Les Cahiers du GERAD ISSN: 0711–2440

A Bloc Heuristic for the Container Loading Problem

F. Chauny

G–2005–87

October 2005

A Bloc Heuristic for the Container Loading Problem

Fabien Chauny

GERAD and
Service de l'enseignement des méthodes quantitatives de gestion
HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7
fabien.chauny@gerad.ca

October 2005

Les Cahiers du GERAD
G–2005–87

Abstract

This paper presents a new heuristic for the container loading problem. This problem arrives when one wants to load a subset of rectangular boxes into a rectangular container such that the volume of the packed boxes is maximized. The basic idea consists to form a lot of blocs composed of identical boxes. A series of blocs are stacked to form piles which are loaded side by side in a section of the container. The container is filled with these sections. Four optimization procedures are utilized along the heuristic: To form the blocs, the piles, the sections and finally to fill the container. The performance of the method is shown by numerical tests over already published data.

Résumé

Dans cet article, on présente une nouvelle heuristique pour le problème de chargement de container. Ce problème consiste à charger des boîtes rectangulaires dans un container rectangulaire de façon à maximiser le volume total des boîtes chargées. On s'intéresse plus particulièrement ici au problème faiblement hétérogène qui correspond au cas où il y a un petit nombre de types de boîtes différentes et pour chaque type un grand nombre de boîtes sont disponibles. L'idée maîtresse de l'algorithme consiste à rassembler les boîtes identiques en blocs. Ces blocs sont ensuite empilés les uns au-dessus des autres pour former des piles de blocs. Ces piles sont placées côte à côte dans les différentes sections du container. Quatre procédures d'optimisation sont utilisées tout au long de l'heuristique : la formation de blocs, l'empilement de blocs, l'ajustement de piles dans les sections et finalement le choix de la séquence de sections. La performance de l'algorithme est comparée à d'autres algorithmes sur des données déjà publiées et les résultats sont supérieurs pour la majorité des cas étudiés.

1 Introduction

The container loading problem arrives when one wants to load a series of rectangular boxes into a container. Identical boxes may be grouped by type. For each type, the number of loaded boxes can not exceed a given quantity. Many versions of this problem appear in the literature according to the fact that we want to minimize the number (or the value) of the containers required to load all the boxes (also called the three dimensional bin packing problem) or to maximize the total volume (or the total value) of all the loaded boxes (the three-dimensional knapsack problem). Another important distinction concerns the number of types. When one single type of boxes is considered, we refer to the homogeneous problem (also called the manufacturer's problem). When the set of boxes is composed of many different types, we have a strongly heterogeneous problem and when there are only a few different types of boxes with many boxes for each type, we call it the weakly heterogeneous problem. In this paper we try to maximize the volume of the loaded boxes in a single container for the weakly heterogeneous problem. As usual for this type of problem, all stowed boxes are placed completely in the container, without overlapping and laid parallel to the container sides. We also suppose that empty spaces can be filled out with some kind of foam to ensure the stability of the load.

The three dimensional packing problem was the subject of several papers for the last thirty years. For the homogeneous problem, many heuristics are based on a pattern similar to the one presented in Figure 1 (see e.g. Steudel (1979), Smith and de Cani (1980), Bischoff and Dowsland (1982), Scheithauer and Terno (1996) and Morabito and Morales (1998)). This pattern is repeated as long as the top of the container is not reached. If the box can be loaded on their bottom side or end surface, the number of layers in each dimension can be obtained with a small knapsack problem (Liu and Hsiao (1997)) or with a combination of patterns along the side or the end wall of the container (Han et al. (1989), George (1992)).

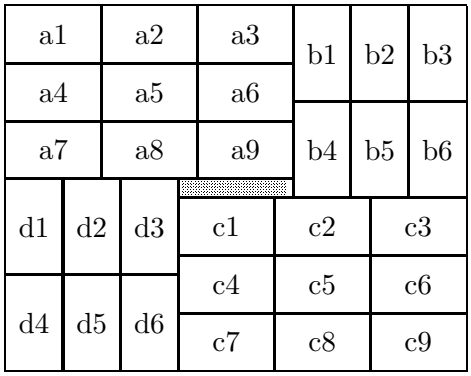

Figure 1: Typical layout

Many greedy heuristics have been proposed for the heterogeneous problem, (See e.g. George and Robinson (1980), Carlo et al. (1985), de Kovel (1986), Ivancic et al (1989), Portman (1990), Li and Cheng (1990, 1992), Mohanty et al. (1994), Ngoi et al. (1994), Heady et al. (1995) and Bischoff and Ratcliff (1995)). More sophisticated methods have also been proposed Eley (2002), Pisinger (2002) and Brunetta and Grégoire (2005) (Sub-optimization methods with enumeration trees), Terno et al. (2000) (Dynamic programming), Bortfeld and Gehring (genetic algorithms (1997 and 2001) and tabu search (1998)).

The algorithm proposed here begins with the formation of blocs of a single box type. These blocs are stacked to form piles of blocs which are loaded side by side in sections. The lengths of the sections are obtained with a search in an enumeration tree. At each step of the algorithm, different optimization procedures are involved with the objective to minimize the lost of space. The organization of this paper is the following: Section 2 describes the problem and the notation. In Section 3, we present the different procedures used in the algorithm. Computational experiments with different test instances taken from the literature are exposed in Section 4. The paper is concluded by some final remarks in Section 5.

2 The Container Loading Problem.

The problem under consideration may be formulated as follows: Consider a container of length L, width W and height H and a given set of n types of boxes. Boxes of types j, (j=1, 2,...,n) have a length l_j, a width w_j and a height h_j. The number of boxes of type j that may be loaded is noted by q_j. When it is possible to rotate a box and to use the length or the width of this box as its height, we will utilize the following parameters to indicate it:

$$p_j^k = \begin{cases} 1 & \text{if dimension } k \text{ may be used as height} \\ 0 & \text{otherwise} \end{cases}$$

where $k = 1$ corresponds to the length, $k = 2$ for the width and $k = 3$ for the height. Note that $p_j^3 = 1, \forall j$.

The volume of boxes of type j is noted by $V_j = l_j \cdot w_j \cdot h_j$. The problem is to find a feasible arrangement of the boxes within the container with the maximum total volume of the loaded boxes. This problem may be seen as a generalisation of the well known knapsack problem, which is the case when $(W/2 < w_j \leq W)$ and $(H/2 < h_j \leq H)\,\forall j$. It is then NP-hard and there exists not any algorithm to solve it to optimality for practical instances.

