UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

The Constrained Compartmentalised Knapsack Problem

Fabiano do Prado Marques
Marcos Nereu Arenales

Nº 86

NOTAS

São Carlos - SP

UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação
ISSN 0103-2577

The Constrained Compartmentalised Knapsack Problem

Fabiano do Prado Marques
Marcos Nereu Arenales

Nº 86

NOTAS

Série Computação

São Carlos – SP
Abr./2005

The Constrained Compartmentalised Knapsack Problem

Fabiano do Prado Marques
Marcos Nereu Arenales

Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
fp_marques@uol.com.br, arenales@icmc.usp.br

ABSTRACT

The Constrained Compartmentalised Knapsack Problem is an extension of the classical Integer Constrained Knapsack Problem which can be stated as the following hypothetical situation: a climber must load his/her knapsack with a number of items. For each item a weight, a utility value and an upper bound are given. 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 by items from the same class). The compartments have flexible capacities which 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 to load it and, in addition, each new compartment introduces a loss of capacity of the original knapsack. The Constrained Compartmentalised Knapsack Problem consists of determining suitable capacities of each compartment and how these compartments should be loaded, such that the total items inside all compartments does not exceed the upper bound given. The objective is to maximise the total utility value minus the cost of the compartments. This kind of problem arises in practice, such as in the cutting of steel or paper reels. The problem is modeled as an integer non-linear optimisation problem and for which some heuristic methods are designed. Finally, computational experiments are given to analyse the methods.
Keywords: Knapsack Problems, Cutting and Packing Problems, Integer and Combinatorial Optimisation.

RESUMO

O Problema da mochila compartimentado restrito é uma extensão do clássico problema da mochila inteiro restrito, o qual pode ser enunciado conforme a seguinte situação hipotética: um alpinista deve carregar sua mochila com vários itens de seu interesse. A cada item são associados um peso, um valor de utilidade e um limite superior, o qual não pode ser ultrapassado. Entretanto, os itens são de classes diferentes (alimento, medicamento, utensílio, etc) de modo que devem ser arranjados em compartimentos separados dentro da mochila (cada compartimento é por si uma mochila a ser carregada). Os compartimentos têm capacidades flexíveis, as quais são limitadas superior e inferiormente. Cada compartimento tem também um custo fixo de ser incluído na mochila original, que depende da classe dos itens a ser carregado e, além disso, cada novo compartimento introduz uma perda de capacidade da mochila original. O problema da mochila compartimentada restrita consiste em determinar capacidades adequadas para os compartimentos, bem como carregá-los de modo que o total de itens dentro de todos os compartimentos não exceda o limitante superior dado e que o valor de utilidade total seja maximizado, descontando-se os custos da inclusão dos compartimentos. Este problema surge em aplicações praticas, tais como no corte de bobinas de aço e papel. O problema é modelado como um problema de otimização não-linear inteiro e alguns métodos heurísticos foram desenvolvidos. Finalmente, apresentamos uma análise computacional do desempenho das heurísticas.
Palavras-chaves: Problemas da mochila, problemas de corte e empacotamento, otimização inteira e combinatória.

1 Introduction

Cutting large objects (e.g., bobbins, plates, boxes) to produce a specified quantity of smaller items, or packing smaller items into larger spaces are identical problems, considering that an item cut from a position can be seen as occupying such a position. Therefore, such problems are referred to as *Cutting and Packing Problems.* Of course, these types of problems can represent abstract arrangements rather than concrete items loaded in objects, such as jobs, resources, projects, etc. (Dyckhoff and Finke, 1992).

The number of possible arrangements of items on an object (each possible arrangement is called *cutting or packing pattern*) is, in general, a very large number, so that the effort of enumerating all of them is in the computational point of view, an infeasible task. An objective function can be defined to measure, for example, waste in case of cutting problems or non-occupied space in packing problems, as well as being able to measure costs, profit, etc. A cutting and packing problem consists of determining a cutting pattern that minimises a given objective function.

In this category of problems are various classical operational research problems, such as knapsack problems and bin-packing, which are in general, NP-complete (Garey and Johnson, 1979).

Various surveys on Cutting and Packing Problems have been published in the last two decades, such as Hinxman (1980), Dyckhoff *et al.* (1985), Dyckhoff (1990), Martello and Toth (1990), Dowsland and Dowsland (1992), Dyckhoff and Finke (1992), Sweeney and Parternoster (1992), Wäscher and Gau (1996), Dyckhoff *et al.* (1997), Lodi *et al.* (2002) as well as special issues of OR journals such as Dyckhoff and Wäscher (1990), Bischoff and Wäscher (1995), Martello (1994a, 1994b), Arenales *et al.* (1999), Hifi (2002), and Oliveira and Wäscher (2005).

A number of general methods of OR were specialised for Cutting and Packing Problems, which belong to the class of combinatorial optimisation problems, such as branch and bound methods, the column generation technique (the simplex method), dynamic programming, Lagrangian relaxation, graph search, and metaheuristics.

In the literature, there is a number of variations for Cutting and Packing Problems, and Dyckhoff (1990) proposed a categorisation for them based on some few characteristics and Wäscher *et al.* (2004) improved it. A first and more basic characteristic concerns the relevant dimensions of the cutting process. Therefore, one can refer to a one-dimensional cutting problem if only one dimensional (width, for example) is relevant in the cutting process as, for example, the cutting of reels or bars (of paper, clothes, plastics, steel, films, etc). The classic knapsack problem can be seen as a one-dimensional cutting problem. Other two-dimensional cutting problems, for which two-dimensions (width and length) are relevant have a number of applications, such as in cutting wooden plates, steel plates, glass, textile, etc. Three or more dimensional cutting problems can also be defined (Dyckhoff, 1990, Dyckhoff and Finke, 1992). The cutting problem focused on in this paper is one-dimensional.

If a demand of items should be met by cutting large objects then the problem is referred to as a *Cutting Stock Problem* which may modeled as an integer linear optimisation problem with a very large number of variables (one for each cutting pattern). Gilmore and Gomory (1961, 1963, 1965) successfully used a column generation technique (simplex method) by relaxing the integer condition on the variables, which provides, in general, good approximate solutions and it is possible to design effective rounding heuristics (Wascher and Gau, 1996, Poldi and Arenales, 2005).

Solution methods for cutting stock problems (either based on column generation or greedy heuristics, see Hinxman, 1980, Wäscher and Gau, 1996, Poldi and Arenales, 2005) depend, in general, on solving a fundamental *cutting sub-problem* which consists of determining the best cutting pattern for a single object in stock. In this sub-problem there is no need to meet the demand (in the case where the demand is low, one should take it into account, otherwise a single object cut could exceed demand, therefore a constrained cutting problem arises). This paper focuses on this

sub-problem of finding the best cutting pattern for a single object, where the quantities of items are limited.

The Compartmentalised Knapsack Problem models a hypothetical situation described in the abstract which has important practical applications. For example, in steel or paper roll cuttings. For the steel roll cutting problem, 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 sub-rolls, which have, of course, the same thicknesses as the original roll. Then, each thickness of the sub-roll is reduced to meet the item thicknesses of a class. Finally, the reduced thicknesses of the sub-rolls are cut into items. Another application arises from the paper industry, where many ordered items are rectangles, say, $l_1 x w_1$, $l_2 x w_2$, ..., which are cut from rolls of width, say, W. Instead of cutting sub-rolls of ordered widths w_1, w_2, ... one first cut intermediate rolls of widths, say, $\overline{w}_1$, $\overline{w}_2$, ..., and then each intermediate roll is cut into ordered items having the same ordered length, *i.e.*, if $l_1=l_2$ then items type 1 and 2 can be combined in the same intermediate roll (this is because transversal knives are on a cylinder that cuts the lengths of items 1 and 2 simultaneously. In practice, to avoid the *compartmentalisation* difficulty, some authors have reduced the intermediate rolls to cut only one type of item.

