem consists of cutting rolls from stock into required items. A special case of the one-dimensional cutting stock problem is the two-stage cutting stock problem. In this case, the roll is cut into intermediate rolls. Then, each intermediate roll is cut into items. This two-stage cutting process can be found in many industries, such as paper (Haessler, 1971; Correia et al., 2004; Zak, 2002a,b), film (Haessler, 1979), and steel (Ferreira et al., 1990; Carvalho and Rodrigues, 1994, 1995; Hoto et al., 2007; Marques and Arenales, 2007).

The need for two stage cutting can be due to the restrictions of production processes and ordered item characteristics, e.g., limited number of knives, coating of materials (coated paper and coated steel ribbons), and thickness reduction (steel rolls). If the items are grouped into classes and only items of the same class can be cut from an intermediate roll, then we have the compartmentalized knapsack problem (Hoto et al., 2007; Marques and Arenales, 2007).

To illustrate the compartmentalized knapsack problem, consider the situation where a knapsack must be loaded with a certain number of items. For each item, a weight, a utility value and an upper bound are given (if no upper bound is given, the problem is called unconstrained). However, the items are of different classes (food, medicine, utensils etc.) and they have to be loaded in separate compartments inside the knapsack (each compartment is itself a knapsack to be loaded with items from the same class). The compartments have flexible capacities, but are lower and upper bounded. Each compartment has a fixed cost to be included inside the knapsack that depends on the class of items chosen. In addition, each new compartment introduces a fixed loss of capacity of the original knapsack. The compartmentalized knapsack problem consists of determining suitable compartment capacities, and how they should be loaded in such a way to maximize the total utility value, given by the sum of the utility value of the items loaded, minus the costs of the compartments. According to the improved typology by Wäscher et al. (2007), the problem can be categorized into one dimensional single large object placement problem and one dimensional single knapsack problem.

In the literature, the constrained compartmentalized knapsack problem has been considered in steel roll cutting, where items are grouped according to their thickness, i.e., items in the same class have the same thickness. A first cut on the original roll in stock produces various intermediate rolls, which have the same thicknesses as the original roll. An intermediate roll is seen as a compartment in the knapsack. Then, the thickness of each sub-roll is reduced to be the same as the item thicknesses of a class. A fixed trim is necessary to eliminate irregularities on the borders of each

* Corresponding author.
E-mail addresses: aasleao@gmail.com (A.A.S. Leão), mari@icmc.usp.br (M.O. Santos), hoto@uel.br (R. Hoto), arenales@icmc.usp.br (M.N. Arenales).

0377-2217/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2011.02.016

intermediate roll, which means a fixed loss of the knapsack capacity. Furthermore, there is a relevant cost of thickness reduction that depends on the percentage of reduction.

The solution methods we have found in the literature for the steel roll cutting problem are heuristics. Ferreira et al. (1990) propose a greedy heuristic to build intermediate objects (or compartments) and cutting patterns. However, the cutting patterns can be infeasible because of overproduction of required items, or the lower and upper bounds of the compartments cannot be satisfied. In this case, other different compartments must be produced. This process is repeated until all the requirement orders are met. Carvalho and Rodrigues (1994, 1995) use the column generation method for the steel roll cutting problem, where each compartment produces one type of item, i.e., a homogeneous solution for each compartment. Marques and Arenales (2007) present an integer non-linear programming formulation of the constrained compartmentalized knapsack problem and propose several heuristics to solve the problem consisting of two steps: in the first step a number of compartments are generated, and in the second step the compartments are selected to load the overall knapsack. Hoto et al. (2007) consider the cutting stock problem where each cutting pattern is restricted to be compartmentalized. To solve the unconstrained compartmentalized knapsack problem, they propose heuristic and exact methods, while for the constrained version a simple heuristic was used. Hoto et al. (2005) introduce a column generation technique for compartmentalized knapsacks, better investigated later by Leão (2009). A comparison between the column generation technique for compartmentalized knapsacks and a heuristic in Marques and Arenales (2007) can be found in Hoto et al. (2010). Xavier and Miyazawa (2006) called the constrained compartmentalized knapsack problem a class constrained shelf knapsack problem and propose a polynomial time approximation and fully polynomial time approximation algorithms.

Other variants of the constrained compartmentalized knapsack problem can be found in the literature such as: the class constrained bin packing problem (Shachnai and Tamir, 2001b, 2004; Xavier and Miyazawa, 2008a, 2009); the online class constrained bin packing problem (Shachnai and Tamir, 2004; Xavier and Miyazawa, 2008a); the temporary class constrained bin packing problem (Shachnai and Tamir, 2004); the class constrained shelf bin packing problem (Xavier and Miyazawa, 2008b, 2009); the class constrained multiple knapsack problem (Shachnai and Tamir, 2001a,b); the fair placement problem (Shachnai and Tamir, 2001a); the generalized class constrained packing problem and the 0–1 class constrained knapsack problem (Shachnai and Tamir, 2001b). The main differences among these variants are the objective function, the fixed loss (if it is considered or not), or a sequence to pack the items. In order to solve these problems, the authors propose approximation algorithms or FFD and BFD based heuristics. The variants model the storage management for multimedia-on-demand (MOD) systems. These services are common in library information retrieval, entertainment, and commercial applications. The compartmentalized structure also arises in flexible manufacture systems where machines are loaded with different tools to process a number of operations. Each operation can be seen as an item and the operations that use the same tool belong to the same class (Özpeynirci and Azizoglu, 2010).

In this paper, the integer non-linear model introduced by Marques and Arenales (2007) is reformulated into an integer linear master program and subproblems, where the subproblems are given by constrained knapsack problems. We give a general framework to build heuristics, which include all the heuristics of Marques and Arenales (2007). Furthermore, the column generation method is used to solve the relaxed master problem, and in the optimality, the restricted and integer master problem is solved by considering the generated columns. When solving the subproblems, one can determine one or many solutions for them. Computational results show that proposed heuristics can find good solutions with low computational effort.

We also propose a new integer linear model, which is solved by solver Cplex 12.1. The paper is organized as follows. Section 2 contains three mathematical formulations for the constrained compartmentalized knapsack problem. Two upper bounds are presented in Section 3. In Section 4, we give a general framework to devise the heuristics and propose heuristics based on static and dynamic column generations. Computational results are reported in Section 5, and the final remarks are in Section 6.

2. Problem formulations

The constrained compartmentalized knapsack problem is well-defined by the following parameters.

