**EngOpt 2008 - International Conference on Engineering Optimization**
Rio de Janeiro, Brazil, 01 - 05 June 2008.

# One Dimensional Cutting Stock Problem with Redevelopment of the Surplus Material

**Silas Sallaume, Marcio T. Mine, Matheus de S. A. Silva, Luiz S. Ochi, Alexandre Plastino**

Universidade Federal Fluminense - Niterói, Rio de Janeiro, Brazil
{ssallaume, mmine, msalves, satoru, plastino}@ic.uff.br

**Marcone Jamilson Freitas Souza**

Universidade Federal de Ouro Preto – Ouro Preto, Minas Gerais, Brazil
marcone@iceb.ufop.br

**Abstract**

This work deals with the One Dimensional Cutting Stock Problem, in which a demand of a set of small pieces named items, gotten from large pieces named objects, has to be produced. It is desirable that the cutting layouts have a small waste or the remaining piece must be big enough to be reused in the future. This is an important problem, since even small improvements in the cutting layouts result in large savings of raw material and energy when the amount of produced material is huge. The Cutting Stock Problem is NP-hard, so a heuristic combining the Modified First Fit Decreasing (FFD) method, an Integer Programming (IP) model and a Data Mining (DM) module is proposed to solve it. The FFD heuristic is used to produce an upper bound. With this upper bound the IP model becomes able to be solved more quickly. Then, the DM module is proposed to extract common parts from the best current solutions obtained from several iterations of the IP model. Finally the extracted common parts are used to guide the search for better solutions in less computational time. Computational results have shown that the proposed method is able to generate good quality and competitive solutions when compared to the ones presented in the related literature.

**Keywords: Cutting Stock Problem; Heuristics; Integer Programming; Data Mining.**

## 1. Introduction

The Cutting Stock Problem (CSP) can be defined as the process of cutting larger pieces (objects), available in stock for the production of smaller parts (items), in a manner to attempt a specific demand. Among the goals to be achieved there is a minimization of wastes, the excesses of production and cutting patterns to be used.

This kind of problem appears in several industrial processes, such as cutting reels of paper and aluminum, steel bars, sheet metal and wood, from printed circuit boards, cardboard boxes, rolls of fabric, among others (see, for example, [9, 10, 11 and 17]). Thus, the CSP is essential for the planning of production in these industries and the reduction of production costs and improving efficiency are often associated with the use of appropriate strategies of cuts.

Cutting stock problems are, in general, formulated as integer linear programming problems. The methods of resolution that has greater impact in the literature were proposed by Gilmore and Gomory [5, 6 and 7]. In [5] the integrality of the variables of decision is relaxed and a method of generating columns to the resulting linear problem is proposed, in which each column represents a pattern of cutting. However, problems such as cutting belong to the class of NP-hard problem, for which the exact techniques (methods of enumeration implicit and variants, for example, branch-and-cut and branch-and-price) have limited applicability. Thus, researches in this area have walked towards developing heuristics techniques.

The Unidimensional Cutting Stock Problem (UCSP) can be defined as follows: given a sufficient number of objects of length $L$ and a demand $d_i$, $i = 1, ..., m$, of items with width $l_i$, $i = 1, ..., m$, where $l_i \leq L$, the problem is to cut the items from objects, satisfying the quantity of items demanded in order to optimize a function of evaluation, which may be the loss of material, the cost of process, the number of objects used, among other goals.

In [16], the UCSP treaty is aiming to reduce the number of larger parts being cut to meet the demand. Determining a pattern of cutting to one dimensional problem represents a classic case of literature, known as the Knapsack Problem, therefore it mainly consists of allocating smaller units within a larger unit in the most valuable possible manner [14]. Three restrictions are considered, namely: (*i*) it is not allowed to over-produce, (*ii*) objects stock should be cut completely, and (*iii*) pieces of cuts that are not demanded items are considered loss. Defined the patterns of cutting, the next step is to define how many times each pattern will be used to meet the demand. This amount of time should be integer and non-negative, which makes the problem difficult to be computationally solved. One way to solve this problem is to relax the condition of completeness and solve the problem relaxed through Simplex Method [4], using the process of generation of columns, proposed by [5]. The optimal solution to the relaxed problem is generally fractional, making heuristics to determine the rounding of the developed solutions. In the proposed heuristic, called Nova, each iteration solves the relaxed cutting stock problem and orders up the patterns in accordance with how often they were used. Each pattern is associated with an $m$-dimensional array $\alpha$, where each component $\alpha_i$ represents the amount of items of type $i$ present in the cut pattern. For each position of the array $\alpha$, the frequency that each item is present in the pattern is rounded to the

integer number above the fractional obtained. After this step, the viability of this solution is evaluated, that is, if excesses of items were not generated. If the solution is infeasible, the frequency is reduced of one until excesses are removed. When the last generated pattern for cutting is examined, the demand is updated, the residual problem is solved and the rounding procedure is repeated until all the demand is satisfied. The Residual Problem achieves an integer solution for the problem from an approximated integer solution. Since this approximated integer solution does not meet the entire demand, several sub-problems (residual) whose demand is the residual demand must be satisfied, that is, the remaining demand to be attempted. These sub-problems can be solved through rounding techniques, and if this technique can not satisfy the demand, then a final residual problem is solved using heuristics that produce integer solutions, such as greedy heuristics. Computational results showed that solving the original problem with the relaxed integrality condition and a rounding strategy is quite efficient. Moreover, it was observed that the heuristic Nova obtained, on average, the lowest number of patterns from the heuristics tested.