Some authors have avoided dealing explicitly with the compartmentalised knapsack problem by using simple heuristics. For example, Carvalho and Rodrigues (1994) reduced the solution space to the homogeneous solutions for each intermediate roll obtained after the first cut (intermediate rolls play the role of compartments). Ferreira *et al.* (1990) used a greedy heuristic to generate a set of cutting patterns without compartmentalisation constraints and then they tried to build compartments (*i.e.*, putting items together of the same class), discarding the ones that had failed. Later Correia *et al.* (2004) used a similar procedure to study a production planning problem in a Portuguese paper mill and a cutting optimisation problem of reels and sheets.

Johnston and Khan (1995) studied a related problem to the Compartmentalised Knapsack Problem, called the Nested Knapsack Problem. This problem arises, for example, in the manufacturing of optical fibre cables. However, the sizes of the compartments are given beforehand, and no loss of capacity is incurred due to the inclusion of compartments. Shachnai and Tamir (2001) studied a kind of multiple knapsack problem, where feasible loading is constrained by limiting the number of different types (colors) of items. Although this problem can be seen as building compartments (one for each color) inside the knapsack, it differs from our problem since the compartments are not bounded, nor is there a loss of capacity to include a new compartment.

Hoto *et al.* (2002) studied the unconstrained version of the compartmentalised knapasck problem. They proposed a mathematical modeling and two methods: an exact and a heuristic. Later, Hoto *et al.* (2004) also proposed a simple heuristic for the constrained case.

Zak (2002) studied a related problem where a sequence of stages in the cutting process causes the roll in stock having to be cut into intermediate rolls before the ordered items are finally cut. This process produces kinds of compartments (intermediate rolls), but there is not a partition of the items into classes (e.g., food, medicine, etc.) as any item can be cut from any intermediate roll.

Several knapsack problems have been studied in the last decades (see Martelo and Toth (1990) and Lin (1998) for surveys), however it is worth noting that the compartmentalised knapsack problem has been ignored.

2. The Constrained Compartmentalised Knapsack Problem

In this section we formalised the constrained compartmentalised knapsack problem and presented an integer non-linear optimisation model. Heuristic methods are given in section 3.

2.1 Problem definition

A climber wants to load his/her knapsack with m available items. For each item i weighing l_i the climber gives a utility value v_i. Let L be the maximum weight that the climber can bear. Furthermore, the items are grouped into classes (e.g., class 1: foods, class 2: medicine, class 3: utensils, class 4: clothes, etc.) which must be in different compartments in the knapsack (*i.e.*, each compartment can contain only items from the same class). The capacities of the compartments are flexible, but they are lower and upper bounded. These bounds can depend on the classes, say: L^k_{min} and L^k_{max} (*i.e.*, the total weight of items from class k loaded in a compartment have to be greater than or equal to L^k_{min} and less than or equal to L^k_{max}). The climber can have bigger compartments for food than clothes, or even have various compartments for food, but the total number of items type i cannot exceed an upper bound, say, d_i, $i=1,...,m$. (If no upper bound is given or d_i is very large then the problem is called *unconstrained*). For each compartment, a cost c_k is also given, in the case of the compartment being loaded with items from class k. Moreover, each compartment loaded with items from class k causes a fixed loss of capacity, say, S_k (*e.g.*, if three compartments are built with items from class k, then the knapsack capacity becomes $L-3S_k$, instead of L).

The Constrained Compartmentalised Knapsack Problem consists of determining suitable capacities for compartments and how these compartments should be loaded, in such a way that the total items inside all compartments does not exceed the upper bound given. The objective is to maximise the total utility value (i.e., the sum of the utility values of each chosen item), minus the cost of the compartments. Note that the classical integer knapsack problem is a special case of the compartmentalised one, if just one class of items is given.

2.2 A Mathematical Model

In this section we provide a mathematical model for the Constrained Compartmentalised Knapsack Problem.

Data:

- $M = \{1,...,m\}$: the set of types of items;
- l_i: the weight (Kg) of item type i, $i=1,...,m$;
- v_i: the utility value of item type i, $i=1,...,m$;
- d_i: the maximum number of items type i allowed in the knapsack, $i=1,...,m$;
- K: the total number of classes;
- A_k: the set that defines the class k, $k=1,...,K$ ($M = A_1 \cup A_2 ... \cup A_K$, $A_i \cap A_j = \varnothing$, for $i \neq j$);
- c_k: the cost of including a compartment loaded with items of class k, where $c_k \geq 0$, $k=1,...,K$;
- L: the capacity (Kg) of the knapsack;
- L^k_{min}: the minimum capacity (Kg) of compartments which are loaded with items from class k;
- L^k_{max}: the maximum capacity (Kg) of compartments which are loaded with items from class k; ($L^k_{min} < L^k_{max} < L$);
- S_k: the waste (Kg) due to the inclusion of a new compartment loaded with items from class k in the knapsack.

Also, let N_k be the total number of possible compartments for class k, $k=1,...,K$ (this is a very large number, just used to state the mathematical model, but only a few compartments will be necessary to solve the problem).

Variables:

- α_{ijk}: the number of items of type i, from class k, in the compartment of type j, i=1,...,m, k=1,...,K and j=1,...,N_k;
- β_{jk}: the number of times that a compartment of type j loaded with items from class k is repeated in the knapsack, k=1,...,K and j=1,...,N_k.

Therefore, the j^{th} compartment loaded with items from class k has:

- its capacity given by:

$$L_{jk} = \sum_{i \in A_k} l_i \alpha_{ijk} + S_k, \; k = 1,...,K \text{ and } j = 1,...,N_k. \quad (1)$$

- and its utility value given by:

$$V_{jk} = \sum_{i \in A_k} v_i \alpha_{ijk}, \; k = 1,...,K \text{ and } j = 1,...,N_k. \quad (2)$$

A mathematical model for the Compartmentalised Knapsack Problem can be written as:

$$\text{Maximise } \sum_{k=1}^{K} \sum_{j=1}^{N_k} (V_{jk} - c_k)\beta_{jk} \quad (3.1)$$

$$\text{Subject to: } V_{jk} = \sum_{i \in A_k} v_i \alpha_{ijk}, \; k = 1,...,K \text{ and } j = 1,...,N_k \quad (3.2)$$

$$L_{jk} = \sum_{i \in A_k} l_i \alpha_{ijk} + S_k, \; k = 1,...,K \text{ and } j = 1,...,N_k \quad (3.3)$$

$$L^k_{min} \le L_{jk} \le L^k_{max}, \; k = 1,...,K \text{ and } j = 1,...,N_k \quad (3.4)$$

$$\sum_{j=1}^{N_k} \alpha_{ijk} \beta_{jk} \le d_i, \; i \in A_k, k = 1,..., K \quad (3.5)$$

$$\sum_{k=1}^{K} \sum_{j=1}^{N_k} L_{jk} \beta_{jk} \le L \quad (3.6)$$

$$\alpha_{ijk} \ge 0, \text{ integer}, \; i = 1,...,m, \; k = 1,...,K \text{ and } j = 1,...,N_k \quad (3.7)$$

$$\beta_{jk} \ge 0, \text{ integer}, \; k = 1,...,K \text{ and } j = 1,...,N_k. \quad (3.8)$$

The objective function (3.1), to be maximised, is the total utility value minus the costs of including compartments; the constraints (3.2) and (3.3) give, respectively, the utility value and the capacity of each compartment (obviously, these constraints can be eliminated by using them inside the others); the constraints (3.4) define the bounds to the compartments; the constraints (3.5) are to limit the number of items in the knapsack (this constraint is necessary for no oversupply if d_i is low), and finally the constraint (3.6) gives the knapsack capacity. Note that (3.1-3.8) is an integer non-linear optimisation model with a very high number of variables.

We may have a particular class of items, called *free items,* which can be included into the knapsack without the need of building a new compartment. In the heuristic methods, we define the last class K as the class of free items. Therefore, for k=K, the constraint (3.4) is unnecessary (or,

$L^K_{min} = 0$ and $L^K_{max} = L$ so that it is redundant) and $c_K = 0$. Furthermore, for the class of free items, we have $S_K = 0$ since no compartment is needed.