The basic approach consists to divide the length of the container in a series of sections where it will be placed one or more piles of blocs of similar boxes. A useful concept allows us to limit the number of possible lengths for a section. It is called the normal cuts (Herz (1972)). A value c is called a normal cut for a rectangle $l \times w$ if there exists two integer numbers α_1 and α_2 such that $c = \alpha_1 \cdot l + \alpha_2 \cdot w$. The set of all normal cuts for the box

of type j, with dimension k for the vertical dimension is noted C_j^k with $C_j^k = \phi$ if $p_j^k = 0$ and $C = \bigcup_{j,k} C_j^k = \{c_1, c_2, \ldots, c_n\}$ is the set of all allowable lengths for a section. We will use the staircase functions $t_j^k(x) = \max\{c \in C_j^k : c \leq x\}$ and $t(x) = \max\{c \in C : c \leq x\}$ to refer to the largest value smaller than or equal to x in the corresponding sets. Note that the set C does not include all the integer linear combinations of all the dimensions of boxes but simply the integer linear combinations of two dimensions of a single box.

Another set of possible lengths is of special interest here. As introduced by Scheithauer and Terno (1996), the set of *raster points* is defined as $R(Y) = \{\langle Y - x\rangle : x \in C; x \leq Y\}$ where $\langle x\rangle$ holds for $\max\{c \in C : c \leq x\}$.

3 The heuristic

In this section we first briefly describe the way the blocs are created. Next a general model will be presented. This model is too difficult to be solved to optimality. It will then be decomposed in three parts. First, we will see how the piles are formed, next how the piles are loaded in sections and finally how the lengths of sections are decided.

3.1 Create blocs

We need to know how many boxes of type j could be inserted in a block of length y and width x when the height of the bloc is its k-th dimension. The objective of this section is then to find the maximum number of rectangles of length l and width w that can be entered in a rectangle of length y and width x. This problem is known as the homogeneous problem, discussed earlier. Different algorithms have been proposed in the recent years to solve this problem. We choose to use the dynamic programming approach proposed by Scheithauer and Terno (1996). The main advantage of this algorithm is that the computations are done for all values of $x, y \in C_j^k$. Let $g_j^k(x, y)$ be the number of boxes of type j put in an x by y rectangle when the k-th dimension of the box is used for the vertical dimension (k=1, 2, 3). The general equation is given by:

$$g_j^k(x, y) = \min\{\max\{N_1, N_2, N_3\}; q_j \cdot p_j^k\}$$

where:

$$N_1 = \max_{x_a \in C_j^k} \{g_j^k(x_a, y) + g_j^k(t_j^k(x - x_a), y)\}$$

$$N_2 = \max_{y_c \in C_j^k} \{g_j^k(x, y_c) + g_j^k(x, t_j^k(y - y_c))\}$$

$$N_3 = \max_{\{x_a, x_b, y_c, y_d\}} \{g_j^k(x_a, y_c) + g_j^k(t_j^k(x - x_a), y_d) + g_j^k(x_b, t_j^k(y - y_c)) + g_j^k(t_j^k(x - x_a), t_j^k(y - y_d))$$

here, x_a and $x_b \leq x \leq W$, y_c and $y_d \leq y \leq L$ are some values from the set C_j^k. The values of N_1 and N_2 are easy to obtain. They correspond to the case where the rectangle

is divided in two parts with the separation parallel to a side of the x by y rectangle. The value of N_3 is more difficult to obtain. It corresponds to a pattern similar to the one of Figure 1, but the four sections are filled with this kind of pattern. Concepts of bounds and symmetries are useful to avoid unnecessary computations. (See Scheithauer and Terno (1996) for more details on the algorithm.) Note that if the k-th dimension is forbidden for the vertical dimension of the box, then $g_j^k(x,y) = 0$. Moreover, $g_j^k(x,y)$ is bounded by q_j. The values of $g_j^k(x,y), j = 1,2,\ldots,n; k = 1,2,3; x,y \in C_j^k$ are computed once at the beginning of the algorithm and stored in memory.

3.2 The general model

The aim of the general model is to minimize the volume of the remaining boxes when they are loaded according to the pattern described earlier. Suppose that the container is divided in P_L sections and that section s contains at most P_W piles. Consider the following variables:
y_s: length of the section s;
x_p^s: width of the pile p of the section s;
z_{pj}^{sk}: number of blocs formed with boxes of type j with its k-th dimension for its vertical dimension in the pile p of section s;
e_j: unloaded boxes of type j; u_j: overloaded boxes of type j.

$$\nu_{cs} = \begin{cases} 1 & \text{if the lentgth of section } s = c \in C \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_{cp}^s = \begin{cases} 1 & \text{if the width of pile } p \text{ in section } s = c \in C \\ 0 & \text{otherwise} \end{cases}$$

Finally, let θ_j^k be the vertical dimension of a box of type j when it is rotated with the k-th dimension placed vertically ($\theta_j^1 = l_j, \theta_j^2 = w_j, \theta_j^3 = h_j$).

Then, the model is:

$$\text{Min} \quad \sum_{j=1}^{n} V_j \cdot e_j \tag{1}$$

$$\text{Subject to} \quad \sum_{s=1}^{P_L} y_s \leq L \tag{2}$$

$$\sum_{p=1}^{s} x_p^s \leq W \qquad \forall s \tag{3}$$

$$\sum_{j=1}^{n} \sum_{k=1}^{3} \theta_j^k \cdot z_{pj}^{sk} \leq H \qquad \forall s, p \tag{4}$$

$$\sum_{s=1}^{P_L}\sum_{p=1}^{P_W}\sum_{k=1}^{3} g_j^k(t_j^k(x_s^p), t_j^k(y_s)) \cdot z_{pj}^{sk} + e_j = q_j + u_j \qquad \forall j \qquad (5)$$