Data:

- L: knapsack capacity;
- n: number of type of items;
- K: number of classes;
- $\{n_0+1,\dots,n_1,n_1+1,\dots,n_2\ ,\dots,n_{K-1}+1,\dots,n_K\}$: set of items, where $n_0=0$ and $n_K=n$. That is, class k consists of $\{n_{k-1}+1,\dots,n_k\}$;
- l_i: length of item i;
- v_i: utility value of including item type i in the knapsack;
- σ_i: limit on the number of item type i in the knapsack (for σ_i large enough, the problem is called unconstrained);
- $L^k_{max}\left(L^k_{min}\right)$: upper (lower) bound to the capacity of compartments for class k;
- c_k: cost of including a new compartment for class k;
- S_k: loss of capacity in the overall knapsack due to the inclusion of a new compartment for class k.

2.1. Integer non-linear model

Let a_{is} be the number of items type i in compartment s, and β_{sk} be the number of times that compartment s loaded with items of class k is repeated in the knapsack. Consider the maximum number of compartments for class k inside the knapsack given by: $\overline{N}_k=\lfloor\frac{L}{L^k_{min}}\rfloor$. The mathematical model of Marques and Arenales (2007) is given as follows.

$$\text{maximize}\ \sum_{k=1}^{K}\sum_{s=1}^{\overline{N}_k}\left(\sum_{i=n_{k-1}+1}^{n_k} v_i a_{is}-c_k\right)\beta_{sk},\tag{1}$$

$$\text{subject to}:\sum_{k=1}^{K}\sum_{s=1}^{\overline{N}_k}\left(\sum_{i=n_{k-1}+1}^{n_k} l_i a_{is}+S_k\right)\beta_{sk}\leqslant L,\tag{2}$$

$$\sum_{s=1}^{\overline{N}_k} a_{is}\beta_{sk}\leqslant\sigma_i,\quad k=1,\dots,K,\ i=n_{k-1}+1,\dots,n_k,\tag{3}$$

$$L^k_{min}\leqslant\sum_{i=n_{k-1}+1}^{n_k} l_i a_{is}+S_k\leqslant L^k_{max},\quad k=1,\dots,K,\ s=1,\dots,\overline{N}_k,\tag{4}$$

$$a_{is}\geqslant 0,\ \text{integer}, k=1,\dots,K,\ i=n_{k-1}+1,\dots,n_k,\ s=1,\dots,\overline{N}_k,\tag{5}$$

$$\beta_{sk}\geqslant 0,\ \text{integer}, k=1,\dots,K,\ s=1,\dots,\overline{N}_k.\tag{6}$$

The objective function (1) to be maximized consists of the total utility value minus the cost of including compartments. Constraint (2) is to satisfy the overall knapsack capacity. Note that compartment s with items of class k is valued by: $\sum_{i=n_{k-1}+1}^{n_k} v_i a_{is}-c_k$, and its size is: $\sum_{i=n_{k-1}+1}^{n_k} l_i a_{is}+S_k$. Constraints (3) limit the number of the items in the knapsack, and constraints (4) limit the capacity of each compartment. Non-linearities can be found in the objective function and also in the constraints (2) and (3).

2.2. Problem reformulation

The integer non-linear model (1)–(6) has shown to be difficult to solve. Thus, we propose a straightforward reformulation by considering all the compartments known, which means all possible solutions for constraints (4) and (5). In order not to build a compartment that cannot be used at least once due to constraints (3), we limit the number of items inside the compartments. Therefore, a feasible compartment for class $k (k = 1,\dots,K)$ is a solution to the following constraints:

$$L^k_{min} \leqslant \sum_{i=n_{k-1}+1}^{n_k} l_i a_i + S_k \leqslant L^k_{max}, \tag{7}$$

$$0 \leqslant a_i \leqslant \sigma_i, \quad \text{integer}, \ i = n_{k-1}+1,\dots,n_k, \tag{8}$$

where a_i represents the number of items type i from class k in the compartment. Let $\hat{a}_{is}$, $s = 1,\dots,N_k$, $i = n_k - 1 + 1,\dots,n_k$ all the possible solutions to constraints (7) and (8), i.e., N_k is the number of all possible compartments loaded with items of class k (N_k is much larger than $\overline{N}_k$). Therefore, problem (1)–(6) is equivalently written as the following problem referred to as a master problem:

$$\text{maximize} \quad \sum_{k=1}^{K}\sum_{s=1}^{N_k}(V_{sk} - c_k)\beta_{sk}, \tag{9}$$

$$\text{subject to}: \sum_{k=1}^{K}\sum_{s=1}^{N_k}(L_{sk} + S_k)\beta_{sk} \leqslant L, \tag{10}$$

$$\sum_{s=1}^{N_k}\hat{a}_{is}\beta_{sk} \leqslant \sigma_i, \quad k = 1,\dots,K, \ i = n_{k-1}+1,\dots,n_k, \tag{11}$$

$$\beta_{sk} \geqslant 0, \quad \text{integer}, \ k = 1,\dots,K, \ s = 1,\dots,N_k, \tag{12}$$

where $L_{sk} = \sum_{i=n_{k-1}+1}^{n_k} l_i\hat{a}_{is} + S_k$ and $V_{sk} = \sum_{i=n_{k-1}+1}^{n_k} v_i\hat{a}_{is}$ are the length and the utility value of compartment s for class k, respectively. Problem (9)–(12) is an integer linear optimization problem with a large number of variables. The matrix structure of the master problem is given in Table 1.

The linear relaxation of master problem (9)–(12) can be solved by the column generation technique, where K knapsack problems (pricing subproblems) restricted to (7) and (8) are solved per iteration. See details in Section 3.2.