2.3 An upper bound to the Compartmentalised Knapsack Problem

In order to obtain upper bounds to a maximization problem, one can relax the original formulation by different ways and solve easier problems. For example, one may ignore the compartmentalisation needs and solve a classical knapsack problem. However, this simple relaxation gives a poor assessment of the problem. A more useful and nontrivial relaxation is given now, with which one can better assess the performance of heuristics described in section 3.

Proposition. Let x_i be the number of items type i in the knapsack, $i = 1,...,m$, and y_k be the number of compartments loaded with items from class k, $k = 1,...,K$. Then the following problem is a relaxation to the constrained compartmentalised knapsack problem:

$$\text{Maximise } \sum_{i=1}^{m} v_i x_i - \sum_{k=1}^{K} c_k y_k \qquad (4.1)$$

$$\text{Subject to: } \sum_{i=1}^{m} l_i x_i \le L - (\sum_{k=1}^{K} S_k y_k) \qquad (4.2)$$

$$y_k . L^k_{min} \le \sum_{i \in A_k} l_i x_i + S_k y_k \le y_k . L^k_{max}, \quad k=1,...,K \qquad (4.3)$$

$$0 \le x_i \le d_i, \text{integer}, \quad i=1,...,m \qquad (4.4)$$

$$y_k \ge 0, \text{ integer}, \quad k=1,...,K. \qquad (4.5)$$

To prove the proposition is enough to show that any feasible solution to problem (3.1-3.8) is also a feasible solution to problem (4.1-4.5), and for any feasible solution to problem (3.1-3.8) the objective function value (4.1) is greater than or equals the objective function value (3.1).

Therefore, consider a feasible solution to problem (3.1-3.8): (α_{ijk}, β_{jk}). Then, by definition, the total number of items type i in the knapsack and the total number of compartments loaded with items from class k (which are the the variables to problem (4.1-4.5) are given by:

$$x_i = \sum_{j}^{N_k} \alpha_{ijk} \beta_{jk}, \text{ for } i \in A_k, k = 1,...,K \qquad (5)$$

$$y_k = \sum_{j=1}^{N_k} \beta_{jk}, \text{ for } k = 1,...,K. \qquad (6)$$

Therefore, the objective function (3.1), by substituting (3.2), is:

$$\sum_{k=1}^{K} \sum_{j=1}^{N_k} (\sum_{i \in A_k} v_i \alpha_{ijk} - c_k) \beta_{jk} = \sum_{k=1}^{K} \sum_{i \in A_k} v_i (\sum_{j=1}^{N_k} \alpha_{ijk} \beta_{jk}) - \sum_{k=1}^{K} c_k \sum_{j=1}^{N_k} \beta_{jk}$$

Since $M=\{1,...,m\} = A_1 \cup A_2 ... \cup A_K$, where $A_i \cap A_j = \varnothing$, for $i \neq j$, and using (5) and (6), it follows the objective function (4.1). That is, the objective functions (3.1) and (4.1) have always the same value, since the variables are related according to (5) and (6).

The constraints (3.4), using (3.3), are: $L^k_{\min} \le \sum_{i \in A_k} l_i \alpha_{ijk} + S_k \le L^k_{\max}$, $k = 1,...,K$. By multiplying each side by $\sum_{j=1}^{N_k} \beta_{jk}$, we get: $L^k_{\min} \sum_{j=1}^{N_k} \beta_{jk} \le \sum_{i \in A_k} l_i \sum_{j=1}^{N_k} \alpha_{ijk} \beta_{jk} + S_k \sum_{j=1}^{N_k} \beta_{jk} \le L^k_{\max} \sum_{j=1}^{N_k} \beta_{jk}$. Then, using (5) and (6), it follows: $L^k_{\min} y_k \le \sum_{i \in A_k} l_i x_i + S_k y_k \le L^k_{\max} y_k$ which means that constraints in (4.3) are met.

From (3.6), substituting (3.3), it follows: $\sum_{k=1}^{K}\sum_{j=1}^{N_k}(\sum_{i\in A_k} l_i\alpha_{ijk}+S_k)\beta_{jk}\le L$, which is equivalent to:

$$\sum_{k=1}^{K}\sum_{i\in A_k} l_i\sum_{j=1}^{N_k}\alpha_{ijk}\beta_{jk}+\sum_{k=1}^{K}S_k\sum_{j=1}^{N_k}\beta_{jk}\le L.$$

By using (5) and (6), it follows that: $\sum_{i=1}^{m} l_i x_i+\sum_{k=1}^{K} S_k y_k \le L$, which means that the constraint (4.2) is met. Finally, (4.4) and (4.5) follow straightforward from (3.5), (3.7) and (3.8), with which the proposition is proved.

3. Heuristics for the Constrained Compartmentalised Knapsack Problem

In this section we proposed four heuristics to solve the Constrained Compartmentalised Knapsack Problem. The heuristics bet on solutions with compartments as large as possible, because the cost of including a new compartment is high in practice. For example, in the steel roll cutting problem this cost is related to reducing the thicknesses of intermediate rolls to meet the item's thicknesses. In section 4, we will analyse the computational performance of these heuristics.

3.1 The Decomposition Heuristic

This heuristic consists of two phases: In the first phase, for each class (except class K of free items), solve *(K-1)* Knapsack Problems with capacities equal to L_{max}, which provide the most valuable compartments, one for each class. In the second phase, another Knapsack Problem is solved by considering the compartments obtained in the first phase as *super-items* (*i.e.*, a combination of items) together with the free items.

Therefore, a feasible solution to problem (3.1-3.8) is built with the following characteristics:

- In order to maximise the objective function (3.1), it is convenient to build V_{jk} as large as possible. Therefore, for each class k, we build only the best compartment (then $N_k = 1$, so that j can be omitted) by maximising (3.2) subject to (3.3), (3.4) and (3.7) (see problem (7.1-7.3)). The value of α_{ik} (j omitted) is then determined;
- With $N_k = 1$ (*i.e.*, only a compartment for class) and α_{ik} already determined: one determines L_k (the size of compartment k, see (3.3), with index j omitted). The value β_k is determined as to optimise (3.1) subject to the other constraints of problem (3.1-3.8) (see problem (8.1-8.5).

The algorithm in the following section summarises this heuristic.

Algorithm – Decomposition Heuristic

Step 1: Determine the most valuable compartment of class k, k=1, ...,*K-1*, by solving the following Knapsack Problem of capacity L_{max}:

$$V_k = \text{Maximum} \sum_{i\in A_k} v_i\alpha_{ik} \qquad (7.1)$$

$$\text{Subject to: } \sum_{i\in A_k} l_i\alpha_{ik}+S_k \le L^{k}_{\max} \qquad (7.2)$$

$$0\le\alpha_{ik}\le d_i \text{ and integer}, \forall\, i\in A_k, k=1,...,K-1. \qquad (7.3)$$

Let $L_k=\sum_{i\in A_k} l_i\alpha_{ik}+S_k$ be the size of compartment k. (note that α_{ik} are fixed in this step).