$$y_s = \sum_{c\in C} c \cdot \nu_{cs} \qquad \forall s \qquad (6)$$

$$x_p^s = \sum_{c\in C} c \cdot \mu_{cp}^s \qquad \forall s, p \qquad (7)$$

$$\sum_{c\in C} \nu_{cs} = 1 \qquad \forall s \qquad (8)$$

$$\sum_{c\in C} \mu_{cp}^s = 1 \qquad \forall s, p \qquad (9)$$

$$z_{pj}^{sk} \text{ integer } \geq 0 \qquad \forall s, p, j, k \qquad (10)$$

$$\nu_{cs} \in \{0, 1\} \qquad \forall c, s \qquad (11)$$

$$\mu_{cp}^s \in \{0, 1\} \qquad \forall c, s, p \qquad (12)$$

$$e_j, u_j \geq 0 \qquad \forall j \qquad (13)$$

$$y_s \geq 0 \qquad \forall s \qquad (14)$$

$$x_s^p \geq 0 \qquad \forall s, p \qquad (15)$$

Constraints (2) to (4) control the size of the piles, the constraints (5) compute the number of loaded boxes, and constraints (6) to (9) limit the dimensions of any pile to be a normal cut. This model is not very useful for practical instances since the number of integer variables is too high and the structure is hard to handle. Then, we propose a few features to obtain a good heuristic solution. These features are:

1. To limit the number of sections. Since the blocs do not have the same dimensions, it is unavoidable to have some lost of space between each pile. If we limit the number of sections, we reduce the number of times this lost occurs. Moreover, numerical tests shown that such a number is often less than four for weakly heterogeneous problems.
2. The lengths of the sections will be determined with a search in a tree. At each node of the tree, the length of a section is evaluated by the volume of the boxes which are not already loaded in the previous sections and that can be loaded with the heuristic. In order to speed up the search, an upper bound will be derived which permits us to prune some nodes.
3. Within each section at most two piles are loaded. The width of the section is divided in two parts, each of them will be filled with a single pile. The location for the border between the two piles is also obtained by a search in a tree and again, an upper bound will permit us to prune some nodes.
4. Each pile is formed by solving a pure integer linear programming problem by a fast branch and bound procedure.

We will now describe in more details each step of the algorithm.

3.3 Construction of piles

Many times during the construction of a solution, we will have to find the most valuable pile with a height no more than H and formed from blocs with a length not larger than y and a width not larger than x. Moreover, if the number of loaded boxes of type j exceeds b_j, only the first b_j boxes are valuable. To see the nature of this problem, let us introduce the following variables:
ζ_j^k = Number of blocs of boxes of type j with the k-th dimension as height.
γ_j = Number of valuable boxes of type j in the pile.
σ_j = Surplus variables for the boxes of type j.

Consider now the model for finding the best pile of blocs when there are at most b_j boxes of type j to load. (Note that for the first pile $b_j = q_j \forall j$):

$$\text{Max} \qquad \sum_{j=1}^{n} V_j \cdot \gamma_j \tag{16}$$

$$\text{S.t.} \qquad \sum_{j=1}^{n} \sum_{k=1}^{3} \theta_j^k \cdot \zeta_j^k \leq H \tag{17}$$

$$\sum_{k=1}^{3} g_j^k(t_j^k(x), t_j^k(y)) \cdot \zeta_j^k = \gamma_j + \sigma_j \quad j = 1, 2, \ldots, n \tag{18}$$

$$\gamma_j \leq b_j \qquad j = 1, 2, \ldots, n \tag{19}$$

$$\gamma_j, \sigma_j, \zeta_j^k \geq 0 \qquad \text{integers}$$

The objective function (16) evaluates the volume of valuable boxes in the pile, the constraint (17) is for the height of the pile and the constraints (18) and (19) control the number of valuable boxes. This is a pure integer linear programming problem with a reasonable number of variables but very similar to a knapsack problem with bounded variables. In fact, if the demands (b_j) are sufficiently large, the surplus variables become useless, and the problem may be rewritten as a multidimensional knapsack problem (MKP), the constraints (18) and (19) may now be replaced by the traditional constraints on demands for any cutting stock problem. A different way to see this problem as an MKP, is to introduce incomplete blocs for each different height. With incomplete blocs, we have an MKP with one linking constraint and n independent constraints. To simplify the notation, suppose that the set of all the indexes S is partitioned in K separate subsets $S_i, i = 1, 2, ..., K$ and we have to solve the following problem:

$$
\begin{aligned}
Z = \text{Max} \ & \sum_{j \in S} c_j \cdot x_j \\
\text{S. t.} \ & \sum_{j \in S} a_j \cdot x_j \leq b_0 \\
& \sum_{j \in S_i} \alpha_{ij} \cdot x_j \leq b_i \quad i = 1, 2, ..., K \\
& x_j \text{ integer } \geq 0
\end{aligned}
$$

The first constraint is called the knapsack constraint and we will consider all the other constraints like special bounding conditions on variables and we will adapt the algorithm for the bounded knapsack problem from Martello and Toth (1990). Note that $\forall j \in S_i, c_j/\alpha_{ij} = V_i$ is the volume of a box of type i. To solve this problem, we first suppose that the variables are already sorted such that $c_1/a_1 \geq c_2/a_2 \geq ... \geq c_m/a_m$ and we denote by m_j the type of box associated to variable x_j. The proposed algorithm is simply a branch and bound procedure. The following proposition gives an idea of the optimal basis for the linear relaxation:

Proposition 1:
If the linear relaxation has an optimal solution then there exists an optimal solution where x_1 is a basic variable.

Proof:
We first note that for the optimal solution of the linear relaxation, if the knapsack constraint is not tight, then all the other constraints must be tight.