2.3. Integer linear formulation

Instead of defining a compartment and deciding how many times it is loaded in the knapsack (a pattern-oriented type modeling), now we build the compartments without thinking in repetition, just if it is or not loaded (an item-oriented type modeling). Of course, a compartment can have the same items as others (implicit repetition). Let a_{is} be the number of items type i, $n_{k-1} + 1 \leqslant i \leqslant n_k$, in the sth compartment for class k included in the knapsack. The maximum number of compartments for class k inside the knapsack is given by: $\overline{N}_k = \lfloor \frac{L}{L^k_{min}} \rfloor$. Let δ^k_s be 1 if compartment s for class k is built, and 0 otherwise. Then, the objective function can be written as follows:

$$\text{maximize} \sum_{k=1}^{K}\sum_{s=1}^{\overline{N}_k}\sum_{i=n_{k-1}+1}^{n_k}\left(v_i a_{is} - c_k\delta^k_s\right).$$

Constraints 3 can be represented as:

$$\sum_{s=1}^{\overline{N}_k} a_{is} \leqslant \sigma_i, \quad k = 1,\dots,K, \ i = n_{k-1}+1,\dots,n_k.$$

In addition, the constraints (2) and (4) are given, respectively by:

$$\sum_{k=1}^{K}\sum_{s=1}^{\overline{N}_k}\sum_{i=n_{k-1}+1}^{n_k}\left(l_i a_{is} + S_k\delta^k_s\right) \leqslant L$$

$$\delta^k_s L^k_{min} \leqslant \sum_{i=n_{k-1}+1}^{n_k} l_i a_{is} + S_k\delta^k_s \leqslant \delta^k_s L^k_{max}, \quad k = 1,\dots,K, \ s = 1,\dots,\overline{N}_k.$$

Therefore, an integer linear mathematical formulation can be written for the constrained compartmentalized knapsack problem:

$$\text{maximize} \sum_{k=1}^{K}\sum_{s=1}^{\overline{N}_k}\sum_{i=n_{k-1}+1}^{n_k}\left(v_i a_{is} - c_k\delta^k_s\right), \tag{13}$$

$$\text{subject to}: \sum_{k=1}^{K}\sum_{s=1}^{\overline{N}_k}\sum_{i=n_{k-1}+1}^{n_k}\left(l_i a_{is} + S_k\delta^k_s\right) \leqslant L, \tag{14}$$

$$\sum_{s=1}^{\overline{N}_k} a_{is} \leqslant \sigma_i, \ k = 1,\dots,K, i = n_{k-1}+1,\dots,n_k, \tag{15}$$

$$\delta^k_s L^k_{min} \leqslant \sum_{i=n_{k-1}+1}^{n_k} l_i a_{is} + S_k\delta^k_s \leqslant \delta^k_s L^k_{max}, \ k = 1,\dots,K, s = 1,\dots,\overline{N}_k, \tag{16}$$

$$a_{is} \geqslant 0, \ \text{integer}, k = 1,\dots,K, i = n_{k-1}+1,\dots,n_k, s = 1,\dots,\overline{N}_k, \tag{17}$$

$$\delta^k_s \in \{0,1\}, \ k = 1,\dots,K, s = 1,\dots,\overline{N}_k. \tag{18}$$

In constraints (16), if $\delta^k_s = 1$, compartment s is loaded in the knapsack with the lenght between L^k_{min} and L^k_{max}, otherwise ($\delta^k_s = 0$) the variables $a_{is} = 0$, $i = n_{k-1} + 1,\dots,n_k$.

Table 1
Matrix structure of the master problem.

$V_{11} - c_1$	…	$V_{1N_1} - c_1$	$V_{21} - c_2$	…	$V_{2N_2} - c_2$	…	$V_{K1} - c_K$	…	V_{KN_K}
L_{11}	…	L_{1N_1}	L_{21}	…	L_{2N_2}	…	L_{K1}	…	L_{KN_K}
$\hat{a}_{(n_0+1)1}$	…	$\hat{a}_{(n_0+1)N_1}$	0	…	0	…	0	…	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
$\hat{a}_{n_1 1}$	…	$\hat{a}_{n_1 N_1}$	0	…	0	…	0	…	⋮
0	…	0	$\hat{a}_{(n_1+1)1}$	…	$\hat{a}_{(n_1+1)N_2}$	…	0	…	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0	…	0	$\hat{a}_{n_2 1}$	…	$\hat{a}_{n_2 N_2}$	…	0	…	0
⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮	⋮	⋮
0	…	0	0	…	0	…	$\hat{a}_{(n_{K-1}+1)1}$	…	$\hat{a}_{(n_{K-1}+1)N_K}$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0	…	0	0	…	0	…	$\hat{a}_{n_K 1}$	…	$\hat{a}_{n_K N_K}$

3. Upper bounds for the constrained compartmentalized knapsack problem

In order to evaluate the quality of the heuristic solutions, upper bounds are used. In this section, we describe two upper bounds for the constrained compartmentalized knapsack problem. The first one was proposed in Marques and Arenales (2007), and the second one is the solution of the relaxed master problem (9)–(12) by the column generation method.

3.1. Super-compartments

Marques and Arenales (2007) provided an upper bound for problem (1)–(6) using a compact integer linear program. Let a_i be the number of items type i in the knapsack, and β_k be the number of compartments loaded with items of class k in the knapsack. The super-compartment model is given as follows.

$$\text{maximize} \quad \sum_{i=1}^{n} v_i a_i - \sum_{k=1}^{K} c_k \beta_k, \tag{19}$$

$$\text{subject to} \quad \sum_{i=1}^{n} l_i a_i \leqslant L - \left(\sum_{k=1}^{K} S_k \beta_k \right), \tag{20}$$

$$L^k_{min} \beta_k \leqslant \sum_{i=n_{k-1}+1}^{n_k} l_i a_i + S_k \beta_k \leqslant L^k_{max} \beta_k, \quad k = 1, \ldots, K, \tag{21}$$

$$0 \leqslant a_i \leqslant \sigma_i, \quad \text{integer}, \ i = 1, \ldots, n, \tag{22}$$

$$\beta_k \geqslant 0, \quad \text{integer}, \ k = 1, \ldots, K. \tag{23}$$

Note that constraint (21) considers one super-compartment with length between $L^k_{min}\beta_k$ and $L^k_{max}\beta_k$ for each class k, which may not be divided into β_k compartments. See Marques and Arenales (2007) for the proof that (19)–(23) is a relaxation of problem (1)–(6).