Step 2: Solve the following Knapsack Problem to pack the compartments obtained in step 1 together with free items:

- Let β_k be the number of times that the compartments with items of class k (just one compartment is created in step 1) are repeated in the knapsack,
- Let γ_k be the number of free items type k packed in the knapsack.

$$\text{Maximise} \quad \sum_{k=1}^{K-1}(V_k - c_k)\beta_k + \sum_{k\in A_K} v_k\gamma_k \tag{8.1}$$

$$\text{Subject to:} \sum_{k=1}^{K-1} L_k\beta_k + \sum_{k\in A_K} l_k\gamma_k \le L \tag{8.2}$$

$$\alpha_{ik}\beta_k \le d_i,\ i \in A_k,\ k = 1,...,K-1 \tag{8.3}$$

$$\gamma_k \le d_k,\ k \in A_K, \tag{8.4}$$

$$\alpha_{ik}, \gamma_k \ge 0 \text{ and integer},\ i \in A_k, k = 1,...,K-1, k \in A_K. \tag{8.5}$$

3.2 The Best Compartment Heuristic

In this heuristic, similar to the previous one, the best compartment, one for each class, is determined in step 1 (i.e., the problem (7.1-7.3) is solved for each class k). Therefore, we have:

$$L_k' = \begin{cases} (L_k + S_k), & k = 1,...,K-1 \\ l_i, & k = (K-1)+i,\ i = 1,...,|A_K| \end{cases}$$
$$V_k' = \begin{cases} V_k - c_k, & k = 1,...,K-1 \\ v_i,\ \forall i \in A_K, & k = (K-1)+i,\ i = 1,...,|A_K| \end{cases} \tag{9}$$

Then, choose the compartment of the best unit value: $V_r'/L_r' = \max\{V_k'/L_k', \forall k\}$ to be packed into the knapsack as much as possible, without exceeding the upper bound on the number of items. Data are updated ($d_i \leftarrow d_i - \alpha_{ir}$, and the remainder capacity: $L \leftarrow L - L_r'$) and the process is repeated until the available knapsack capacity is too small. Therefore, the problem (8.1-8.5) is avoided. Note that now we might have different compartments for the same class, since when the process is repeated, another compartment might be built for the same class, something not considered by the decomposition heuristic, which allows only the repetition of the best compartment obtained as a solution to the problem (7.1-7.3). The algorithm in the following section summarises the heuristic.

Algorithm – Best Compartment Heuristic

Step 1: $L\_util \leftarrow L$; {*L_util* keeps the available capacity in the knapsack}

Step 2: For $k = 1$ to K-1, do:

2.1 If ($L\_util \ge L_{max}$)

then *L_compartment* = $L^k_{\max}$ {in this case, a maximum compartment is created}

else If ($L\_util < L^k_{\min}$)

then *L_compartment*= *L_util* and only free items in (10.1-10.3) are considered, {no more compartments can be loaded}

else (i.e., $L_{min} \le L\_util < L^k_{\max}$) *L_compartment* = *L_util*

Remark. If *L_compartment* < $L^k_{\min}$ or if $d_i = 0$, $\forall i \in A_k$, then $V_k^{'} = 0$, k=1,...,K-1, only free items to be packed remain.

2.2 Choose the best compartment with class k by solving the following Knapsack Problem with capacity equals *L_compartment*:

$$V_k^{'} = \text{Maximum} \sum_{i \in A_k} v_i \alpha_{ik} \qquad (10.1)$$

$$\text{Subject to: } \sum_{i \in A_k} l_i \alpha_{ik} + S_k \le L\_compartment \qquad (10.2)$$

$$0 \le \alpha_{ik} \le d_i \text{ and integer, } i \in A_k, k = 1,...,K\text{-}1. \qquad (10.3)$$

Step 3:

3.1 Choose r such that: $\text{Max}\{\frac{V_k^{'} - c_k}{L_k^{'}},\ k = 1,\cdots,K-1;\ \frac{v_k}{l_k},\ k \in A_K\}$

3.2 Updating:
If r in step (3.1) corresponds to a compartment (remember it might correspond to a free item)

then $L\_util \leftarrow L\_util - L_r^{'}$ {i.e., a compartment is allocated}
$d_i \leftarrow d_i - \alpha_{ir}$

else $L\_util \leftarrow L\_util - l_r$ {i.e., a free item is allocated}
$d_r \leftarrow d_r - 1$

3.3 If $L\_util \ge$ Min {Min { $L_k^{'}$, k=1,...,*K-1*}, Min{l_i, $\forall\ i \in A_K$}}, go to Step 2, otherwise, stop.

3.3 The z-Best Compartment Heuristic

Basically this heuristic consists of determining the best z compartments for each class, in other words, the best z solutions to the Knapsack Problem for each class. This leads to an integer linear optimisation problem, whose solution produces a cutting pattern to the compartmentalised knapsack problem. The heuristic is summarised as follows.

Algorithm – z-Best Compartment Heuristic

Step 1: Determine the best z solutions to the problem (7.1-7.3), for each class k. Let (α_{ijk}, $i \in A_k$) be the j^{th} solution, and V_{jk} be its corresponding objective function value, that is, $V_{1k} \ge V_{2k} \ge ... \ge V_{zk}$.

Step 2: Solve the following integer linear optimisation problem involving ((K-1)*z) compartments obtained in step 1 and the $|A_K|$ free items:

- Let β_{jk} be the number that the j^{th} compartment of class k is allocated in the knapsack.
- Let γ_k be the number of free items type k packed in the knapsack.

$$\text{Maximise} \sum_{k=1}^{K-1}\sum_{j=1}^{z} (V_{jk} - c_k)\beta_{jk} + \sum_{k\in A_K} v_k\gamma_k \qquad (12.1)$$

$$\text{Subject to:} \sum_{k=1}^{K-1}\sum_{j=1}^{z} L_{jk}\beta_{jk} + \sum_{k\in A_K} l_k\gamma_k \le L - S \qquad (12.2)$$

$$\sum_{j=1}^{z} \alpha_{ijk}\beta_{jk} \le d_i,\ i \in A_k \text{ and } k = 1,...,K\text{-}1. \qquad (12.3)$$

$$0 \le \gamma_k \le d_k,\ k \in A_K. \qquad (12.4)$$

$$\beta_{jk} \ge 0,\ \text{integer},\ k = 1,...,K-1 \text{ and } j = 1,...,z. \qquad (12.5)$$

where $L_{jk} = \sum_{i\in A_k} l_i\alpha_{ijk} + S_k, \quad k = 1,...,K-1.$

Remark: The Decomposition Heuristic is obtained with $z = 1$. However, in this case, the integer linear optimisation problem (12.1-12.5) becomes the Knapsack Problem (8.1-8.5). In order to solve the Knapsack Problem that arises in the heuristics, we used some modifications in the implicit enumeration method for Knapsack Problems from Gilmore and Gomory (1963). Such modifications are briefly discussed in the next section.

3.4 The *w*-capacity heuristic

In a first phase, this heuristic basically consists of determining the best compartment for w different capacities, for each class, which generates $w^*(K\text{-}1)$ compartments. In a second phase similar to the z-best compartment heuristic, it is necessary to solve an integer linear optimisation problem in order to determine a compartmentalised cutting pattern. The heuristic is summarised as follows:

Algorithm:

Step 1: Determine the best compartment for each one of w capacities for class k. For k = 1,...,K-1, do:

1.1 Let $L\_compartment = L^k_{max}$,

1.2 For j = 1,...,w, do:

If $L\_compartment \ge L^k_{min}$

then 1.2.1 Solve:

$$V_{jk} = \text{Maximum} \sum_{i\in A_k} v_i\alpha_{ijk} \qquad (13.1)$$

$$\text{Subject to:} \sum_{i\in A_k} l_i\alpha_{ijk} + S_k \le L\_compartment \qquad (13.2)$$

$$0 \le \alpha_{ijk} \le d_i, \text{integer}, i \in A_k. \qquad (13.3)$$

1.2.2 Let $L_{jk} = \sum_{i\in A_k} l_i\alpha_{ijk} + S_k$,

1.2.3 Let $L\_compartment = L_{jk} - 1$,

else $V_{jk} = 0$.

step 2: Solve the integer linear optimisation problem (12.1–12.5) by considering the ((*K*-1)**w*) compartments obtained in step 1 and the $|A_K|$ free items.