Consider the basic variable with the smallest index. Let x_k be this variable, and suppose now that $k > 1$. There exists two possibilities for m_1 and m_k: $m_1 = m_k$ or $m_1 \neq m_k$.
If $m_1 = m_k$, then the reduced cost of x_1 is $\bar{c}_1 = c_1 - \mu_0 \cdot a_1 - \mu_{m_1} \cdot \alpha_{m_1 1}$ where μ_0 is the shadow price of the knapsack constraint and μ_i is the shadow price of the (i+1)-th constraint. The reduced cost of x_k is given by $\bar{c}_k = c_k - \mu_0 \cdot a_k - \mu_{m_k} \cdot \alpha_{m_k k} = 0$ since x_k is a basic variable, then $\frac{\bar{c}_1}{a_1} = \frac{c_1}{a_1} - \mu_0 - \frac{\alpha_{m_1 1}}{a_1} \left(\frac{c_k - \mu_0 a_k}{\alpha_{m_k k}} \right)$. Using the relation between c_j and α_{ij}, $\frac{\bar{c}_1}{a_1} = -\mu_0 + \frac{\alpha_{m_1 1}}{a_1} \left(\frac{\mu_0 a_k}{\alpha_{m_k k}} \right)$. We know that μ_0 is non negative, $a_1 > 0$ and $\frac{\alpha_{m_1 1}/a_1}{\alpha_{m_k k}/a_k} \geq 1$ then the reduced cost of x_1 must be positive. The variable x_1 may be introduced in the basis without destroying the optimality.
If $m_1 \neq m_k$, x_1 non basic implies that all the variables x_j such that $\alpha_{m_1 j} > 0$ must be non basic (it is the case $m_1 = m_k$), the constraint for this type of box is not tight and its shadow price is 0. Moreover, the knapsack constraint is tight with a positive shadow price. The reduced costs for x_k and x_1 are $\bar{c}_k = c_k - \mu_0 \cdot a_k - \mu_{m_k} \cdot \alpha_{m_k k} = 0$ and

$\bar{c}_1 = c_1 - \mu_0 \cdot a_1 = c_1 - \frac{c_k - \mu_{m_k k} \cdot \alpha_{m_k k}}{a_k} \cdot a_1$. Then, $\frac{\bar{c}_1}{a_1} = \left(\frac{c_1}{a_1} - \frac{c_k}{a_k}\right) + \frac{\mu_{m_k} \cdot \alpha_{m_k k}}{a_k}$ and we can conclude that $\bar{c}_1 \geq 0$ and the variable x_1 may be introduced in the basis without destroying the optimality.

Proposition 1 allows to find fastly the next branching variable in a branch and bound algorithm. Now we find an upper bound as follows. Suppose that all the variables x_1 through x_{j-1} have been fixed and we want to evaluate the possibility to set the value of some variables x_k, $k \geq j$ to a value greater than 0. Let $\bar{b} = b_0 - \sum_{k<j} x_k \cdot a_k$ the remaining height over the pile after that the first blocs are loaded and $\bar{z} = \sum_{k<j} c_k \cdot x_k$ the value of the current partial solution. The quantity $u_1 = \bar{z} + \frac{c_j \cdot \bar{b}}{a_j}$ is clearly an upper bound for the solution of our problem when the first $j-1$ variables have been fixed to these values. This bound may be improved. Let $\nu = \text{Min}\left\{\frac{b_{m_j}}{\alpha_{m_j j}}; \frac{\bar{b}}{a_j}\right\}$. We have possibly a better bound as $u_2 = \bar{z} + c_j \cdot \nu + \frac{c_{j+1} \cdot \left(\bar{b} - \nu \cdot a_j\right)}{a_{j+1}}$. If $\nu = \frac{\bar{b}}{a_j}$ then $u_1 = u_2$, else $u_1 > u_2$. Here is the detail of the algorithm for the knapsack problem with special structure for the bounds (KSSB). The input is b_0 and vector $\mathbf{b}$, which are the right hand side of the constraints, the vector $\mathbf{c}$, which corresponds to the coefficients of the objective function, the vector $\mathbf{a}$ and the matrix α which are the coefficients of the constraints. The output is the vector of variables $\mathbf{X}^*$, the optimal volume Z^* and the vector $\mathbf{N}^*$ for the number of boxes of each type loaded in the pile. The values of b_0 and $\mathbf{b}$ will be modified during the algorithm. They correspond to residual capacities when some variables are fixed:

Procedure KSSB $(b_0, \mathbf{b}, \mathbf{c}, \mathbf{a}, \alpha)$;

Step 1: Initialisation:
$\mathbf{X}^* = \mathbf{X} = 0; Z^* = Z = 0; \mathbf{N}^* = 0;$
$L_{Min}(k) = \min_{k+1 \leq i \leq n}\{a_k\}, k = 1, 2, ...n-1; j = 1.$

Step 2: Forward:
While $((b_0 \geq L_{Min}(j))$ and $(j \leq n))$ do
$x_j = \min\left\{\lceil \frac{b_{m_j}}{\alpha_{m_j j}} \rceil, \lfloor \frac{b_0}{a_j} \rfloor\right\}; q_j = \min\{x_j \cdot \alpha_{m_j j}; b_{m_j}\};$
$z = z + q_j \cdot V_{m_j}; b_0 = b_0 - x_j \cdot a_j; b_{m_j} = b_{m_j} - q_j;$
$j = j + 1.$
Update $\mathbf{Z}^*, \mathbf{N}^*$ and $\mathbf{X}^*$;

Step 3: Backtrack:
If $(j = n)$ then
$z = z - q_j \cdot V_{m_j}; b_0 = b_0 + x_j \cdot a_j; b_{m_j} = b_{m_j} + q_j; x_j = 0.$
Find the largest value of j such that $x_j > 0$.
If $(j = 0)$ stop; Return the optimal solution: $\mathbf{Z}^*, \mathbf{N}^*$ and $\mathbf{X}^*$.
Remove the last bloc from the pile and adjust the values of x_j, z, b_0 and b_{m_j}.
Compute u_2.
If $(u_2 \leq \mathbf{Z}^*)$
If $(x_j > 0)$ then set $x_j = 0$ and adjust the values of z, b_0 and b_{m_j}.

Go to step 3.
Else
Set $j = j + 1$ and go to step 2.

At step 2, the values of $x_k, k = 1, 2, ..., j-1$ have all been fixed. We obtain a lower bound by increasing the values of $x_j, x_{j+1}, ..., x_n$ as high as possible in this order. According to Proposition 1, we know that there exists an optimal solution for the linear relaxation where x_j is a basic variable. This variable is chosen to be the next branching variable in a branch and bound scheme. Its value corresponds to the largest possible value which respects all the constraints. It is then useless to try to branch to a greater integer value. At step 3, the value of the variable with a strictly positive value with the largest index is decreased by one unit. It should be paid attention for the last bloc could be incomplete. Next the upper bound is computed. If this bound shows that it is impossible to obtain a better solution, the branching variable is directly set to 0 since all the lower values for this variable will leave a lower upper bound. We may backtrack again.