In [16], the one dimensional integer cutting stock problem is treated with supply restrictions. It is considered in this work that the demand is small and there are diverse types of bars in supply available in limited amounts. The resolution of the problem is made by means of the constructive heuristics FFD (First-Fit-Decreasing) and Gulosa and of residual heuristics FFD, Gulosa, Nova and Nova version 2. According to the computational tests, the authors had identified that the residual heuristics, in the special one the heuristics proposed by them (Nova and Nova version 2), had presented better results of the ones purely constructive. Thus, on the basis of these results, the authors had opposed the affirmation made in [19] of that residual heuristics are not appropriate for problems of low demand.

The FFD classic heuristic consists of putting the biggest item in a cutting pattern as many times as possible, until there are no more space to place this item, or until their demand has already been attempted (the larger items are placed in firstly, because they are more difficult to be combined). So, when it is not possible or necessary to include the largest item, the second largest item is considered and so on. When any new item can be included, a cut pattern is made and repeated as many times as possible without exceeding that demand. This procedure to generate a good cutting pattern and repeat it as often as possible is known in the literature as exhaustive repetition heuristic [13] and it is employed in the cutting and packing.

In [2], it is shown, also, another way to build a good cut pattern for the repetition exhaustive heuristic, that consists in generate a cutting pattern with minimal loss through the resolution of the Knapsack Problem.

In traditional methods of solving the UCSP, it is considered a loss each cut piece that is not a demanded item. These pieces are usually discarded or reused as raw material. But many cutting problems permit reuse, for a future demand, the pieces not used. Also low losses are always desirable. The possibility of reuse introduces a change in the criteria for a solution selection. An alternative to solve this problem is to form cutting patterns that the surplus concentrated in a few standards and are large enough to return to stock and use again. This is the so-called problem of one dimensional cuts of stock with reuse of the surplus material, introduced in [2] and [3].

To generate a set of ideal patterns, or at least acceptable, the works [2] and [3] introduce changes in FFD heuristic, naming it FFD modified heuristic. The heuristic proposes to apply the FFD heuristic for obtaining cutting patterns and apply them to attempt the demand of the items. For each cutting pattern generated by FFD is considered the loss/over generated. If the loss/over is within acceptable limiting (defined previously), the next cut standard is considered. Otherwise, the pattern is subject to a correction procedure, based on the resolution of the knapsack problem [16].

This paper presents a new heuristic method to solve the One-dimensional Cutting Stock Problem with Redevelopment of the Surplus Material. The proposed method generates an initial solution based on FFD Modified heuristic [2] and uses this value as an upper bound for a model of Integer Linear Programming and the module of Data Mining. These techniques work together to generate a pool of solutions and at the end of the process the best solution is returned.

Data mining refers to the extraction of new and potentially useful knowledge from datasets [12]. The key motivation to incorporate a data mining process in heuristics is that it could be used to extract common parts of good solutions, which represent features of sub-optimal solutions. Then, these parts could be used to guide the search for better solutions in less computational time. Recent applications of this kind of hybridization have achieved promising results [18], [20] and [21]. In this work, we incorporate a data mining algorithm to improve the performance of the proposed heuristic.

This paper is organized as follows. In section 2, the problem under study is described in detail. In section 3, a methodology to solve it is presented. In section 4, the results are presented and analyzed. Section 6 concludes the work and makes suggestions for improving the method proposed.

## 2. Description of the approach problem

The One-dimensional Cutting Stock Problem of with Redevelopment of Surplus Material (UCSP-RSM) consists in attempt a demand of a set of parts (items), obtained from cutting larger pieces (objects), such that the losses arising from the cutting of the objects are small enough or large enough to be redeveloped. In the latter case, the losses are considered surplus.

For the definition of a small loss and a surplus, let $L$ the length of a larger piece (object), $\beta$ the maximum acceptable loss percentage, $\beta L$ the size of the acceptable loss and $\delta$the size of minimum acceptable surplus.

Thus, a cutting pattern is acceptable if the unused part is less or equal to $\beta L$ or whether that part is greater than or equal to $\delta$. Otherwise, the standard is considered unacceptable.

For a better understanding of the UCSP-RSM, consider the following example, taken from [1], which must be cut 2 items of 5 meters and 2 of 3 meters from objects of 10 meters. Consider that the stock is 2 objects of 10 meters and that the parameters $\beta L$ and $\delta$ are, respectively, 2 and 4 meters. Figures 1 and 2 shows two solutions that can be obtained for this example. Consider, in these pictures, that the dark piece represents an unused part of the pattern (loss or surplus).

