UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Dealing with Small Demand in Integer Cutting Stock Problems with Limited Different Stock Lengths

Kelly Cristina Poldi
Marcos Nereu Arenales

Nº 85

NOTAS

São Carlos - SP

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

Dealing with Small Demand in Integer Cutting Stock Problems with Limited Different Stock Lengths

Kelly Cristina Poldi
Marcos Nereu Arenales

Nº 85

NOTAS

Série Computação

São Carlos – SP
Mar./2005

Dealing with small demand in integer cutting stock problems with limited different stock lengths

Kelly Cristina Poldi and Marcos Nereu Arenales
kelly@icmc.usp.br, arenales@icmc.usp.br

Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Av. do Trabalhador São-Carlense, 400 - Centro
CEP 13560-970 - São Carlos - SP - Brazil

Abstract. This paper deals with the classical one-dimensional integer cutting stock problem, which consists of cutting a set of available objects in stock in order to produce smaller ordered items in such a way as to optimize a given objective function (e.g., minimizing waste). We have studied the case in which there are several stock lengths available in limited quantities. Moreover, we have focused on the special case in which the quantity demanded of each item is low. Some heuristic methods are given in order to obtain an integer solution, which can be classified as constructive (greedy heuristics) or residual (based on column generation). These heuristics are empirically analyzed by solving a set of randomly generated instances. The analysis showed that residual heuristics are much superior than constructive ones. In particular, a new proposed residual heuristic obtained better results than the others. An important by-product from the empirical analysis contradicts the folklore in the cutting and packing area, which believes that constructive heuristics must be used if one is faced with low demand problems.

O problema de corte de estoque com baixa demanda e com diferentes tipos de objetos em estoque em quantidade limitada

Kelly Cristina Poldi e Marcos Nereu Arenales
kelly@icmc.usp.br, arenales@icmc.usp.br

Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Av. do Trabalhador São-Carlense, 400 - Centro
CEP 13560-970 - São Carlos - SP - Brasil

Resumo. Este trabalho trata do clássico problema de corte de estoque inteiro, que consiste em cortar um conjunto de objetos disponíveis em estoque a fim de produzir itens menores demandados por clientes, de forma a otimizar uma certa função objetivo (por exemplo, minimizar perda). Estudamos o caso no qual há vários tipos (comprimentos) de objetos em estoque disponíveis em quantidades limitadas. Focamos nosso estudo em problemas com baixa demanda de itens. São apresentados alguns métodos heurísticos para obtenção de solução inteira, estas heurísticas são classificadas em construtivas (gulosas) ou residuais (baseado em geração de colunas). Estas heurísticas são analisadas empiricamente com base na solução de exemplos gerados aleatoriamente. Esta análise mostra que as heurísticas residuais são melhores que as construtivas. Em particular, a heurística nova proposta foi a que apresentou os melhores resultados. Um sub-produto importante desta análise contradiz o folclore na área de problemas de corte e empacotamento que diz que deve-se usar heurísticas construtivas para problemas com baixa demanda.

1 Introduction

A cutting stock problem consists of cutting a set of available objects in stock into smaller pieces, by optimizing a certain objective, for example, minimizing the total waste. Problems like this are essential when planning the production, in several industries such as paper industry, steel industry, plastic, wood and so on (Dyckhoff and Finke [1], Stadtler [13]).

This problem may be formulated as an integer linear optimization problem, which is very difficult to be solved, mainly because of the large number of cutting patterns (columns). In order to deal with these difficulties, in 1961, Gilmore and Gomory [4] relaxed the integrality constraint and proposed an efficient method of column generation to solve the linear optimization problem.

In this way, we have an optimal solution which may be non-integer and so, unfeasible to the cutting stock problem. Therefore, we have to find out an integer solution to the problem. In this paper some heuristic methods are reviewed from literature and new others are proposed. The heuristic solutions and computational effort are empirically analyzed based on randomly generated instances with small demand and few limited stock lengths to be cut.

Some heuristics analyzed in this paper are extensions of those studied by Wäscher and Gau [14] for solving the integer cutting stock problem with only one standard stock length available in limitless quantity. They compiled several heuristics from literature and proposed others, which are classified as basic, residual and composed, accepting oversupply of some items. Wäscher and Gau also analyzed the heuristics with relatively high demand and here we focus on instances with low demand.

Holthaus [9] studies the cutting stock problem with different stock lengths but without limitation on the stock availability. The author solves the relaxed problem by the column generation technique and uses two specific methods to reduce the residual problem. The final residual problem is solved by an ILP-Solver. The author states that this technique is suitable for solving medium-size and large instances of the one dimensional cutting stock problem.

The outline of this paper is as follows: in section 2 we briefly review the one-dimensional cutting stock problem and its mathematical formulation. In section 3 we describe two classical greedy heuristic procedures, commonly used when one is faced with low demand problems. Other approaches tested are based on the column generation technique, which are described in section 4. In section 5 there are some ideas on the implementation. In section 6, we describe the computational experiments. Finally, in section 7 we give some final remarks and future proposals.

2 Problem Definition and Mathematical Formulation

The *integer one-dimensional cutting stock problem* is stated as follows.

> Assume that we have available in stock K types of objects of given lengths $L_k, k = 1, \ldots, K$. Each stock length is available in quantity $e_k, k = 1, \ldots, K$. On the other hand, we have a set of m smaller ordered items of lengths $l_i, i = 1, \cdots, m$ that are required in quantities d_i, $i = 1, \cdots, m$ (item lengths are such as: $l_i \leqslant L_k$, for some k). The problem consists of producing the required items by cutting the available stock in such a way as to minimize the total waste of material. (Of course other objective functions could be stated, as well as the problem could be a multi-objective one, e.g., the waste and the number of changes in the cutting machine should be minimized.)

To summarize,

Demand data:

- m : the number of items;
- l_i : the length of the item $i, i = 1, \ldots, m$;
- d_i : demand for the item $i, i = 1, \ldots, m$.

Stock data:

- K : the number of different stock lengths available;
- L_k : the length of the stock object type $k, k = 1, \ldots, K$;
- e_k : the available quantity of stock length $k, k = 1, \ldots, K$.

The specific way of cutting a stock object into items is called *cutting pattern*, and let N_k be the number of cutting patterns for the stock object type $k, k = 1, \ldots, K$.

Decision variables (frequency):

- $\mathbf{x}_{jk}$: the number of stock lengths type k cut according to the cutting pattern $j, j = 1, \ldots, N_k, k = 1, \ldots, K$.

Assume no over production of any item, i. e., the total number of items type i produced equals d_i. A stock object is cut completely or left unused. Any piece of material that is not ordered item is considered waste.

It is associated with any cutting pattern, for a stock object k, an $m-$dimensional vector,

$$\mathbf{a_k} = (\alpha_{1k}, \alpha_{2k}, \cdots, \alpha_{mk})^T \tag{1}$$

where α_{ik} is the number of times that the item i appears in a cutting pattern for the stock object type k.