3.4 Filling a section

The aim of this portion of the algorithm is to fill as well as possible a section of the container with piles of blocs of boxes. Each section is filled with one or two piles. Let $c \in C$ be the length of the section to be filled. We first tentatively divide the section in two rectangular parts with dimensions $(x \times c)$ and $((W - x) \times c)$ where $x \in R(W), x \geq W/2$. These two rectangular spaces are filled in turn with boxes which were not loaded yet in the preceding sections using the procedure KSSB. The boxes in the first pile are not admissible for the second pile. At this point we have two adjacent piles. If at least one stage in each pile is composed of the same type of box with the same vertical orientation, these two stages are merged together to possibly load more boxes of this type. This procedure is repeated for all the values of $x \in R(W), x \geq W/2$, and the value of x which presents the best overall volume for the loaded boxes in the whole section is preserved if a length c is adopted for this section.

There are two characteristics which make it possible to save time even if the quality of the solution would be slightly worse. First, the volume of the second pile is bounded by $t(W - x) \cdot c \cdot H$. If the volume of the first pile plus this bound is less than or equal to the best known volume for the whole section, the value of x for the width of the first pile is automatically rejected. Note that the value of x is rejected even if the merging procedure would give a better result. Next, we consider only the values greater than or equal to $W/2$ for x. Since, when we are filling the second pile, the choice of boxes becomes different, one cannot suppose that the solution will be the same one if we would try two piles with dimensions $((W - x) \times c)$ and $(x \times c)$.

3.5 Choose the length of the sections

To determine the length of the sections, several strategies are possible. We analyse, in this paper, four of them.

The 2-sections strategy
The first strategy consists in dividing the length of the container into two sections of lengths y_1 and y_2, such that $y_1 + y_2 \leq L$. The values for y_1 and y_2 are two elements of the set of raster points $R(L)$. We start by evaluating the total volume of the boxes which can be placed in a section of length $y_1, \forall y_1 \in R(L)$ using at most q_i boxes of type $i, i = 1, 2, ..., n$. This volume is noted by $V(y_1, \mathbf{q})$ and is obtained by using the procedure of the Section 3.4. Next, the length of the second section is evaluated by $y_2 = \max\{y \in R(L) : y \leq L - y_1\}$. Since, $y_2 \in R(L)$ there is immediately a good idea of the maximum volume of boxes which it will be possible to place in the second section. Let Z^* be the value of the best known solution for the problem, then if $V(y_1, \mathbf{q}) + V(y_2, \mathbf{q}) < Z^*$, the optimization for the second section is abandoned, the value of y_1 for the length of the first section is automatically eliminated. Otherwise, we evaluate the maximum volume of the boxes which it is possible to place in the second section by using only the boxes which are not loaded in the first section.

A greedy strategy
The second strategy, named the greedy strategy, consists to evaluate for the section $s, (s \geq 1)$, the total volume of the remaining boxes which it is possible to place in a section of length c, for all the feasible values of $c, \left(0 < c \in C \leq L - \sum_{j<s} y_j\right)$. The value of y_s is the value of c which maximizes the ratio $\frac{V(c,\mathbf{q}^s)}{c}$ (where $\mathbf{q}_s$ denotes the set of remaining boxes). This procedure is repeated until the remaining length becomes too small to give an interesting packing. We then use another strategy to fill out the last section of the container. When the remaining length is less than W, we use the 2-sections strategy to fill out the section of length W, width $L - \sum_{j \leq s} y_j < W$ and height H with the remaining boxes.

The 3-sections strategy
The third approach consists to use at most three sections. We first compute $V(c, \mathbf{q})$, $\forall c \in C$ using the procedure of Section 3.4. The remainder of the container is filled using the 2-sections strategy with the remaining boxes. Here again, one suspects that the result of the 2-sections strategy will be lower than u_1, where

$$u_1 = \max_{\{c_1 \in C : c + c_1 \leq L\}} \{V(c_1, \mathbf{q}) + V(t(L - c - c_1), \mathbf{q})\}.$$

But with the amalgamation between various stages of the two piles, it seldom happens that this is not true. It so much rarely arrives that the 2-sections strategy presents a better result, that we decided to use this quantity as a bound and we make the following test

before calling the 2-sections procedure: If $Z^* \geq V(c, \mathbf{q}) + u_1$ then the value of c is rejected for the length of the first section.

The accelerated 3-sections strategy
The difference between this strategy and the preceding one is in the evaluation of u_1. If $c_i < c_j$, it is known that very often $V(c_i, \mathbf{q}) \leq V(c_j, \mathbf{q})$. Let us note now $\overline{V}(c_i, \mathbf{q})$ the total volume of the boxes packed by the 2-sections strategy in the remaining part of the container when the first part, with length c_i, has already been packed with a subset of boxes from the set $\mathbf{q}$. We note that very often $\overline{V}(c_i, \mathbf{q}) \geq \overline{V}(c_i, \mathbf{q})$. This suggests the following test: If $Z^* \geq V(c_i, \mathbf{q}) + \overline{V}(c_{i-1}, \mathbf{q})$ then the value of c_i is rejected for the length of the first section without applying the 2-sections strategy for the remaining of the container.

3.6 An example

We present now an example. The data comes from Bischoff and Ratcliff [3] and is available from the web site of OR-Library [1]. Figures 2 to 5 present a typical layout of the boxes. The length of the container is 587, its width is 233 and its height is 220. There are three types of boxes. The dimensions are presented in the Table 1.

All dimensions except the length of the first type of box can be used like the vertical dimension of the box. The results for the 2-sections strategy appear in Table 2.