Figure 1. Solution 1 of the UCSP-RSM

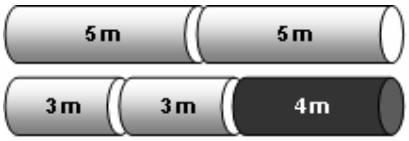

Figure 2. Solution 2 of the UCSP-RSM

For the conventional one dimensional cutting stock problem, both Solutions 1 and 2 are equivalent because of the same total loss (4 meters). But in the view of the UCSP-RSM, the Solution 2 is preferable than Solution 1, as the losses are concentrated in one part and therefore is a surplus that can be reused to attempt future demands.

## 3. Metodology

### 3.1. Solution Representation

A solution $s = (s_{ij})$ of the problem is represented by a matrix, where the lines represents the cutting patterns and the columns are the information about each pattern. The first column (Pattern) refers to the identification of the pattern, the second column (Object) says which object was used to cut the items of that pattern. The third column (Item) represents how many times each item is used in that cutting layout. The penultimate column (Rest), relates to the unused space in the object and the last column (NAP) is the number of applications of that pattern.

The Table 1 shows a solution for the PCEU-RSM, considering 5 types of items with sizes of 240, 180, 170, 140 and 125 cm, whose demands are, respectively, 20, 18, 25, 10 and 15 units. In this example, the following object supply is considered: 7 units of Object 1, which has 3100 cm of length and 5 units of Object 2, which has 2200 cm of length.

Table 1. Solution representation for the PCEU-RSM

| Pattern | Object | Items | | | | | Rest | NAP |
|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | | |
| 1 | 1 | 11 | 1 | 0 | 2 | 0 | 0 | 1 |
| 2 | 1 | 8 | 5 | 0 | 2 | 0 | 0 | 1 |
| 3 | 2 | 0 | 8 | 2 | 3 | 0 | 0 | 1 |
| 4 | 1 | 0 | 4 | 14 | 0 | 0 | 0 | 1 |
| 5 | 2 | 0 | 0 | 9 | 3 | 2 | 0 | 1 |
| 6 | 1 | 1 | 0 | 0 | 5 | 8 | 260 | 1 |

To sample the referring information about a pattern belonging to the solution, consider the line 3 of Table 1: The pattern 3 will be cut using the object of type 2 (3100 cm). In this pattern eight items of type 2, two items of the type 3 and three items of type 4 will be cut. This pattern does not have any rest and will be applied only once (NAP equal 1), that is, an object of type 2 will be used to cut pattern 3.

### 3.2. Objective function

A solution $s$ is evaluated by the function $f$ give by Formula 1, which should be minimized:

$$f(s) = \sum_{i \in K} nap_i \times (waste_i + surplus_i \times \omega) + \alpha \times \sum_{j \in M} productionExcess_j + \sum_{r \in C} \gamma_r \times inv_r \quad (1)$$

where
- $f(s)$: evaluation function of the solution $s$;
- $K$: set of patterns of the solution $s$;
- $M$: set of items that should be cut;
- $C$: set of constraints;
- $nap_i$: number of applications of the pattern $i \in K$;
- $perda_i$: waste of the pattern $i \in K$;
- $sobra_i$: surplus of the pattern $i \in K$;
- $\omega$: constant between 0 and 1 that establishes the weight of the surplus on the waste;
- $\alpha$: penalty for exceeding the demand of the items;
- $productionExcess_j$: quantity of item $j \in M$ build beyond the demand;
- $\gamma_j$: penalty for disregarding the constraint $j \in C$;
- $inv_j$: times that the constraint $j \in C$ is disregarded.

The objective function of this problem is still an opened concept, especially with regard to the constant $\omega$, as it is completely dependent on the practical application of the solution. Were not found in the literature, information regarding the cost of this

relationship between waste and surplus ( $\omega$ ), but it is believed that this value is very small.

### 3.3. Initial Solution Generation

The initial solution that can be generated by methods which will be described in sections 3.3.1 and 3.3.2 will be used as an upper bound for the algorithm proposed in section 3.6.

#### 3.3.1. Classical FFD Heuristic

The classical FDD heuristic[16] or FFD-C, is, initially, put the largest item in a cutting pattern the maximum times possible, that is, until there is no more space to place this item or until your demand has been answered. So when is no longer possible or necessary to include the largest item, the second largest item is considered and so on. When it is not possible to include more items, the pattern is ready and it is applied the maximum possible times, according to the restrictions of stock of the object and not exceeding the demand of the items in the pattern. Then the demands of the items and the stock of objects are updated and a new pattern is built in the same way. The method stops when the demands are fully satisfied.