3.2. Relaxed linear master problem: column generation method

The column generation method was initially used by Manne (1958), Dantzig and Wolfe (1960) and Gilmore and Gomory (1961). In this section, the method devised by Gilmore and Gomory is applied to solve the linear relaxation of the master problem (9)–(12). For each class k ($k = 1, \ldots, K$) there arises a pricing problem as follows. Let λ and π be the dual variables associated with constraints (10) and (11), respectively:

$$(\lambda, \pi) = (\lambda, \pi_{n_0}, \ldots, \pi_{n_1}, \pi_{n_1+1}, \ldots, \pi_{n_2}, \ldots, \pi_{n_{K-1}+1}, \ldots, \pi_{n_K}).$$

A column of the master problem corresponding to class k is given by (see Table 1, omitting index s):

$$\alpha_k = (L_k, 0, \ldots, 0, a_{n_{k-1}+1}, \ldots, a_{n_k}, 0, \ldots 0)^T.$$

Thus, the best reduced cost for each class k, $k = 1, \ldots, K$, is the following:

$$max \left\{ (V_k - c_k) - \lambda L_k - (\pi_{n_{k-1}+1}, \ldots, \pi_{n_k}) \begin{pmatrix} a_{n_{k-1}+1} \\ \vdots \\ a_{n_k} \end{pmatrix} \right\}.$$

Since $V_k = \sum_{i=n_{k-1}+1}^{n_k} v_i a_i$ and $L_k = \sum_{i=n_{k-1}+1}^{n_k} l_i a_i + S_k$, the pricing problem for class k, $k = 1, \ldots, K$ is given as follows.

$$\text{maximize} \quad \sum_{i=n_{k-1}+1}^{n_k} (v_i - \lambda l_i - \pi_i) a_i, \tag{24}$$

$$\text{subject to}: L^k_{min} \leqslant \sum_{i=n_{k-1}+1}^{n_k} l_i a_i + S_k \leqslant L^k_{max}, \tag{25}$$

$$0 \leqslant a_i \leqslant \sigma_i, \quad i = n_{k-1} + 1, \ldots, n_k, \ \text{integer}. \tag{26}$$

In each iteration of the simplex method, K subproblems (24)–(26) are solved to provide columns to the master problem. If $\sum_{i=n_{k-1}+1}^{n_k} (v_i - \lambda l_i - \pi_i) a_i - (c_k + \lambda S_k) \leqslant 0$ for all $k = 1, \ldots, K$, then an optimal solution of the linear relaxation of the master problem is achieved.

4. Solution methods

Two types of solution methods are described in this section. The first method is based on several heuristics from the literature (static column generation), and the second one uses those heuristics in the column generation method (dynamic column generation).

4.1. Restricted master problem: static column generation

Several heuristics have been proposed by Marques and Arenales (2007) who use ideas of decomposition, consisting of two stages. In the first stage many subproblems are solved to provide columns for the master problem. These subproblems are given by (24)–(26), but the $\lambda = 0$, $\pi_i = 0$ for all i, and different solutions are taken into account, not only the best one, but the second best solution, the third best etc. Moreover, by varying the right hand side in (25), solutions are obtained with different lengths of compartments.

4.1.1. Z best compartment heuristic (Marques and Arenales, 2007)

Stage 1: For class k, $k = 1, \ldots, K$ determine the Z-best solutions for the following subproblem:

$$\text{maximize} \quad \sum_{i=n_{k-1}+1}^{n_k} v_i a_i, \tag{27}$$

$$\text{subject to}: \sum_{i=n_{k-1}+1}^{n_k} l_i a_i + S_k \leqslant L^k_{max}, \tag{28}$$

$$0 \leqslant a_i \leqslant \sigma_i, \quad i = n_{k-1} + 1, \ldots, n_k, \ \text{integer}. \tag{29}$$

Let $(\hat{a}_{iz})_{i=n_{k-1}+1}^{n_k}$ be the zth best solution, and $L_{zk} = \sum_{i=n_{k-1}+1}^{n_k} l_i \hat{a}_{iz} + S_k$ and $V_{zk} = \sum_{i=n_{k-1}+1}^{n_k} v_i \hat{a}_{iz}$ be the compartment length and the utility value. If $L_{zk} < L^k_{min}$ the solution is discarded.
Stage 2: Solve the restricted master problem.

$$\text{maximize} \quad \sum_{k=1}^{K} \sum_{z=1}^{Z} (V_{zk} - c_k) \beta_{zk}, \tag{30}$$

$$\text{subject to}: \sum_{k=1}^{K} \sum_{z=1}^{Z} L_{zk} \beta_{zk} \leqslant L, \tag{31}$$

$$\sum_{z=1}^{Z} \hat{a}_{iz} \beta_{zk} \leqslant \sigma_i, \ k = 1, \ldots, K, i = n_{k-1} + 1, \ldots, n_k, \tag{32}$$

$$\beta_{zk} \geqslant 0, \ \text{integer}, k = 1, \ldots, K, z = 1, \ldots, Z. \tag{33}$$

Remark: For $Z = 1$, problem (30)–(33) is a knapsack problem.

4.1.2. W capacity heuristic (Marques and Arenales, 2007)

Considering W as the number of different capacities for the compartments, this heuristic can be described as follows.

Stage 1: For class k, $k = 1, \ldots, K$ and for $w = 1, \ldots, W$, solve the following subproblem, where $L^1 = L^k_{max}$ is the first capacity:

$$\text{maximize} \quad \sum_{i=n_{k-1}+1}^{n_k} v_i a_i, \tag{34}$$

$$\text{subject to}: \sum_{i=n_{k-1}+1}^{n_k} l_i a_i + S_k \leqslant L^w, \tag{35}$$

$$0 \leqslant a_i \leqslant \sigma_i, \quad i = n_{k-1} + 1, \ldots, n_k, \ \text{integer}. \tag{36}$$

And determine $(\hat{a}_{iw})_{i=n_{k-1}+1}^{n_k}$, $L^{w+1} = \left(\sum_{i=n_{k-1}+1}^{n_k} l_i\hat{a}_{iw} + S_k\right) - 1$. If $L_{wk} < L_{min}^k$ the solution is discarded and returns with w different capacities.

Stage 2: Let $L_{wk} = \sum_{i=n_{k-1}+1}^{n_k} l_i\hat{a}_{iw} + S_k$ and $V_{wk} = \sum_{i=n_{k-1}+1}^{n_k} v_i\hat{a}_{iw}$ be the length and the utility value of wth capacity. Solve the restricted master problem.

$$\text{maximize} \sum_{k=1}^{K}\sum_{w=1}^{W}(V_{wk} - c_k)\beta_{wk}, \tag{37}$$

$$\text{subject to} \sum_{k=1}^{K}\sum_{w=1}^{W} L_{wk}\beta_{wk} \leqslant L, \tag{38}$$

$$\sum_{w=1}^{W}\hat{a}_{iw}\beta_{wk} \leqslant \sigma_i, \quad k = 1,\ldots,K,\ i = n_{k-1}+1,\ldots,n_k, \tag{39}$$

$$\beta_{wk} \geqslant 0, \quad \text{integer}, k = 1,\ldots,K,\ w = 1,\ldots,W. \tag{40}$$

4.1.3. A hybrid heuristic

The Z best compartment and W capacity heuristic is now combined. That is, consider the Z best solutions for each different capacity.

