UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

THE UNIDIMENSIONAL CUTTING STOCK PROBLEM WITH USABLE LEFTOVER – A HEURISTIC APPROACH

ADRIANA CRISTINA CHERRI
MARCOS NEREU ARENALES
HORACIO HIDEKI YANASSE

Nº 90

NOTAS

São Carlos - SP

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

THE UNIDIMENSIONAL CUTTING STOCK PROBLEM WITH USABLE LEFTOVER – A HEURISTIC APPROACH

ADRIANA CRISTINA CHERRI
MARCOS NEREU ARENALES
HORACIO HIDEKI YANASSE

Nº 90

NOTAS

Série Computação

São Carlos – SP
Maio/2007

THE UNIDIMENSIONAL CUTTING STOCK PROBLEM WITH USABLE LEFTOVER – A HEURISTIC APPROACH

Adriana Cristina Cherri
Marcos Nereu Arenales
Horacio Hideki Yanasse
adriana@icmc.usp.br, arenales@icmc.usp.br
Instituto de Ciências Matemáticas e de Computação – ICMC
Universidade de São Paulo – USP
Av. Trabalhador São-Carlense, 400 - Caixa Postal 668
13560-970 – São Carlos – SP
horacio@lac.inpe.br
Laboratório Associado de Computação e Matemática Aplicada – LAC
Instituto Nacional de Pesquisas Espaciais – INPE
Av dos Astronautas, 1.758 – Jd. Granja
12227-010 – São José dos Campos – SP

Abstract: In this work we consider a one-dimentional cutting stock problem in which the non-used material in the cutting patterns may be used in the future, if large enough. This feature introduces difficulties in comparing solutions of the cutting problem, for example, up to what extent a minimum leftover solution is the most interesting one when the leftover may be used? Some desirable characteristics of good solutions are defined and classical heuristic methods are modified, so that cutting patterns with undesirable leftover (not large enough to be used, nor too small to be acceptable waste) are redesigned. The performance of the modified heuristics is observed by solving instances from the literature, practical instances and randomly generated instances.
Keywords: cutting stock problems, usable leftover.

1 Introduction

Cutting stock problems (CSP) consist in cutting large pieces (*objects*), available in stock, into a set of smaller pieces (*items*) in order to fulfill their requirements, optimizing a certain objective function, for instance, minimizing the total number of objects cut, minimize waste, minimize the cost of the objects cut etc. These problems are relevant in the production planning of many industries such as the paper, glass, furniture, metallurgy, plastics and textile industries.

In the last four decades cutting stock problems have been studied by an increasing number of researchers. The interest on these problems can be explained by their practical motivation and the challenge they offer to the academia for, despite their apparent simplicity, they are, in general, computionally difficult to solve.

Due to the diversity of situations where CSP arise, we are always faced with new constraints and/or objectives for which the available methods are of limited value. Hence, the use of simple heuristics has been observed in practice, many without any evaluation of their perfomance.

Although freqüently arising in practical situations, we could not find many articles in the literature that consider the situation where the leftover material may be used to cut future demands, if large enough. We call leftover any piece cut that is not a required item. To the best of our knowledge only Gradisar *et al.* (1997), Gradisar *et al.* (1999a), Gradisar *et al.* (1999b), Gradisar and Trkman (2005) and Abuabara (2006) consider this possibility. In 1997, Gradisar *et al.* proposed a heuristic

(denoted by COLA) to optimize roll cutting in the textile industry with the objective of creating a cutting plan with reduced letfovers or to concentrate them in a single object. All objects have different lengths and they propose a bi-objective function that minimizes the number of unfulfilled item demands and the total loss (sum of the leftover smaller or equal to a pre-defined value). In 1999, Gradisar *et al.* proposed a modified COLA (denoted by CUT) and in 2005, Gradisar and Trkman developed an algorithm to find a solution to general unidimensional cutting stock problems with distinct objects, starting from the solution obtained by CUT and replanning patterns that do not satisfy some criteria. In 2006, Abuabara modified the mathematical model proposed by Gradisar *et al.* (1997), decreasing its size, that is, reducing the number of constraints and variables in the model.

In this work we present some characteristics of a desirable solution (we avoid "optimal solution" since a criterion to compare solutions is not defined) to the cutting stock problem with usable leftover (CSPUL). Modifications on classical heuristic methods to solve CSP are suggested aiming to find a solution that satisfies those characteristics.

This article is organized as follows. In Section 2, the CSPUL is defined. Some methods to solve it are presented in Sections 3 and 4. Computational tests are presented in Section 5 and conclusion remarks and future works are presented in Section 6.

2 Definition of the cutting stock problem with usable leftover

During the cutting process, unavoidable leftover occur that are often discarded. Some industries, however, have the possibility of using the leftover to cut future demanded items, as long as their sizes are sufficiently large. In this situation, the simple objective of minimizing the leftover may not be appropriate.

Many of the solution methods to solve cutting problems aim to minimize leftover (alternative objectives may be defined but low amount of leftover must also be pursued). Although a low amount of leftover is an objective to pursue, the possibility of using them introduces a new condition to evaluate a solution. In this new problem, planning cutting patterns that concentrate the leftover in fewer patterns seems to be a good alternative to pursue since it increases the chances that these leftovers will be sufficiently large to go back to stock to be used to cut future demanded items.

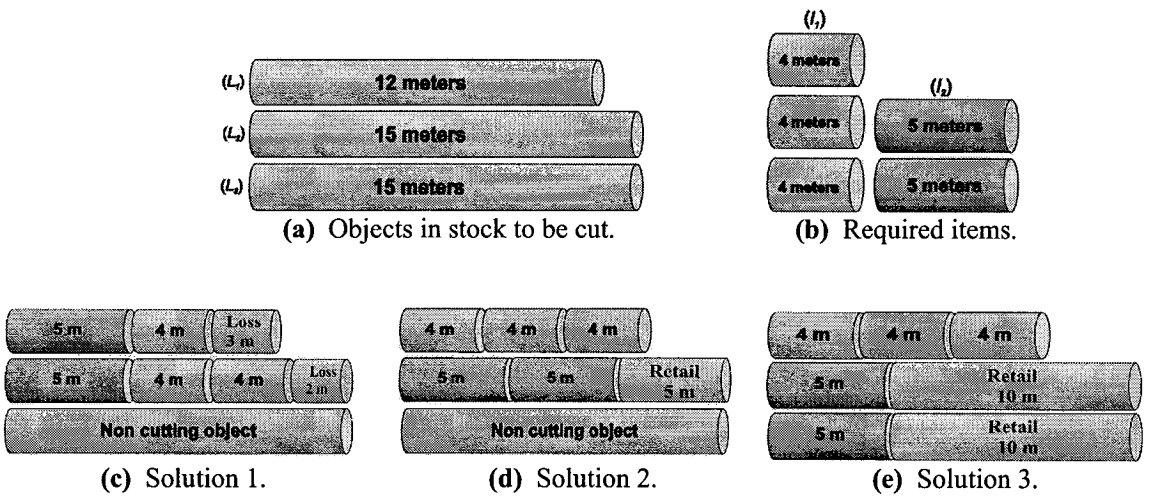
Hence, we present the unidimensional CSPUL as:

"A set of pieces (items) must be produced by cutting large units (objects) of standard sizes (objects bought from suppliers) or non standard (objects that are leftover of previous cuts). The demand of the items and the availability of the objects are given. Demand must be met by cutting the available objects such that the leftover are "small" (denoted by scrap) or "sufficiently large" (denoted by retail) to return to stock, but in a reduced number".

This high level definition aims to capture the main elements of the CSPUL but it lacks details that are going to be completed next.

The "sufficiently large" length or, equivalently, the minimum acceptable length for retail is a choice of the decision maker. Some possible choices include the length of the shortest demanded item, the average lengths of the demanded items or the length of the longest demanded item. Gradisar *et al.* (1997), Gradisar *et al.* (1999a), Gradisar *et al.* (1999b) and Gradisar and Trkman (2005) considered a retail any leftover with length greater or equal to the shortest demanded item. This choice may not be interesting in cases where the portfolio of demanded items includes a small item that is not typical because it is likely that retails that are seldomly used will be stocked. On the other hand a particular portfolio may include only large items, and retails with sizes smaller than the smallest of the items are acceptable.

In the classical CSP we find objective functions like minimize the total waste, minimize the number of objects cut, minimize the costs, and so on. In the CSPUL our objective is to have little or no scraps (as in the classical problem) and/or a reduced number of retails. Therefore, two solutions with the same leftover may be different as illustrated in Figure 1. In this example, a leftover piece of size larger or equal to 4 meters is considered retail.



(a) Objects in stock to be cut. **(b)** Required items.

(c) Solution 1. **(d)** Solution 2. **(e)** Solution 3.

Figure 1: A cutting stock problem data and alternative solutions

For the CSPUL, Solution 2 (Fig **1 - d**) is better than Solution 1 (Fig **1 - c**), since it concentrates the leftover of a size superior to 4 meters (a retail) in a single object (Solution 1 has 5 m of scrap while Solution 2 has zero scrap and a retail of 5 m). For the CSPUL we can say that Solution 1 is an *undesirable* solution compared to the *ideal* Solution 2. Another *undesirable* solution (compared with Solution 2) is Solution 3, given in Figure **1 - e**, for although it does not generate scraps, it generates a larger number of retails.

Due to the difficulty in defining a single objective function that differentiates such solutions we begin qualifying the solutions according to the following definition.

Definition 1: *The solutions of a CSPUL are defined as:*

- *Ideal solution: when a small number of objects have little scraps and none of the objects have not so little scraps. In case there are retails, they must be concentrated in a very small number of cut objects;*
- *Acceptable solution: when a small number of objects present not so little scraps and a small number of objects present retails;*
- *Undesirable solution: when several cut objects present not so little scraps or present several retails.*

Observe that an ideal solution is always acceptable but the reverse is not true.

This definition (that depends on quantifying terms like *small*, *very small* or *several objects*, *little scrap* or *not so little scrap* and *retail*), tries to incorporate general features of the solutions for the CSPUL. By not so little scrap we mean a leftover material that is larger than a little scrap but it is not big enough to be a retail.

The sizes of little scrap, not so little scrap or retail are defined by the user (decision maker). The decision maker can define these values by his/here experience. Also he/she may use parameters to define them, like:

- θ: fraction that defines the largest size for a leftover material to be considered a *little scrap* for standard sized objects, that is, θL_k is the maximum size for a leftover material to be considered little scrap in a standard object of lenght L_k, $k = 1, ..., \bar{k}$, where $\bar{k}$ is the quantity of standard object types in stock;
- β: fraction that defines the largest size for a leftover material to be considered a *little scrap* for non standard sized objects, that is, βL_k is the maximum size for a leftover material to be

considered little scrap in a non standard object of lenght L_k, $k = \bar{k} + 1, ..., K$ (the objects of type $\bar{k}+1,..., K$ are retails);

- δ: smallest size of a leftover to be considered a retail (for example, δ is the average length of the item types demanded). Any leftover larger or equal to δ is considered retail, independent of the object type.

Observe that with the parameters θ and β we make the scrap dependent on the object type. The additional parameter β allows the decision maker to define larger "little scraps" for non-standard objects, making them more prone to be used compared to the standard objects.

The quantities "*small*", "*very small*" and "*several*" in definition 1 are also defined by the user (decision maker). The decision maker can define them by his/her experience or he/she can use, for instance, two parameters ξ_1 and ξ_2, with $0 < \xi_1 < \xi_2 < 1$ and set:

- *Very small number of objects cut:* up to $\lceil \xi_1 \eta \rceil$ of the objects cut;
- *Small number of objects cut:* up to $\lceil \xi_2 \eta \rceil$ of the objects cut;
- *Several objects cut:* above $\lceil \xi_2 \eta \rceil$ of the objects cut;

where η is the total number of objects cut in the solution.

For simplicity, from now on, we use the term *acceptable leftover* when the leftover is a small scrap or retail.

With the aim of generating an ideal solution, or at least an acceptable one, we introduce modifications in some well-known heuristics of the literature to solve the unidimensional CSP so that solutions with several objects having not so small scraps are avoided. These are described in the next Sections 3 and 4.

3 Constructive Heuristics

One heuristic used in the solution of the CSP is the exhaustive repetition (Hinxman, 1980):

Algorithm – Exhaustive repetition heuristic

***Step* 1:** Build a good cutting pattern for each object type k, $k = 1, ..., K$;

***Step* 2:** Select one of the cutting patterns generated in Step 1 (a selection criterion, can be, for example, minimum waste. This selected pattern is associated with an object type);

***Step* 3:** Use the cutting pattern chosen in Step 2 as much as possible, without exceeding the required demand of the items and the availability of the associated object;

***Step* 4:** Update the demand of the items and the stock of the objects;

***Step* 5:** If the demand of the items is all satisfyed or there are no more objects available, STOP. Otherwise, go to Step 1.

The exhaustive repetition heuristic relies fundamentally on a procedure that generates "good" cutting patterns (Step 1). Two very well known procedures to generate cutting patterns are FFD (*First Fit Decreasing*) and Greedy.