The pseudo-code of the FFD-C is shown in Figure 3. In this figure, *I* is the list of items placed in descending order of size, $pattern_k$ is the *k-th* pattern generated by FFD-C heuristic and $nap_k$ refers to the number of times that the k-th pattern is applied. The *rest* means the space available to allocate an item in the current pattern, *L* is the size of the object and $l_j$ and $d_j$ are, respectively, the length and demand of the *j-th* item.

1) $k \leftarrow 1$
2) **while** ( $|I| \neq 0$ ) **do**
   a) $item \leftarrow$ first element of the list *I*
   b) $pattern_k \leftarrow \phi$ ; $nap_k \leftarrow 0$; $rest \leftarrow L$
   c) **while** ( $rest \geq \min(l_j \mid j \in I)$ ) **do**
      i) **if** ( $rest \geq l_{item}$ ) **then**
         (1) $pattern_k \leftarrow pattern_k \cup \{item\}$
         (2) $rest \leftarrow rest - l_{item}$
      **ii) else**
         *(1)* $item \leftarrow$ next element of the list *I*
   d) $nap_k \leftarrow \min\{\lfloor d_j / \alpha_j \rfloor \forall j \in I\}$, where $\alpha_j$ is the number of times that *j* is in the pattern *k*.
   e) $I \leftarrow I - \{item\}$
   f) $k \leftarrow k + 1$
   g) update the demand of the items
3) return as a solution the sets *pattern and nap*

Figure 3. Algorithm of the classical FDD heuristic.

To better understand this heuristic, consider the example below:

Be the one-dimensional CSP composed of objects of 7 meters to be cut to items of 4, 3 and 2 meters. Consider that the demand of the items is: 89 of 4 m, 59 of 3 m and 92 of 2 m. Suppose that the availability of objects is enough to answer all the demand. Initially, the items are ordered in descending order of length (4 m> 3 m> 2 m). Then, a pattern considering the bar (object) of 7 m is build up, placing the item of greater size (4 m) until there is no more space or until its demand is reached. Thus, the pattern will be formed, initially, by an item of 4m. As the remaining space (3 m) is larger than the smallest item (2 m), whose demand has not been satisfied yet, then it tries to put the next item (3 m). Finally, it is obtained the first pattern (A), since there is no space to add any more item. This pattern is shown in Figure 4.


Figure 4. First cutting pattern (A) generated by the Classical FFD heuristic.

Established a pattern, it should be applied at the most times possible, without exceeding the demands of the items involved (in this case, the pattern A it is applied 59 times). Thus, the demand of the item 3 will be fully answered.

As the total demand has not been satisfied yet, it is necessary to update the demands and repeat the process, but without consider the item 3, because its demand has been completely answered. The new demands are: 89 - 59 = 30 for the 4 m item e 92 to the item of 2 m. Thus, the second pattern (B) generated by heuristic is shown in Figure 5.


Figure 5. Second cutting pattern (B) generated by the Classical FFD heuristic.

In Figure 5, the highlighted item represents the rest (1m). As the lower demand to be met is of a 4m item (30), it is necessary to apply the pattern B 30 times to accommodate their demands. Thus, there remains unsatisfied demand of the item of 2 m (i.e. it is necessary to produce 62 items of this size). Therefore, the final pattern (C) generated is shown in Figure 6.

Figure 6. Last cutting pattern (C) generated by the Classical FFD heuristic.

Applying the pattern of the Figure 6, the final solution will be, the application of 59 times the pattern A, 30 times the pattern B and 21times the pattern C. Then the total waste generated will be 50 x 0m + 30 x 1 m = 51 m and the total surplus will be 2 m.

### 3.3.2. Modified FFD Heuristic

The Modified FFD heuristic [1], or FFD-M, consists in generate a solution to the PCEU-RSM. In this problem, the unused part in the pattern generated can be classified as waste or surplus. Will be waste if the length of this part is equal or less than $\beta L$, being $\beta$ a percentage of the object of size $L$. It will be surplus when that length is greater or equal to a value $\delta$. If this length is not waste neither surplus, then the pattern will be classified as unacceptable. Otherwise, the pattern will be considered acceptable.

Basically, the FFD-M generates a pattern using the classical FFD (FFD-C). If the pattern generated is acceptable, applies the pattern until, at least, one of the demands of the items belonging to the pattern is met or until there are no more stock of object used by the pattern. If the generated pattern is unacceptable, takes place the following procedure: removes one occurrence of biggest item in the pattern. Then, build up a model of mathematical programming for the acquisition of complementary part of the pattern generated so far. This model is formulated in accordance with the Knapsack Problem, where the capacity is the remaining space in the pattern. If the pattern generated by the Knapsack Problem is unacceptable, this procedure is repeated in order to generate an acceptable standard. If the pattern continues even undesirable, then one occurrence of the smallest item is removed from the last pattern generated by the previous procedure. This process is repeated until the rest of the pattern is considered a waste.