For each stock object type k, $k = 1, \ldots, K$, an associated vector to any cutting pattern has to be such as,

$$\begin{array}{l} l_1\alpha_{1k} + l_2\alpha_{2k} + \ldots + l_m\alpha_{mk} \leqslant L_k \\ 0 \leqslant \alpha_{ik} \leqslant d_i, \text{ and } \alpha_{ik} \text{ integer }, i = 1, \ldots, m \end{array} \tag{2}$$

Note that the cutting pattern has to be restricted ($\alpha_{ik} \leqslant d_i$), otherwise it could be useless.

After defining the cutting patterns, the next step is to determine the number of times that each cutting pattern should be used to solve the problem. Therefore, the mathematical modelling of a cutting stock problem is built in two steps,

1. define all the possible cutting patterns, for each stock object type k, $k = 1, \ldots, K$;

2. determine the number of times each cutting pattern should be used (frequency) to meet demand: this is an integer and non-negative number.

Let the associated vectors to cutting patterns for stock object type k be

$$\mathbf{a}_{1k} = \begin{pmatrix} \alpha_{11k} \\ \alpha_{21k} \\ \vdots \\ \alpha_{m1k} \end{pmatrix}, \quad \mathbf{a}_{2k} = \begin{pmatrix} \alpha_{12k} \\ \alpha_{22k} \\ \vdots \\ \alpha_{m2k} \end{pmatrix}, \quad \cdots, \quad \mathbf{a}_{N_k k} = \begin{pmatrix} \alpha_{1N_k k} \\ \alpha_{2N_k k} \\ \vdots \\ \alpha_{mN_k k} \end{pmatrix}, \quad k = 1, \ldots, K.$$

where α_{ijk} is the number of times the item i is present in the pattern j to the stock object type k, $i = 1, \ldots, m, j = 1, \ldots, N_k, k = 1, \ldots, K$, ($\mathbf{a}_{jk}$ is the vector which corresponds to the jth cutting pattern of the stock object type k).

We consider an additional information to describe the mathematical model: c_{jk} is the waste produced by the cutting pattern j to the stock object type k, given by $c_{jk} = L_k - \sum_{i=1}^{m} l_i \alpha_{ijk}$.

The problem can be formulated as follows,

$$\text{minimize } f(x_{11}, x_{12}, \ldots) = \sum_{j=1}^{N_1} c_{j1} x_{j1} + \sum_{j=1}^{N_2} c_{j2} x_{j2} + \ldots + \sum_{j=1}^{N_K} c_{jK} x_{jK} \tag{3}$$

$$\text{subject to: } \sum_{j=1}^{N_1} \mathbf{a}_{j1} x_{j1} + \sum_{j=1}^{N_2} \mathbf{a}_{j2} x_{j2} + \ldots + \sum_{j=1}^{N_K} \mathbf{a}_{jK} x_{jK} = \mathbf{d} \tag{4}$$

$$\begin{aligned} \sum_{j=1}^{N_1} x_{j1} &\leqslant e_1 \\ \sum_{j=1}^{N_2} x_{j2} &\leqslant e_2 \\ \ddots \quad &\vdots \\ \sum_{j=1}^{N_K} x_{jK} &\leqslant e_K \end{aligned} \tag{5}$$

$$x_{jk} \geqslant 0, \quad \text{and integer} \quad j = 1, \ldots, N_k, \quad k = 1, \ldots, K. \tag{6}$$

The objective function (3) consists of the total waste of material. The equality constraints in (4) assure that the total quantity of items produced meets demand. The constraints (5) assure that the number of each stock object type k cut does not exceed its availability e_k.

Note that a column of the constraints matrix is a vector associated with the cutting pattern of a stock object type k, but with additional K components with 1 in the $(m + k)$ position and 0 in the others. That is, a column of the mathematical model (3)-(6) is given by,

$$\begin{pmatrix} \alpha_{1jk} & \ldots & \alpha_{mjk} & 0 & \ldots & 1 & \ldots & 0 \end{pmatrix}^T. \tag{7}$$

For sake of simplicity to present the heuristics we write the mathematical model (3)-(6) in matrix notation,

$$\text{minimize} \quad f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} \tag{8}$$

$$\text{subject to:} \quad \mathbf{Ax} = \mathbf{d} \tag{9}$$

$$\mathbf{Ex} \leqslant \mathbf{e} \tag{10}$$

$$\mathbf{x} \geqslant \mathbf{0}, \text{ integer} \tag{11}$$

where $\mathbf{A}$ is the matrix of the cutting patterns in (4) and $\mathbf{E}$ is a matrix of 0's and 1's of the constraints in (5).

Integrality constraint (6) is commonly found in real-world cutting problems and, it makes the problem (3)-(6) difficult, if not impossible, to be exactly solved even for medium-size instances. Thus, heuristic approaches have to be devised to find integer solutions. We focus on a class of heuristics which first determine an optimal solution to the continuous relaxation, by applying the *column generation procedure* developed by Gilmore and Gomory [4, 5, 6]. This is a variant of the simplex method that has been adopted for the continuous relaxation of the cutting stock problem. First, we include the slack variables and the phase 1 of the simplex method is performed, with a basic matrix $((m + K) \times (m + K))$ and, in some cases, it will be necessary that an artificial problem be solved (details in section 5). At each simplex iteration, one of the basic patterns is replaced by a new cutting pattern that improves the current basic solution. Such cutting pattern (column) is determined by solving a knapsack problem with the constraints given in (2). From the optimal solution to the continuous relaxation of (3)-(6), which generally is non-integer, heuristics procedures have been developed to determine integer solutions (Wäscher and Gau [14], Holthaus [9], Stadtler [13], Poldi [10]).

3 Constructive Heuristics

Now, we present two well-known heuristics often used in practice (Wäscher and Gau [14], Stadtler [13]): *FFD (First-Fit-Decreasing)* and a *Greedy* heuristic. Hinxman [8] call this approach as *repeated exhaustion reduction.* These heuristics are used to solve the integer cutting stock problem as follows.

General constructive algorithm:

Step 1: Construct a good cutting pattern (FFD or greedy heuristics);
Step 2: Use the cutting pattern constructed in step 1 as many as possible, without exceeding demand of any item;
Step 3: Update demand and stock data, and repeat until demand is met.

This type of algorithm has been used to solve integer one dimensional cutting stock problems with low demand. Applying FFD or greedy heuristics to the cutting stock problem with different stock lengths consists of generating a cutting pattern for each type of stock object, and as the objective is minimizing waste, we choose among the generated cutting patterns the one with the smallest waste.

3.1 FFD Heuristic

The main idea of the *FFD heuristic* is to put in a cutting pattern the biggest items first an then fill the remaining space with smaller items.