FFD Procedure

In the FFD procedure we initially cut the largest item as much as possible or until its demand is satisfied (the largest items are chosen first since they are, generally, more difficult to be combined).

When it is not possible anymore to cut the largest item, the second largest item is considered and, so on, until we reach the smallest item. When no new item can be cut, the pattern is complete.

Greedy Procedure

In the greedy heuristic the cutting patterns are generated solving a sequence of knapsack problems of the form:

$$z(b) = \text{maximize } \ell_1 a_1 + \ell_2 a_2 + \ldots + \ell_m a_m \qquad (1)$$

$$\text{subject to:}$$

$$\ell_1 a_1 + \ell_2 a_2 + \ldots + \ell_m a_m \le b \qquad (2)$$

$$0 \le a_i \le r_i \,, i = 1, \ldots, m \quad \text{and integer} \qquad (3)$$

where ℓ_i is the length of item i, $i = 1, \ldots, m$, and r_i is the residual demand of item i, $i = 1, \ldots, m$, updated in Step 4. Initially, $r_i = d_i$, $i = 1, \ldots, m$, the required number of items type i.

These two classical procedures have distinct phylosophies. In the FFD procedure, there is an excessive concern in producing the largest items as earlier as possible since they are harder to combine, while in the Greedy procedure the best possible cutting patterns are generated first, with no concern about the quality of the future cutting patterns. Next we modify these two procedures to deal with the case of usable leftover.

3.1 FFD_L procedure

We modify the FFD procedure aiming to avoid not so small scraps, i.e., trying to obtain at least, acceptable solutions.

The FFD_L procedure applies FFD to generate a pattern and, just after the generation, the leftover is analysed. If it is an acceptable leftover (a small scrap or a retail) the pattern is accepted. Otherwise, we take out an item (the largest one) of the pattern. With the resulting empty space we solve the knapsack problem (1)-(3), with the capacity b equal to the leftover material of the generated pattern plus the size of the item withdrawn. After solving the knapsack problem, the resulting leftover is analysed and if it is not acceptable, an additional item (second largest) of the original pattern is withdrawn. Again, with the space generated a new knapsack problem (1)-(3) is solved. In case we have withdrawn an item of each size among all items in the pattern, we withdraw again another piece of the largest item. This procedure is repeated until the leftover obtained is acceptable or the initial cutting pattern becomes null. In this latter case, the cutting pattern is the solution of knapsack problem (1)-(3).

Next, we present the main steps of algorithm FFD_L. We denote by e_k the quantity of objects type k, $k = 1, \ldots, K$, available in stock.

FFD_L algorithm

$k = 1$

***Step* 1:** {*Start*}

If $e_k = 0$ then $k = k + 1$, go to Step 9.

else

Apply the FFD procedure for an object type k and obtain a FFD pattern, whose associated vector is α_k^{FFD}.

***Step* 2:** {*leftover analysis: FFD pattern*}

If the leftover is acceptable (i.e., small scrap or retail) then
Begin
pattern α_k^{FFD} is accepted {temporarily stored in list B}.
$k = k + 1$, go to Step 9.
End

***Step* 3:** {*Item to be withdrawn from the FFD pattern*}
i = index of the longest item in α_k^{FFD}.

***Step* 4:** {*Change pattern with undesirable leftover*}
Withdraw one unit of item i from α_k^{FFD};
SPACE = leftover of pattern FFD + ℓ_i;
Solve knapsack problem (1)-(3) with capacity b = SPACE, considering the items that are already in the pattern FFD and obtain the knapsack pattern, whose associate vector is denoted by $\alpha^{knapsack}$;

***Step* 5:** {*Selecting a pattern when all items are withdrawn from the FFD pattern*}
If $\alpha_k^{FFD} = 0$, then
Begin
While the leftover is not acceptable do:
Begin
Let p = index of the longest item in pattern $\alpha^{knapsack}$.
Withdraw one unit of item p of $\alpha^{knapsack}$.
End
If $\alpha^{knapsack} \neq 0$ then
Begin
pattern $\alpha^{knapsack}$ is accepted {temporarily stored in list B}
End
Else
Begin
pattern $\alpha^{knapsack}$ obtained in Step 4 is accepted {temporarily stored in list C}
End
$k = k + 1$, go to Step 9
End

***Step* 6:** {*New pattern* }
New pattern (FFD + Knapsack): $\alpha_k^{FFD} + \alpha^{knapsack}$;

***Step* 7:** {*evaluation of the leftover material: pattern FFD + Knapsack*}
If the leftover is acceptable then
Begin
pattern $\alpha_k^{FFD} + \alpha^{knapsack}$ is accepted {temporarily stored in list B};
$k = k + 1$, go to Step 9;
End

***Step* 8:** {*Item to withdraw from pattern FFD - index update*}
i = next item to be withdrawn from α_k^{FFD} according to Procedure Withdraw_items (the pattern FFD + Knapsack is disconsidered);
Go to Step 4.

***Step* 9:** *{Determination of the best cutting pattern, frequency, demand and stock update}*

If $k = K + 1$ then

Begin

If list B = ∅ then

Begin

Select among the cutting patterns in list C the one that presents the smallest scrap and store;

End

Else

Begin

Select among the cutting patterns in list B the one that presents the smallest scrap or if there is no pattern with scrap, the one that presents the smallest retail and store;

End

Determine the frequency of the chosen pattern (under demand and stock restrictions) and store;

Update d_i, $i = 1, ..., m$ and e_k, $k = 1, ..., K$.

If $d_i = 0$, $i = 1, ..., m$ (feasible solution found) or $e_k = 0$, $k = 1, ..., K$ (demand not fulfilled) the STOP.

Else $k = 1$ and empty lists B and C.

End

Go to Step 1.

Procedure Withdraw_items

Withdraw the largest items in non-increasing order, one type at a time. In case one item of each type is withdrawn, among all item types in the original pattern, withdraw the remaining largest items in non-decreasing order, one type at a time. Repeat following the same reasoning.

For each object type k in stock, a cutting pattern is built and the one with the smallest scrap is used with a frequency defined in Step 9. If there is no pattern with scrap, the selected pattern is the one having the smallest retail. The pattern with the smallest retail is chosen in order to avoid solutions as observed in Figure **1. e** – Solution 3.

3.2 $Greedy_L$ procedure

The $Greed_L$ procedure consists in applying the Greedy procedure to obtain a cutting pattern and observe the leftover. If acceptable (small scrap or retail), the pattern is accepted, otherwise the largest item in the pattern is withdrawn and the leftover is analysed again. If the pattern is still unacceptable the second largest item is withdrawn. This process is repeated until we have an acceptable pattern or a null pattern. If the pattern is null, we choose among the original patterns, the one that presents the smallest leftover (this is not a typical situation but it can occur, for instance, when the stock is formed only by retails).

Next, we present the mais steps of the $Greedy_L$ algorithm.

$Greedy_L$ algorithm

$k = 1$

***Step* 1:** *{Start}*

If $e_k = 0$ then $k = k + 1$, go to Step 3.

Else

Apply the Greedy procedure for an object type k and obtain a greedy pattern, whose associated vector is α_k^{greedy}.

***Step* 2:** {*Leftover evaluation: pattern* α_k^{greedy}}

If the leftover is acceptable then
Begin
pattern α_k^{greedy} is accepted {temporarily stored in list B}.
End
Else {*modification on Greedy pattern*}
Begin
While leftover is not acceptable do:
Begin
Let i = index of the largest item in pattern α_k^{greedy}.
Withdraw one unit of item i from α_k^{greedy}.
End
If $\alpha_k^{greedy} \neq 0$ then
Begin
pattern α_k^{greedy} is accepted {temporarily stored in list B}.
End
Else
Begin
pattern α_k^{greedy} obtained in Step 1 is accepted {temporarily stored in list C}.
End
$k = k + 1$, go to Step 3.
End

***Step* 3:** {*Determination of the best cutting pattern, frequency, demand and stock update*}

If $k = K + 1$ then
Begin
If list B = ∅ then
Begin
Select among the cutting patterns in list C the one that presents the smallest leftover and store;
End
Else
Begin
Select among the cutting patterns in list B the one that presents the smallest scrap or if there is no pattern with scrap, the one that presents the smallest retail and store;
End
Determine the frequency of the chosen pattern (under demands and stock constraints);
Update d_i, $i = 1, ..., m$ and e_k, $k = 1, ..., K$.
If $d_i = 0$, $i = 1, ..., m$ (feasible solution found) or $e_k = 0$, $k = 1, ..., K$ (demand not fulfilled) then STOP.
Else $k = 1$ and empty lists B and C.
End
Go to Step 1.

In general, in the FFD_L and $Greedy_L$ procedures, the last selected patterns are composed with large items, that are more difficult to combine with the other items. These patterns generally present large leftover, that is, retails, that are not totally undesirable.

4 Residual Heuristics

Residual heuristics find an integer solution for the unidimensional CSP from the linear relaxation of the Integer Programming problem

$$\text{Minimize } f(\mathbf{x}) = \sum_{k=1}^{K}\sum_{j=1}^{N_k} c_{jk}x_{jk} \qquad (4)$$

subject to:

$$\sum_{k=1}^{K}\sum_{j=1}^{N_k} \alpha_{ijk}x_{jk} = d_i, \qquad i = 1, ..., m \qquad (5)$$

$$\sum_{j=1}^{N_k} x_{jk} \le e_k, \qquad k = 1, ..., K \qquad (6)$$

$$x_{jk} \ge 0 \text{ and integer} \qquad j = 1, ..., N_k, k = 1, ..., K \qquad (7)$$

where

c_{jk} is the leftover material in cutting pattern j for the object type k in stock, given by $c_{jk} = L_k - \sum_{i=1}^{m} \ell_i \alpha_{ijk}$, $j = 1,..., N_k$, $k = 1,..., K$, and

x_{jk} is the decision variable that represents the number of objects type k cut according to pattern j, $j = 1, ..., N_k$, $k = 1, ..., K$;

$\boldsymbol{\alpha}_{jk} = (\alpha_{1jk}, \alpha_{2jk}, ..., \alpha_{mjk})$ is a cutting pattern, where α_{ijk} is the number of items type i in cutting pattern j of object type k, $i = 1, ..., m$, $j = 1, ..., N_k$, $k = 1, ..., K$. In the unidimensional case, for each object type k, any cutting pattern must satisfy

$$\begin{cases} \ell_1\alpha_{1jk} + \ell_2\alpha_{2jk} + ... + \ell_m\alpha_{mjk} \le L_k \\ 0 \le \alpha_{ijk} \le d_i \text{ , } i = 1, ..., m \text{ and } \alpha_{ijk} \text{ integer.} \end{cases} \qquad (8)$$

In model (4)-(7), the objective function (4) minimizes the total leftover material; constraints (5) guarantee that the quantity of items cut is exactly the required demand (any piece cut that is not a required item is considered *leftover*). Constraints (6) guarantee that the number of objects type k cut does not exceed its availability e_k, $k = 1, ..., K$ and finally, constraints (7) guarantee that the number of times each pattern is cut is a non-negative integer number.

For simplicity of notation, model (4)-(7) is written in matricial form as:

$$\text{Minimize } f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} \qquad (9)$$

subject to

$$\mathbf{Ax} = \mathbf{d} \qquad (10)$$

$$\mathbf{Ex} \le \mathbf{e} \qquad (11)$$

$$\mathbf{x} \ge \mathbf{0} \text{ and integer} \qquad (12)$$

where **A** is the cutting patterns matrix in (5) and **E** is a matrix of 0's and 1's on constraints (6).

The integrality condition on the variables x_{jk} complicates the solution of the problem as m increases (it is already difficult when m is of the order of a few dozens). Gilmore e Gomory (1961) proposed to solve the problem using *column generation* after relaxing the integrality constraints on the decision variables x_{jk}. From the optimal solution of the relaxed problem, that usually is not integer, we determine an integer solution for the original CSP. This integer solution can be determined by heuristics that have been developed by several researchers, like Wäscher and Gau (1996), Poldi and Arenales (2005), Stadtler (1990), among others. Some of these heuristics are presented in subsections 4.1, 4.2 and 4.3.

Definition 2: *Let* **x** *be an optimal fractional solution for the linear realxation problem* (9)-(12) *and let* **y** *be a vector of non-negative integer numbers, close in some sense to* **x**, *such that:*

$$\mathbf{Ay} \leq \mathbf{d} \tag{13}$$
$$\mathbf{Ey} \leq \mathbf{e} \tag{14}$$

The vector **y**, *obtained from* **x**, *is called an approximate integer solution of* **x**.

A possible way to obtain an approximate integer solution **y** of **x** is by a simple truncation:

$$\mathbf{y} = (\lfloor x_1 \rfloor, \lfloor x_2 \rfloor, \ldots, \lfloor x_n \rfloor) \tag{15}$$

that satisfies (13)–(14) since all coefficients of **A** and **E** are non-negative, **x** satisfies $\mathbf{Ax} = \mathbf{d}$ and $\mathbf{Ex} \leq \mathbf{e}$.