Table 1: Data for the example

Type	Length	Width	Height	Quantity
1	55	50	26	160
2	48	42	37	167
3	47	34	26	149

Table 2: Results for the example

	Length								
	550				37				
Width	Type	Height	Qty	Nbr of stages	Type	Height	Qty	Nbr of stages	Width
155	1	26	31	5	2	42	3	4	146
	2	48	53	1	3	26	3	2	
	2	42	45	1					
78	3	47	48	2	2	48	2	1	87
	3	34	34	1	3	47	3	3	
	2	48	26	1	3	26	1	1	
	2	42	22	1					

The length of the container is divided in two parts of lengths 550 and 37 respectively. The width of part one is also divided in two parts of widths 155 and 78 respectively while the part two is divided in two widths of 146 and 87. The algorithm KSSB proposes to do five stages with 31 boxes of type 1 with a height of 26 each and two stages with boxes of type two, the first one with 53 boxes by using its length as vertical dimension and for the other stage to put 45 boxes by using its width for the vertical dimension. The remainder of the table is read in the same way. The two piles of length 550 have a stage formed with boxes of type 2 with a height of 42 which can be merged together. It is the same situation for the stage with a height of 48. The merging process suggests putting 71 boxes with the height 42 (Figure 2) and 81 boxes with the height 48 (Figure 3). The two piles of the

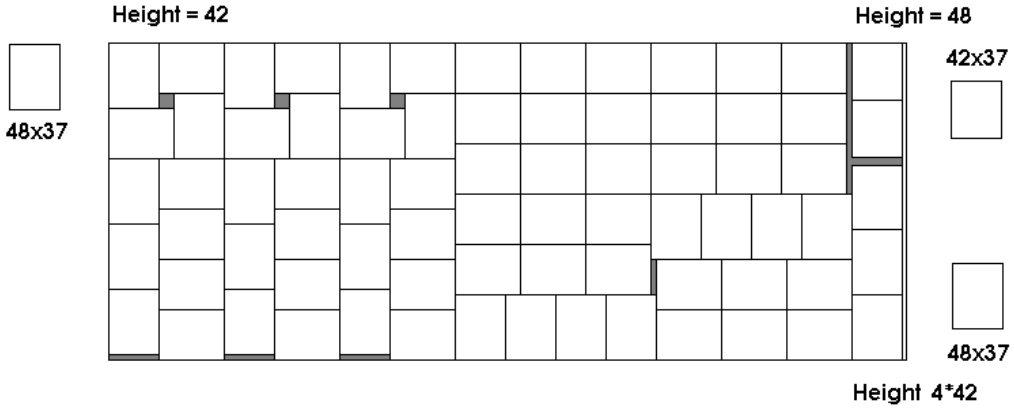

Figure 2: First floor

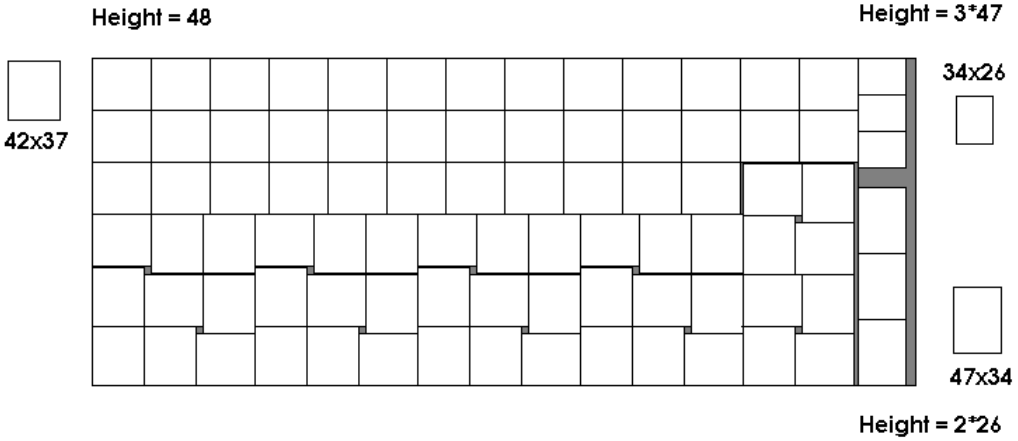

Figure 3: Second floor

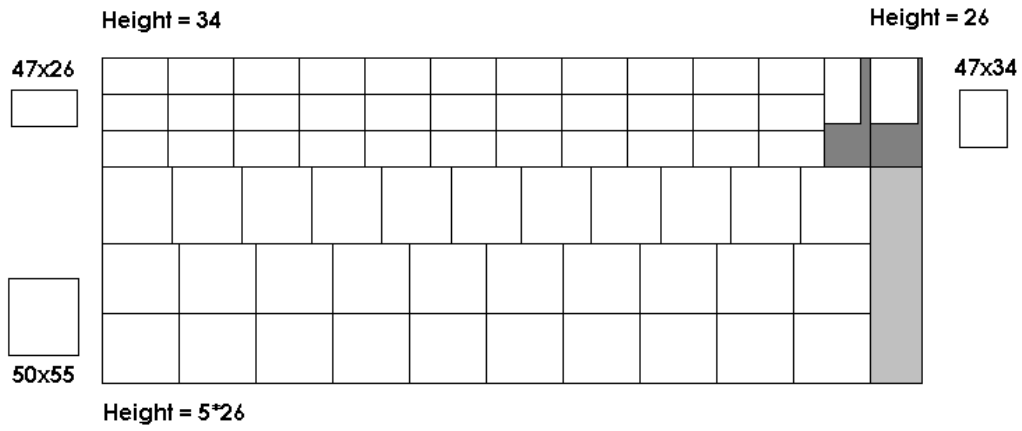

Figure 4: Third floor

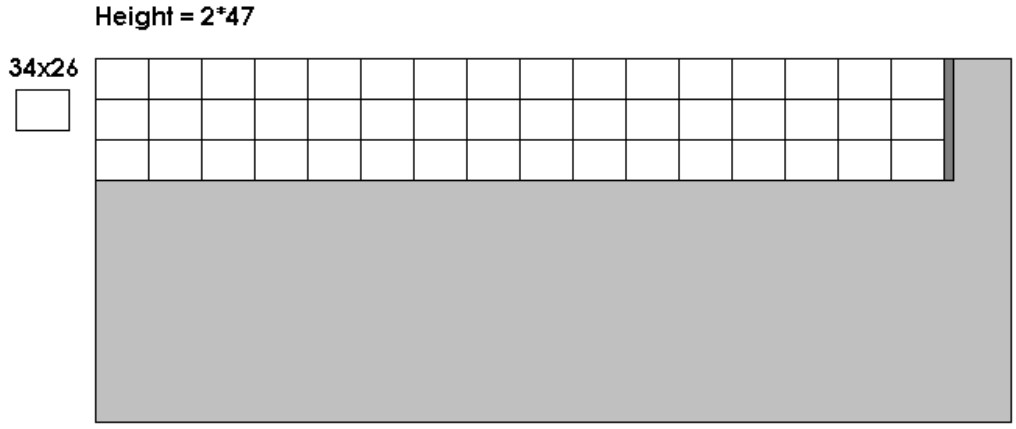

Figure 5: Fourth floor

second section also have a common height, but no gain is possible in this case. Figure 4 presents the lay out of the five stages with the boxes of type 1 (Bottom left) and the stage with 34 boxes of type 3 (Top left). Figure 5 presents the final stages with 48 boxes of type 3 each. The right part of Table 2 is rather obvious. The container is filled at 98.14% with this packing.