Stage 1: For class k, $k = 1,\ldots,K$ determine the Z best solutions for problem (34)–(36), where $L^1 = L_{max}^k$, and obtain $(\hat{a}_i^{zw})_{i=n_{k-1}+1}^{n_k}$, $V_k^{zw} = \sum_{i=n_{k-1}+1}^{n_k} v_i\hat{a}_i^{zw}$ and $L_k^{zw} = \sum_{i=n_{k-1}+1}^{n_k} l_i\hat{a}_i^{zw} + S_k$. In order to determine L_{w+1}^k (the $(w+1)$th capacity), we choose $L_{w+1}^k = min\{L_k^{zw}, z = 1,\ldots,Z\}$.

Stage 2: Solve the restricted master problem:

$$\text{maximize} \sum_{k=1}^{K}\sum_{z=1}^{Z}\sum_{w=1}^{W}(V_k^{zw} - c_k)\beta_k^{zw}, \tag{41}$$

$$\text{subject to:} \sum_{k=1}^{K}\sum_{z=1}^{Z}\sum_{w=1}^{W} L_k^{zw}\beta_k^{zw} \leqslant L, \tag{42}$$

$$\sum_{z=1}^{Z}\sum_{w=1}^{W}\hat{a}_i^{zw}\beta_k^{zw} \leqslant \sigma_i, \quad k = 1,\ldots,K,\ i = n_{k-1}+1,\ldots,n_k, \tag{43}$$

$$\beta_k^{zw} \geqslant 0, \ \text{integer}, k = 1,\ldots,K, z = 1,\ldots,Z, w = 1,\ldots,W. \tag{44}$$

4.2. Restrict master problem: dynamic column generation method

In this section, a method based on the column generation method is proposed. The method consists of solving the relaxed master problem (9)–(12), but for each iteration of the simplex method a set of columns is found for each subproblem (24)–(26), by using the hybrid heuristic. In addition, when an optimal solution has been found for the linear relaxation of the master problem, the integer restricted master problem with all columns is solved. To find an initial solution, we use slack variables, and consequently, $\lambda = \pi_i = 0$. Therefore, the solution from the first iteration in the simplex method is the same found by the hybrid heuristic. All new columns with positive reduced costs are added into the restricted master problem, and they are kept until stopping. A feasible solution can be obtained by the column generation procedure stated as follows.

Algorithm: Dynamic column generation

```
Stage 1: Generating columns
Phase 1: Generate starting solution using slack variables.
    Stop = False.
Phase 2: While Stop = False do
      Solve the linear relaxed master problem (9)–(12) and
      determine the dual variables $(\lambda, \pi)$.
      Generate new columns using Stage 1 of the hybrid
      heuristic for subproblem (24)–(26).
      If $\sum_{i=n_{k-1}+1}^{n_k}(v_i - \lambda l_i - \pi_i)\hat{a}_i^{11} - (c_k + \lambda S_k) \leqslant 0$, $k = 1,\ldots,K$,
        then the solution to the restricted master problem is
        an optimal solution for the
          master problem.
        Stop = True.
      Else
        Add all new generated columns with positive
        reduced costs to the new restricted
          master problem.
      End While.
Stage 2: Solve the integer restricted master problem
```

5. Computational results

In this section, we report computational experiments with the methods presented earlier, which were implemented in C. The code was run on an Intel Core2 Duo with 2.33 GHz and the memory of 2 Gbyte. To determine the best and the Z best solutions of a knapsack problem, we used a branch-and-bound algorithm based

Table 2
Characteristics of the problems

Problem	K	Number of items	Correlation	Problem	K	Number of items	Correlation
1	5	20	1	19	15	20	1
2	5	20	2	20	15	20	2
3	5	20	3	21	15	20	3
4	5	40	1	22	15	40	1
5	5	40	2	23	15	40	2
6	5	40	3	24	15	40	3
7	5	80	1	25	15	80	1
8	5	80	2	26	15	80	2
9	5	80	3	27	15	80	3
10	10	20	1	28	20	20	1
11	10	20	2	29	20	20	2
12	10	20	3	30	20	20	3
13	10	40	1	31	20	40	1
14	10	40	2	32	20	40	2
15	10	40	3	33	20	40	3
16	10	80	1	34	20	80	1
17	10	80	2	35	20	80	2
18	10	80	3	36	20	80	3

Table 3
Average gaps to upper bounds in Section 3.

Problem	GAP (%)	Problem	GAP (%)	Problem	GAP (%)	Problem	GAP (%)
1	1.83	10	1.25	19	1.58	28	1.17
2	0.16	11	0.29	20	0.32	29	0.29
3	0.31	12	0.30	21	0.18	30	0.24
4	1.69	13	1.31	22	0.99	31	0.97
5	0.22	14	0.16	23	0.21	32	0.18
6	−0.11	15	−0.02	24	0.03	33	0.07
7	0.95	16	1.01	25	1.18	34	1.53
8	0.22	17	0.24	26	0.18	35	0.13
9	−0.46	18	0.12	27	−0.05	36	−0.23

on the algorithm by Gilmore and Gomory (1963), and the others were solved by the CPLEX 12.1 package.

5.1. The instances

The instances were generated as follows. The knapsack length is set to 1100 mm. To generate data related to items, we based on Pisinger (1999). For this, we choose a number R and consider three categories:

(1) Uncorrelated instances: utility values are uniformly distributed in a finite interval,
(2) Weakly correlated instances: utility values are uniformly distributed in the interval $\left[l_i - \frac{R}{10}, l_i + \frac{R}{10}\right]$,
(3) Strongly correlated instances: utility values are given by: $l_i + 10$.

Then, based on these three categories, the lengths of the items are uniformly and randomly generated as: $l_i \in [L^k_{min}, L^k_{max}]$. The utility values are such that:

(1) $v_i \in [L^k_{min}, L^k_{max}]$,
(2) $v_i \in \left[l_i - \frac{L^k_{max}}{30}, l_i + \frac{L^k_{max}}{30}\right]$,
(3) $v_i = l_i + 10$.