Definition 3: (Residual Problem): *Let* **y** *be an approximate integer solution of* **x**, $\mathbf{r} = \mathbf{d} - \mathbf{Ay}$ *the residual demand and* $\mathbf{s} = \mathbf{e} - \mathbf{Ey}$ *the residual stock of the available objects. The residual problem is defined as* (9) – (12) *with* $\mathbf{d} = \mathbf{r}$ *and* $\mathbf{e} = \mathbf{s}$.

In residual heuristics we solve a linear relaxation of problem (9)-(12) and we obtain an approximate integer solution. We then solve a linear relaxation of the residual problem (definition 3) and we obtain an approximate integer solution and, so on successively, until the residual demand is null or the approximate integer solution is null. In the latter case, we apply some method (heuristic or exact) to solve the residual problem with just a few items. Next, we present a general structure of these heuristics.

Residual Algorithm (from Poldi and Arenales (2005))

***Step* 1:** {*Start*}
Do $\ell = 0$, $\mathbf{r}^0 = \mathbf{d}$, $\mathbf{s}^0 = \mathbf{e}$;

***Step* 2:** {*Determining the continuous optimal solution*}
Solve the residual problem with $\mathbf{r} = \mathbf{r}^\ell$ and $\mathbf{s} = \mathbf{s}^\ell$;
Let $\mathbf{x}^\ell$ be the continuous solution (column generation is used);
If $\mathbf{x}^\ell$ is integer, then STOP.

***Step* 3:** {*Determining an approximate integer solution*}
Determine $\mathbf{y}^\ell$, the approximate integer solution of $\mathbf{x}^\ell$.
If $\mathbf{y}^\ell$ is a null vector, then go to the Step 5.

***Step* 4:** {*Update*}
Determine the new residual demand
$\mathbf{r}^{\ell+1} = \mathbf{r}^\ell - \mathbf{A}\,\mathbf{y}^\ell$;
$\mathbf{s}^{\ell+1} = \mathbf{s}^\ell - \mathbf{E}\,\mathbf{y}^\ell$;
$\ell = \ell + 1$.
Go to Step 2.

***Step* 5:**
Solve the final residual problem with a few items by some method, heuristic or exact.

4.1 Residual Heuristics FFD, Greedy, FFD_L and $Greedy_L$

In order to completely define the previous residual algorithm, it is necessary to specify how $\mathbf{y}^{\ell}$, the approximate integer solution of $\mathbf{x}^{\ell}$, is detemined in Step 3, and how to solve the residual problem in Step 5. We denote the residual heuristics FFD, Greedy, FFD_L and $Greedy_L$ those obtained applying the residual algorithm where in Step 3 the approximate integer solution is defined by (15) (that is, $\mathbf{y}^{\ell}$ is determined by the floor of the fractional numbers obtained) and, in Step 5, if there are still items with demand to be satisfied, we use heuristics FFD, Greedy, FFD_L or $Greedy_L$, respectively.

4.2 Residual Heuristic by Greedy Rounding (RGR)

Poldi and Arenales (2005) developed a greedy rounding procedure to obtain an approximate integer solution of a continuous solution **x** in Step 3 of the residual algorithm.

In Poldi and Arenales's greedy rounding procedure, Step 3 of the residual algorithm is divided in two parts: *Pre-processing* Step and *Rounding* Step. For these heuristics, refered to as RGR heuristics, the Step 5 of the residual heuristic never occurs for all demand is satisfied since at least for one pattern its frequency will be rounded up to its nearest integer and, hence, $\mathbf{y}^{\ell}$ will never be null in Step 3.

In the *Pre-processing* Step we order the cutting patterns of the continuous solution $\mathbf{x}^{\ell}$ (obtained in Step 2 of the residual algorithm) according to RGR 1, RGR 2 and RGR 3, described ahead. To simplify the notation, consider T cutting patterns with non null frequencies obtained in Step 2, enumerated by 1, 2, ..., T and k_j the object associated to cutting pattern $j, j = 1,..., T$.

- **RGR 1**: the cutting patterns are ordered by their frequencies values in non-increasing order, that is, $x_{1k_1} \geq x_{2k_2} \geq ... \geq x_{Tk_T}$;
- **RGR 2**: the cutting patterns are ordered by their leftover values in non-decreasing order, that is, $c_{1k_1} \leq c_{2k_2} \leq ... \leq c_{Tk_T}$;
- **RGR 3**: let $f_{jk} = x_{jk} - \lfloor x_{jk} \rfloor$ the fractional part of x_{jk}. The cutting patterns are ordered by their corresponding fractional parts in non-increasing order, that is, $f_{1k_1} \geq f_{2k_2} \geq ... \geq f_{Tk_T}$.

In the *Rounding* Step, we start with the first cutting pattern, following one of the previous orderings, and we round up its frequency, that is, $y_{1k_1} = \lceil x_{1k_1} \rceil$. The other frequencies are set to 0, that is, $y_{jkj} = 0, j = 2, ..., T$. Conditions (13) and (14) are tested and, in case of violation, y_{1k_1} is reduced successively by a unit, that is, $y_{1k_1} = y_{1k_1} - 1$, unitl we get an approximate integer solution. The value of y_{1k_1} is fixed and we repeat the procedure with the second pattern. We determine a new approximate integer solution y_{1k_1}, y_{2k_2} and $y_{jkj} = 0, j = 3, ..., T$. We repeat the procedure for all cutting patterns.

To solve the CSPUL, we adapted this greedy rounding to obtain heuristics RGR_L – versions 1, 2 and 3.

4.3 RGR Heuristics for the CSPUL

In the RGR_L heuristics we use a bound for the maximum acceptable fraction for a scrap, that is defined from the approximate integer solution obtained using one of the versions of the RGR heuristic (in the constructive heuristics, this bound is defined by the user, for standard and non-standard objects).

Computation of the Maximum Acceptable Fraction for a Scrap

1. Consider an approximate integer solution obtained by one of the versions of the RGR heuristic (Section 4.2).
2. Determine the total leftover, excluding the retails (that is, leftover material in a cutting pattern larger or equal to δ):

$$\sigma = \sum_{i/c_{ik_i}<\delta} c_{ik_i} x_{ik_i}$$

3. Determine the total length of the objects cut with scraps:

$$\gamma = \sum_{i/c_{ik_i}<\delta} L_{ik} x_{ik_i}$$

4. The maximum acceptable fraction (small scrap) is given by:

$$\rho = \frac{\sigma}{\gamma}$$

Let $\lambda_{ik_i} = \frac{c_{ik_i}}{L_{k_i}}$ be the fraction lost in pattern i, i = 1,..., T of object k_i. The main steps of heuristics RGR_L versions 1, 2 and 3 are:

RGR_L Algorithm

***Step* 1:** {*Start*}
Determine an approximate integer solution using the greedy rounding procedure according to criterion RGR (versions 1, 2 or 3).

***Step* 2:** {*Evaluation of the leftover material in the patterns*}
Analyse the leftover material of all patterns generated in Step 1;
If the leftover is *acceptable*, that is, $\lambda_{ik_i} \leq \rho$ (obtained from the computation of the maximum acceptable fraction for a scrap) then
pattern i of object type k_i is accepted and stored;
Else
pattern i is rejected and demand of the items in pattern i and stock of the object type k_i are updated;

***Step* 3:**
Apply the FFD_L procedure to solve the residual problem formed by the items of the rejected patterns in Step 2 and the remaining objects.

Other procedures to solve the residual problem (Step 3) can be used to generate different heuristics. We suggest the FFD_L procedure because, as we will see later in the computational test results, this heuristic generates less scrap when compared with the FFD and Greedy heuristics, and a smaller quantity of objects cut with retail compared to $Greed_L$ heuristic.

5 Computational Tests

To evaluate the heuristics described in sections 3 and 4, 16 classes of instances were considered. For each class, 20 instances were randomly generated. For these randomly generated

classes, we also present the results obtained using the COLA algorithm developed by Gradisar *et al.* (1997). Algorithm COLA minimizes the loss of material and/or tries to concentrate them in an object so that they become retail. We consider retail any material left with length larger or equal to the average of the lengths of the required items. Computational tests are also presented with instances from Trkman (2005) and practical instances from Abuabara (2006).

To classify the solutions (*ideal*, *acceptable*, or *undesirable*) obtained according to definition 1, we use the values $\xi_1 = 0.03$ and $\xi_2 = 0.1$, except for instances 4 to 6, that present small demand.

To classify the solutions according to definition 1 we require only the number of objects cut with small scraps, not so small scraps or with retails. But we present other results that may be useful for the user/decision maker to choose a solution (the concept of *ideal*, *acceptable* or *undesirable* solution may always be revised by the user/decision maker and include other features). The additional data we present are: the number of objects cut (standard and non-standard), total length lost (sum of the scraps) and the total length of the non-standard objects generated (sum of the retails). Note that, it is desirable that the total length lost is small, but not necessarily this is valid for the total length of the retails. Observe that with these data it is possible to compute the total length of the objects used which may be also useful information.

All the heuristics including the COLA algorithm were implemented in DELPHI 6. The experiments were executed in a Pentium IV (3 GHz, 2 GB RAM) microcomputer. The implicit enumeration method described in Gilmore and Gomory (1961) was used to solve the knapsack problems that arise in the heuristics and in the column generation.

5.1 Results using Instances of the Literature

In this section we present numerical instances from Trkman (2005) and practical instances from Abuabara (2006).

In the instances of Trkman (2005) we have several types of objects in stock but only one of each type, that is, $e_k = 1$ for all k. This is a very special situation where the variables x_{jk} are binary. Due to the special characteristic of these instances, we treated the objects as non-standards and we set only the parameter β to 0.005.

For these instances we have information about the solution given by the CUT algorithm, therefore, we used the same criteria adopted by Gradisar *et al.* (1997) for *scrap* and *retail*, that is, all the material left after cutting an object that is greater or equal to the smallest item demanded is considered *retail*, otherwise it is *scrap*.

In the following tables we classify the solutions according to definition 1 and we use **ID** to denote an *ideal solution*, **AC** to denote an *acceptable solution*, and **UND** to denote an *undesirable solution*. Also we use Obj.Cut. to denote the quantity of objects cut, Tot.Length to denote the total length of the objects in stock cut, Total Loss to denote the total length of scraps, Total Ret. to denote the total length of the retails, OSScrap to denote the number of objects cut with small scrap, ONSScrap to denote de number of objects cut with not so small scrap, and ORetail to denote the number of objects cut with retail.

Instance 1. $K = 20$ types of objects with lengths between 2,200 and 6,000 cm; availability of one unit of each type of object and $m = 5$ items demanded according to Table 1.

Table 1 - **Data of instance 1 – Items**

Item	Length (cm)	Demand
1	235	4
2	200	51
3	347	42
4	471	16
5	274	37

From Table 1, $\delta = 200$ cm since this is the length of the smallest item. In Table 2 we present the computational test results obtained.

Table 2 – **Solution of Instance 1**

		Constructive				Residual									
	CUT	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
Obj.Cut.	11	11	13	12	12	11	10	11	11	10	10	10	10	11	10
Tot.Length	44136	44027	48506	44715	46104	46243	43803	46243	46234	44079	45245	44079	45245	47141	46507
Total Loss	5	639	8	39	1	331	3	131	3	0	0	0	0	19	0
Total Ret.	743	0	5110	1288	2715	2524	412	2724	2852	691	1857	691	1857	3734	3119
OSScrap	3	1	3	1	1	0	2	1	1	0	0	0	0	0	0
ONSScrap	0	10	0	3	0	3	0	1	0	0	0	0	0	1	0
ORetail	1	0	7	1	4	1	1	1	2	1	1	1	1	2	2
Solution	AC	UND	UND	UND	UND	UND	AC	AC	AC	**ID**	**ID**	**ID**	**ID**	AC	AC

From Table 2 we observe that the heuristics RGR_L – versions 1, 2 use less objects than the CUT algorithm but the total length of the objects is longer, that is, they use longer objects compared to CUT. This was expected since CUT gives priority to smaller objects first. We also notice that similar to algorithm CUT, heuristics RGR_L – versions 1 e 2 concentrate the leftover in a single object. With respect to losses, none of these two heuristics generate losses performing better than algorithm CUT with respect to this criterion. Note that, in general, the derived heuristics with usable leftover perform better compared to their original version by definition 1. For this instance, the classification of the solutions obtained by each algorithm is given in the last row of the Table, using definition 1 and the parameters defined previously.

Instance 2: $K = 20$ types of objects with lengths between 2,100 and 5,000 cm; availability of one unit of each type of object and $m = 5$ items demanded according to Table 3.

Table 3 – **Data of instance 2 – Items**

Item	Length (cm)	Demand
1	549	39
2	433	27
3	207	43
4	308	39
5	583	2

For this instance, $\delta = 207$ cm, the length of the smallest item.

Table 4 – **Solution of Instance 2**

		Constructive				Residual									
	CUT	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
Obj.Cut.	15	15	16	16	16	15	15	15	15	15	15	16	16	16	14
Tot.Length	56302	56581	58356	57100	57100	56256	56256	56256	56256	56861	57554	57313	58159	59929	56395
Total Loss	104	925	17	69	17	47	9	9	9	114	6	69	3	17	7
Total Ret.	1017	475	3158	1850	1902	1028	1066	1066	1066	1566	2367	2063	2972	4731	1207
OSScrap	7	2	8	8	8	2	5	5	5	4	4	4	4	5	5
ONSScrap	3	12	0	1	0	3	0	0	0	1	0	2	0	1	0
ORetail	1	1	3	1	2	1	1	1	1	2	2	2	3	3	2
Solution	UND	UND	UND	AC	AC	UND	AC	AC	AC	AC	AC	AC	AC	UND	AC

In this instance the residual heuristics $Greedy_L$, and FFD_L presented a better solution than algorithm CUT, according to definition 1. The majority of the solutions generated by the other residual heuristics proposed also present acceptable solutions.