4 Numerical results

In order to investigate the computational behaviour of the proposed algorithms, numerical tests were carried out. All the programs are written in Fortran 77 and run on a Sun

Workstation Sun4u SunFire 4800, 1200 MHz. The approaches were compared on two series of instances, that makes it possible to compare the performance with known existing methods.

The first series (Loh and Nee 1992) is composed of 15 instances. For all of these instances, the vertical dimension of any boxes is fixed since the boxes can not be rotated. We give in Table 3 the volume utilization rate (in percent) for four of these instances. The other eleven instances do not really present a challenge, most of methods manage to load all the boxes and this is the case for all the heuristics presented here. The first column of Table 3 gives the problem name with the number of different types and the average number of boxes per type. The second column presents the best known solution, which corresponds to the percentage of utilised space in the container. These solutions were always obtained by the tabu search algorithm from Bortfeldt and Ghering (1998). For the problems LN07 and LN13, it is the optimal solution since all the boxes were loaded. The four other columns give the results of the methods proposed in this paper. As we can see, the performance of the proposed methods is just behind the tabu search algorithm for these problems.

The second series was made on 600 instances proposed by Bischoff and Ratcliff (1995). There are six series composed of 100 instances, each with different degrees of heterogeneity for the consignment. According to the series, there are 3, 5, 8, 10, 12 and 15 types of boxes. The total volume of boxes is slightly less than the volume of the container. For a box, when a dimension is larger than twice its smallest dimension, it can not act as the vertical dimension. It is the only restriction for this characteristic. Tables 4, 5 and 6 present the results for these instances. In Table 4, we present the performance of other known algorithms for the same instances. The first column gives the name of the series, the second column presents the average volume utilization (%) for the 100 instances of the series for the hybrid genetic algorithm of Bortfeldt and Gehring (2001). The column BR-Comb presents the minimum, the average and the maximum volume utilization (%) for the 100 instances of the series for the combined approach of Bischoff and Ratcliff (1995). The other columns are for the genetic algorithm of Bortfeldt and Ghering (1997), the Tabu search algorithm of Bortfeldt and Ghering (1998) and the algorithm of Terno et al (2000). Eley (2002) has also used this data to test its algorithm, but did not obtain results as good as those from the Tabu search. Tables 5 and 6 present the same statistics for the methods

Table 3: Results for four instances from Loh and Nee

Problem	Best known solution	2-sections	Greedy	3-sections	Accelerated 3 sections
LN02 (8,25)	96.6%	96.1%	96.4%	96.1%	96.1%
LN06 (8,25)	96.2%	95.0%	94.9%	95.5%	95.0%
LN07 (8,25)	84.7%	84.7%	84.7%	84.7%	84.7%
LN13 (7,17.6)	85.6%	85.6%	84.6%	85.6%	85.6%

Table 4: Known results for instances from Bischoff and Ratcliff

Set	BG	BR-Comb. (1995)			GB (1997)			BG (1998)			TSSR (2000)		
	2001	Min	Ave.	Max	Min	Ave	Max	Min	Ave	Max	Min	Ave	Max
B3	87.8	73.7	85.4	94.4	76.7	85.8	94.3	83.6	92.4	97.0	75.7	89.9	95.9
B5	89.4	73.8	86.3	93.8	78.4	87.3	95.2	86.8	92.3	96.5	81.9	89.6	94.7
B8	90.5	75.3	85.9	92.6	81.1	88.1	92.9	87.5	92.0	96.5	83.2	89.2	93.0
B10	90.6	78.4	85.1	90.1	82.7	88.0	91.6	87.0	91.3	94.5	83.1	88.9	92.7
B12	90.7	78.7	85.2	90.4	81.7	87.9	92.6	86.6	90.4	93.0	83.0	88.3	91.6
B15	90.7	75.2	83.8	89.2	84.1	87.9	92.5	86.3	89.6	92.2	82.3	87.4	90.5

Table 5: Results for the first three methods

Set	2-sections				Greedy				3-sections			
	Min	Ave.	Max	Time	Min	Ave	Max	Time	Min	Ave	Max	Time
B3	84.0	93.6	98.1	3.83	72.1	90.8	97.6	4.03	84.0	94.1	98.1	4.66
B5	85.4	93.3	96.7	1.32	79.6	92.0	96.5	2.54	87.8	94.1	96.9	6.20
B8	86.3	92.2	95.0	3.99	83.4	92.5	96.7	8.96	90.2	93.8	96.2	48.4
B10	86.1	91.4	94.7	7.39	86.2	92.7	96.0	17.3	88.9	93.3	95.7	118.5
B12	86.5	90.4	93.3	12.6	82.9	92.5	95.0	29.3	89.3	92.6	94.8	255.6
B15	85.7	88.8	92.1	23.6	85.3	92.3	94.4	53.8	89.7	91.9	93.8	582.6

Table 6: Results for the last method

Set	Accelerated 3-sections			
	Min	Ave.	Max	Time
B3	83.2	94.0	98.1	4.25
B5	87.8	94.0	96.7	3.98
B8	90.2	93.5	96.2	26.40
B10	88.9	93.3	95.7	60.04
B12	88.8	92.4	94.5	124.8
B15	89.1	91.6	93.5	267.4

proposed here as well as average time (in seconds) necessary to solve the hundred problems of the series.

Clearly, for this benchmark, our methods outperformed the existing ones. When the number of types is very small (3 or 5) the solutions for the 3-sections are very good and the computing times are sufficiently low. As the number of types increases, the computing times increase in a fulgurating way. The accelerated 3-sections presents solutions which are slightly below the 3-sections method but the computing times are lower when the number of types becomes greater than 8. The computing times for the 2-sections method are very low, but the solutions are fairly worse, even they are comparables with those proposed

by the Tabu Search algorithm. When the number of types becomes larger, the greedy heuristic gives the best results with a reasonable amount of time.