FFD algorithm:

Step 0: { *Inicialization* }

Step 0.1:
Sort the demanded items in non-increasing order of length. To simply notation, assume that:

$$l_1 \geqslant l_2 \geqslant \ldots \geqslant l_m.$$

Step 0.2: { *Set the parameters* }
$r_i = d_i,\ i = 1, \ldots, m$ { r_i *is the residual demand for* $i = 1, \ldots, m$ }
$s_k = e_k,\ k = 1, \ldots, K$ { s_k *is the residual stock object type* $k, k = 1, \ldots, K$ }
$j = 1$ { *first cutting pattern* }

Step 1: { *Construct a good cutting pattern for each stock object* $k, k = 1, \ldots, K$ }

For $k = 1, \ldots, K$ do: { *building a good cutting pattern for stock object* }
If $s_k > 0$ then
$i = 1$;
$SPACE = L_k$;
{ $SPACE$ *means the remaining space in the stock object for which a cutting pattern is being built*}
While $\left(i \leqslant m \quad \text{and} \quad SPACE \geqslant l_m\right)$
{ *i. e., while there are free space remaining* }, do:

$$\alpha_{ik} = \min\left\{\left\lfloor \frac{SPACE}{l_i} \right\rfloor,\ r_i\right\};$$

{ $\mathbf{a}_k = (\alpha_{1k}, \ldots, \alpha_{mk})^T$ *is the vector associated with the cutting pattern*}
$SPACE = SPACE - (\alpha_{ik} l_i)$;
$i = i + 1$;
End-while
End-if
$w_k = L_k - SPACE$; { *waste in this cutting pattern* }
End-for

Step 2: { *Determine the best cutting pattern and the number of times it is used* }

Determine p such that:
{ *a simple greedy criterium is to choose the pattern with the minimal waste* }

$$w_p = \min\{w_k, \text{ such that } s_k > 0, \quad k = 1, \ldots, K\}.$$

Determine the number of times the cutting pattern is used:

$$x_{jp} = \min \left\{ s_p, \left\lfloor \frac{r_i}{\alpha_{ip}} \right\rfloor, \text{ where } \alpha_{ip} \neq 0, \quad i = 1, \ldots, m \right\}.$$

Step 3: { *Update demand and stock availability* }

$r_i = r_i - x_{jp}\alpha_{ip}, \quad i = 1, \ldots, m;$
$s_p = s_p - x_{jp};$
$j = j + 1;$
End.

3.2 Greedy Heuristic

The *greedy heuristic* consists of solving a knapsack problem in **Step 1** of the general constructive algorithm. Its algorithm is the same as described above, in section 3.1, for the FFD heuristic, but for the greedy heuristic it is not necessary **Step 0.1**, and **Step 1** changes as follows:

Greedy algorithm:

Step 1: { *Construct a good cutting pattern for each stock object* $k, k = 1, \ldots, K$ }

For $k = 1, \ldots, K$ do:
  If $s_k > 0$, then
    Solve the knapsack problem:

$$v_k = \text{maximize} \quad l_1\alpha_{1k} + l_2\alpha_{2k} + \ldots + l_m\alpha_{mk}$$
$$\text{subject to:} \begin{cases} l_1\alpha_{1k} + l_2\alpha_{2k} + \ldots + l_m\alpha_{mk} \leqslant L_k \\ 0 \leqslant \alpha_{ik} \leqslant r_i, \alpha_{ik} \text{ integer}, i = 1, \ldots, m; \end{cases} \tag{12}$$

    Do: $w_k = L_k - v_k$; { *waste in this cutting pattern* }
  End-if
End-for

4 Residual Approaches

Residual procedures generate an integer solution from an optimal solution to the continuous relaxation of the cutting stock problem (8)-(11). We assume that at least one component of the solution vector is non-integer, otherwise, the integer cutting stock problem would be already solved.

Definition 4.1 *(***Approximated integer solution***) Let* $\mathbf{x}$ *be a solution to the relaxation of (8)-(11), that is*

$$\mathbf{Ax} = \mathbf{d}$$
$$\mathbf{Ex} \leqslant \mathbf{e}.$$

Let $\mathbf{y}$ *be a vector of integers close, in some sense, to* $\mathbf{x}$*, an integer approximation of* $\mathbf{x}$ *such that:*

$$\mathbf{A}\mathbf{y} \leqslant \mathbf{d}$$
$$\mathbf{E}\mathbf{y} \leqslant \mathbf{e}.$$

Vector $\mathbf{y}$ *is called an approximated integer solution to* $\mathbf{x}$.

For instance, a simple technique to obtain an approximated integer solution is to round down $\mathbf{x}$: $\mathbf{y} = (\lfloor x_1 \rfloor, \lfloor x_2 \rfloor, \ldots, \lfloor x_n \rfloor)$. This simple approximated integer solution does not work well when demand is low, because $\mathbf{y}$ tends to be a null vector. In section 4.4 we give another procedure to determine an approximated integer solution. In authors' opinion, this is why the Gilmore and Gomory approach fails for problems with low demand. We will see that different approximated integer solutions to the continuous solution to the problem (8)-(11) produce better results.

Definition 4.2 *(***The Residual Problem***)*

Let $\mathbf{y}$ *be an approximated integer solution to* $\mathbf{x}$*, and* $\mathbf{r} = \mathbf{d} - \mathbf{A}\mathbf{y}$*, the residual demand, and* $\mathbf{s} = \mathbf{e} - \mathbf{E}\mathbf{y}$*, the residual availability of stock objects. The residual problem is given by:*

$$\textit{minimize} \quad \mathbf{c}^T\mathbf{x} \tag{13}$$

$$\textit{subject to:} \quad \mathbf{A}\mathbf{x} = \mathbf{r} \tag{14}$$

$$\mathbf{E}\mathbf{x} \leqslant \mathbf{s} \tag{15}$$

$$\mathbf{x} \geqslant \mathbf{0}, \textit{ integer.} \tag{16}$$

4.1 Residual Heuristics - A General Framework

In this section we present a general framework for residual heuristics, and in the next others, we specify some undefined steps.

The residual algorithm:

Step 1: { *Initialization* }

Let $\ell = 0$, $\mathbf{r}^0 = \mathbf{d}$ and $\mathbf{s}^0 = \mathbf{e}$ be the data for the starting residual problem (indeed, the original problem).

Step 2: { *Determining a continuous optimal solution* }

Solve the problem (8)-(11) with $\mathbf{r}^\ell$ instead of $\mathbf{d}$, and $\mathbf{s}^\ell$ instead of $\mathbf{e}$.

Let $\mathbf{x}^\ell$ be the continuous solution (the column generation technique is used).

If $\mathbf{x}^\ell$ is an integer solution, **then** STOP.

Step 3: { *Determining an approximated integer solution* }

Determine an approximated integer solution to $\mathbf{x}^\ell$ and denote it as $\mathbf{y}^\ell$.

If $\mathbf{y}^\ell$ is a null vector, **then** STOP.

Step 4: { *Updating* }

Determine the new residual demand and stock:

$\mathbf{r}^{\ell+1} = \mathbf{r}^{\ell} - \mathbf{A}\mathbf{y}^{\ell}$;
$\mathbf{s}^{\ell+1} = \mathbf{s}^{\ell} - \mathbf{E}\mathbf{y}^{\ell}$;
$\ell = \ell + 1$.
Repeat **Step 2**.