Instance 3: $K = 90$ types of objects with lengths between 3,000 and 9,000 cm; availability of one unit of each type of object and $m = 15$ item types demanded according to Table 5.

Table 5 – **Data of instance 3 – Items**

Item	Length (cm)	Demand
1	569	34
2	718	26
3	520	25
4	540	12
5	492	30
6	547	2
7	632	6
8	430	36
9	750	7
10	387	20
11	804	3
12	389	32
13	835	18
14	684	39
15	687	10

For this instance, δ is equal to 387 cm.

Table 6 - **Solution of Instance 3**

		Constructive				Residual									
	CUT	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
Obj.Cut.	27	27	29	30	30	26	26	25	25	22	22	22	22	24	22
Tot.Length	170504	172055	170343	175473	173814	174176	174176	173896	173896	170621	172114	170621	172114	175742	170989
Total Loss	2	2009	14	224	0	1350	5	19	0	0	0	0	0	0	0
Total Ret.	1456	1000	1283	6203	4768	3780	5125	4831	4850	1575	3098	1575	3098	6696	1943
OSScrap	2	7	9	0	0	0	4	0	0	0	0	0	0	0	0
ONSScrap	0	17	0	4	0	12	0	1	0	0	0	0	0	0	0
ORetail	1	1	1	2	6	1	2	2	3	2	1	2	1	3	1
Solution	**ID**	UND	AC	UND	UND	UND	AC	AC	AC	AC	**ID**	AC	**ID**	AC	**ID**

For this larger instance, we observe from Table 6 that the solutions presented by the residual heuristics RGR$_L$ – verions 1, 2 and 3 and by the CUT algorithm were considered *ideal*, according to definition 1. Also, if we compare the solutions obtained with the modified heuristics compared with the ones obtained with the original procedures, we clearly observe improvements in quality, following definiton 1.

We presented here the computational test results of 3 instances only, although we received and tested a total of eight instances. In all the 8 instances we observed a similar behaviour, i.e., the performance of the modified heuristics were better or equivalent to the performance of algorithm CUT, according to definiton 1. The results of the other instances can be seen in Cherri (2006).

The next set of instances is from Abuabara (2006), whose work is based on the models proposed by Gradisar *et al.* [3]. The instances are from the portfolio of demands of a small Brazilian agricultural airplane industry that cuts methalic tubes to build its airplanes whose structure are formed with lattice porticoes.

The instances presented, which characterize small industries, are not classified according to definition 1 since the values $\xi_1 = 0.03$ and $\xi_2 = 0.1$ that we planned to use are not appropriate because the total number of objects cut is small (small demand) therefore yielding a very small bound for the quantity of objects with small scraps, not so small scraps and retails. The choice of the best solution according to definition 1 can be made after the quantities "*small*", "*very small*" and "*several*" are defined by the decision maker.

In the following examples, we have a single type of object in stock, that is, $e_k = 1$, $k = 1$. Since the stock has standard objects only we define solely the parameter θ and we use $\theta = 0.005$. The minimum size of the retails is $\delta = \min\{\ell_i, i = 1, ..., m\}$.

Instance 4: The length of the objects in stock is 3,000 cm and there are 10 of them. $m = 5$ item types demanded according to Table 7.

Table 7 – **Data of instance 4 – Items**

Item	Length (cm)	Demand
1	250	2
2	273	2
3	285	4
4	525	4
5	1380	4

Any leftover material on a cutting pattern larger than or equal to 250 cm is considered a retail.

Table 8 – **Solution to Instance 4**

	Constructive				Residual									
	FFD	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
Obj.Cut.	4	5	4	5	4	5	4	5	4	5	4	5	4	5
Tot.Length	12000	15000	12000	15000	12000	15000	12000	15000	12000	15000	12000	15000	12000	15000
Total Loss	525	0	240	0	240	0	240	0	240	0	240	0	244	4
Total Ret.	1669	5194	1954	5194	1954	5194	1954	5194	1954	5194	1954	5194	1950	5190
OSScrap	0	0	0	0	0	0	0	0	0	0	0	0	1	1
ONSScrap	3	0	1	0	1	0	1	0	1	0	1	0	1	0
ORetail	1	3	1	3	1	3	1	3	1	3	1	3	1	3

From Table 8 we observe that the RGR$_L$ algorithms – versions 1, 2, do not generate scraps but they present large quantities of retails when compared to the original heuristics. This occurs since the usable leftover heuristics tend to eliminate the not so small scraps.

In this instance and the next ones, we are faced with solutions where we have loss of material but a single retail or no loss of material but a larger number of retails. This is a typical situation that we face when we solve this problem; we rarely obtain a solution that is good considering all the criteria. So, the choice of the best solution is of the decision maker since he/she knows better the reality of the firm.

Instance 5: The length of the objects in stock is 6,000 cm and there are 10 of them. $m = 4$ item types demanded according to Table 9.

Table 9 - **Data of instance 5 – Items**

Item	Length (cm)	Demand
1	370	5
2	905	5
3	910	5
4	930	5

Any leftover material on a cutting pattern larger than or equal to 370 cm is considered a retail.

Table 10 – **Solution to Instance 5**

	Constructive				Residual									
	FFD	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
Obj.Cut.	3	4	3	3	3	4	3	3	3	3	3	3	3	3
Tot.Length	18000	24000	18000	18000	18000	24000	18000	18000	18000	18000	18000	18000	18000	18000
Total Loss	250	0	250	0	250	0	250	0	515	150	515	150	515	150
Total Ret.	2175	8425	2175	2425	2175	8425	2175	2425	1910	2275	1910	2275	1910	2275
OSScrap	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ONSScrap	2	0	2	0	2	0	2	0	2	1	2	1	2	1
ORetail	1	4	1	3	1	4	1	3	1	2	1	2	1	2

As we can observe, the original heuristics generate only a single retail to stock but all of them present larger losses compared to the losses of the modified heuristics. In the RGR$_L$ heuristics–

versions 1, 2 and 3, it was not possible to eliminate all the not so small scraps, but the quantity of retails is smaller compared to the other modified heuristics. The choice of the best solution is not trivial since it involves the simultaneous analysis of several features. The option of generating only a single retail leads to significantly larger losses while the solutions with no loss have a significant number of retails. Again, the instance shows the conflicting nature of the objectives and the decision maker must make his/her choice.

Instance 6: The length of the objects in stock is 6,000 cm and there are 15 of them. $m = 7$ item types demanded according to Table 11.

Table 11 – **Data of Instance 6 – Items**

Item	Length (cm)	Demand
1	350	12
2	540	3
3	705	3
4	735	6
5	760	6
6	890	6
7	900	3

Any leftover material on a cutting pattern larger than or equal to 350 cm is considered a retail.

Table 12 – **Solution to Instance 6**

	Constructive				Residual									
	FFD	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
Obj.Cut.	5	5	5	5	5	5	5	5	5	5	5	5	5	5
Tot.Length	30000	30000	30000	30000	30000	30000	30000	30000	30000	30000	30000	30000	30000	30000
Total Loss	455	30	0	0	455	0	140	0	305	0	105	0	110	0
Total Ret.	4600	5025	5055	5055	4600	5055	4915	5055	4750	5055	4950	5055	4945	5055
OSScrap	1	2	0	0	0	0	0	0	0	0	0	0	0	0
ONSScrap	3	0	0	0	2	0	1	0	1	0	1	0	1	0
ORetail	1	2	2	2	1	2	1	2	1	2	1	2	1	2

To satisfy all demand only a small number of objects needed to be cut in this instance (at most 5 objects), as can be observed from Table 12. The heuristics modified to deal with usable leftover produced solutions with reduced losses compared with the solutions given by the original procedures.

For these instances, solutions with the mathematical models of Gradisar et al. (1997) or Abuabara (2006) can be obtained. In these models constraints on the number of retails are imposed. The solutions obtained with these models are, in general, similar to the solutions obtained using the modified heuristics. When we consider a great variety of object types in stock with large availability as well as a large variety of item types with large demands, the mathematical models present a large number of variables and advanced solvers like CPLEX, often are not able to produce a solution.

5.2 Results using Randomly Generated Instances

In this section we present the computational results for randomly generated instances according to the instance generator program described in Poldi and Arenales (2005). The data for the instances and for the generator are:

- *Number of standard object types:* $\bar{k} = 2$;
- *Number of non-standard object types:* $k = 3$, 5 and 7;
- *Total number of object types in stock:* $K = 5$, 7 and 9;
- *Number of item types:* $m = 10$, 20 and 40;
- *Availability of standard object types:* $e_1 = e_2 = 100$ objects;

- *Availablilty of non-standard object types:* e_k, $k = 3, \ldots, K$, are randomly generated in the interval [1, 10];
- *Lengths of the standard objects:* $L_1 = 1{,}000$ and $L_2 = 1{,}100$;
- *Length of the items:* The lenght ℓ_i of item type i is randomly generated in the interval $[v_1 L, v_2 L]$, where L is the average value of L_k, $k = 1, 2$, $v_1 = 0.01$, $v_2 = 0.2$ and 0.8. Combining these values, classes of instances are generated. For small items we consider $v_2 = 0.2$ and for average sized items $v_2 = 0.8$. Obviously, small items can be defined using other parameters;
- *Acceptable size of a retail:* $\delta = \left\lfloor \frac{1}{m} \sum_{i=1}^{m} \ell_i \right\rfloor$;
- *Length of the non-standard objects:* L_k, $k = 3, \ldots, K$ are randomly generated in the interval $\left[\delta, \frac{L_1}{2}\right]$;
- *Demand:* d_i is randomly generated in the interval $\left[0.02 \frac{\sum_{k=1}^{K} e_k L_k}{\sum_{i=1}^{m} \ell_i}, \frac{\sum_{k=1}^{K} e_k L_k}{\sum_{i=1}^{m} \ell_i} \right]$, $i = 1, \ldots, m$.

  Combining these values, we guarantee that the total length of the demanded items does not exceed the total length of the objects in stock;
- *Maximum size of small scrap for standard objects:* $0.005 L_k$, $k = 1, 2$, that is, $\theta = 0.005$;
- *Maximum size of small scrap for non-standard objects:* $0.05 L_k$, $k = 3, \ldots, K$, that is, $\beta = 0.05$.

In the generated instances we defined a larger percentagewise acceptable loss for the non-standard objects, increasing the number of possible patterns considered acceptable for these objects. With this, we expect to increase the chances that a pattern using a non-standard object appears in a solution.

We considered 16 classes of instances combining the parameters K = (5, 7 or 9), m = (10, 20 or 40), $v_2 = 0.2$ (small items: S) and $v_2 = 0.8$ (average sized items: A). In Table 13 these 16 classes of the generated instances are described. For example, class 1 contains instances with $K = 5$ types of objects, that are cut to produce $m = 10$ item types, whose lengths are randomly generated in the interval $[0.01L\ 0.2L]$, with $L = (L_1 + L_2)/2 = 1{,}050$. Class 2 instances have the same size of the instances in Class 1 in terms of number of object types and item types, except the lengths of the items are randomly generated in the interval $[0.01L\ 0.8L]$.

Table 13 – Description of the classes

	Parameters		
Class	K	m	Items
1	5	10	S
2	5	10	A
3	5	20	S
4	5	20	A
5	5	40	S
6	5	40	A
7	7	10	S
8	7	10	A
9	7	20	S
10	7	20	A
11	7	40	S
12	7	40	A
13	9	10	S
14	9	10	A
15	9	20	S
16	9	20	A

20 instances of each one of the classes in Table 13 were generated and the average computational test results obtained for the 320 instances (16 classes X 20 instances) are presented in the next tables. In these tables we present:

i. The average total number and average total length of non-standard and standard objects used (Tables 14, and 15);
ii. The average total loss, average total length of the losses in the standard objects and in the non-standard objects (Table 16);
iii The average total length of the retails, average total length of the retails in the standard objects and in the non-standard objects (Table 17);
iii. The average number of objects cut with: retail, small scrap, and not so small scraps (Table 18);
iv. The average percentage of objects cut with: small scrap, not so small scrap, and retail (Table 19);
v. The average execution time (Table 20);
vi. The general classification of the heuristics (Table 21).

In each row of the tables, the largest and smallest values obtained are marked in bold and in italic, respectively. The detailed values of the test results of each table are given in the Appendix.