The number of classes K was 5, 10, 15 and 20. For each class k, $k = 1,\ldots,K$, we consider the number of items as 20, 40 and 80 with the demand for each item fixed to 1. The compartment length is limited to $\left[L^k_{min}, L^k_{max}\right] = [100mm, 300mm]$. Finally, the costs were generated randomly, such as $c_k \in [1,100]$. The results described in this section are for $S_k = 0$, however, some observations for $S_k \neq 0$ are made at the end of the section. Table 2 shows the characteristics for each problem type, for which 20 instances were generated.

5.2. Upper bounds: the relaxed master problem versus super-compartments

The two upper bounds presented in Section 3 are now compared against each other. The gap is given by: $\frac{f_{CG} - \bar{f}}{\bar{f}}$, where f_{CG} is the optimal objective function value of the relaxed master problem given by (9), and $\bar{f}$ is the value of the objective function (19). Table 3 shows the gaps.

Note that, the super-compartment upper bound is tighter than the linear relaxation of the master problem, but not dominant. The column generation formulation was better only for 19.31% of the instances. Moreover, the average run time to solve problem (19)–(23) was 0.12 of a second, while for the column generation method it was 0.47 of a second, which took an average 9.4 iterations.

5.3. Integer linear formulation

Considering all 720 generated instances, for only two of them the CPLEX did not prove the optimality of the solution of model (13)–(18), but with gaps of 0.24% and 0.25%. The total average run time is 87.28 seconds, but the average run time for the problems range between 1.56 and 1410.43 seconds. The run time is very unstable and does not depend on the size of the instance. For some instances, it can take less than 1 second, and for a few others it can take more than 7 hours. The average standard deviation of time is 357.19 seconds for all instances, and the standard deviation range between 0.47 seconds and 6235.81 seconds. We consider model (19)–(23) to estimate the gap of the integer linear formulation, which is given by: $\frac{\bar{f} - f}{f}$, where f is the objective function value (13) and $\bar{f}$ is the objective function value (19) met by CPLEX. For almost 50% of the instances the gap is 0%, and the average gap is 0.20%. The super-compartment upper bound given by problem (19)–(23) was solved by CPLEX, within an average run time of 0.12 of a second, and the minimum and maximum run time are 0.01 and 3.02 seconds. Thus, the super-compartment provides a good upper bound and can be solved with low computational time.

5.4. Hybrid heuristic and dynamic column generation

A number of tests were run for the hybrid heuristic and dynamic column generation approaches by setting the parameters $Z = 1, 5, 7$ and $W = 1, 5, 7$, that is, the Z best solutions and the W different capacities for the compartments. For each instance, we calculated the gap as: $\frac{f - \bar{f}}{f}$, where $\bar{f}$ is the value of the objective function (13) provided by integer linear formulation, and f the value of the objective function of the hybrid heuristic or the dynamic column

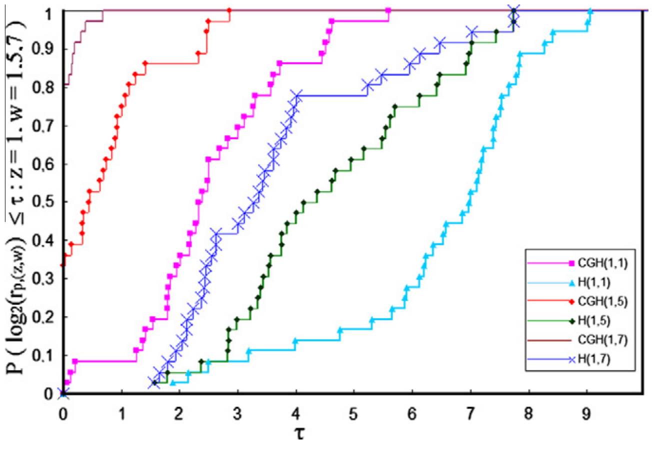

Fig. 1. Gap evaluation profiles ($Z = 1$).

Table 4
Average computational time.

Z, W	$H(Z,W)$	$\mathrm{CGH}(Z,W)$	Z, W	$H(Z,W)$	$\mathrm{CGH}(Z,W)$	Z, W	$H(Z,W)$	$\mathrm{CGH}(Z,W)$
1, 1	0.02	0.49	5, 1	0.03	1.51	7, 1	0.03	1.98
1, 5	0.04	1.31	5, 5	0.09	9.62	7, 5	0.10	9.48
1, 7	0.05	1.65	5, 7	0.09	9.62	7, 7	0.12	14.75

Table 5
Gaps for $Z = 1$.

	CGH(1,1) (%)	CGH(1,5) (%)	CGH(1,7) (%)	H(1,1) (%)	H(1,5) (%)	H(1,7) (%)
Worst	9.84	6.51	6.40	24.28	19.60	19.57
Better	0.17	0.08	0.02	10.04	0.43	0.25
Average	2.13	0.90	0.72	19.22	10.02	6.68

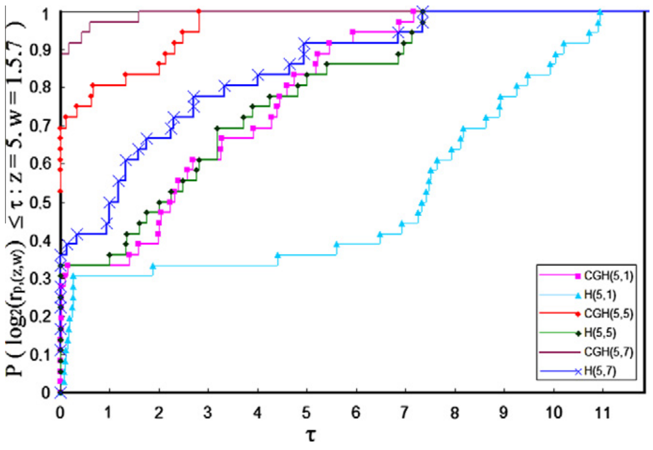

Fig. 2. Gap evaluation profiles ($Z = 5$).

generation. These results from heuristics are visualized using performance profiles proposed by Dolan and Moré (2002). We denote $t_{p,\,(Z,W)}$ as the average gap of the objective function for problem type p solved by the hybrid heuristic (or dynamic column generation) with parameters Z and W. The function evaluation $r_{p,\,(Z,W)}$ is a normalization of the average gap given by:

$$r_{p,(Z,W)} = \frac{t_{p,(Z,W)}}{\min\{t_{p,(Z,W)}, \text{for all parameters } Z \text{ and } W \text{ analyzed}\}}. \tag{45}$$