Final Step:

If the procedure is stopped in **Step 2**,
then a feasible integer solution to problem (8)-(11) is obtained;
otherwise, (stopped in **Step 3**) a residual problem lasts.

This algorithm will be completely defined by specifying how to determine $\mathbf{y}^{\ell}$, the approximated integer solution in **Step 3** and how to solve the final residual problem in the **Final Step**.

4.2 Residual Heuristics by Rounding Down

The three next heuristics: *residual FFD heuristic*, *residual greedy heuristic* and *residual bin-packing heuristic*, have the same **Step 3**, in the general algorithm. They differ one from each other only in the **Final Step** of the residual algorithm, section 4.1.

The approximated integer solution $\mathbf{y}^{\ell}$ in **Step 3** is determined by rounding down the continuous solution $\mathbf{x}^{\ell}$.

Step 3: { *Determining an approximated integer solution* }
Round down the continuous solution:

$$\mathbf{y}^{\ell} = \lfloor \mathbf{x}^{\ell} \rfloor$$

The differences will appear only in the **Final Step**, as we will see in the next sections.

4.2.1 Residual FFD Heuristic

Final Step:
Solve the remaining residual problem by the FFD heuristic (section 3.1).

4.2.2 Residual Greedy Heuristic

Final Step:
Solve the remaining residual problem by the greedy heuristic (section 3.2).

4.2.3 Residual Bin-packing Heuristic

Final Step:
Solve the remaining residual problem as a multiple sized bin-packing packing problem which can be stated as follows.

The remaining stock objects and the remaining ordered items are enumerated one by one, even in case of repetition. For sake of simplicity, let n be the total quantity of remaining stock objects (possibly, $n = \sum_{k=1}^{K} s_k^{\ell}$, where ℓ is the last iteration in the residual algorithm), and m' be the total quantity of remaining ordered items ($m' = \sum_{i=1}^{m} r_i^{\ell}$). The length of the stock object type j is denoted by L_j and the length of item type i is denoted by l_i.

Let the decision variables be defined as,

$$y_j = \begin{cases} 1, & \text{if object } j \text{ is used;} \\ 0, & \text{otherwise.} \end{cases}$$

$$x_{ij} = \begin{cases} 1, & \text{if item } i \text{ is in the object } j; \\ 0, & \text{otherwise.} \end{cases}$$

$$\begin{aligned} \text{minimize} \quad & \sum_{j}^{n} L_j y_j \\ \text{subject to:} \quad & \sum_{i=1}^{m'} l_i x_{ij} \leqslant L_j y_j, \quad j = 1, \ldots, n; \\ & \sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1, \ldots, m'; \\ & y_j \in \{0,1\}, \qquad j = 1, \ldots, n; \\ & x_{ij} \in \{0,1\}, \qquad j = 1, \ldots, n, i = 1, \ldots, m'. \end{aligned} \tag{17}$$

Note: The *residual bin-packing heuristic* consists of solving the residual problem in the **Final Step** of the residual algorithm given in section 4.1 to optimality, as a bin-packing problem. This is an extension of a heuristic in Wäscher and Gau [14], who studied the cutting problem with only one type of standard stock length. In their paper, the residual problem was solved as a classical bin-packing problem, which is not an easy integer programming problem. In the case with different types in stock, the bin-packing problem has to be changed to include the several types of stock object, and this turns the bin-packing problem even more difficult to solve.

4.3 Stadtler's Heuristic

Stadtler [13] proposed a residual heuristic for cutting stock problems with only one standard stock length available in limitless quantity. His procedure was extended here to consider more than one type of stock object.

All components of vector $\mathbf{x}$, which are integers, are fixed at those values. The component of $\mathbf{x}$ with the largest fractional value is rounded up to the next integer. Note that at least one component is fixed in each iteration.

However, some ordered items may be undersupply, giving rise to a residual problem. Stadtler solves this residual problem heuristicly by the application of the FFD algorithm, section 3.1. Wäscher and Gau [14] solve this residual problem by an exact bin-packing algorithm. In the results presented in this paper, the residual problems were solved by the FFD heuristic.

Stadtler used a modified approximated integer solution which may not match the definition 4.1, since $\mathbf{Ay} \leqslant \mathbf{d}$ might not be met. The Stadtler's approximated integer solution is given as follows.

Step 3 in the residual algorithm presented in section 4.1 we have,

(i) let $y_{ij} = x_{ij}$ for all (i,j) such that x_{ij} is integer;

(ii) let $y_{ij} = \lceil x_{ij} \rceil$ for (i,j) such that $\lceil x_{ij} \rceil - x_{ij} = \max\{\lceil x_{pq} \rceil - x_{pq}, \forall(p,q)\}$;

(iii) let $y_{ij} = \lfloor x_{ij} \rfloor$ for all others (i,j).

Therefore, the approximated integer solution $\mathbf{y}^\ell$, in **Step 3** of the residual algorithm, determined as described above might be such that $\mathbf{Ay}^\ell \nleq \mathbf{r}^\ell$, so the updating in **Step 4** must be slightly modified to have $\mathbf{r}^{\ell+1} \geqslant 0$, let

$$\widetilde{\mathbf{r}}^{\ell+1} = \mathbf{r}^\ell - \mathbf{Ay}^\ell,$$

and,

$$\mathbf{r}_k^{\ell+1} = \max\left\{0, \widetilde{\mathbf{r}}_k^{\ell+1}\right\}.$$

Stopping criterium can be modified since $\mathbf{y}^\ell = \mathbf{0}$ in **Step 3** only if $\mathbf{r}^\ell = \mathbf{0}$. When the residual algorithm stops, it is usual oversupply occurs. Therefore, before going to **Final Step**, it is necessary to eliminate the oversupply. Stadtler proposed the following procedure.

(i) Identify the cutting pattern which contains the largest part of items in oversupply.

(ii) Reduce the frequency of this cutting pattern by one.

(iii) Repeat steps (i) and (ii) until there is no more oversupply.

After eliminating oversupplying as described above, there is a remaining residual problem to be solved. So,

Final Step:

Solve the remaining residual problem by the FFD heuristic.

4.4 A New Residual Heuristic

Now we present a new residual heuristic with a different way to obtain the approximate integer solution, in **Step 3** of the residual algorithm, in section 4.1. Residual FFD and residual greedy heuristics round down all frequencies. Note that it is unfeasible to round up all frequencies, but some of them can be rounded up and others rounded down. The Stadtler's heuristic [13], presented in section 4.3, also proceeds in this way, but it always rounds up only the frequency with the biggest fractional part in each iteration, which may cause oversupply.