5 Conclusions

In this paper, a new bloc heuristic was developed to solve the container loading problem. The problem is decomposed in four sub problems. At the upper level, the length of the container is divided in sections. Each section is filled with at most two piles of homogenous blocs. Four methods are proposed on the way of dividing the container into sections. According to the degree of heterogeneity of the consignment and the available amount of time, the user may choose between four methods. When the numbers of different types is very low (five or less) the 3-sections method performs very well. As the number of types increases, the greedy heuristic seems to be a better choice. These methods give better solutions than those obtained by the tabu search of Bortfeldt and Ghering especially when the boxes may be rotated in any direction. Two more general situations arise in practical situations. The first one is the three dimensional bin packing problem when one wants to minimize the number of containers utilized to load all the boxes. The second one is when the order demands for the boxes are given with lower and upper bounds only. The author is currently working on these issues.

References

[1] Beasley, J.E., (1990), "OR-Library. Distributing Test Problems by Electronic Mail", *Journal of the Operational Research Society*, 41, 1069–1072.

[2] Bischoff, E., Dowsland, W.B., (1982), "An Application of the Micro to Product Design and Distribution", *Journal of the Operational Research Society*, 33, 271–280.

[3] Bischoff, E.E., Ratcliff, M.S.W., (1995), "Issues in the Development of Approaches to Container Loading", *Omega*, 23, 377–390.

[4] Bortfeld, A., Gehring, H., (1997), "A genetic Algorithm for Solving the Container Loading Problem", *Int. Trans. Opl Res.*, 4, 401–418.

[5] Bortfeldt, A., Gehring, H., (2001), "A hybrid genetic algorithm for the container loading problem", *European Journal of Operational Research*, 131, 143–161.

[6] Bortfeldt, A., Gehring, H., (1998), "Ein Tabu search-Verfahren fr Containerbeladeprobleme mit schwach herogenem Kistenvorrat", *OR Spektrum*, 20, 237–250.

[7] Brunetta, L., Grégoire, P., (2005), "A General Purpose Algorithm for Three-Dimensional Packing", *INFORMS J. on Comp.*, 17, 328–338.

[8] Carlo, H., Hodgson, T.J., Martin-Vega, L.A., Stern, E.R., (1985), "Micro-IPLS: Pallet Loading on a Microcomputer", *Computers and Industrial Engineering*, 9, 29–34.

[9] De Kovel, W.F., (1986), "An Algorithm for the Loading of Containers with Block-shape Parcels of Arbitrary Sizes", *Proc. of the 3rd int. Conf. on Automated Material Handling*, 274–261.

[10] Eley, M., (2002), "Solving Container Loading problems by Block Arrangement", *European Journal of Operational Research*, 141, 393–409.

[11] George, J.A., (1992), "A Method for Solving Container Packing for a Single Size of Box", *Journal of the Operational Research Society*, 43, 307–312.

[12] George, J.A., Robinson, D.F., (1980), "A Heuristic for Packing Boxes into a Container", *Comp. and Op. Res.*, 7, 147–156.

[13] Han, C.P., Knott, K., Egbelu, P.J., (1989), "A Heuristic Approach to the Three-dimensional Cargo-loading Problem", *Int. J. Prod. Res.*, 27, 757–774.

[14] Heady, R.B., Toma, A.G., Ristroph, J.H., (1995), "Evaluation of Carton Packing for High Volume Operations", *J. of Operations Management*, 13, 59–66.

[15] Herz, J.C., (1972), "Recursive Computational Procedure for Two-dimensional Stock Cutting", *IBM J. Res. Develop.*, 16, 462–469.

[16] Ivancic, N., Mathur, K., Mohanty, B.B., (1989), "An Integer Programming Based Heuristic Approach to the Three-dimensional Packing Problem", *J. Mfg. Oper. Mgt.*, 2, 268–298.

[17] Li, H., Cheng, K.H., (1990), "On Three Dimensional Packing", *SIAM J. Computing.*, 19, 847–867.

[18] Li, H., Cheng, K.H., (1992), "Heuristic Algorithm for On-Line Packing in Three Dimensions," *J. of Algorithms*, 13, 589–605.

[19] Liu, F.-H., Hsiao, C.-J., (1997), "A three-dimensional pallet loading method for the single-size boxes", *Journal of the Operational Research Society*, 48, 726–735.

[20] Loh, H.T., Nee, A.Y.C., (1992), "A Packing Algorithm for Hexahedral Boxes", *Proc. of the Industrial Automation'92 Conf.*, Singapore, 115–126.

[21] Martello, S., Toth, P., (1990), *Knapsack Problems*, John Wiley and Sons.

[22] Mohanty, B.B., Mathur, K., Ivancic, N.J., (1994), "Value Considerations in Three-dimensional Packing- A Heuristic Procedure Using the Fractional Knapsack Problem", *European Journal of Operational Research*, 74, 143–151.

[23] Morabito, R., Morales, S., (1998), "A Simple and Effective Recursive Procedure for the Manufacturer's Pallet Loading Problem", *Journal of the Operational Research Society*, 49, 819–828.

[24] Ngoi, B.K.A., Tay, M.L., Chua, E.S., (1994), "Applying Spatial Representation Techniques to the Container Packing Problem", *Int. J. Prod. Res.*, 32, 111–123.

[25] Pisinger, D., (2002), "Heuristics for the Container Loading Problem", *European Journal of Operational Research*, 141, 382–392.

[26] Portman, M.C., (1990), "An Efficient Algorithm for Container Loading", *Methods of Oper. Res.*, 64, 563–572.

[27] Scheithauer, G., Terno, J., (1996), "The G4-Heuristic for the Pallet Loading Problem", *Journal of the Operational Research Society*, 47, 511–522.

[28] Smith, A., De Cani, P., (1980), "An Algorithm to Optimize the layout of Boxes in Pallets", *Journal of the Operational Research Society*, 31, 573–578.

[29] Steudel, H.J., (1979), "Generating Pallet Loading Patterns: A Special Case of the Two-Dimensional Cutting Stock Problem", *Management Science*, 25, 997–1004.

[30] Terno, J., Scheithauer, G., Sommerweiss, U., Riehme, J., (2000), "An efficient approach for the multi-pallet loading problem", *European Journal of Operational Research*, 123, 372–381.