If the denominator is zero, then the values of gaps are shifted by 1. The general performance of the heuristics in the problems is defined by a function $\rho : \mathfrak{R} \to [0,1]$ given by:

$$\rho_{(Z,W)}(\tau) = \frac{\text{number of problems so that } \log_2 r_{p,(Z,W)} \leqslant \tau}{\text{total number of problems}}. \tag{46}$$

This function is called the performance profile and it represents the fraction of problems solved by the heuristic with a performance $\log_2 r_{p,\,(Z,W)}$ within a factor τ of the best possible performance obtained. This is equivalent to the probability of the heuristic solving a problem with a performance $\log_2 r_{p,\,(Z,W)}$ within a factor τ of the best possible ratio. Then, $\rho_{(Z,W)}$ can be seen as the cumulative distribution function for the performance ratio, given by:

$$\rho_{(Z,W)} = P(\log_2 r_{p,(Z,W)} \leqslant \tau \text{ for all parameters } Z \text{ and } W \text{ analyzed}). \tag{47}$$

Fig. 1 shows the performance profile for the hybrid heuristic ($H(Z,W)$) and dynamic column generation ($\mathrm{CGH}(Z,W)$) for $Z = 1$ and $W = 1, 5, 7$ represented by (Z,W). It indicates that the dynamic column generation with $Z = 1$ and $W = 7$ (CGH(1,7)) resulted in the best overall performance. This method obtained the smallest gap for about 79% of the problems and solved each problem with a gap no worse than $2^{0.9}$ of the best gap obtained by the other methods. It is evident from Fig. 1 that for $Z = 1$, the dynamic column generation dominates the hybrid heuristic. Even for $Z = 1$ and $W = 1$, the dynamic column generation dominates the hybrid heuristic for $Z = 1$ and $W = 7$. As value Z increases, the performance is better for both methods. However, the run time increases as shown in the results presented in Table 4. To complete the results, Table 5 shows the gaps for both methods considering $Z = 1$ and $W = 1, 5, 7$. These results demonstrate that the dynamic column generation obtains better gaps than the hybrid heuristic. Note that if $Z = 1$ in the hybrid heuristic, it gives the W capacity heuristic.

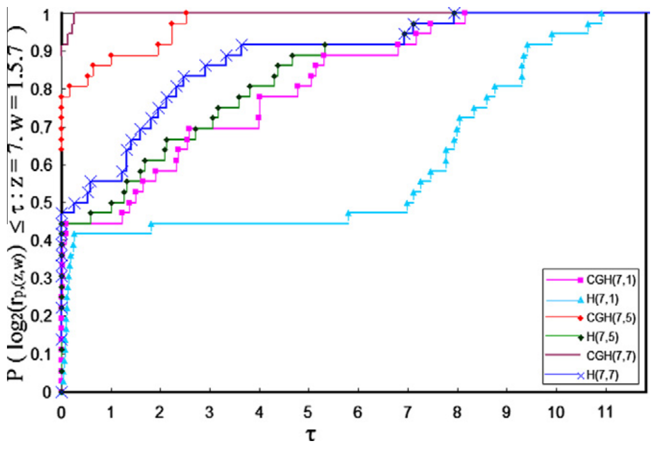

Fig. 3. Gap evaluation profiles ($Z = 7$).

Table 6
Gaps for $Z = 5$.

	CGH(5,1) (%)	CGH(5,5) (%)	CGH(5,7) (%)	H(5,1) (%)	H(5,5) (%)	H(5,7) (%)
Worst	7.28	6.28	6.27	23.10	19.58	19.57
Better	0.04	0.00	0.00	4.65	0.01	0.00
Average	1.51	0.41	0.25	15.64	6.04	3.55

Table 7
Gaps for $Z = 7$.

	CGH(7,1) (%)	CGH(7,5) (%)	CGH(7,7) (%)	$H(7,1)$ (%)	$H(7,5)$ (%)	$H(7,7)$ (%)
Worst	8.50	6.26	6.27	22.00	19.57	19.57
Better	0.03	0.00	0.00	3.06	0.01	0.00
Average	1.51	0.29	0.22	13.76	2.65	2.29

Fig. 2 depicts the performance profile for the hybrid heuristic ($H(Z,W)$) and dynamic column generation (CGH(Z,W)) for $Z = 5$ and $W = 1$, 5, 7. Note that $W = 1$ in the hybrid heuristic gives the Z best compartment heuristic. In the same manner as $Z = 1$, the dynamic column generation with $Z = 5$ and $W = 7$ (CGH(5,7)) obtained the best overall performance. In addition, it obtained the smallest gap for about 89% of the problems and solved each problem with a gap no worse than $2^{1.7}$ of the best gap obtained by the other methods. On the other hand, the dynamic column generation for $(Z,W) = (5,1)$ has a similar performance profile to hybrid heuristic for $(Z,W) = (5,5)$. While that hybrid heuristic for $(Z,W) = (5,7)$ dominates dynamic column generation for $(Z,W) = (5,1)$ for factors smaller than $2^{5.5}$. The results in Table 6 show that for $Z = 5$, the gaps are smaller than $Z = 1$, but the behavior is similar.

Fig. 3 depicts the performance profile for hybrid heuristic ($H(Z,W)$) and the dynamic column generation (CGH(Z,W)) for $Z = 7$ and $W = 1,5,7$. The perfomance of methods for $Z = 7$ is similar to $Z = 5$. The main difference is in the perfomance for the dynamic column generation for $W = 7$, which solved each one of the problems with a gap 2^x, such as x is close to zero, of the best gap obtained by the other methods. Table 7 shows the gaps for both methods considering $Z = 7$ and $W = 1,5,7$. Note that, CGH(7,7) was worse than CGH(7,5) in the worst case, because the columns are different, and the columns that improve the objective function value were not generated for some instances. Compared to Table 6, when we increase $Z = 5$ to $Z = 7$, the average gap of the hybrid heuristic decreases more than the dynamic column generation.