1) Be $pattern_k$ the initial pattern for the object $k$ (method parameter)
2) while ( exists any item in the $pattern_k$) do
   a) remove one occurrence of the biggest item in the $pattern_k$
   b) $R \leftarrow$ rest of the $pattern_k$
   c) $dr_i \leftarrow$ residual demand of the item $i$
   d) solve the Knapsack Problem with capacity $R$ generating a pattern $\gamma$:

$$z(s) = \max\ l_1 s_1 + l_2 s_2 + \cdots + l_m s_m$$

$$subject\ to \begin{cases} l_1 s_1 + l_2 s_2 + \cdots + l_m s_m \le R \\ 0 \le s_i \le dr_i, i = 1, \ldots, m \end{cases}$$

   e) $\rho \leftarrow pattern_k \cup \gamma$
   f) if ( pattern $\rho$ is acceptable ) then
      i) return the pattern $\rho$
3) return the pattern with the minor waste/surplus for the object $k$ by the resolution of the Knapsack Problem with capacity $R$, where $R$ is the length of the object $k$

Figure 8. Procedure of patterns generation by the Knapsack Problem.

The FFD-M generates patterns until the entire demand of the items be answered or until all the stock of the objects be used. Figure 7 presents the pseudo-code of modified FFD heuristic and Figure 8 shows the procedure whose purpose is to try to generate a pattern with the minimum waste or surplus through the resolution of the Knapsack Problem.

## 3.4. Mathematical formulation for the PCEU-RSM

In this section is proposed a model of Integer Linear Programming (Figure 9), able to solve the PCEU-RSM. As can be seen in Figure 9, given a set of patterns, the demands of the items and the objects in the stock, the model founds the best combination of patterns in a manner to minimize the total waste/surplus.

Minimize

$$f(x) = \sum_{i \in P} (p_i \times x_i + \omega \times s_i \times x_i) \quad (1)$$

Subject to:

$$\sum_{i \in P} a_{ij} \times x_i = d_j \forall j \in I \quad (2)$$

$$\sum_{i \in P} c_{ik} \times x_i \leq e_k \forall k \in O \quad (3)$$

$$x_i \in Z^+ \forall i \in P \quad (4)$$

where
- $f(x)$: evaluation function of the solution $x$;
- $P$: set of cutting patterns;
- $I$: set of items that should be cut;
- $O$: set of objects in the stock;
- $p_i$: waste of the pattern $i \in P$;
- $s_i$: surplus of the pattern $i \in P$;
- $\omega$: constant between 0 and 1 that establishes the weight of the surplus on the waste;
- $x_i$: number of times that the pattern $i$ is used;
- $a_{ij}$: number of times that the item $j$ is in the pattern $i$;
- $d_i$: demand of the item $j$;
- $c_{ik}$: 1 if the pattern $i$ uses the object of the type $k$ and 0 otherwise;
- $e_k$: supply of the object $k$;

Figure 10. Mathematic model proposed to solve the PCEU-RSM

In (1) is the evaluation function, where the waste and the surplus are taken into consideration. Although the weight of the surplus is multiplied by the constant $\omega$. The function was thus defined since that the reduction of the waste is the main goal of the problem and $\omega$ is used to represent the importance of the surplus in relation to the loss. The constraints in (2) make the model, answer all the demand. In (3) is restricted to the stock of objects available. Finally, (4) are defined the constraints of completeness and non-negativity of the model.

### 3.5. PLDM-PCEU-RSM Algorithm

To solve the PCEU-RSM, an algorithm called PLDM-PCEU-RSM is proposed. Basically, given the available objects and the demanded items, this algorithm generates the set of all possible cutting patterns. Then, it generates an initial solution and uses it as an upper bound, which allows the extraction of a small part of the set which contains all possible patterns. With a smaller group of patterns, it is possible to use the exact method. Figure 11 shows the pseudo-code of the PLDM-PCEU-RSM algorithm.

```
1   poolOfSolutions = createPool( sizePool )
2   s0 ← ModifiedFFD( )
3   upperboundWaste ← s0.getTotalWaste( )
4   upperboundSurplus ← s0.getTotalSurplus( )
5   P ← generateAllPossiblePatterns( )
6   P' ← getPatternsBetterThanTheUpperBounds(P, upperboundWaste, upperboundSurplus )
7   iter ← 0;
8   while (P'.size( ) > solverCapacity && iter < iterMax) do
9     P' ← randomicPattern(P', solverCapacity, minedCommonParts)
10    s ← solveMathModel(P')
11    poolofSolutions.update(s )
12    upperboundWaste ← poolofSolutions.getUpperBoundWaste( )
13    upperboundSurplus ← poolofSolutions.getUpperBoundSurplus( )
14    P' ← getPatternsBetterThanTheUpperBounds(P, upperboundWaste, upperboundSurplus )
15    if (iter mod freqMine == 0) do
16      minedCommonParts = minePool(minimumSupport)
17    end if
18    iter ← iter + 1;
19  end while
20  if (P'.size( ) <= solverCapacity) do
21    s ← solveMathModel(P')
22    poolofSolutions.update(s )
23  end if
24  s* ← poolOfSolutions.getBest( )
25  return s*
```

Figure 11. PLDM-PCEU-RSM Algorithm proposed to solve the PCEU-RSM.