We developed three versions of the new heuristic. The components of the continuous solution $\mathbf{x}^\ell$ obtained in **Step 2** is sorted in a specific way that will be described later, the way of sorting is the difference among the three versions. Beginning with the first cutting pattern, its frequency x_1 is rounded up and its feasibility is tested (no oversupply is allowed): $y_{1j} = \lceil x_{1j} \rceil$. If

$\mathbf{y} = (y_{1j}, 0, \ldots, 0)$ is such that $\mathbf{Ay} \geqslant \mathbf{d}$, i.e., oversupply occurred, then the solution is reduced by one until oversupply is eliminated, $y_{1j} = y_{1j} - 1$. Next, with the first component already rounded up or down, we repeat for the second cutting pattern, $\mathbf{y} = (y_{1j}, y_{2k}, 0, \ldots, 0)$, and go on up to the last generated cutting pattern. This produces a vector $\mathbf{y}$ which is an integer approximate solution. It should be noted that the column generation procedure is such that each cutting pattern generated can be used at least once (see (2)), so that at least one frequency could be rounded up. This assures that the residual demand becomes smaller in the next iteration, and so, at the end, the residual demand is null.

Notation: Up to now, for sake of simplicity, we have denoted $\mathbf{x}_{jk}$ the frequency of cutting pattern j for stock object k. But, as we have a list of cutting patterns, each one associated with its stock object, we use the following notation:

$$\mathbf{x}_{jk_j}$$

to denote the frequency of cutting pattern j for stock object type k_j. Therefore, we enumerate all cutting patterns; instead of referring $(\mathbf{x}_{11}, \mathbf{x}_{13}, \ldots)$: cutting pattern 1 for stock object type 1, cutting pattern 1 for stock object type 3, etc., we say $(\mathbf{x}_{11}, \mathbf{x}_{23}, \ldots)$, i.e., $j = 1, k_1 = 1; j = 2, k_j = 3; \ldots$

To summarize, for this heuristic, we have to specify the **Step 3** of the residual algorithm in section 4.1. Here, this step is divided in two. **Step 3.1** is a preprocessing and **Step 3.2** is the rounding procedure.

The new residual algorithm:

Step 3: { *Getting an approximated integer solution* }

Step 3.1: { *preprocessing* }
Sort the frequency vector $(\mathbf{x}^{\ell})$ of the cutting patterns obtained in **Step 2**.
Later on, we will give criteria to sort them.

Step 3.2: { *rounding* }
Let T be the total number of cutting patterns generated in the optimal continuous solution in **Step 2**.
For $i = 1, \ldots, T$ do:
$y_{ik_i} = \min\{\lceil x_{ik_i} \rceil,\ s_{k_i}\}$
$s_{k_i} = s_{k_i} - y_{ik_i}$ { *update residual stock availability* }
While $\mathbf{Ay} \not\leqslant \mathbf{r}$, do: { *oversupply* }
$y_{ik_i} = y_{ik_i} - 1$
$s_{ik_i} = s_{ik_i} + 1$

End.

4.4.1 A New Residual Heuristic - version 1

To completely define the heuristic, we state the **Step 3.1**. Sort the vector $\mathbf{x}$ by non-increasing order, i.e, we give priority to the most used cutting patterns.

Step 3.1: { *preprocessing: high frequency priority* }
Sort the components of $\mathbf{x}$ such that,

$$x_{1k_1} \geqslant x_{2k_2} \geqslant \ldots \geqslant x_{Tk_T}.$$

Example:

Consider we have a cutting stock problem, with $K = 2$ types of stock objects: object type 1 lengths $L_1 = 130$ and object type 2 lengths $L_2 = 105$. Suppose that we have to produce $m = 3$ types of items, which lengths and demands are given on Table 1.

Table 1: *Demanded Items.*

Item	Length	Demand
1	$l_1 = 42$	$d_1 = 35$
2	$l_2 = 37$	$d_2 = 20$
3	$l_3 = 23$	$d_3 = 50$

First, we solve the relaxed problem (8)-(11) by the simplex method with column generation. We get a continuous solution, shown on Table 2.

Table 2: *Continuous solution obtained by column generation:* $\mathbf{d} = (35, 20, 50)^t$.

$\mathbf{x}^1$	Object	Cutting Pattern
9.375	1	$(2, 0, 2)^t$
3.750	1	$(0, 1, 4)^t$
16.250	2	$(1, 1, 1)^t$

Then, sort the solution vector $\mathbf{x}^1 = (16.25; 9.375; 3.75)^t$. Try to round up the first component, $x^1_{12} = y^1_{12} = \lceil 16.25 \rceil = 17$. We can easily note that this integer solution $\mathbf{y}^1 = (17; 0; 0)^t$ is a feasible solution. Then, go to the second component, $x^1_{21} = y^1_{21} = \lceil 9.375 \rceil = 10$. Note that the integer solution $\mathbf{y}^1 = (17; 10; 0)^t$ is infeasible, it produces $(\mathbf{37}; 17; 37)^t$ which exceeds the production of item type 1. So, decrease the frequency by 1, getting the integer solution $\mathbf{y}^1 = (17; 9; 0)^t$, which is a feasible solution because its production is $(35; 17; 35)^t$ (no oversupply). Finally, we deal with the third component, $x^1_{31} = y^1_{31} = \lceil 3.75 \rceil = 4$. Again the integer solution $\mathbf{y}_1 = (17; 9; 4)^t$, is infeasible because it causes oversupply of items 2 and 3 (production is $(37; \mathbf{21}; \mathbf{51})^t$). Decrement the frequency by one, getting the integer solution $\mathbf{y}_1 = (17; 9; 3)^t$. Update demand and solve the problem again by the method simplex with column generation, but now, considering the residual demand: $\mathbf{r} = (0, 0, 3)^t$. The continuous solution in the second iteration is given by one cutting pattern, and it is cut one stock object type 2 by the cutting pattern $(0, 0, 3)^t$. So the final integer solution is given on Table 3.

Table 3: *Final integer solution.*

$\mathbf{y}$	Object	Cutting Pattern
17	2	$(1, 1, 1)^t$
9	1	$(2, 0, 2)^t$
3	1	$(0, 1, 4)^t$
1	2	$(0, 0, 3)^t$

4.4.2 A New Residual Heuristic - version 2

This version uses another criterium of sorting the cutting patterns. Let w_{jk} be the waste in the cutting pattern j of the stock object k.

Step 3.1: { *preprocessing: low waste priority* }
Sort the components of **w** such that,

$$w_{1k_1} \leqslant w_{2k_2} \leqslant \ldots \leqslant w_{Tk_T}.$$

4.4.3 A New Residual Heuristic - version 3

This version uses another criterium of sorting the cutting patterns. Let $f_{jk} = x_{jk} - \lfloor x_{jk} \rfloor$ be the fractional value of x_{jk}.

Step 3.1: { *preprocessing: high fractional value priority* }
Sort the components of **f** such that,

$$f_{1k_1} \geqslant f_{2k_2} \geqslant \ldots \geqslant f_{Tk_T}.$$

5 Implementation Issues

All the algorithms described in sections 3 and 4 were implemented in DELPHI 5, and ran on a Pentium III (866MHz - 256MB RAM). A home made simplex method with column generation as proposed by Gilmore and Gomory [4] was used. A branch-and-bound method was used for solving the constrained knapsack problem (it is a modification of Gilmore and Gomory [5] to take into account of upper bounds). The bin-packing problem (17), was solved by CPLEX 7.5.