Table 14 – Average number and average total length of non-standard objects used

		Constructive Heuristics				Residual Heuristics									
	COLA	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
Average number	**28.1**	11.0	8.6	8.4	9.6	9.1	5.8	5.0	4.8	3.7	4.3	4.0	4.4	*3.5*	4.5
Average length	**7065.3**	2445.0	1820.4	1865.5	2510.3	1827.0	1058.1	911.9	1014.7	841.3	1025.4	884.8	1041.6	*800.9*	1052.5

Different from the modified heuristics developed, in algorithm COLA the objects are ordered in non-decreasing length and, for each object in this sequence a cutting pattern is built. Therefore, algorithm COLA gives priority to non-standard objects and uses the largest number and length of these objects in the solution (Table 14). Among the original and modified heuristics, constructive FFD and Greedy$_L$ are the ones that use more non-standard objects, and RGR – versions 1, 2 and 3 use less of these objects.

We must point out that the use of non-standard objects was not a priority in the heuristics developed. Just a larger tolerance for small scraps was considered. In case it is relevant to reduce the non-standard objects in stock, the heuristics can be revised, possibly in detriment of other desirable features.

Table 15 - Average number and average total length of standard objects used

		Constructive Heuristics				Residual Heuristics									
	COLA	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
Average number	104.0	105.8	108.5	109.0	**110.9**	101.8	102.7	*101.6*	102.9	102.3	102.6	102.2	102.5	102.3	102.6
Average length	*105120.9*	109978.1	111161.9	110649.7	**113147.2**	109886.3	110724.1	110720.9	110816.9	110612.5	110932.2	110589.7	110881.9	110655.0	110926.9

From Table 15 we note that the constructive Greedy$_L$ heuristic uses the largest number and largest total length of standard objects. This can be explained since in the Greedy heuristic it is more likely to get good cutting patterns with longer objects. Algorithm COLA presented the least amount of total length used although it does not use the least number of standard objects. As pointed out earlier, algorithm COLA gives priority to the shortest objects, therefore, it uses first smaller sized objects.

Table 16 –Average total loss, average total length of the losses in the standard objects and in the non-standard objects

		Constructive Heuristics				Residual Heuristics									
	COLA	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
Total loss	126.3	**577.2**	37.1	575.5	29.2	61.3	15.6	73.9	16.3	60.1	*11.5*	65.3	11.8	66.1	12.5
Standard objects	114.9	**552.1**	32.2	456.3	14.7	13.6	11.8	14.7	11.4	16.2	*6.2*	15.7	6.6	21.5	7.0
Non-std objects	11.4	23.5	4.9	**119.3**	14.4	37.1	*3.8*	59.3	4.9	43.4	5.3	49.6	5.3	46.6	5.1

From Table 16, we observe that the modified residual heuristics present better solutions when the losses are analysed. A fact that we must point out is that even allowing a larger loss for the non-standard objects (a way so that they are preferred to standard objects) the solution presented by the usable leftover heuristics are, among the residual heuristics, the ones that present the smallest losses. Taking the algorithm COLA, we observe that in almost all the classes (Table A5) this algorithm present larger losses compared to the usable leftovers heuristics developed. For this algorithm, the best solutions are in the odd classes that are composed of small items where the limit of the size of retail is small (the average of the lengths of the items). But the reduction in loss is balanced with an increase of retails (new non standard objects) that can compromise the quality of the solution, according to definition 1, becoming an undesirable solution.

Table 17 – Average total length of the retails, average total length of the retails in the standard objects and in the non-standard objects

		Constructive Heuristics				Residual Heuristics									
	COLA	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
Total retail	685.8	489.6	1592.6	652.5	**4269.6**	293.1	401.7	204.0	455.2	*34.9*	587.4	39.0	552.8	49.6	601.9
Standard objects	663.3	488.5	1489.2	492.2	**3688.0**	267.9	222.7	108.6	207.5	4.4	383.5	*2.5*	342.0	8.0	383.8
Non-std objects	*0.0*	1.1	103.5	159.1	**581.5**	25.2	178.9	95.3	249.0	30.5	195.7	36.5	210.9	41.6	217.5

According to definition 1, the total length of the retails does not qualify a solution as good or bad since a solution is *ideal* or *acceptable*, as long as the retails are concentrated in a few objects. The information about the lengths of the retails in Table 17 are interesting to observe the effect produced by the heuristics with the modification made to avoid not so small scraps. For example, the construtive FFD$_L$ heuristic reduces the loss, on average, from 577.2 to 37.1 (Table 16), but the total length of the retails increase, on average, from 489.6 to 1592.6 (Table 17).

From Table 17 we observe that constructive heuristic Greedy$_L$ produces on average, the longest retails. This was expected since in this algorithm, when an undesirable loss in a pattern is observed, it is transformed into retail.

Algorithm COLA does not generate retails in non-standard objects because these objects are cut first, when the demand of all or most of the items are not fulfilled yet. Therefore, it is likely that good patterns for these objects are produced at these earlier stages.

In terms of definition 1, it is more relevant the amount of objects that present retails rather than their total length. In definition 1 this quantity was assumed relevant because it dictates the amount of non-standard objects in the future. The choice of the best heuristic to solve the CSPUL is not trivial since it must consider many conflicting factors simultaneously.

Table 18 – Average number of objects cut with: retail, small scraps and not so small scraps

		Constructive Heuristics				Residual Heuristics									
	COLA	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
Retail	3.4	1.0	5.6	3.9	**17.6**	0.7	1.8	0.9	2.5	*0.3*	2.6	*0.3*	2.6	0.5	2.8
Small Scrap	9.0	**45.7**	16.6	8.5	8.9	6.7	3.5	3.3	3.4	*3.1*	3.2	*3.1*	3.2	3.3	3.6
Not Sml Scrap	4.3	**22.5**	*0.0*	11.4	*0.0*	1.9	0.6	2.2	0.6	1.8	0.1	2.0	0.1	2.0	0.1

From Table 18, we observe that the modified heuristics present a larger number of objects with retail compared to their original versions. In general, this occurs because the patterns with not so small scraps are modified by the modified heuristics. Observe, for instance, RGR heuristic – version 1 that presents, on average, 0.3 cut objects with retail while the corresponding modified version, RGR_L – version 1, presents 2.6 cut objects. Observe that although the modfied heuristics do not eliminate all the non-standard objects in stock (Table 14), they generate, on average, a smaller quantity of retails (Table 18) with respect to what was used (except the constructive heuristic $Greedy_L$), hence, the number of non standard objects tends to decrease.

For the randomly generated instances algorithm COLA presented on average, larger values of cut objects with retail, larger values of cut objects with small scraps (objects cut with no loss is not included), and larger values of cut objects with not so small scraps, compared with the values observed of the residual heuristics.

Percentagewise, the average number of objects cut with retail, small scrap and not so small scrap for the 320 randomly generated instances are given in Table 19.

Table 19 – Percentage of objects cut with: retail, small scrap and not so small scrap

		Constructive Heuristics				Residual Heuristics									
	COLA	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
Retail	2.6	0.9	4.8	3.3	**14.6**	0.6	1.7	0.9	2.3	*0.3*	2.4	*0.3*	2.4	0.5	2.6
Small Scrap	6.8	**39.2**	14.2	7.2	7.4	6.0	3.2	3.1	3.2	*2.9*	3.0	*2.9*	3.0	3.1	3.4
Not Sml Scrap	3.3	**19.3**	*0.0*	9.7	*0.0*	1.7	0.6	2.0	0.6	1.7	< 0.1	1.9	< 0.1	1.9	< 0.1

From Table 19 we observe that the original heuristics present a smaller percentage of objects cut with retail. The modified heuristics present a larger percentage of retails because they try to avoid undesirable losses (not so small scraps). This fact can be observed when we analyse the percentage of cut objects with not so small scraps (line Not Sml Scrap in Table 19), where the modified heuristics present improved solutions compared with the ones obtained with the original heuristics and the COLA algorithm.

Table 20 – Average execution time (in seconds)

		Constructive Heuristics				Residual Heuristicss									
	COLA	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
Average	44.85	*0.001*	0.073	0.323	0.367	14.15	15.21	14.25	14.25	22.36	22,55	24.53	22.74	81,60	**81.85**

In Table 20 we present the average execution times observed to solve all the 320 randomly generated instances. We observe that the constructive heuristics are much faster to run than the residual heuristics but all the execution times observed seem to be acceptable in practice.

From the results presented in the previous tables we can classify the heuristics, according to definition 1. This classification is presented in Table 21. In this classification we relaxed the no objects cut with not so small scraps to an percentage of at most 0.1%, for ideal solutions and we used $\xi_1 = 0.03$ and $\xi_2 = 0.10$.

Table 21 – Classification of the heuristics

		Constructive Heuristics				Residual Heuristics									
	COLA	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
Ideal											**X**		**X**		**X**
Acceptable	X		X	X		X	X	X	X	X		X		X	
Undesirable		X			X										

We observe that heuristics RGR_L – versions 1, 2 and 3 were classified as *ideal*, while the other residual heuristics and algorithm COLA were classified as acceptable. Since we have three heuristics with ideal solutions we can use other criteria, defined by the decison maker, to untie them. This criterion may be, for example, the total length of the losses generated, the quantity of new retails generated, etc.

6 Conclusions

In this article we considered the cutting stock problem with usable leftover, that is, if the resulting leftover material of a cut object is large enough it can be used again to cut future demanded items. To deal with this problem, we modified some heuristics of the literature that minimizes the trim loss and we included the possibility of retails (large leftover) that are not computed as losses. A set of desirable characteristics was used to define solutions as *ideal*, *acceptable* and *undesirable*. Still, there exist difficulties in pointing out which method performed better for the solutions present important and conflicting characteristics, like retail, small scrap, not so small scrap, together with their distribution in the cut objects. Other characteristics like the total length of the losses generated, the quantity of new retails generated, can also be included.

The use of mathematical models like those proposed by Gradisar *et al.* (1997, 1999a, 1999b, 2005) and Abuabara (2006) are practically suitable for solving problems with a small quantity of objects and items of moderate sizes but they are computational time consuming for instances with large quantities of objects and items with large demands. The modified heuristics can handle these instances without much effort.

Acknowledgments

The authors are indebted to Dr. Kelly Cristina Poldi for her valuable help on implementations. This work was partially financed by FAPESP and CNPq.

References

ABUABARA, A., (2006), *Otimização no corte de tubos estruturais: aplicação na indústria aeronáutica agrícola*. MS Dissertation, DEP - UFSCar, São Carlos, SP, Brazil.

CHERRI, A. C., (2006), *O problema de corte de estoque com reaproveitamento da sobras de material*. MS Dissertation, ICMC - USP, São Carlos, SP, Brazil.

GRADISAR, M., JESENKO, J., RESINOVIC, C., (1997), *Optimization of roll cutting in clothing industry*. Computers & Operational Research, 10: 945-953.

GRADISAR, M., KLJAJIC, M., RESINOVIC, C., JESENKO, J., (1999a), *A sequential heuristic procedure for one-dimentional cutting*. European Journal of Operational Research, 114: 557-568.

GRADISAR, M., RESINOVIC, C.,KLJAJIC, M., (1999b), *A hybrid approach for optimization of one-dimentional cutting*. European Journal of Operational Research, 119: 719-728.

GRADISAR, M., TRKMAN, P., (2005), *A combined approach to the solution to the general one-dimentional cutting stock problem*. Computers and Operations Research, 32: 1793-1807.

GILMORE, P. C., GOMORY, R. E., (1961), *A linear programming approach to the cutting stock problem*. Operations Research, 9: 848-859.

HINXMAN, A., (1980), *The trim-loss and assortment problems: a survey*. European Journal of Operational Research, 5: 8-18.

TRKMAN, P., (2005), Private Communication (09/11/2005).

POLDI, K. C., (2003), *Algumas extensões do problema de corte de estoque*. MS Dissertation, ICMC - USP, São Carlos, SP, Brazil.

POLDI, K. C., ARENALES, M. N., (2005), *Dealing with small demand in integer cutting stock problems with limited different stock lengths*. Notas do ICMC - Série Computação, 85, ICMC – USP, São Carlos, SP, Brazil.

STADTLER, H., (1990), *A one-dimensional cutting stock problem in the Aluminium Industry and its solution*. European Journal of Operational Research, 44: 209-223.

WÄSCHER, G., GAU, T., (1996), *Heuristics for the integer one-dimensional cutting stock problem: a computational study*. OR Spektrum, 18: 131-144.