3.5 Solving Sub Problems embedded in the Heuristics

Gilmore and Gomory (1963) proposed a branch and bound method to solve Integer Knapsack Problems (no upper bound on the variables). This method is quite efficient in solving Knapsack Problems in the size that arises in a cutting problem context, that is, dozens of items (in fact, we have never found a case that involves hundreds or thousands of types of items in the cutting and packing practice). The implementation uses the *backtracking* strategy in such a way that the solution space is searched in a simple way by basically using an m-vector $\alpha = (\alpha_1\ \alpha_2 \ldots\ \alpha_k\ 0 \ldots 0)$, where α_i is the number of items type *i*, and *k* is the last item packed in the knapsack ($\alpha_k > 0$). This vector is sufficient to define a search tree node, so that the previous node is well identified by $\alpha' = (\alpha_1\ \alpha_2 \ldots\ \alpha_k\text{-}1\ 0 \ldots 0)$. Simple rules to include new items into the knapsack (hence, investigating new solutions or new nodes of the tree) are easily designed, given the simplicity of the constraints of the knapsack problem. Once a more valuable solution is identified, its value is kept, discarding the previous one. However, the knapsack problem that we need to solve embedded in each heuristic presents additional features that can be included in the searching process, they are: *i)* upper bound on the number of items in the knapsack, *ii)* determination of the *z*-best solutions and *iii)* additional constraints given by (12.3). The inclusion of additional constraints in the searching process of the implicit enumeration method of the Gilmore and Gomory method is widely used, but has scarcely been published. For example, Gilmore and Gomory (1963) also suggest that simple constraints, such as limitation on the number of knives, can be included in the searching process. Morabito and Garcia (1998) also included other nontrivial constraints, difficult to be mathematically described, but easy to be included in the searching process and which presented very good results. In the following section, a brief description of how to include the features of the knapsack problem embedded in the heuristics in sections 3.1 to 3.4 will be given.

3.5.1 The Constrained Knapsack Problem

The Constrained Knapsack Problem, which appears in (7.1-7.3), (8.1-8.5) and (10.1-10.3), were solved by simple modifications in the branch and bound algorithm of Gilmore and Gomory (1963). In order not to exceed the upper bound on the variable α_i, when its value is first calculated (which consists of a depth first search) one has to use the formula:

$$\alpha_i = \text{Minimum} \left\{ d_i,\ \left\lfloor \frac{\overline{L}}{l_i} \right\rfloor \right\},\quad i = 1,\ldots,m. \qquad (14)$$

where $\overline{L}$ is the remaining capacity which is updated in each node (remember that a node is well defined by an m-vector $\alpha = (\alpha_1\ \alpha_2 \ldots\ \alpha_k\ 0 \ldots 0)$ then $\overline{L} = L - \sum_{j=1}^{k} \alpha_j l_j$) and $\lfloor x \rfloor$ is the largest integer less or equals *x*.

The upper bound evaluation of the objective function is also simply modified to consider the upper bounds on the variables. The other steps are similar to the original version of the algorithm.

3.5.2 The *z*-best solutions

In order to identify the *z*-best solutions to the constrained knapsack problem, not only the optimal one, it is sufficient to keep the *z* solutions during the searching process which have the best

objective function values. Therefore, every time a new solution is determined, the corresponding objective function value, say V, is compared with the z values previously kept in memory (suppose: $V_1 \geq V_2 \geq \ldots \geq V_z$). If $V \leq V_z$ then the new solution is discarded, or otherwise V_z is discarded and V is kept. Of course, we may find $V_1 = V_2 = \ldots = V_z$, however the solutions will be different from each other, since the searching process avoids solution duplications. An alternative pseudo-polynomial method to determine the z-best solutions for the Knapsack Problem can be found in Yanasse *et al.* (2000).

3.5.3 The Problem (12.1-12.5)

Among the sub problems embedded in the heuristics, the problem (12.1-12.5) is the most difficult, because constraints (12.3) are added to the knapsack constraint (12.2). We proceed similarly to the constrained case, by introducing a modification in the search process in such a way as to guarantee the constraints (12.3) and (12.4). Since we are dealing with super-items (i.e., compartments that contain items of the same class) and considering that we have z super-items associated with the same class, it is necessary to check the searching process to verify if the maximum quantities d_i, i=1,...,m, are not exceeded (see constraints in (12.3)). In order to avoid such a violation, it is necessary to limit firstly the inclusion of the super-item (j, k), j = 1,...,z and k = 1,...,K-1 (i.e., the j^{th} compartment associated with class k obtained by the problem (11.1-11.3)) by:

$$\beta_{jk} \leq \text{Minimum}\left\{\left\lfloor \frac{d_i}{\alpha_{ijk}} \right\rfloor, \ \forall \ i \in A_k\right\}. \qquad (15)$$

This reduces the problem to the constrained case already discussed, but it is not sufficient. For example, if $\beta_{1k} > 0$ and $\beta_{2k} > 0$ (i.e., the first and second compartment with items from class k are loaded in the knapsack), and furthermore, $\alpha_{i1k} > 0$ and $\alpha_{i2k} > 0$ (i.e., item i is inside both compartments) then the total of items type i already included in the knapsack is given by: $\alpha_{i1k}\,\beta_{1k} + \alpha_{i2k}\,\beta_{2k}$ (see constraint (12.3)). Therefore, the quantity of a new super-item, for example type 3, is limited by:

$$\beta_{3k} \leq \left\lfloor \frac{\overline{d_i}}{\alpha_{i3k}} \right\rfloor, \qquad (16)$$

where $\overline{d_i} = d_i - \alpha_{i1k}\beta_{1k} - \alpha_{i2k}\beta_{2k}$ is the slack in constraint (12.3). The search is similar to section (3.5.1), but considering such a limitation on the number of compartments.

4. Computational Experiments

The methods in section 3 were implemented in Delphi 5 and run on a microcomputer Intel Celeron 650MHz with 320 MB RAM. Furthermore, the CPLEX 7.5 package was used to solve the integer linear program (4.1-4.5) to obtain the upper bounds.

A number of randomly generated instances were solved to analyse the performance of the heuristics. The instances were classified according to the following four features:

i) **Number of Classes: K**
1: few classes: $K \in [3,10]$.
2: several classes: $K \in [11,20]$.

ii) **Number of type of items per class: $|A_k|$, $k=1,...,K-1$**
3: few items: $|A_k| \in [1,6]$, $k = 1,...,K-1$.
4: several items: $|A_k| \in [7,15]$ $k = 1,...,K-1$.

iii) **Number of types of free items: $|A_K|$**
5: few free items: $|A_K| \in [1,7]$.
6: several free items: $|A_K| \in [8,20]$.

iv) **Upper bounds on the number of number of items: d_i, $i = 1,...,m$**
7: small bound: $d_i \in [1, 3]$, $i \in A_k$, $k = 1,...,K$.
8: large bound: $d_i \in [4, 15]$, $i \in A_k$, $k = 1,...,K$.

Therefore, 16 sets of instances were generated, e.g., see set A in Table 1 which consists of instances with the following features: **1**: few classes, **3**: few items per class, **5**: few free items and **7**: small upper bounds on the number of items. Moreover, 20 instances with such features were generated and the tables and the graphs in the following section show the average objective function values for the solutions obtained by each specificied method.

Furthermore, some other parameters were fixed (usually found in the practice of cutting steel rolls) or randomly generated as follows.

- **Given parameters:**
  1. Knapsack capacity: $L = 1188$ mm;
  2. Lower and upper bounds for the compartment lengths: $L^k_{\min} = 154$ mm and $L^k_{\max} = 456$ mm, $k=1,...,K-1$.
  3. Waste due to the inclusion of a new compartment: $S_k = 8$ mm, $k=1,...,K-1$.

- **Randomly generated parameters:**
  4. The length of the items: $l_i \in [20, 444]$, $i \in A_k$, $k = 1,..., K$.
  5. The utility value is given by: $v_i = l_i + \sigma$, where: $\sigma \in [0,l_i]$.
  6. The cost of including a new compartment with items from class k: $c_k \in [1,100]$.