The initial basis was built as follows. Let $\mu_k \geqslant 0$ be the slack variables of constraints (5) and the model (3)-(6) becomes,

$$\begin{array}{lllll}
\sum_{j=1}^{N_1} \mathbf{a}_{j1} x_{j1} + & \sum_{j=1}^{N_2} \mathbf{a}_{j2} x_{j2} + & \ldots + & \sum_{j=1}^{N_K} \mathbf{a}_{jK} x_{jK} & = \mathbf{d} \\
\sum_{j=1}^{N_1} x_{j1} & & & +\mu_1 & = e_1 \\
& \sum_{j=1}^{N_2} x_{j2} & & +\mu_2 & = e_2 \\
& & \ddots & \vdots & \\
& & & \sum_{j=1}^{N_K} x_{jK} \quad +\mu_K & = e_K
\end{array} \tag{18}$$

Assume that $L_1 \geqslant L_k, k = 2, \ldots, K$, i.e., the stock object type 1 is the largest one. Also, assume that $l_i \leqslant L_1, i = 1, \ldots, m$, otherwise, the problem would be infeasible. So, we can build m columns associated with simple cutting patterns, which have $m + K$ components,

$$\mathbf{a}_{j1} = (0, \ldots, b_{jj}, 0, \ldots, 1, 0, \ldots, 0)^T \tag{19}$$

where $b_{jj} = \min\left\{\left\lfloor \frac{L_1}{l_j} \right\rfloor, \quad d_j\right\}, \quad j = 1, \ldots, m.$

The formula to calculate b_{jj} above is needed because demand may be so low that $\left\lfloor \frac{L_1}{l_j} \right\rfloor > d_j$, that is, a single stock object cut causes oversupply of item j.

Note that any basis has $m + K$ columns so we need more K columns, which may be columns associated with the slack variables. Also note that when considering the system (18) with only the columns in (19), i. e., setting $x_{jk} = 0$ for the other variables (non-basic variables), the $m-$first equations are enough to well-determine the value of $x_{j1}, j = 1, \ldots, m$:

$$x_{j1} = \frac{d_j}{b_{jj}}, \quad j = 1, \ldots, m. \tag{20}$$

We have to analyze wether this solution is feasible (case 1) or infeasible (case 2).

Case 1: If $\sum_{j=1}^{m} x_{j1} \leqslant e_1$, then we have a feasible solution and the initial basic matrix $\mathbf{B}$, of dimension $(m + K)$, can be written as:

$$\mathbf{B} = \begin{bmatrix} b_{11} & \ldots & 0 & 0 & \ldots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & b_{mm} & 0 & \ldots & 0 \\ 1 & \ldots & 1 & 1 & \ldots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & 0 & 0 & \ldots & 1 \end{bmatrix} \tag{21}$$

Note that the column related to the slack variable $\mu_{\mathbf{k}}$ is:

$$(0, \ldots, 0, 0, \ldots, 1, \ldots, 0)^T$$

and the value 1 is in the $(m + k)$ position.

Note, also, that stock objects type $2, 3, \ldots, K$ were not used and the constraints (18) are satisfied with:

$$\mu_1 = e_1 - \sum_{j=1}^{m} x_{j1} \qquad \text{and}$$
$$\mu_j = e_j, \quad j = 2, \ldots, K,$$

all slack variables are basic variables.

Case 2: If $\sum_{j=1}^{m} x_{j1} > e_1$, then the solution given by (20) is infeasible and it is necessary a phase I of the simplex method. The big-M artificial problem has just one artificial variable μ_1^a, related to the violated constraint $m + 1$ (first stock constraint):

$$\text{minimize} \quad f_a = \sum_{j=1}^{N_1} c_{j1}x_{j1} + \sum_{j=1}^{N_2} c_{j2}x_{j2} + \ldots + \sum_{j=1}^{N_K} c_{jK}x_{jK} + M\mu_1^a \tag{22}$$

$$\text{subject to:} \quad \sum_{j=1}^{N_1} \mathbf{a}_{j1}x_{j1} + \sum_{j=1}^{N_2} \mathbf{a}_{j2}x_{j2} + \ldots + \sum_{j=1}^{N_K} \mathbf{a}_{jK}x_{jK} = \mathbf{d} \tag{23}$$

$$\begin{aligned}
\sum_{j=1}^{N_1} x_{j1} \qquad\qquad\qquad\qquad & +\mu_1 \quad -\mu_1^a = e_1 \\
\sum_{j=1}^{N_2} x_{j2} \qquad\qquad & +\mu_2 \qquad = e_2 \\
\ddots \qquad\qquad & \qquad\qquad \vdots \\
\sum_{j=1}^{N_K} x_{jK} \quad & +\mu_K \qquad = e_K
\end{aligned} \tag{24}$$

$$x_{jk} \geqslant 0 \text{ and integer}, \ \mu_j \geqslant 0, \ \mu_1^a \geqslant 0, \ j = 1, \ldots, N_k, \ k = 1, \ldots, K. \tag{25}$$

We consider the same basis for the artificial problem as in (21), but the slack variable $\mu_1 < 0$ is replaced by the artificial variable μ_1^a. The initial basis for the artificial problem is given by:

$$\mathbf{B} = \begin{bmatrix}
b_{11} & \ldots & 0 & 0 & \ldots & 0 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & \ldots & b_{mm} & 0 & \ldots & 0 \\
1 & \ldots & 1 & -1 & \ldots & 0 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & \ldots & 0 & 0 & \ldots & 1
\end{bmatrix}$$

Note that $\mu_1^a = -e_1 + \sum_{j=1}^{m} x_{j1}$ (it is $-\mu_1$ in case 1) and the other variables have the same values as before. If in the optimal solution to the artificial problem (22)-(25), $\mu_1^a > 0$ then it means that there is no feasible basic solution to the original cutting stock problem. If, by the other hand, $\mu_1^a = 0$ at any simplex iteration, then the artificial variable μ_1^a is dropped.

6 Computational Tests

In order to evaluate the heuristic procedures described in sections 3 and 4, 240 randomly generated instances were solved. These instances are grouped into 12 classes defined below.

6.1 The Random Generator

A random generator was created based on some ideas found in Foerster and Wäscher [2] and Gau and Wäscher [3]. These authors created a generator for cutting stock problems with only one standard stock length available in limitless quantity. Here we have some extra input parameters which have to be generated:

- *Number of types of stock objects:* $K = 3, 5$ and 7.
- *Stock object lengths:* The values of $L_k, k = 1, \ldots, K$ were randomly generated in the interval $[10, 100]$.
- *Availability in stock:* The values $e_k, k = 1, \ldots, K$ were randomly generated in the interval $[1, 100\frac{m}{2}]$.
- *Number of types of ordered items:* $m = 5, 10$ and 20;
- *Item lengths:* The item lengths l_i were randomly generated in the interval $[v_1 L, v_2 L]$, where L is the average value among the $L_k, k = 1, \ldots, K$, as used in Wäscher and Gau [14]: $v_1 = 0.01$ and, $v_2 = 0.2$ and 0.8. Combining these values, classes of instances were generated with *small items* ($v_2 = 0.2$) and *medium items* ($v_2 = 0.8$).
- *Demand:* $d_i, i = 1, \ldots, m$ were randomly generated in the interval $[1, 10]$.