The algorithm starts creating a pool of size *sizePool* with the elite solutions. In line 2, the method proposed in section 3.3.2 is used to create an initial solution, $s_0$. The value of the total waste of the solution $s_0$ is assigned to the the *upperboundWaste* variable and the value of the total surplus of the solution $s_0$ is assigned to the *upperboundSurplus* variable. After this, a set of all possible cutting patterns, *P*, is created. Then, a subset of *P*, which contains only the patterns with waste less or equal to the *upperboundWaste* and surplus less or equal *upperboundSurplus*, is placed in *P'*. Since the size of *P'* is greater than the maximum number of patterns which solver can deal with, *solverCapacity*, and the number of iterations, *iter*, is less than *iterMax*, *solverCapacity* patterns of *P'* are randomly chosen and the mathematic model is solved (section 3.4) for these patterns. Then the pool is updated with the new solution and the values of *upperboundWaste* and *upperboundSurplus* variables are also changed. In lines 15 to 17, in each *freqMine* iteration, the procedure of data mining is called. This procedure uses the FPMax* algorithm [11] to mine common parts of the solutions. To be mined, these common parts must occur in *minimumSupport* solutions of the pool. These parts, obtained by the data mining strategy, are placed in the set *minedCommonParts*. These mined common parts are used during the random choice of cutting patterns done in line 9. In this procedure, different cutting patterns are chosen randomly, however some are pre-defined using the set of parts extracted by the data mining module. The intention of this procedure is to find good solutions more quickly. Upon exiting the loop of line 8 to 19, if the size of the set *P'* is less or equal to the maximum number of patterns that the solver can manage, the model is solved for the *P'* set. Then the pool is updated with the obtained solution and finally, on lines 24 and 25, the best solution is retrieved from the pool and it is returned.

## 4. Computational Experiments

This section presents the computational results obtained by PLDM-PCEU-RSM method to solve the UCSP-RSM. The algorithm was developed in C++ language using the Borland C++ Builder environment, version 6.0, and tested in an Intel Core 2 Duo 1.6 GHz computer with 1.5 GB of RAM memory, with the Windows Vista operational system. To validate it, five instances were used, which are available in [1], [2] and [3], and which contain data of the One Dimensional Cutting Stock Problem with a limited object supply.

For each instance, 15 runs were performed, each one starting from a different seed of random numbers. The parameters adopted in the PLDM-PCEU-RSM method were: *sizePool* = 10, *solverCapacity* = 400, *freqMine* = 30 and *minimumSupport* = 0.3. Tables 15, 16, 17, 18 and 19 show a comparison between the PLDM-PCEU-RSM method results and the ones obtained in [1], [2] and [3]. In these tables, *Method* represents the heuristic used to solve the problem, *#CuttingLayouts* refers to the number of cutting layouts used, *TotalWaste* and *TotalSurplus* are related to the total waste and the total surplus provided from the object cutting, *#Objects* represents the number of objects used from supply, *#SCL* is the number of surplus cutting layouts, and *#WCL* is the number of waste cutting layouts.

Table 15. Comparison of methodologies for the CA2005A instance [2]

| Method | #CuttingLayouts | TotalWaste | TotalSurplus | #Objects |
|---|---|---|---|---|
| Heuristica FFD Pura [2] | 8 | 500 | 4550 | 20 |
| Heuristica FFD Modificada [2] | **6** | 5 | 1205 | **12** |
| Heurística Gulosa Pura [2] | 9 | 133 | 667 | 15 |
| Heurística Residual FFD [2] | 8 | 105 | 5005 | 16 |
| Heurística Residual FFD Modificada [2] | **6** | 9 | 1201 | **12** |
| Heurística Residual Gulosa [2] | 7 | 10 | **600** | **12** |
| PLDM-PCEU-RSM | **6** | **0** | 1210 | **12** |

Table 16. Comparison of methodologies for the CA2005B instance [3]

| Method | #CuttingLayouts | TotalWaste | TotalSurplus | #Objects |
|---|---|---|---|---|
| Heuristica FFD Pura [3] | 6 | 192 | 2905 | 11 |
| Heuristica FFD Modificada [3] | 6 | 8 | 946 | 11 |
| Heurística Gulosa Pura [3] | 8 | 90 | 753 | 9 |
| Heurística Gulosa Modificada [3] | 7 | **0** | 843 | 9 |
| Heurística Residual FFD [3] | 6 | 78 | 876 | 11 |
| Heurística Residual FFD Modificada [3] | 6 | 8 | 946 | 11 |
| Heurística Residual Gulosa [3] | 6 | 206 | 753 | 11 |
| Heurística Residual Gulosa Modificada [3] | 5 | **0** | 843 | 9 |
| Heurística Nova [3] | 7 | 23 | 931 | 11 |
| Heurística Nova Modificada [3] | 6 | 4 | 950 | 11 |
| PLDM-PCEU-RSM | **3** | **0** | **752** | **7** |