4.1 Results

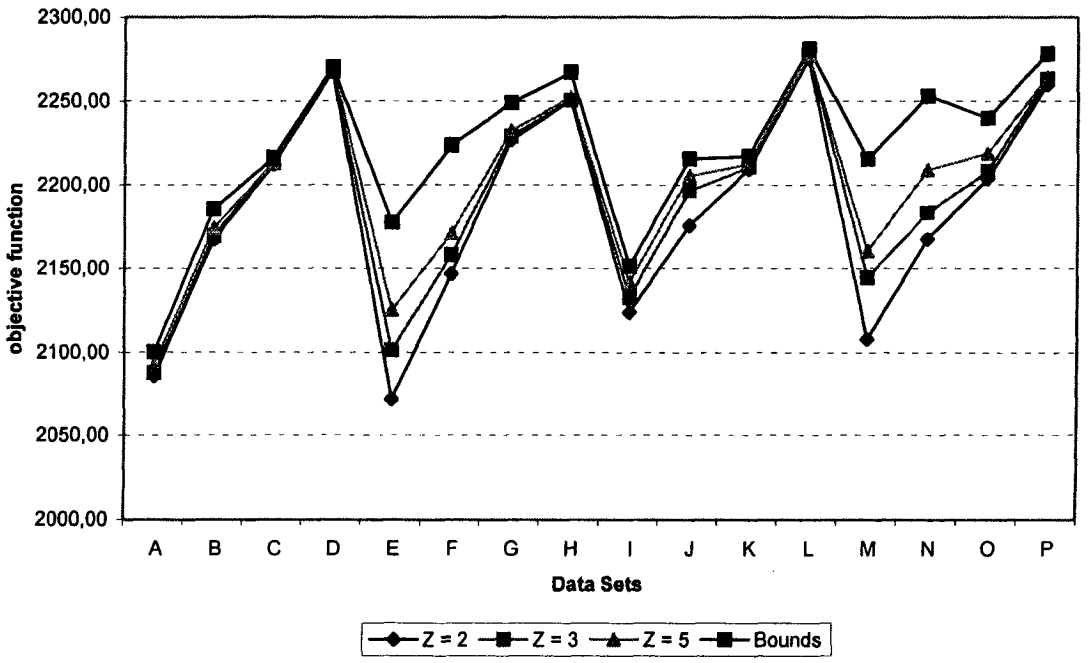
We start by showing the results obtained with the z-best compartments and w-capacities heuristics. In column "obj. func. z = 2" the averages of the objective function values (for 20 instances per set) of the solutions obtained by the 2-best compartment heuristic are shown. Similarly to "z = 3" and "z = 5". The column "upper bound" shows the average of the objective function values of solutions to the problem 4 (section 2.3) obtained by the CPLEX 7.5 package, and the columns "GAP" are the difference in percentage between the values obtained by the heuristics and the upper bound.

Table 1: Averages of the objective function values for z-best compartment heuristic (z = 2, 3 and 5)

Sets	Features	obj. func. z = 2	GAP z=2	obj. func. z= 3	GAP z=3	obj. func. z = 5	GAP z=5	Upper Bound
A	1,3,5,7	2086.17	0.69%	2088.10	0.60%	2092.33	0.40%	2100.63
B	1,3,5,8	2167.40	0.84%	2169.53	0.74%	2174.17	0.53%	2185.70
C	1,3,6,7	2212.43	0.18%	2213.30	0.14%	2213.30	0.14%	2216.43
D	1,3,6,8	2267.93	0.12%	2267.93	0.12%	2270.13	0.02%	2270.60
E	1,4,5,7	2072.50	4.83%	2101.77	3.49%	2125.67	2.39%	2177.70
F	1,4,5,8	2147.17	3.45%	2158.27	2.95%	2171.27	2.37%	2223.97
G	1,4,6,7	2226.93	0.99%	2229.20	0.89%	2232.50	0.74%	2249.17
H	1,4,6,8	2250.70	0.73%	2250.70	0.73%	2252.53	0.65%	2267.27
I	2,3,5,7	2124.30	1.26%	2133.33	0.84%	2141.90	0.45%	2151.50
J	2,3,5,8	2175.50	1.82%	2196.53	0.87%	2205.63	0.46%	2215.73
K	2,3,6,7	2209.47	0.36%	2210.77	0.30%	2212.77	0.21%	2217.47
L	2,3,6,8	2275.27	0.26%	2277.90	0.15%	2277.90	0.15%	2281.30
M	2,4,5,7	2108.10	4.86%	2144.77	3.21%	2160.50	2.50%	2215.80
N	2,4,5,8	2167.70	3.80%	2183.43	3.10%	2209.17	1.96%	2253.27
O	2,4,6,7	2203.97	1.61%	2208.47	1.41%	2219.17	0.93%	2240.07
P	2,4,6,8	2260.27	0.79%	2263.53	0.65%	2264.17	0.62%	2278.37
Mean		2184.74	1.66%	2193.60	1.26%	2201.44	0.91%	2221.56

Graph 1 depicts these results where we can see the hardest classes better and how the solutions improve when z increases.

Graph 1: Performance of the z-best compartment heuristic



We may see that classes E, F, M and N are the hardest ones, and they have features 4 and 5 in common, that is, several items per class and only few free items.

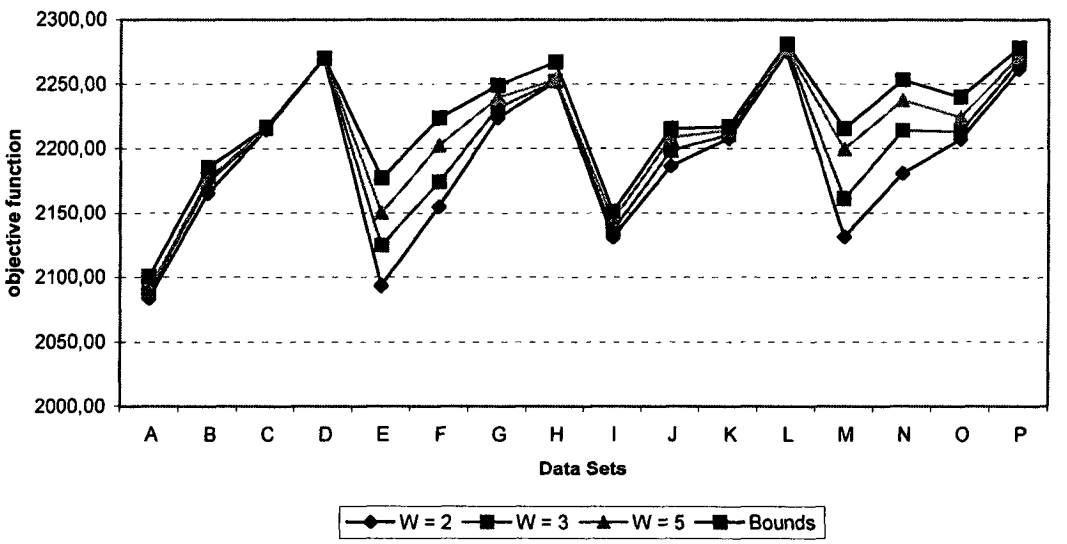
Table 2 shows the performance of the *w*-capacity heuristic run on the same instances used above.

Table 2. Averages of the objective function values for w-capacity heuristic

Sets	features	obj. func. w = 2	GAP w=2	obj. func. w = 3	GAP w=3	obj. func. w = 5	GAP w=5	upper Bound
A	1,3,5,7	2084.03	0.79%	2088.23	0.59%	2090.23	0.50%	2100.63
B	1,3,5,8	2165.80	0.91%	2174.00	0.54%	2176.57	0.42%	2185.70
C	1,3,6,7	2214.63	0.08%	2215.57	0.04%	2215.57	0.04%	2216.43
D	1,3,6,8	2270.13	0.02%	2270.13	0.02%	2270.13	0.02%	2270.60
E	1,4,5,7	2093.70	3.86%	2125.10	2.42%	2150.53	1.25%	2177.70
F	1,4,5,8	2155.37	3.08%	2174.37	2.23%	2202.83	0.95%	2223.97
G	1,4,6,7	2224.27	1.11%	2232.03	0.76%	2239.80	0.42%	2249.17
H	1,4,6,8	2252.30	0.66%	2252.40	0.66%	2253.93	0.59%	2267.27
I	2,3,5,7	2131.53	0.93%	2136.83	0.68%	2145.07	0.30%	2151.50
J	2,3,5,8	2186.73	1.31%	2198.83	0.76%	2208.73	0.32%	2215.73
K	2,3,6,7	2208.00	0.43%	2211.07	0.29%	2214.80	0.12%	2217.47
L	2,3,6,8	2275.33	0.26%	2277.97	0.15%	2278.23	0.13%	2281.30
M	2,4,5,7	2131.80	3.79%	2161.33	2.46%	2199.90	0.72%	2215.80
N	2,4,5,8	2181.13	3.20%	2214.20	1.73%	2237.90	0.68%	2253.27
O	2,4,6,7	2207.50	1.45%	2213.57	1.18%	2224.43	0.70%	2240.07
P	2,4,6,8	2261.60	0.74%	2267.57	0.47%	2271.60	0.30%	2278.37
Mean		2190.24	1.41%	2200.83	0.93%	2211.27	0.46%	2221.56

Graph 2 depicts the results in Table 2.