Therefore, 12 classes of instances were created by the parameter combination K, m, l_i, $K = (3, 5$ and $7)$, $m = (5$ and 20 or 10 and 20) and l_i (small and medium). 20 instances were randomly generated for each one of these classes. These classes are described in Table 4

6.2 Results

Table 5 shows the average number of cutting patterns in each one of the classes. Table 6 shows the total waste (in average) occurred in each class, which is the objective to be minimized. In these tables the value in italic is the worst and in boldface is the best for each class. The computational times, given in seconds, are shown in Table 7.

Table 4: *Parameters which describe the classes.*

	Parameters		
Class	K	m	Items
1	3	5	S
2	3	5	M
3	3	20	S
4	3	20	M
5	5	10	S
6	5	10	M
7	5	20	S
8	5	20	M
9	7	10	S
10	7	10	M
11	7	20	S
12	7	20	M

In Table 5, we can note that the new heuristics and Stadtler's heuristics presented the best solutions in terms of the number of cutting patterns. On the other hand, the residual

Table 5: *Average of the 20 instances in each one of the 12 classes.*

	Number of cutting patterns								
	Constructive		**Residual**						
	FFD	*Greedy*	*FFD*	*Greedy*	*RBP*	*Stadtler*	*New 1*	*New 2*	*New 3*
1	4.85	4.85	*4.90*	4.85	4.15	3.95	4.00	4.00	**3.90**
2	6.95	6.55	6.45	6.20	*8.60*	5.90	**5.20**	5.25	5.75
3	14.10	13.25	13.90	13.35	*16.75*	9.20	**9.15**	**9.15**	**9.15**
4	21.40	20.20	*24.65*	21.05	21.70	18.45	16.05	**16.00**	17.70
5	*7.90*	7.60	*7.90*	7.60	7.70	**5.30**	**5.30**	**5.30**	**5.30**
6	11.85	11.45	*13.75*	12.20	10.65	11.15	**9.00**	9.15	9.85
7	15.40	14.75	15.15	14.35	*17.05*	**8.05**	8.40	8.40	8.45
8	21.60	19.25	*26.00*	21.30	21.65	18.30	15.40	**15.30**	18.05
9	*9.75*	9.05	9.65	9.00	9.05	**5.05**	5.35	5.35	5.40
10	12.30	11.50	*13.75*	11.90	11.00	10.20	**8.55**	8.70	9.45
11	14.20	13.55	13.90	13.65	*14.75*	8.35	8.30	8.35	**8.25**
12	21.40	19.05	*22.85*	18.80	18.95	17.25	14.60	**14.55**	16.55

FFD and residual bin-packing tend to use more cutting patterns than the others heuristics. Minimizing the number of cutting problems is not the objective proposed in this article, but it is an interesting by-product because it is important in practice since changing cutting patterns can cause undesired setup time.

Table 6: *Average of the 20 instances in each one of the 12 classes.*

	Total waste								
	Constructive		**Residual**						
	FFD	*Greedy*	*FFD*	*Greedy*	*RBP* *	*Stadtler*	*New 1*	*New 2*	*New 3*
1	*346.05*	314.85	340.80	312.30	**66.00**	150.95	166.85	166.85	152.40
2	*2353.80*	1161.10	1225.75	738.80	450.35	815.85	455.90	**443.65**	471.10
3	306.65	424.55	298.30	*434.00*	151.65	144.80	121.60	121.60	**94.45**
4	6030.65	6030.65	*8938.15*	6802.75	192.50	1916.70	**168.90**	171.50	216.60
5	160.35	*232.75*	160.35	*232.75*	**49.40**	87.05	84.45	85.45	85.45
6	3486.45	2334.40	*3954.75*	1919.15	**270.55**	791.60	334.55	387.75	344.15
7	1174.65	684.90	*1190.80*	693.50	144.90	**77.65**	95.95	100.15	90.80
8	6416.05	5508.75	*9227.10*	4123.95	191.75	420.95	197.65	171.15	**140.40**
9	587.80	397.10	*593.95*	395.75	60.30	57.20	56.95	51.10	**43.50**
10	4187.15	2368.25	*4480.25*	1862.05	**73.35**	783.00	125.50	115.35	136.05
11	280.00	286.80	282.87	*287.30*	223.25	**83.30**	96.30	91.85	99.30
12	*8427.35*	2838.50	7483.45	2511.50	188.20	1219.25	95.35	**60.45**	88.70

* The bin-packing problems (17) were solved by CPLEX 7.5, with running time limited in 10 minutes.

Minimizing waste was the objective proposed and Table 6 shows that FFD and greedy heuristics (constructive and residual) produce very poorer solutions than the others. Curiously, Stadtler's heuristic may present the best solution to some classes and very poor for others.

Table 7: *Computational times (in seconds) for solving 240 instances.*

	Constructive		Residual						
	FFD	*Greedy*	*FFD*	*Greedy*	*RBP*	*Stadtler*	*New 1*	*New 2*	*New 3*
each	0.01	0.09	2.48	2.73	567.59	7.17	2.57	2.59	2.17
total	3.20	22.00	596.60	657.00	136221.49	1721.20	617.20	623.60	508.00

In the "**each**" row in Table 7 are the average computational times to solve each instance in the total of 240 instances for each method. In the "**total**" row are the total computational times to solve all the 240 instance by each method. Note that 567.59 seconds corresponds to 9.46 minutes. We limited the CPLEX processing time in 10 minutes to solve the residual bin-packing problem, and 210 from among 240 instances stopped because of the time limit. We can note that the computational time for the Residual bin-packing heuristic is impracticable, although the results in Table 6 could be improved for RBP.

For a better comparative analysis between constructive and residual approaches we evaluated the average values given in previous tables for the two constructive and called it *constructives*. We also evaluated the average values for the three versions of the new heuristic and we will refer to them as *new*. So, our analysis proceeds in constructive versus residual heuristic approaches.

As we can see in Figure 1, although the residual bin-packing heuristic performs well in terms of waste, it produces higher numbers of cutting patterns.