Table 17. Comparison of methodologies for the instance *Exemplo 1* [1]

| Method | TotalWaste | TotalSurplus | #SCL | #WCL | #Objects |
|---|---|---|---|---|---|
| CUT | 62 | 2408 | **1** | 7 | 14 |
| Heurística FFD Modificada | 18 | 6907 | 7 | 8 | 18 |
| Heurística Gulosa Modificada | 3 | 5106 | 9 | 1 | 17 |
| Heurística Residual FFD Modificada [2] | 2 | 3276 | 2 | 2 | 16 |
| Heurística Residual Gulosa Modificada [2] | 2 | 3276 | 2 | 2 | 16 |
| Heurística Residual RAG1 [2] | 1 | 2225 | 2 | 1 | 15 |
| Heurística Residual RAG2 [2] | **0** | 2264 | 2 | **0** | 15 |
| Heurística Residual RAG3 [2] | 1 | **2202** | 4 | 1 | 16 |
| PLDM-PCEU-RSM | **0** | **2239** | **7** | **0** | **12** |

Table 18. Comparison of methodologies for the instance *Exemplo 2* [1]

| Method | TotalWaste | TotalSurplus | #SCL | #WCL | #Objects |
|---|---|---|---|---|---|
| CUT | 5 | **743** | **1** | 3 | 11 |
| Heurística FFD Modificada | 10 | 4765 | 7 | 5 | 13 |
| Heurística Gulosa Modificada | 1 | 2715 | 4 | 1 | 12 |
| Heurística Residual FFD Modificada [2] | **0** | 2125 | 3 | **0** | 11 |
| Heurística Residual Gulosa Modificada [2] | 3 | 2852 | 2 | 1 | 11 |
| Heurística Residual RAG1 [2] | **0** | 1857 | **1** | **0** | **10** |
| Heurística Residual RAG2 [2] | **0** | 1857 | **1** | **0** | **10** |
| Heurística Residual RAG3 [2] | **0** | 1560 | 2 | **0** | **10** |
| PLDM-PCEU-RSM | **0** | **1333** | **6** | **0** | **10** |

Table 19. Comparison of methodologies for the instance *Exemplo 3* [1]

| Method | TotalWaste | TotalSurplus | #SCL | #WCL | #Objects |
|---|---|---|---|---|---|
| CUT | 104 | **1017** | **1** | 10 | **15** |
| Heurística FFD Modificada | 11 | 4618 | 7 | 5 | 16 |
| Heurística Gulosa Modificada | 17 | 1902 | 2 | 8 | 16 |
| Heurística Residual FFD Modificada [1] | 3 | 3132 | 3 | 3 | **15** |
| Heurística Residual Gulosa Modificada [1] | 9 | 1066 | **1** | 5 | **15** |
| Heurística Residual RAG1 [1] | 6 | 2367 | 2 | 4 | **15** |
| Heurística Residual RAG2 [1] | 3 | 3071 | 4 | 3 | 16 |
| Heurística Residual RAG3 [1] | 4 | 4901 | 3 | 4 | **15** |
| PLDM-PCEU-RSM | **0** | **2120** | **10** | **0** | **16** |

Based on the results obtained, it can be observed the following solution aspects generated by the PLDM-PCEU-RSM in relation to the heuristics proposed by [1], [2] and [3]. From the experiment with CA2005A instance, it can be observed that the solution obtained by the proposed method was better in relation to the total waste, equal in the numbers of cutting layouts and objects used and satisfactory in the total surplus. It is important to note that the proposed method is superior, when compared to each heuristic. In relation to the CA2005B instance, the PLDM-PCEU-RSM is equivalent in the total waste and superior in all other attributes. For the instances *Exemplo* 1, 2 and 3, the proposed methodology performed well and the solutions obtained had always a minimum waste and a reasonable surplus.

The computational results demonstrate that the PLDM-PCEU-RSM method can produce high quality solutions with little waste and objects from supply. These aspects influence directly in the reduction of the costs involved in the acquisition of the objects used.

Furthermore, it can be verified that the proposed method used a few number of cutting layouts (#CuttingLayouts). In real cases, this contributes to speed up the production process, since the setup time, i.e., the time needed to configure the cutting machine, according to the cutting layout used, increases proportionally to the number of cutting layouts used.

Thus, we can be concluded that the proposed method obtained better results when compared to the ones in the literature.

## 5. Conclusions and Future Works

This paper presents a hybrid method, which uses heuristics, integer linear programming and data mining for the solution of the One Dimensional Cutting Stock Problem with Redevelopment of Surplus Material. For the generation of an initial solution *s*, it was used the Modified FFD Heuristic, proposed by [1], which generates a solution to the problem in a deterministic manner. The value of the objective function of *s* was used as upper bound for the other steps of the method.

With the use of a good solution as upper bound, it was possible to greatly reduce the number of potentially promising patterns, which made it possible to use the exact model. The module of data mining was useful since it helped the method to find good solutions more quickly.