Appendix

Table A1 – Average number of non-standard object used

		Constructive Heuristics				Residual Heuristics									
	COLA	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
C_1	**17.7**	11.0	8.0	7.5	8.6	8.9	7.7	5.4	5.8	*2.8*	4.8	3.9	4.9	3.4	4.5
C_2	**16.0**	8.5	4.5	4.3	4.5	4.9	4.5	4.7	5.1	3.8	4.2	4.2	4.4	*3.2*	4.3
C_3	**18.1**	9.2	8.8	7.8	8.7	9.5	6.0	4.8	4.6	3.9	4.5	4.3	4.5	*3.8*	4.5
C_4	**18.0**	8.7	8.0	8.1	4.4	7.1	4.7	4.0	2.9	3.7	4.1	3.9	4.2	*2.4*	3.5
C_5	**18.1**	8.8	6.7	6.7	5.5	7.8	5.6	5.4	4.6	3.2	3.9	*3.5*	3.9	3.9	4.2
C_6	**15.5**	10.0	7.0	8.9	6.1	6.4	4.3	3.7	3.6	*2.3*	3.4	2.8	3.4	2.8	3.0
C_7	**30.4**	10.8	7.0	8.1	11.7	9.5	5.9	4.8	5.1	4.8	4.6	5.2	4.6	*3.8*	4.8
C_8	**26.5**	11.2	10.6	8.8	11.2	9.2	6.0	4.8	4.8	3.8	6.2	4.1	6.1	*3.5*	5.4
C_9	**30.1**	11.3	7.1	7.9	9.9	9.5	7.6	5.0	5.1	4.2	4.2	4.5	4.2	*3.5*	4.3
C_{10}	**30.0**	11.6	10.3	6.2	11.6	11.1	5.7	*2.6*	4.1	3.8	4.1	3.8	4.2	2.7	3.5
C_{11}	**30.1**	7.4	6.9	7.0	8.7	6.8	7.1	5.4	6.4	*3.1*	3.8	3.3	3.7	4.7	6.2
C_{12}	**28.9**	14.2	9.9	7.8	10.1	10.7	3.7	4.6	4.9	2.3	2.6	*2.2*	2.6	3.1	3.8
C_{13}	**42.8**	11.1	8.0	9.0	12.0	9.7	5.7	5.4	5.5	4.2	4.4	4.6	4.4	4.2	*3.6*
C_{14}	**39.6**	14.6	12.5	10.5	13.5	10.4	5.4	*4.2*	4.8	4.8	5.6	4.8	6.0	*4.2*	6.8
C_{15}	**43.9**	11.6	9.4	11.1	13.5	9.7	6.7	4.9	4.9	*3.9*	4.0	4.3	4.2	4.2	5.0
C_{16}	**43.7**	15.2	12.6	14.6	14.3	13.9	6.4	9.7	5.0	4.1	4.9	4.1	4.9	*3.1*	3.8
Average	**28.1**	11.0	8.6	8.4	9.6	9.1	5.8	5.0	4.8	3.7	4.3	4.0	4.4	*3.5*	4.5

Table A2 – Average total length of non-standard objetcs used

		Constructive Heuristics				Residual Heuristics									
	COLA	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
C_1	**3000.9**	1519.4	983.3	1170.3	1475.9	1184.2	873.3	*586.4*	733.6	589.0	796.5	661.1	840.9	758.2	900.8
C_2	**5050.7**	2853.0	1662.5	1492.6	1765.7	1626.3	1745.6	1609.2	2006.1	1342.5	1581.5	1486.1	1622.2	*1113.4*	1595.5
C_3	**3447.0**	1496.3	1165.4	1054.0	1620.5	1256.3	*604.9*	593.1	669.7	653.3	718.1	686.5	761.9	701.1	796.5
C_4	**5812.7**	2934.7	2778.5	2858.5	1740.2	2228.8	1715.6	1403.1	1076.7	1224.5	1448.4	1318.7	1515.3	*836.7*	1276.6
C_5	**3474.3**	1175.3	702.9	969.0	1225.4	1004.3	681.6	659.9	746.0	*542.3*	646.3	544.8	649.1	593.2	621.2
C_6	**5509.5**	3456.6	2446.3	2446.3	2630.1	2088.1	1392.1	1247.9	1504.5	*735.5*	1152.1	897.8	1152.1	902.7	1093.9
C_7	**7585.0**	1998.5	644.5	1393.5	2313.1	1412.0	607.3	496.7	547.7	*678.4*	683.7	681.4	678.6	708.8	837.4
C_8	**7653.4**	3042.3	3074.8	2443.8	3877.8	2333.4	1703.8	1287.7	1488.1	*1071.3*	1874.2	1175.8	1810.5	1117.8	1644.5
C_9	**6414.6**	1655.6	629.1	1076.7	1651.2	1134.9	761.7	*540.1*	578.5	732.2	781.5	742.6	783.2	628.2	668.1
C_{10}	**8637.5**	3472.7	2899.3	1997.2	4362.2	2861.2	1606.2	797.5	1417.4	1116.3	1319.4	1105.7	1356.7	*770.3*	1139.1
C_{11}	**4975.3**	812.4	498.6	645.3	1369.1	670.5	488.7	*417.7*	506.4	449.5	511.2	485.4	494.6	727.6	871.1
C_{12}	**9454.3**	4533.2	2997.3	2761.2	3948.4	3365.3	1033.3	1372.9	1725.2	682.9	776.6	*681.4*	786.0	1037.3	1217.1
C_{13}	**10433.4**	1489.0	839.4	1323.6	1828.3	1312.0	579.3	*415.4*	515.3	673.0	674.2	685.5	675.6	709.5	738.8
C_{14}	**11072.6**	3347.3	3824.1	2825.9	4115.0	2331.4	1249.9	*968.8*	1120.0	1266.6	1522.7	1270.6	1615.0	1103.4	1678.7
C_{15}	**9559.6**	1705.8	1058.4	1512.9	2057.0	1408.6	618.5	*439.9*	461.7	646.8	689.6	658.8	694.0	586.1	738.1
C_{16}	**10964.1**	3627.1	2921.5	3876.4	4185.0	3015.1	1267.9	1754.6	1137.6	1056.8	1230.4	1075.2	1230.4	*518.2*	1023.0
Average	**7065.3**	2445.0	1820.4	1865.5	2510.3	1827.0	1058.1	911.9	1014.7	841.3	1025.4	884.8	1041.6	*800.9*	1052.5

Table A3 – Average number of standard objects used

		Constructive Heuristics				Residual Heuristics									
	COLA	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
C_1	108.5	107.7	108.3	**110.3**	**110.3**	*102.5*	102.9	103.0	103.0	102.6	102.6	*102.5*	102.6	102.6	102.7
C_2	114.2	111.0	123.3	116.7	**129.5**	*110.3*	110.6	110.4	110.7	110.4	113.7	*110.3*	113.3	110.5	113.6
C_3	107.3	105.5	108.0	**109.6**	109.4	101.5	102.1	102.1	102.1	*101.3*	101.5	*101.3*	101.5	*101.3*	101.5
C_4	102.5	100.5	103.9	105.8	**116.4**	*99.1*	100.1	100.1	101.0	100.0	101.4	99.9	101.1	100.3	101.5
C_5	99.35	103.9	106.0	**106.3**	106.2	98.5	98.8	98.8	98.8	98.0	98.0	98.0	98.0	*97.9*	98.0
C_6	106.7	105.6	109.0	108.7	**113.2**	*102.0*	103.0	102.8	103.0	102.6	102.5	102.5	102.5	102.6	102.8
C_7	103.1	105.6	107.9	**108.6**	108.2	*101.9*	102.8	102.9	102.9	102.2	102.4	102.2	102.4	102.3	102.3
C_8	107.3	109.0	111.4	113.1	**114.4**	*106.7*	107.5	107.9	107.8	107.7	107.5	107.6	107.5	107.8	107.6
C_9	104.9	107.7	**109.8**	109.6	109.4	101.9	102.3	102.6	102.7	*101.8*	*101.8*	*101.8*	*101.8*	*101.8*	101.9
C_{10}	98.1	100.0	102.0	**105.2**	104.2	*96.3*	97.7	98.3	98.0	97.6	97.6	97.6	97.6	98.0	97.9
C_{11}	105.1	107.5	**108.7**	108.6	108.3	*101.3*	101.6	101.6	101.8	100.7	100.8	100.7	100.8	100.4	100.4
C_{12}	101.1	102.0	105.5	**107.2**	107.2	*98.6*	101.0	100.8	100.6	100.5	100.5	100.5	100.5	100.3	100.3
C_{13}	104.2	109.8	111.8	112.4	**112.5**	*105.7*	106.3	106.4	106.6	105.8	106.0	105.8	106.0	*105.7*	105.8
C_{14}	*92.3*	99.0	98.1	99.1	**101.0**	93.4	95.1	95.3	95.3	94.7	94.6	94.7	94.6	94.9	94.7
C_{15}	*104.7*	110.1	111.9	**112.1**	111.9	104.9	105.8	106.0	106.0	105.0	105.1	105.0	105.1	105.0	105.1
C_{16}	103.9	108.1	110.3	110.6	**112.7**	103.7	105.5	*85.9*	105.7	105.2	105.3	105.2	105.3	105.5	105.4
Average	104.0	105.8	108.5	109.0	**110.9**	101.8	102.7	*101.6*	102.9	102.3	102.6	102.2	102.5	102.3	102.6

Table A4 – Average total length of the standard objects used

		Constructive Heuristics				Residual Heuristics									
	COLA	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
C_1	*109545*	111380	111525	111630	**111750**	111360	111695	111705	111705	111570	111620	111525	111625	111450	111530
C_2	*116025*	117590	127955	120065	**135655**	116680	116900	116665	116985	116725	120010	116605	119495	116945	119875
C_3	*108170*	110175	110460	110600	110495	110300	**110855**	110690	110740	110480	110680	110480	110630	110465	110605
C_4	*103440*	106350	107545	108010	**120930**	105940	106885	106770	107760	106870	108195	106760	107920	107285	108370
C_5	*104765*	107085	**107620**	107130	107090	107160	107385	107165	107215	107220	107270	107220	107270	107185	107285
C_6	*107510*	109870	110980	110295	**115945**	110830	111675	111435	111710	111910	111755	111750	111755	111775	112025
C_7	*104190*	109705	**111170**	110270	109910	110040	110975	110940	110940	110570	110770	110570	110770	110545	110590
C_8	*108640*	114310	115065	115170	**116995**	113205	113980	114375	114270	114235	113995	114140	114010	114235	114115
C_9	*105515*	110335	**111430**	110815	110540	110655	110965	111125	111275	110840	110890	110840	110890	110930	111030
C_{10}	*98455*	104440	104505	106895	**107045**	103885	105230	105735	105485	105375	105380	105380	105385	105735	105590
C_{11}	*105885*	110010	**110285**	109620	109290	109935	110155	109950	110150	109880	109980	109880	110030	109595	109645
C_{12}	*101590*	106600	107855	108440	108575	107370	109650	109285	109135	**109815**	**109815**	**109815**	**109815**	109520	109520
C_{13}	*105155*	114130	114750	114030	114195	114110	114755	114745	**114945**	114435	114635	114435	114635	114425	114475
C_{14}	*92775*	101790	100645	101270	**103440**	100935	102115	102315	102320	101860	101750	101865	101710	102045	101780
C_{15}	*105655*	113610	114230	113655	113505	113610	**114535**	114485	114485	114150	114200	114200	114200	114220	114320
C_{16}	*104620*	112270	112570	112500	**114995**	112165	113830	114150	113950	113865	113970	113970	113970	114125	114075
Average	*105120.9*	109978.1	111161.9	110649.7	113147.2	109886.3	110724.1	110720.9	110816.9	110612.5	110932.2	110589.7	110881.9	110655.0	110926.9

Table A5 – Average total loss

		Constructive Heuristics				Residual Heuristics									
	COLA	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
C_1	17.5	**284.3**	26.4	158.6	15.0	29.9	5.4	20.7	*4.3*	16.0	7.8	31.9	7.1	22.3	6.6
C_2	925.9	2406.4	98.7	**2712.7**	*40.2*	326.3	157.2	331.7	156.3	323.3	87.6	379.1	97.3	328.9	93.8
C_3	4.2	105.1	12.7	**163.5**	12.6	10.9	3.7	47.1	*2.6*	28.6	3.2	31.5	3.3	22.2	3.4
C_4	371.5	889.5	69.4	**1850.9**	32.3	168.7	35.6	217.4	27.9	138.8	18.8	123.0	*18.7*	165.9	24.4
C_5	*0.8*	44.5	6.2	**132.4**	8.5	7.1	1.7	64.0	4.2	25.9	3.5	32.5	3.5	26.0	1.7
C_6	24.0	**354.9**	28.4	531.0	74.5	47.9	*5.7*	133.4	11.9	71.6	7.4	75.8	7.4	114.7	6.7
C_7	6.4	**118.8**	31.6	55.1	11.0	15.2	1.9	8.9	*1.4*	18.4	2.4	19.1	2.4	12.3	3.3
C_8	297.0	**1630.9**	102.5	795.6	54.3	89.3	*9.4*	63.4	10.9	69.7	19.4	79.2	17.0	84.9	21.4
C_9	2.3	**57.1**	21.6	42.3	11.1	10.0	3.1	17.3	3.3	12.3	*1.2*	13.7	*1.2*	9.6	1.9
C_{10}	136.9	951.9	56.4	**1124.2**	20.1	120.0	7.1	58.6	*4.4*	96.7	8.1	98.7	7.4	84.0	8.8
C_{11}	*0.2*	13.0	*9.1*	**44.7**	11.7	3.6	1.5	20.5	2.5	9.4	1.6	10.1	1.7	22.3	3.9
C_{12}	13.5	169.5	11.9	**294.1**	37.8	26.4	*2.3*	96.3	6.8	39.5	4.0	36.3	3.8	45.2	2.8
C_{13}	5.8	**98.0**	17.1	76.0	16.3	8.2	1.8	15.7	3.1	8.3	1.1	8.4	*1.0*	10.4	1.1
C_{14}	155.0	**1460.3**	63.0	593.4	49.6	52.1	*6.5*	43.6	7.7	50.9	11.5	46.1	12.0	55.0	12.5
C_{15}	1.5	36.6	9.5	**42.1**	20.9	4.2	*1.1*	9.7	1.9	9.1	1.4	11.2	1.3	10.6	2.0
C_{16}	58.2	**614.4**	29.0	592.0	50.6	61.7	6.1	33.8	11.3	43.4	*4.3*	48.5	*4.3*	44.3	5.6
Average	126.3	**577.2**	37.1	575.5	29.2	61.3	15.6	73.9	16.3	60.1	*11.5*	65.3	11.8	66.1	12.5