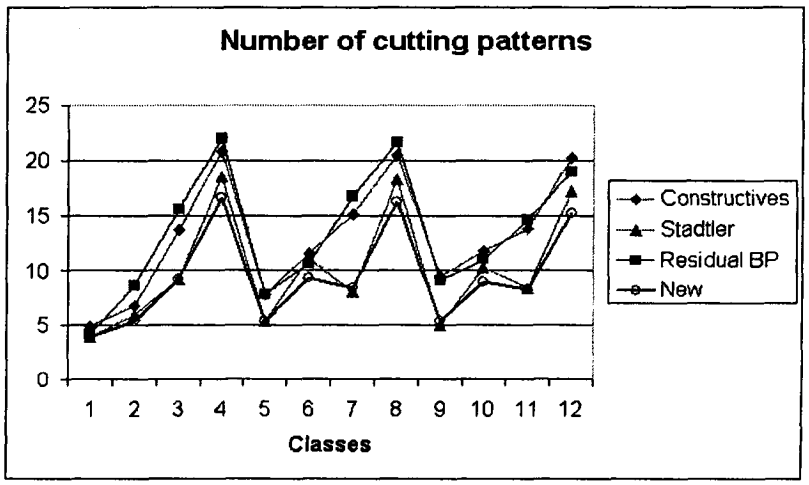

Figure 1: *Average number of cutting patterns.*

Figure 2 clearly shows that constructive heuristics may produce very poor solutions.

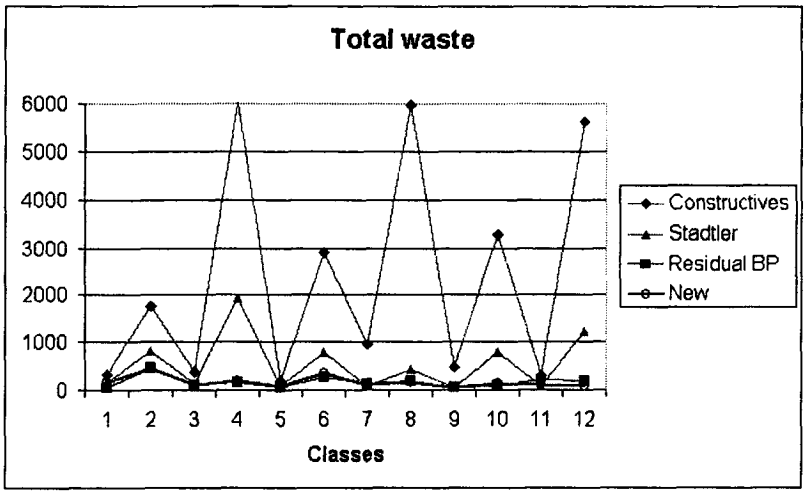

Figure 2: *Total waste (average).*

Figure 3 depicts the performance of the residual bin-packing and the new heuristics, where we can see a similar behavior of both heuristics but the new ones are hundred times faster than residual bin-packing.

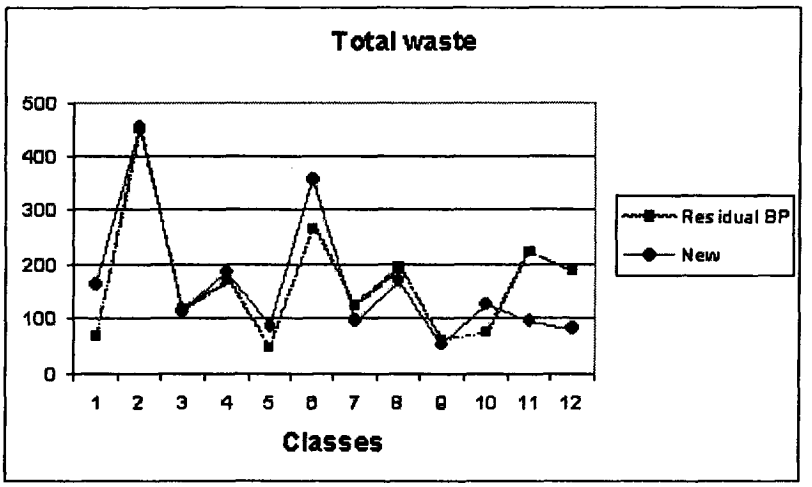

Figure 3: *Residual bin-packing versus new residual heuristics*

7 Conclusions

In this paper we studied some heuristics to solve the integer one dimensional cutting stock problem with different stock lengths. The heuristics were classified in two classes: constructive and residual. In order to analyze the heuristics behavior, we randomly generated instances that included low demand for ordered items and low availability of stock objects. On the opposition to

the cutting and packing area folklore, which suggests that constructive heuristics should be used when one is faced with low demand instances (Riehme *et al.* [11]), we found out that residual heuristics (based on the linear programming model of Gilmore and Gomory [4, 5]) are better. In our opinion the reason why researchers have disregarded residual heuristics is that the rounding down procedure was used to find an approximated integer solution, which produced a null vector due to low demand. So, all the cutting patterns generated worth nothing (frequencies are less than one). The new residual heuristics overcame this drawback. Furthermore, the new residual heuristics can be straightforward extended to two or more dimensions cutting stock problems, to be investigated in a future work.

8 Acknowledgment

The authors thank the support of FAPESP and CNPq.

References

[1] DYCKHOFF, H., FINKE, U., (1992) "*Cutting and packing in production and distribution: typology and bibliography*". Springer-Verlag, Heidelberg.

[2] FOERSTER, H., WÄSCHER, G., (2000) "*Pattern reduction in one-dimensional cutting-stock problems*". International Journal of Production Research, 38: 1657-1676.

[3] GAU, T., WÄSCHER, G., (1995) "*CUTGEN: A problem generator for the standard one-dimensional cutting stock problem*". European Journal of Operational Research, 84: 572-579.

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

[5] GILMORE, P. C., GOMORY, R. E., (1963) "*A linear programming approach to the cutting stock problem - Part II*". Operations Research, 11: 863-888.

[6] GILMORE, P. C., GOMORY, R. E., (1965) "*Multi-stage cutting stock problems of two and more dimensions*". Operations Research, 13: 94-120.

[7] HAESSLER, R. W., (1980) "*A note on computational modifications to the Gilmore-Gomory cutting stock algorithm*". Operations Research, 28: 1001-1005.

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

[9] HOLTHAUS, O., (2002) "*Decomposition approaches for solving the integer one-dimensional cutting stock problem with different types of standard lengths*". European Journal of Operational Research, 141: 295-312.

[10] POLDI, K. C., (2003) "*Algumas extensões do problema de corte de estoque*". Master's dissertation, ICMC - USP.

[11] RIEHME, J., SCHEITHAUER, G., TERNO, J., (1996) "*The solution of two-stage guillotine cutting stock problems having extremely varying order demands*". European Journal of Operational Research, 91: 543-552.

[12] SCHEITHAUER, G., TERNO, J., (1995) "*The modified Integer Round-up Property of the One-dimensional Cutting Stock Problem*". European Journal of Operational Research, 84: 562-571.

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

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

NOTAS DO ICMC

SÉRIE COMPUTAÇÃO

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

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

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

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

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

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

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

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

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

075/2004 NONATO, L.G.; CASTELO FILHO, A.; CAMPOS, J.E.P.P.; BÍSCARO, H.; MINGHIM, R. – Topological tetrahedron characterization with applications in volumetric reconstruction.