According to the results, we can conclude that the algorithm PLDM-PCEU-RSM is an efficient alternative for the solution of the UCSP-RSM. This is due to the fact that the solutions generated by the method have characteristics that are superior or at least similar to those obtained by [1], [2] and [3], namely the total waste, the number of used patterns, the quantity of patterns with surplus, and the number of objects used in the stock.

For future work, maybe using the solutions stored in the pool could be a good approach. For instance the Path Relinking Heuristic [8] can execute this issue.

*Acknowledgements*

The authors thank FAPEMIG (process TEC 679/06), FAPERJ (process E-26/101.023-2007), and CNPq for the support given to the development of this study.

**6. References**

1. A. C. Cherri, (2006), O problema de corte de estoque com reaproveitamento da sobras de material. MS Dissertation, ICMC - USP, São Carlos, SP, Brazil.
2. A. C. Cherri, M. N. Arenales, (2005a). O Problema de Corte de Estoque com Reaproveitamento das Sobras de Material – Heurística FFD Modificada. In: Anais do XXXVII Simpósio Brasileiro de Pesquisa Operacional, Gramado, p. 1700-1711.
3. A. C. Cherri, M. N. Arenales, (2005b). Heurísticas para o problema de corte com reaproveitamento das sobras de material. In: XXVIII Congresso Nacional de Matemática Aplicada e Computacional, 2005, São Paulo. Anais do XXVIII SBMAC. v. 1, p. 1-6.
4. G. B. Dantzig, (1963). Linear Programming and Extensions, Princeton University Press, Princeton, New Jersey.
5. P. Gilmore, R. Gomory, (1961). A linear programming approach to the cutting stock problem. *Operations Research*, v.9, p.849-859.
6. P. Gilmore, R. Gomory, (1963). A linear programming approach to the cutting-stock problem II. *Operations Research*, v.11, p.863-888.
7. P. Gilmore, R. Gomory, (1965). Multistage cutting stock problems of two and more dimensions. *Operations Research*, v.14, p.94-120.
8. F. Glover, G. A. Kochenberger, (2003) *Handbook of Metaheuristics*, Kluwer Academic Publishers, Boston.
9. M. Gradisar.; J. Jesenko.; C. Resinovic, (1997) Optimization of roll cutting in clothing industry. *Computers and Operational Research*, v. 10, p. 945-953.
10. M. Gradisar,; C. Resinovic,; M. Kljajic, (1999) A hybrid approach for optimization of one-dimensional cutting. *European Journal of Operational Research*, v. 119, p. 719-728.
11. G. Grahne, J. Zhu, (2003) Efficiently using prefix-trees in mining frequent itemsets. IEEE ICDM FIMI Workshop.
12. J. Han, M. Kamber, (2006) Data Mining: Concepts and Techniques, Morgan Kaufmann.
13. A. Hinxman, (1980) The trim-loss and assortment problems: a survey. European Journal of Operational Research, v. 5, p. 8-18.
14. G. C. F. Pileggi; R. Morabito; M. N. Arenales, (2002) Duas abordagens para o problema integrado de geração e seqüenciamento de padrões de corte. In: XXXIV Simpósio Brasileiro de Pesquisa Operacional, Rio de Janeiro. Anais do XXXIV SBPO.
15. K. C. Poldi, (2002) Heurísticas para o problema de corte de estoque unidimensional inteiro. In: XXXIV Simpósio Brasileiro de Pesquisa Operacional, 2002, Rio de Janeiro. Anais do XXXIV SBPO.
16. K. C. Poldi, (2003) Algumas extensões do problema de corte de estoque. Dissertação de mestrado, ICMC – USP.
17. K. C. Poldi,; M. N. Arenales, (2003) O problema de corte de estoque unidimensional inteiro com restrições de estoque. In: XXXV Simpósio Brasileiro de Pesquisa Operacional, 2003, Natal/RN. Anais do XXXV SBPO, p. 1498-1509.
18. M. H. Ribeiro, A. Plastino, S. L. Martins, (2006) Hybridization of GRASP Metaheuristic with Data Mining Techniques, Journal of Mathematical Modeling and Algorithms, v. 5, p. 23-41.
19. J. Riehme, G. Scheithauer, J. Terno, (1996) The solution of two-stage guillotine cutting stock problems having extremely varying order demands. European Journal of Operational Research, v. 91, p. 543-552.
20. L. F. M. Santos, R. Milagres, C. Albuquerque, A. Plastino, S. L. Martins, (2006) A Hybrid GRASP with Data Mining for Efficient Server Replication for Realiable Multicast, Proceedings of the IEEE GLOBECOM Conference.
21. L. F. M. Santos, M. H. Ribeiro, A. Plastino, S. L. Martins, (2005) A Hybrid GRASP with Data Mining for the Maximum Diversity Problem, Proceedings of the International Workshop on Hybrid Metaheuristics, p. 69-78.