Table A6 – Average total length lost in the standard objects

		Constructive Heuristics				Residual Heuristics									
	COLA	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
C_1	17.3	**271.2**	21.5	89.2	8.0	3.8	0.3	0.4	*0.0*	3.5	0.2	2.0	0.1	1.9	0.4
C_2	840.3	2339.1	93.1	**2481.4**	*34.5*	115.8	153.6	164.9	151.6	202.5	81.8	208.8	88.5	224.7	87.4
C_3	4.2	**100.3**	10.1	66.1	7.4	2.8	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	0.3	0.1
C_4	353.2	837.5	57.2	**1397.8**	24.6	53.7	25.7	31.4	24.3	31.8	*11.5*	26.9	11.6	74.0	15.7
C_5	0.8	**43.9**	5.4	10.5	2.9	3.5	0.2	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*
C_6	24.0	**337.2**	21.2	314.6	31.2	0.7	0.7	5.2	1.5	6.7	*0.1*	1.3	*0.1*	15.6	0.6
C_7	5.7	**112.2**	31.4	20.2	0.4	0.3	0.3	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	0.3	0.4
C_8	263.6	**1592.7**	88.7	710.8	41.8	4.2	4.2	5.4	3.6	11.7	*3.2*	9.8	3.3	17.1	6.5
C_9	2.3	**54.6**	20.7	13.2	0.5	1.7	0.3	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*
C_{10}	124.1	913.5	50.1	**1094.0**	15.4	1.1	1.1	14.0	0.5	1.4	*0.1*	1.2	0.2	2.9	0.4
C_{11}	0.2	**12.7**	8.4	8.4	2.1	1.3	0.7	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	0.1	*0.0*
C_{12}	13.5	**156.7**	6.5	**174.1**	20.9	4.4	0.1	12.2	0.6	*0.0*	*0.0*	*0.0*	*0.0*	0.2	*0.0*
C_{13}	5.8	**91.4**	15.4	38.8	4.4	1.1	0.5	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	0.1	0.1
C_{14}	130.4	**1354.2**	56.7	449.0	20.1	12.4	0.7	0.7	*0.0*	1.8	1.4	1.6	1.4	3.2	1.0
C_{15}	1.5	**32.3**	8.1	16.2	1.8	0.8	0.7	0.0	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*
C_{16}	50.9	**583.9**	20.9	416.2	19.6	9.8	0.3	0.2	0.2	*0.0*	0.1	*0.0*	0.1	4.3	*0.0*
Average	114.9	**552.1**	32.2	456.3	14.7	13.6	11.8	14.7	11.4	16.2	*6.2*	15.7	6.6	21.5	7.0

Table A7 –Average toal length lost in the non-standard objects used

		Constructive Heuristics				Residual Heuristics									
	COLA	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
C_1	*0.2*	13.1	4.9	**69.4**	7.0	26.2	5.1	20.4	4.3	12.5	7.7	29.9	6.7	46.9	6.2
C_2	85.6	67.3	5.6	**231.3**	5.7	115.8	3.7	166.8	*4.7*	120.9	5.8	170.3	9.5	104.3	6.4
C_3	*0.0*	4.9	2.6	**97.4**	5.2	8.1	3.7	47.1	2.6	28.6	3.2	31.5	3.3	21.9	3.4
C_4	18.3	52.0	12.3	**453.1**	7.7	115.0	10.0	186.0	*3.7*	107.0	7.3	96.1	7.2	92.0	8.7
C_5	*0.0*	0.6	0.8	**121.9**	5.6	3.6	1.5	64.0	4.2	25.9	3.5	32.5	3.5	26.0	1.7
C_6	*0.0*	17.8	7.3	**216.5**	43.3	17.1	5.0	128.3	10.5	65.0	7.3	74.5	7.3	104.2	6.1
C_7	0.75	6.7	*0.3*	**34.9**	10.6	12.2	1.6	8.9	1.4	18.4	2.4	19.1	2.4	12.0	3.0
C_8	33.4	38.2	13.8	**84.9**	12.5	66.4	*5.2*	58.1	7.4	58.0	16.2	69.4	13.7	67.9	14.9
C_9	*0.0*	2.5	0.9	**29.1**	10.6	8.3	2.9	17.3	3.3	12.3	1.2	13.7	1.2	9.6	1.9
C_{10}	12.8	38.4	6.3	30.2	4.7	94.0	6.1	44.6	*4.0*	95.3	8.0	**97.6**	7.2	81.1	8.4
C_{11}	*0.0*	0.3	1.1	**36.3**	9.6	2.3	0.8	20.5	2.5	1.6	1.6	10.1	1.7	22.2	3.9
C_{12}	*0.0*	12.8	5.4	**120.0**	16.9	22.0	2.3	84.1	6.2	39.5	4.0	36.3	3.8	45.1	2.8
C_{13}	*0.0*	6.7	1.7	**37.2**	12.0	7.1	1.3	15.7	3.1	8.3	1.1	8.4	1.0	10.3	1.1
C_{14}	24.6	106.2	6.4	**144.4**	29.5	39.7	*5.9*	42.9	7.7	49.1	10.1	44.6	10.6	51.8	6.1
C_{15}	*0.0*	4.3	1.4	**25.9**	19.1	3.5	0.4	9.7	1.9	9.1	1.4	11.2	1.3	10.6	2.0
C_{16}	7.3	*3.5*	8.2	**175.8**	31.0	52.0	5.8	33.6	11.1	43.4	4.2	48.5	4.2	40.1	5.6
Average	11.4	23.5	4.9	**119.3**	14.4	37.1	*3.8*	59.3	4.9	43.4	5.3	49.6	5.3	46.6	5.1

Table A8 – Average total retail

		Constructive Heuristics				Residual Heuristics									
	COLA	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
C_1	479.4	566.1	432.9	592.7	**1161.9**	465.2	513.9	221.7	385.3	*94.0*	359.6	105.2	409.8	136.9	375.1
C_2	2437.8	324.6	11806.8	1132.9	**19668.5**	268.0	776.4	230.5	1122.8	32.2	3791.9	*0.0*	3307.6	17.5	3664.7
C_3	536.9	490.3	536.9	414.6	**1027.0**	469.6	380.3	160.1	331.2	*28.9*	319.1	59.1	312.7	68.0	322.2
C_4	858.3	479.5	2298.4	1061.9	**14682.2**	44.4	612.3	*0.0*	853.0	*0.0*	1668.8	*0.0*	1460.8	*0.0*	1666.5
C_5	530.0	508.3	**609.2**	259.1	599.4	449.7	357.4	53.4	249.3	*28.9*	205.3	24.8	208.0	44.7	197.0
C_6	515.8	492.1	918.3	576.7	**6021.0**	390.6	581.7	69.9	722.9	94.3	420.1	92.4	420.1	*83.4*	632.6
C_7	577.1	393.2	591.5	417.0	**1020.7**	245.3	389.0	236.4	294.9	*38.5*	259.9	40.9	254.8	50.1	232.6
C_8	633.8	523.9	2939.8	1620.7	**5621.0**	251.6	476.9	401.8	549.7	*39.1*	652.4	*39.1*	606.0	70.4	540.6
C_9	410.2	416.4	520.4	332.3	**662.9**	262.7	206.4	130.7	333.2	42.7	153.2	51.7	154.8	*31.5*	179.1
C_{10}	568.6	573.8	961.0	1381.0	**5000.1**	239.3	442.1	86.9	511.0	7.7	304.4	*0.0*	347.3	34.4	333.3
C_{11}	571.0	**520.3**	485.4	231.5	358.3	312.8	353.1	58.1	364.8	30.9	200.5	39.1	233.8	*11.2*	223.1
C_{12}	617.3	550.2	426.8	493.6	**2072.1**	295.3	267.4	148.0	439.9	*44.8*	174.0	46.6	183.6	98.5	320.7
C_{13}	500.5	439.0	490.3	195.6	**925.0**	331.9	250.5	122.7	355.2	*17.7*	226.1	30.0	227.6	42.1	130.6
C_{14}	628.0	612.3	1341.4	437.8	**4440.7**	149.6	293.7	175.5	367.6	*11.1*	196.5	24.8	248.3	28.8	381.5
C_{15}	461.2	527.4	527.0	374.0	**789.3**	262.5	300.6	163.4	192.9	*35.9*	136.4	45.7	140.8	43.6	204.2
C_{16}	647.2	416.0	595.9	917.8	**4262.8**	251.8	225.2	1004.2	209.6	*11.8*	329.5	25.1	329.5	32.2	225.8
Average	685.8	489.6	1592.6	652.5	**4269.6**	293.1	401.7	204.0	455.2	*34.9*	587.4	39.0	552.8	49.6	601.9

Table A9 – Average total length of the retail generated in the standard objects

		Constructive Heuristics				Residual Heuristics									
	COLA	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
C_1	479.4	566.1	419.8	555.3	**889.4**	460.3	381.7	206.4	244.8	*38.7*	183.6	39.9	222.8	46.9	233.6
C_2	2419.1	324.6	11709.6	1132.9	**19129.3**	268.0	404.1	230.5	542.4	32.2	3434.8	*0.0*	2999.9	17.5	3277.4
C_3	536.6	490.3	500.9	347.2	**630.2**	458.0	320.2	141.1	182.6	*0.0*	150.4	*0.0*	134.3	*0.0*	154.1
C_4	353.2	479.5	2144.5	767.5	**14090.9**	18.9	285.0	*0.0*	607.0	*0.0*	1406.2	*0.0*	1160.6	*0.0*	1362.3
C_5	531.0	508.3	**593.3**	220.4	320.2	446.5	298.1	40.5	85.8	*0.0*	47.0	*0.0*	47.0	*0.0*	66.7
C_6	514.5	492.1	749.2	395.4	**5988.7**	367.8	207.4	47.8	264.2	*0.0*	61.9	*0.0*	61.9	33.9	230.4
C_7	577.1	391.5	575.0	415.3	**616.9**	226.7	335.6	216.9	207.4	*0.0*	160.7	*0.0*	155.6	*0.0*	134.4
C_8	795.0	516.0	2550.1	1045.8	**4509.3**	188.6	102.6	110.7	117.8	*0.0*	261.5	*0.0*	223.5	18.8	243.6
C_9	410.0	416.4	**517.0**	315.2	395.9	246.6	160.4	101.2	224.2	*0.0*	48.8	*0.0*	48.8	*0.0*	85.7
C_{10}	565.6	573.8	876.3	672.8	**3416.7**	223.3	78.2	*0.0*	*0.0*	*0.0*	19.4	*0.0*	7.7	11.3	11.3
C_{11}	**571.0**	518.9	483.5	227.2	150.8	301.4	332.1	47.8	231.4	*0.0*	98.7	*0.0*	145.8	*0.0*	86.9
C_{12}	614.6	550.2	227.6	201.5	**1157.2**	213.8	17.9	53.7	52.8	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	54.1
C_{13}	497.7	439.0	486.0	146.3	**647.7**	309.8	217.6	107.9	287.1	*0.0*	172.0	*0.0*	173.3	*0.0*	43.2
C_{14}	628.0	606.6	1058.4	412.4	**3300.7**	94.9	120.4	125.2	132.4	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	65.1
C_{15}	461.2	527.4	520.3	325.0	**613.5**	245.0	272.9	141.4	140.0	*0.0*	51.0	*0.0*	51.0	*0.0*	91.7
C_{16}	659.3	416.0	415.0	694.5	**3150.4**	217.3	29.6	167.2	*0.0*	*0.0*	40.2	*0.0*	40.2	*0.0*	*0.0*
Average	663.3	488.5	1489.2	492.2	**3688.0**	267.9	222.7	108.6	207.5	4.4	383.5	*2.5*	342.0	8.0	383.8

Table A10 – Average total length of the retail generated in the non-standard objects