When we analyze all the results, the greater Z and W are, the better the gap is. Moreover, increasing W influences the gap more than increasing Z. Despite the fact that hybrid heuristic provides gaps higher than the dynamic column generation, the run time is lower and more stable. The average standard deviation of the run time for the hybrid heuristic is 0.04 of a second, while for the dynamic column generation it is 3.89 seconds. This occurs, because during the column generation many columns can be added and the restricted master problem is finally solved by setting the variable as an integer. In addition, when we increase Z and W the integer solution for the restricted master problem from the dynamic column generation can be worse, because the columns are different, then the method cannot generate the column which improves the objective function value. The average iteration number is about 5.15. Comparing the dynamic column generation with the integer linear formulation (13)–(18), despite the fact that the dynamic column generation does not find all the optimal solutions, its run time is much smaller than the time the CPLEX takes to solve the problem (13)–(18). For instance, for $Z = 7$ and $W = 7$, the gap for the column generation with the hybrid heuristic is 0.25% and the average run time is 14.75 seconds, while the integer linear formulation is 87.28 seconds. For the same instances, some tests were carried out with $S_k \neq 0$, and the computational results, in terms of run times and gaps, are very close to those ones with $S_k = 0$, but the CPLEX takes twice as long to solve problem (13)–(18).

6. Conclusions

This paper has presented two new formulations for the constrained compartmentalized knapsack problem. One formulation is a decomposition in the master problem and subproblems for a known non-linear formulation, and the other is an integer linear formulation. To solve the decomposed formulation, we combine heuristics from the literature. The linear relaxation of the master problem was solved by column generation. Furthermore, we combined the column generation method with the heuristic. After solving the relaxed master problem, we found an integer solution by defining the variables as integers. The Hybrid Heuristic proposed was able to find good solutions (on average about 2% when $Z = 7$ and $W = 7$) within low running times. If the heuristic is included in the column generation, the average gap is reduced by 10 times, but the average running time increases by about 10 times. The integer linear formulation provides most of the optimal solutions when we use the Cplex package, but it can take a very high running time for several instances.

Acknowledgments

The authors thank Pedro A. Munari Jr. for providing the software to plot the performance profile, available on: http://www.lot.icmc.usp.br/webfacil/publico/File/perfis_v1.2.xls. The authors would like also to thank the anonymous referees for their constructive comments, which have significantly improved the presentation of this paper. This research was supported by Fapesp and CNPq.

References

Carvalho, J.M.V., Rodrigues, A.J.G., 1994. A computer-based interactive approach to a 2-stage cutting stock problem. INFOR: Information Systems and Operational Research 32 (4), 243–252.
Carvalho, J.M.V., Rodrigues, A.J.G., 1995. An lp-based approach to a 2-stage cutting stock problem. European Journal of Operational Research 84 (3), 580–589.
Correia, M.H., Oliveira, J.F., Ferreira, J.S., 2004. Reel and sheet cutting at a paper mill. Computers & Operations Research 31 (8), 1223–1243.
Dantzig, G.B., Wolfe, P., 1960. Decomposition principle for linear-programs. Operations Research 8 (1), 101–111.
Dolan, E.D., Moré, J.J., 2002. Benchmarking optimization software with performance profiles. Mathematical Programming 91 (2), 201–213.
Ferreira, J.S., Neves, M.A., Castro, P.F., 1990. A 2-phase roll cutting problem. European Journal of Operational Research 44 (2), 185–196.
Gilmore, P.C., Gomory, R.E., 1961. A linear-programming approach to the cutting-stock problem. Operations Research 9 (6), 849–859.
Gilmore, P.C., Gomory, R.E., 1963. A linear-programming approach to the cutting stock problem – part 2. Operations Research 11 (6), 863–888.
Haessler, R.W., 1971. Heuristic programming solution to a nonlinear cutting stock problem. Management Science 17 (12), 793–802.
Haessler, R.W., 1979. Solving the 2-stage cutting stock problem. Omega-International Journal Of Management Science 7 (2), 145–151.
Hoto, R., Spolador, F., Marques, F., 2005. Resolvendo mochilas compartimentadas restritas (In Portuguese). In: XXXVII Brazilian Symposium on Operational Research. Gramado, Rio Grande do Sul, Brazil.
Hoto, R., Arenales, M.N., Maculan, N., 2007. The one dimensional compartmentalised knapsack problem: A case study. European Journal of Operational Research 183 (3), 1183–1195.
Hoto, R., Maculan, N., Borssoi, A., 2010. A study of the compartmentalized knapsack problem with additional restrictions (In Portuguese). IEEE Latin America Transactions 8 (3), 269–274.
Leão, A.A.S. 2009. MTtodos geração de colunas para problemas de corte em duas fazes (In Portuguese). Master's thesis; Instituto de Ciências Matemáticas e de Computação - Universidade de São Paulo.
Manne, A.S., 1958. Programming of economic lot sizes. Management Science 4 (2), 115–135.
Marques, F.P., Arenales, M.N., 2007. The constrained compartmentalised knapsack problem. Computers & Operations Research 34 (7), 2109–2129.

Özpeynirci, S., Azizoglu, M., 2010. A lagrangean relaxation based approach for the capacity allocation problem in flexible manufacturing systems. Journal of the Operational Research Society 61 (5), 872–877.

Pisinger, D., 1999. Core problems in knapsack algorithms. Operations Research 47 (4), 570–575.

Shachnai, H., Tamir, T., 2001a. On two class-constrained versions of the multiple knapsack problem. Algorithmica 29 (3), 442–467.

Shachnai, H., Tamir, T., 2001b. Polynomial time approximation schemes for class-constrained packing problems. Journal of Scheduling 4 (6), 313–338.

Shachnai, H., Tamir, T., 2004. Tight bounds for online class-constrained packing. Theoretical Computer Science 321 (1), 103–123.

Wäscher, G., Haußner, H., Schumann, H., 2007. An improved typology of cutting and packing problems. European Journal of Operational Research 83, 1109–1130.

Xavier, E., Miyazawa, F., 2006. Approximation schemes for knapsack problems with shelf divisions. Theoretical Computer Science 352 (1-3), 71–84.

Xavier, C.E., Miyazawa, F.K., 2008a. The class constrained bin packing problem with applications to video-on-demand. Theoretical Computer Science 393 (1–3), 240–259.

Xavier, C.E., Miyazawa, F.K., 2008b. A one-dimensional bin packing problem with shelf divisions. Discrete Applied Mathematics 156 (7), 1083–1096.

Xavier, C.E., Miyazawa, F.K., 2009. A note on dual approximation algorithms for class constrained bin packing problems. Rairo-Theoretical Informatics and Applications 43 (2), :239–248.

Zak, E.J., 2002a. Modeling multistage cutting stock problems. European Journal of Operational Research 141 (2), 313–327.

Zak, E.J., 2002b. Row and column generation technique for a multistage cutting stock problem. Computers & Operations Research 29 (9), 1143–1156.