Graph 2: Performance of the w-capacity heuristic

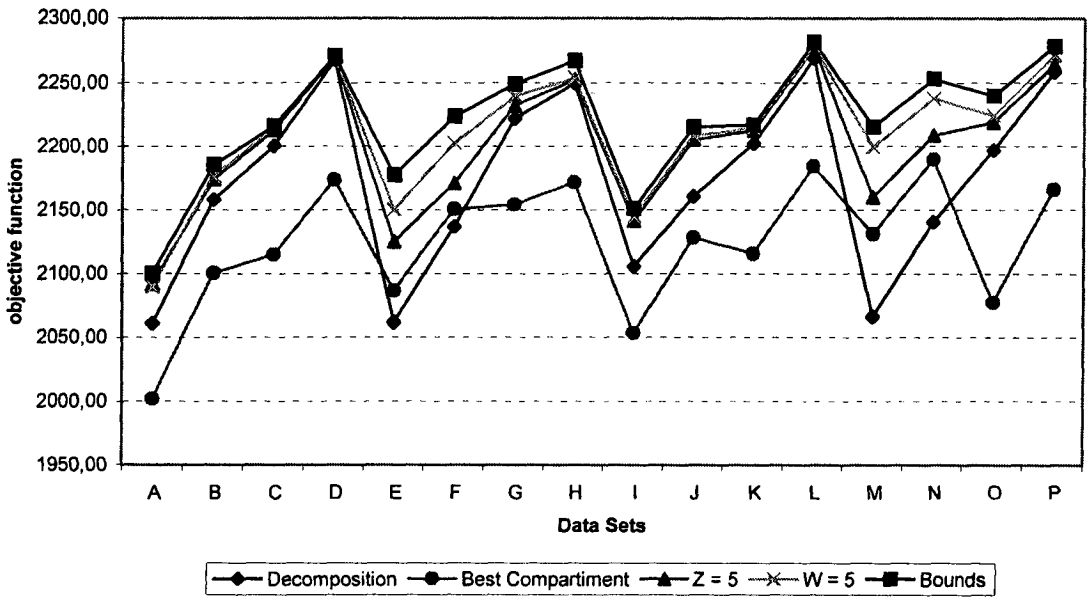


We can see in Graphs 1 and 2 that the w-capacity heuristic performed better than the z-best compartment heuristic. Finally, Table 3 and Graph 3 show the results of all heuristics together.

Table 3. Averages of the objective function values of the 4 heuristics

		Decomposition		Best Compartiment		z-best (5)		w-capacity (5)		
Sets	Features	DEC	GAP	BC	GAP	Z = 5	GAP	W = 5	GAP	Upper Bound
A	1,3,5,7	2060.90	1.89%	2002.47	4.67%	2092.33	0.40%	2090.23	0.50%	2100.63
B	1,3,5,8	2158.27	1.26%	2100.67	3.89%	2174.17	0.53%	2176.57	0.42%	2185.70
C	1,3,6,7	2200.83	0.70%	2115.17	4.57%	2213.30	0.14%	2215.57	0.04%	2216.43
D	1,3,6,8	2267.73	0.13%	2174.00	4.25%	2270.13	0.02%	2270.13	0.02%	2270.60
E	1,4,5,7	2061.97	5.31%	2086.93	4.17%	2125.67	2.39%	2150.53	1.25%	2177.70
F	1,4,5,8	2137.07	3.91%	2150.90	3.29%	2171.27	2.37%	2202.83	0.95%	2223.97
G	1,4,6,7	2222.30	1.19%	2154.57	4.21%	2232.50	0.74%	2239.80	0.42%	2249.17
H	1,4,6,8	2249.97	0.76%	2171.87	4.21%	2252.53	0.65%	2253.93	0.59%	2267.27
I	2,3,5,7	2105.90	2.12%	2053.70	4.55%	2141.90	0.45%	2145.07	0.30%	2151.50
J	2,3,5,8	2160.83	2.48%	2128.87	3.92%	2205.63	0.46%	2208.73	0.32%	2215.73
K	2,3,6,7	2202.67	0.67%	2116.17	4.57%	2212.77	0.21%	2214.80	0.12%	2217.47
L	2,3,6,8	2269.10	0.53%	2184.63	4.24%	2277.90	0.15%	2278.23	0.13%	2281.30
M	2,4,5,7	2066.43	6.74%	2131.57	3.80%	2160.50	2.50%	2199.90	0.72%	2215.80
N	2,4,5,8	2140.73	4.99%	2190.03	2.81%	2209.17	1.96%	2237.90	0.68%	2253.27
O	2,4,6,7	2197.37	1.91%	2077.60	7.25%	2219.17	0.93%	2224.43	0.70%	2240.07
P	2,4,6,8	2258.57	0.87%	2166.10	4.93%	2264.17	0.62%	2271.60	0.30%	2278.37
Mean		2172.54	2.21%	2125.33	4.33%	2201.44	0.91%	2211.27	0.46%	2221.56

Graph 3. Heuristics Performance



4.2 Remarks

The results show the importance of working with different compartment capacities (w=5 performed better than others), since if one has only few free items, then the combination of compartments (which can be seen as large items) tends to become hard. Despite this, 86.67% of the solutions obtained with the Decomposition Heuristic were 5% far from upper bound (therefore, from optimality). On the other hand, only 61.88% of the solutions obtained with the Best

Compartment Heuristic were 5% far from optimality. It shows that greedy heuristics (see step 3 in section 3.2) dealing with inner knapsack problems should be avoided.

For the z-best Compartment Heuristic, one may generate less than z different compartments, since alternative optimum solutions might produce the same capacity compartment (i.e., in step 2 of the algorithm in section 3.3 one may have: $L_{1k}=L_{2k}=...$). However, if z was large enough, the optimal solution would be found, but the problem (12.1-12.5) would be unbearably large. For the z-best Compartment Heuristic the following percentages of the solutions 5% far from the optimality were found:

- 91.46% for z = 2,
- 94.17% for z = 3, and
- 97.71% for z = 5.

Finally, the w-Capacity Heuristic showed the best results because it handles w different capacities for the compartments. However, there is no guarantee of obtaining the optimal solution if w increases, since it does not enumerate all possible solutions. For the w-Capacity Heuristic the following percentage of solutions 5% far from the optimality were found:

- 93.13% for w = 2,
- 97.08% for w = 3, and
- 100% for w = 5.

The running time for each heuristic to solve an instance is only a fraction of a second. Therefore it is feasible to implement the approaches proposed here associated with the column generation technique in practice, which may need to solve hundreds of constrained compartmentalised knapsack problems.

5. Conclusions and future perspectives

In this paper the Constrained Compartmentalised Knapsack Problem has been studied. Although it is an extension to the classical knapsack problem (for which only one class exists) the mathematical formulation changes significantly and a non-linear and integer optimisation problem arises. Various heuristics were proposed and empirically analysed by comparing each other and with a non trivial upper bound. The extension of the heuristics to deal with different lengths of objects in stock is straighforward and they are quick enough to be used in the column generation technique to solve cutting stock problems.