		Constructive Heuristics				Residual Heuristics									
	COLA	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
C_1	*0.0*	*0.0*	13.1	37.4	**272.5**	5.0	132.2	15.3	140.5	55.3	46.1	65.4	187.0	90.1	141.6
C_2	*0.0*	*0.0*	97.3	*0.0*	**539.2**	*0.0*	372.3	*0.0*	580.4	*0.0*	357.2	*0.0*	307.7	*0.0*	387.4
C_3	*0.0*	*0.0*	36.0	67.5	**396.8**	11.6	60.1	19.1	148.6	28.9	168.7	59.1	178.5	68.0	158.2
C_4	*0.0*	*0.0*	153.9	294.4	**591.3**	25.6	327.6	*0.0*	246.1	*0.0*	262.6	*0.0*	300.3	*0.0*	304.2
C_5	*0.0*	*0.0*	15.9	38.8	**279.2**	3.3	59.3	12.9	163.5	28.9	158.4	24.8	161.1	44.7	130.3
C_6	*0.0*	*0.0*	**169.1**	181.3	32.3	22.8	374.4	22.1	458.7	94.3	358.2	92.4	358.2	49.5	402.3
C_7	*0.0*	1.7	16.5	1.7	**403.8**	18.6	53.4	19.5	87.5	38.5	99.3	40.9	99.3	50.1	98.3
C_8	*0.0*	7.9	389.8	554.9	**1111.7**	63.0	374.4	291.1	432.0	39.1	390.9	39.1	382.6	51.6	297.1
C_9	*0.0*	*0.0*	3.4	17.1	**267.0**	16.1	46.1	29.5	109.0	42.7	104.4	51.7	106.0	31.5	93.4
C_{10}	*0.0*	*0.0*	84.7	708.2	**1583.5**	16.0	364.0	86.9	511.0	7.7	285.0	***0.0***	339.6	23.1	322.0
C_{11}	*0.0*	2.4	1.9	4.3	**207.5**	11.4	21.0	9.5	133.4	30.9	101.9	39.1	88.0	11.2	136.2
C_{12}	*0.0*	*0.0*	199.3	293.1	**914.9**	81.2	249.5	94.3	387.1	44.8	174.0	46.6	183.6	98.5	266.6
C_{13}	*0.0*	*0.0*	4.1	49.3	**277.3**	22.1	32.9	14.8	88.1	17.7	54.0	30.0	54.4	42.1	87.4
C_{14}	*0.0*	5.8	283.0	25.5	**1140.0**	54.8	173.4	50.3	235.3	11.1	196.5	24.8	248.3	28.8	316.5
C_{15}	*0.0*	*0.0*	6.7	48.9	**174.8**	17.5	25.7	22.0	52.9	35.9	85.4	45.7	89.9	43.6	112.5
C_{16}	*0.0*	*0.0*	180.9	223.3	**1112.4**	34.5	195.6	837.0	209.6	11.8	289.3	25.1	289.3	32.2	225.8
Average	*0.0*	1.1	103.5	159.1	**581.5**	25.2	178.9	95.3	249.0	30.5	195.7	36.5	210.9	41.6	217.5

Table A11 – Number of objects cut with retail

		Constructive Heuristics				Residual Heuristics									
	COLA	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
C_1	3.0	1.0	1.6	2.7	**10.1**	0.8	2.5	*0.6*	2.2	*0.6*	2.2	*0.6*	2.3	1.0	2.0
C_2	13.1	0.6	36.0	4.2	**53.6**	0.4	2.4	0.4	3.4	0.1	11.3	*0.0*	9.3	0.1	10.8
C_3	2.2	1.0	1.6	2.1	**8.6**	0.9	1.5	0.5	2.0	*0.3*	2.2	0.5	2.2	0.8	2.5
C_4	4.3	0.8	8.0	4.1	**36.8**	0.2	1.9	*0.0*	2.3	*0.0*	4.9	*0.0*	4.8	*0.0*	4.7
C_5	1.1	1.0	1.2	1.6	**4.8**	0.9	1.5	*0.3*	2.1	0.5	2.1	*0.3*	2.2	0.6	2.0
C_6	5.1	0.9	2.9	1.8	**17.8**	0.9	2.1	0.2	2.6	0.4	1.6	0.4	1.6	*0.3*	2.1
C_7	2.0	0.9	1.5	3.9	**11.8**	0.8	1.8	0.6	2.0	*0.5*	2.1	0.6	2.1	0.6	1.6
C_8	4.2	0.9	16.7	9.0	**27.5**	0.7	2.8	2.0	2.8	*0.3*	3.1	*0.3*	3.0	0.5	3.3
C_9	1.2	1.0	1.1	2.6	**7.8**	0.8	1.3	*0.6*	2.8	*0.6*	1.5	*0.6*	1.5	*0.6*	1.8
C_{10}	2.4	0.9	3.1	7.2	**26.6**	0.4	2.0	0.6	2.8	0.1	1.5	*0.0*	1.7	0.2	1.6
C_{11}	1.1	1.0	1.0	0.6	**5.4**	0.9	1.1	0.4	3.7	0.5	1.8	0.6	1.7	*0.2*	2.6
C_{12}	5.3	0.9	1.8	2.8	**9.3**	0.8	1.6	1.0	2.4	*0.3*	1.1	*0.3*	1.2	0.6	1.7
C_{13}	3.2	1.0	1.6	3.2	**11.4**	0.8	1.5	0.5	2.6	*0.3*	1.7	0.4	1.7	0.6	1.5
C_{14}	3.0	1.4	7.4	1.3	**22.7**	0.6	1.7	0.7	2.3	*0.1*	1.8	0.2	2.1	0.3	2.8
C_{15}	1.2	1.0	1.1	3.0	**7.6**	0.8	1.1	0.6	1.6	*0.5*	1.3	*0.5*	1.4	0.6	2.2
C_{16}	2.5	0.9	2.3	*4.1*	**20.5**	0.5	1.7	5.9	1.6	*0.1*	2.1	0.2	2.1	0.3	1.5
Average	3.4	1.0	5.6	3.9	**17.6**	0.7	1.8	0.9	2.5	*0.3*	2.6	*0.3*	2.6	0.5	2.8

Table A12 – Average number of objects cut with small scrap

		Constructive Heuristics				Residual Heuristics									
	COLA	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
C_1	3.35	**68.8**	13.0	6.1	6.2	5.8	2.2	2.1	2.2	*1.3*	2.1	1.5	2.0	1.5	2.3
C_2	23.8	**34.8**	38.8	*14.2*	14.5	25.1	25.2	24.2	24.3	24.2	26.2	23.7	26.5	25.0	26.2
C_3	1.75	**56.7**	7.1	4.2	4.4	5.1	1.2	1.3	1.3	1.1	1.0	1.0	1.1	*0.8*	1.4
C_4	18.3	**48.6**	27.8	11.0	11.5	11.9	9.1	7.6	7.7	8.2	7.6	8.1	*7.2*	8.8	8.9
C_5	*0.4*	**32.1**	3.4	2.1	2.1	3.7	1.2	0.8	0.9	1.0	1.0	1.0	0.9	0.8	0.8
C_6	5.4	**64.9**	12.0	15.3	17.0	7.1	2.0	2.0	2.2	*0.8*	1.2	1.0	1.2	1.5	1.4
C_7	3.2	**50.8**	17.0	2.3	2.5	3.7	1.1	*0.6*	*0.6*	1.0	0.9	1.0	0.9	1.2	1.4
C_8	25.6	**39.9**	37.2	16.7	17.2	7.7	3.6	*2.8*	2.9	3.1	3.2	3.2	3.0	4.3	4.8
C_9	1.0	**38.1**	10.7	2.6	3.0	3.3	1.5	1.0	1.1	*0.6*	*0.6*	*0.6*	*0.6*	0.9	0.8
C_{10}	12.6	**48.4**	25.0	7.2	7.3	7.7	2.1	*1.1*	1.2	1.2	1.5	1.2	1.4	*1.1*	1.5
C_{11}	*0.2*	**11.6**	4.6	3.5	4.0	1.6	1.0	1.0	1.0	0.5	0.6	0.6	0.7	1.1	1.4
C_{12}	5.2	**64.4**	4.9	11.8	11.9	4.8	*0.7*	1.7	1.7	*0.7*	0.9	*0.7*	0.8	*0.7*	*0.7*
C_{13}	2.8	**50.4**	11.2	5.2	5.5	3.2	0.8	1.6	1.7	0.7	0.5	0.5	*0.4*	0.8	0.7
C_{14}	22.7	**37.0**	31.6	15.6	16.1	6.9	2.0	*1.9*	*1.9*	2.4	2.4	2.4	2.3	*1.9*	3.1
C_{15}	0.8	**27.4**	5.8	4.7	5.5	1.1	*0.7*	1.0	1.0	*0.7*	0.8	0.8	0.8	0.8	0.8
C_{16}	11.7	**57.4**	15.2	13.7	13.7	7.9	1.9	2.2	2.3	1.4	*1.3*	*1.3*	*1.3*	*1.3*	*1.3*
Average	9.0	**45.7**	16.6	8.5	8.9	6.7	3.5	3.3	3.4	*3.1*	3.2	*3.1*	3.2	3.3	3.6

Table A13 – Average Number of objects cut with not so small scrap

		Constructive Heuristics				Residual Heuristics									
	COLA	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*FFD*	*FFD$_L$*	*Greedy*	*Greedy$_L$*	*RGR 1*	*RGR$_L$ 1*	*RGR 2*	*RGR$_L$ 2*	*RGR 3*	*RGR$_L$ 3*
C_1	1.4	**16.5**	*0.0*	6.4	*0.0*	1.6	*0.0*	1.3	*0.0*	0.6	*0.0*	1.4	*0.0*	0.9	*0.0*
C_2	29.3	**77.2**	*0.0*	36.6	*0.0*	11.1	7.8	10.2	7.8	10.6	1.0	11.0	1.9	10.8	2.3
C_3	0.1	1.0	*0.0*	**6.0**	*0.0*	0.3	*0.0*	1.7	*0.0*	1.3	*0.0*	1.7	*0.0*	1.0	*0.0*
C_4	14.1	**47.5**	*0.0*	26.2	*0.0*	4.9	1.2	3.8	1.2	2.9	*0.0*	2.7	*0.0*	3.5	*0.0*
C_5	*0.0*	0.1	*0.0*	**4.5**	*0.0*	0.1	*0.0*	2.7	*0.0*	1.1	*0.0*	1.5	*0.0*	1.2	*0.0*
C_6	2.1	**22.9**	*0.0*	16.0	*0.0*	1.9	*0.0*	2.4	*0.0*	1.0	*0.0*	1.0	*0.0*	1.5	*0.0*
C_7	*0.0*	2.7	*0.0*	**4.9**	*0.0*	1.1	*0.0*	1.1	*0.0*	1.6	*0.0*	2.0	*0.0*	0.7	*0.0*
C_8	9.7	**60.9**	*0.0*	15.3	*0.0*	1.7	*0.0*	1.1	*0.0*	1.1	*0.0*	1.1	*0.0*	2.2	*0.0*
C_9	0.1	0.1	*0.0*	**3.9**	*0.0*	0.3	*0.0*	2.2	*0.0*	1.1	*0.0*	1.2	*0.0*	0.7	*0.0*
C_{10}	3.8	**42.9**	*0.0*	15.1	*0.0*	2.6	*0.0*	1.1	*0.0*	1.7	*0.0*	1.8	*0.0*	1.2	*0.0*
C_{11}	*0.0*	*0.0*	*0.0*	**3.9**	*0.0*	0.1	*0.0*	2.2	*0.0*	0.9	*0.0*	0.8	*0.0*	1.5	*0.0*
C_{12}	0.4	3.9	*0.0*	**4.3**	*0.0*	0.3	*0.0*	1.3	*0.0*	0.7	*0.0*	0.6	*0.0*	0.7	*0.0*
C_{13}	*0.0*	1.6	*0.0*	**5.5**	*0.0*	0.7	*0.0*	1.9	*0.0*	1.0	*0.0*	1.3	*0.0*	1.4	*0.0*
C_{14}	5.3	**53.5**	*0.0*	17.0	*0.0*	0.9	*0.0*	1.2	*0.0*	1.2	*0.0*	1.1	*0.0*	1.7	*0.0*
C_{15}	*0.0*	0.5	*0.0*	**3.1**	*0.0*	0.2	*0.0*	0.9	*0.0*	0.9	*0.0*	1.0	*0.0*	1.2	*0.0*
C_{16}	1.7	**28.7**	*0.0*	14.7	*0.0*	2.0	*0.0*	0.7	*0.0*	1.1	*0.0*	1.1	*0.0*	1.0	*0.0*
Average	4.3	**22.5**	*0.0*	11.4	*0.0*	1.9	0.6	2.2	0.6	1.8	0.1	2.0	0.1	2.0	0.1

NOTAS DO ICMC

SÉRIE COMPUTAÇÃO

089/2007 TOLEDO, F.M.B.; ARMENTANO, V.A. – Branch-and-bound algorithms for capacitated lot-sizing in parallel machines

088/2007 TOLEDO, F.M.B.; SANTOS, M.O.; ARENALES, M.N.; SELEGHIM JÚNIOR, P. - Logística de distribuição de água em redes urbanas - racionalização energética.

087/2005 GOIS, J.P; ESTÁCIO, K.C.; OISHI, C.M. BERTTONI,V.; BOTTA, V.A.; NAGAMINE, A.; KUROKAWA, F.A.; FEDERSON, F. – Aplicação de volumes finitos na simulação numérica de contaminação em lençóis freáticos.

086/2005 MARQUES, F.P.; ARENALES, M.N. – The constrained compartmentalized knapsack problem.

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

084/2005 PRADO, T.A.S.; NUNES, M.G.V. – A statistical generative model for unsupervised learning of web 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.; FERREIRA, V.; NAVARRO, H. A. - Upwinding finite-difference schemes for convection dominated problems. Part I: theoretical results