Finally, the compartimentalisation ideas here developed for the one dimensional cutting problem may be extended to two or three dimensional cutting/packing problems. For example, a rectangular plate which is cut into intermediate rectangles and follows different production processes (e.g. for paintings, baths). Finally, the intermediate rectangles are cut into ordered items. Items that follow the same production process define a class. Analagously, 3-dimensional items of different clients should not be packed together in a compartment.

Acknowledgments

The authors would like to thank Dr. Robison Hoto for his useful comments. This research was sponsored by FAPESP and CNPq.

References

Arenales, M.N., Morabito, R., Yanasse H. (eds.) (1999), Cutting and Packing Problems, *Pesquisa Operacional*, v. 19, n. 2.

Bischoff, E. and Wäscher, G. (eds.) (1995), Cutting and Packing, *European Journal of Operational Research*, v. 84, n. 3, special issue.

Carvalho, J.M.V. and Rodrigues, A.J.G. (1994), A Computer Based Interactive Approach to a Two-stage Cutting Stock Problem, *INFOR*, v. 32, p.243-252.

Correia, M.H., Oliveira, J. F. and Ferreira, J. S. (2004), Reel and Sheet Cutting at a Paper Mill, Computers and Operations Research, 31, 1223, 1243.

Dowsland, K.A. and Dowsland, W.B. (1992), Packing Problems, *European Journal of Operational Research*, v. 56, p.2-14.

Dyckhoff, H., Abel, D. and Gal, T. (1985), Trim Loss and Related Problems, *Omega 13*, p. 59-72.

Dyckhoff, H. (1990), A Typology of Cutting and Packing Problems, *European Journal of Operational Research*, v. 44, p. 145-159.

Dyckhoff, H. and Wäscher, G. (eds.) (1990), Cutting and Packing, *European Journal of Operational Research*, v. 44, n. 2, special issue.

Dyckhoff, H. and Finke, U. (1992), *Cutting and Packing in Production and Distribuition: Typology and Bibliography*. Heidelberg: Springer-Verlag Co.

Dyckhoff, H., Scheithauer, G. and Terno, J. (1997), Cutting and Packing, *Annotated Bibliographies in Combinatorial Optimization*, editado por M. Amico, F. Maffioli, F. e S. Martello (John Wiley & Sons, New York), p. 393-414.

Ferreira, J.S., Neves, M.A. and Castro, P.F. (1990), A Two-phase Roll Cutting Problem, *European Journal of Operational Research*, v. 44, p. 185-196.

Garey, M.R. and Johnson, D.S. (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Co., San Francisco, 1979.

Gilmore, P. and Gomory, R. (1961), A Linear Programming Approach to the Cutting Stock Problem, *Operations Research*, v. 9, p. 849-859.

Gilmore, P. and Gomory, R. (1963), A Linear Programming Approach to the Cutting Stock Problem, parth II, *Operations Research*, v. 14, p. 94-120.

Gilmore, P. and Gomory, R. (1965), MultiStage Cutting Stock Problems os Two and More Dimensions, *Operations Research*, v. 14, p. 1045-1074.

Hifi, M. (2002) (ed.). Special issue on cutting and packing. Studia Informatica Universalis, 2, 1-161.

Hinxman, A.I. (1980), Trim-Loss and Assortment Problems: A Survey, *European Journal of Operational Research*, v. 5, p. 8-18.

Hoto, R.S.V. (2001), O Problema da Mochila Compartimentada Aplicado no Corte de Bobinas de Aço. Tese de Doutoramento, COPPE-UFRJ, Rio de Janeiro, Rio de Janeiro.

Hoto, R, Arenales, M. and Maculan, N. (2004), The compartmentalised Knapsack Problem: A case Study, Technical Report n. 81, ICMC-USP.

Hoto, R., Maculan, N., Arenales, M. and Marques, F. P. (2002), Um novo procedimento para o cálculo de mochilas compartimentadas, Investigação Operacional, 22, 213-234.

Johnston, R.E. and L. R. Khan (1995). Bounds for nested knapsack problems. European Journal of Operational Research 81, 154-165.

Lin, E.Y.H. (1998), A Bibliographical Survey on Some Well-Known Non-Standard Knapsack Problems. *INFOR*, 4, 36, p. 274-317.

Lodi, A. Martello S. and Monaci, M. (2002), Two-Dimensional Packing Problems: A Survey, *European Journal of Operational Research*, 141, p. 241-252,

Martello, S. (1994a), Special issue: Knapsack, packing and cutting, Part I: One dimensional knapsack problems. INFOR, 32(3).

Martello, S. (1994b), Special issue: Knapsack, packing and cutting, Part II: Multidimensional knapsack and cutting stock problems. INFOR, 32(4).

Martello, S. and Toth, P. (1990), *Knapsack Problems: Algorithms and Computer Implementations*, Chichester: John Wiley & Sons.

Morabito, R. and Garcia, V. (1998), The Cutting Stock Problem in a Hardboard Industry: A Case Study, *Computers and Operations Research*, 25, 6, p. 469-485.

Oliveira, J. F. and Wäscher, G. (2005), Forthcoming special issue on Cutting, Packing and related Problems, *European Journal of Operational Research.*

Poldi, K. C. and Arenales, M. (2005). Dealing with small demand in integer cutting stock problems with limited different stock lengths. Technical Report n. 85, ICMC, University of Sao Paulo.

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

Sweeney, P. and Paternoster, E. (1992), Cutting and Packing Problems: A Categorized, Aplication-Oriented Research Bibliography, *Journal of the Operational Research Society*, v. 43, p. 691-706.

Wäscher, G. and Gau, T. (1996,) Heuristics for the Integer One-Dimensional Cutting Stock Problem: A Computacional Study. *OR Specktrum*, v. 18, p. 131-144.

Wäscher, G., Haussner, H. and Schumann, H. (2004), An Improved Typology of Cutting and Packing Problems. *Wsorking Paper 24*, Faculty of Economicss and Management, Otto von Guericke University Magdeburg.

Yanasse, H. H.; Soma, N.Y., and Maculan, N. (2000), An algorithm for determining the K-best solutions of the one-dimensional knapsack problem. *Pesquisa Operacional*, v. 20(1), p.117-134.

Zak, E. J.(2002), Modeling multistage cutting stock problems. *European Journal of Operational Research*, v. 141, p. 313-327.

NOTAS DO ICMC

SÉRIE COMPUTAÇÃO

085/2005 POLDI, K.; ARENALES, M.N. – Dealing with small demand in integer cutting stock problems with limeted different stock lengths.

084/2005 PRADO, T.A.S.; NUNES, M.G.V. – A statistical generative model for unsupervides learning of verb argument structures.

083/2005 POLTRONIERE, S.C.; ARENALES, M.N.; TOLEDO, F.M.B.; POLDI, K.C. – Coupling cutting stock and dot sizing problems in the paper industry.

082/2004 OLIVEIRA, P.R.; ROMERO, R.A.F. – Modelo de misturas ICA aperfeiçoado para classificação não supervisionada.

081/2004 HOTO, R.; ARENALES, M.; MACULAN, N. – The compartmentalized knapsack problem: a case study.

080/2004 KAIBARA, M.K.; FERREIRA, V.G.; NAVARRO, H.A. – Upwinding finite-difference schemes for convection dominated problems – Part I: theoretical results.

079/2004 PAIVA, D. M. B.; FREIRE, A. P.; FORTES, R. P. M. – Web engineering process – a case study from academic development

078/2004 SOUZA, R.; SILVA, C.; ARENALES, M.N. – Método do tipo dual simplex para problemas de otimização linear canalizados: teoria

077/2004 SILVA, A M.P.; NUNES, M.G.V. - Using multiword lists for lexically aligning brazilian portuguese and english texts..

076/2004 BÍSCARO, H.H; CASTELO FILHO, A.; NONATO, L.G. – A topological approach to curve reconstruction from scattered points.