Journal of Combinatorial Optimization (2021) 42:657–674
https://doi.org/10.1007/s10878-021-00695-4

Modified Greedy Heuristic for the one-dimensional cutting stock problem

Gonçalo R. L. Cerqueira[1] · **Sérgio S. Aguiar**[1] · **Marlos Marques**[1]

Accepted: 23 December 2020 / Published online: 18 February 2021

Abstract
One can get an integer solution for the one-dimensional cutting stock problem by means of the constructive or residual heuristic. In this work we propose a change in the Constructive Greedy Heuristic that consists in building a cutting pattern by sorting in descending order the items of pair or odd length, priority being given to those which appear more frequently in the problem, cut from objects in stock with pair or odd length respectively, with the aim of minimizing the quantity of cut objects. Computing tests that were carried out showed the efficiency of the proposed heuristic when compared with other methods in the literature that generate an integer solution for the problem and will be presented at the end of this work.

Keywords Optimization · Linear programming · Heuristic · Cutting pattern · Bin packing · Column generation

1 Introduction

Cutting problems appear in several practical situations such as cutting steel sheets, wood, electric wires, paper rolls etc. There are many industries where such problems come up such as: car factories, aero-spatial, mechanics, naval, mobile factories, those of tubes and connections, petrol and gas etc; a lot of them being often applied to such areas as odontology, engineering, logistics, agronomy, fashion and designer, robotics, among others; for applications in these sectors we suggest to look up (Stadtler 1990;

✉ Gonçalo R. L. Cerqueira
goncalorenildo@uesb.edu.br

Sérgio S. Aguiar
sergioaguiar@uesb.edu.br

Marlos Marques
marlos@uesb.edu.br

[1] Department of Exact and Technologic Sciences, State University of Southwest Bahia, Vitória da Conquista 45083-900, BA, Brazil

Farley 1990; Cui 2005; Johnson et al. 1997). These problems may also be seen as bin-packing problems present in food, pharmacy and construction materials industries etc. Specifically in the one-dimensional case, what we have is an optimization combinatorial problem that consists in cutting pieces longer than length L (objects) in order to produce shorter pieces of length l_i (items) so that one may supply the d_i (demand) orders in which $i = 1, \ldots, m$ trying to optimize any objective function, which may be, for instance, the minimization of costs or the minimization of the quantity of cut objects. For this paper, we considered only one objective to be attained, but it is common in the literature to consider two or more objectives for the problem (see Wascher 1990). A solution for the problem consists in determining a set of patterns of cuttings and their frequency, that is, the quantity of times that each of them must be cut so that the demands of the items are respected. Be n a quantity of patterns in the solution, L the size of the object on stock, with utilization cost c_j and J the set of indexes of all variable patterns. The objective is to minimize the quantity of cut objects. Let's assume that $c_j = 1, \forall j \in J$. If α_{ij} is the decision variable that indicates the quantity of items type i in the cutting pattern j with frequency x_j, for $i = 1, \ldots, m$ and $j = 1, \ldots, n$, then the problem may be formulated as the following problem of integral linear programming:

$$\begin{aligned} &minimize: \sum_{j \in J} x_j \\ &subject\ to: \sum_{j \in J} \alpha_{ij} x_j = d_i, \\ &\sum_{i=1}^{m} \alpha_{ij} l_i \leq L, \\ &x_j; \alpha_{ij} \in Z_+; \quad j \in J; \quad i = 1, \ldots, m \end{aligned} \tag{1}$$

Be $A = [\mathbf{a_1}\ \mathbf{a_2}\ \cdots\ \mathbf{a_n}]$ the matrix whose columns represent the n-possible patterns and $\mathbf{d} = (d_1, \ldots, d_i, \ldots, d_m)^T$ the demand vector of the items. Each column-j vector of A, $\mathbf{a_j} = (\alpha_{1j}, \ldots, \alpha_{ij}, \ldots, \alpha_{mj})^T$, represents a variable cutting pattern, and must therefore satisfy the following restriction of the knapsack:

$$\begin{aligned} &l_1\alpha_{1j} + l_2\alpha_{2j} + \cdots + l_m\alpha_{mj} \leq L \\ &0 \leq \alpha_{ij} \leq d_i \text{ and integers}, \quad i = 1, \ldots, m.j \in J \end{aligned} \tag{2}$$

This problem is NP-hard (McDiarmid 1999), and which makes it difficult to solve are the integrality conditions attributed to the decision variables, added to the fact that the number of patterns to explicit increases a lot as the number of different items and their demands increases.

2 Relaxed solution for the cutting problem

The first efficient methods to solve the cutting stock problem (CSP) appeared the 1960's with the works by Gilmore and Gomory (1961, 1963),who presented the method simplex generation of columns, a resolution technique in which the integrality requirement of the decision variables is abandoned, thus considering this a relaxed model. At each

iteration of the simplex method (Bazaraa et al. 1990), a column in the model is generated (Barnhart et al. 1994). The algorithm starts with a basic initial solution for the one-dimensional cutting stock problem(1CSP), formed only by homogeneous patterns of the kind $\mathbf{a_j} = (0, \ldots, \left\lfloor \frac{L}{l_i} \right\rfloor, \ldots, 0)^T, i = 1, \ldots, m;\ \ j \in J$. The basic matrix, in this case, is diagonal, in which the only non-null element of each column indicates the cut quantity of the item $i, i = 1, \ldots, m$:

$$\mathbf{B} = \begin{bmatrix} \left\lfloor \frac{L}{l_1} \right\rfloor & 0 & \ldots & 0 \\ 0 & \left\lfloor \frac{L}{l_2} \right\rfloor & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \left\lfloor \frac{L}{l_m} \right\rfloor \end{bmatrix}$$

2.1 Column generation

By following the procedure suggested in Gilmore and Gomory (1961) the column $\mathbf{a_k}$ to enter the basis is corresponding to the variable x_k with the least relative cost, which means the resolution of the following subproblem:

$$(c_k - \pi^T \mathbf{a_k}) = min\ \left\{(c_j - \pi^T \mathbf{a_j}), j = 1, \ldots, n\right\}. \tag{3}$$

If $(c_k - \pi^T \mathbf{a_k}) \geq 0$ thus the current basic solution $\mathbf{x_B} = \mathbf{B}^{-1}\mathbf{d}$ is optimal, in which π is the simplex multiplying vector of the current iteration, $\mathbf{B}$ is the basic matrix $m \times m$, $\mathbf{x_B}$ is the vector of the basic variables, $\mathbf{d}$ is the vector of the demands. We need to determine a non-basic column $\mathbf{a_j} = (\alpha_{1j}, \ldots, \alpha_{ij}, \ldots, \alpha_{mj})^T$, corresponding to a feasible cutting pattern $j \in J$, which is a candidate to enter the basis. Thus, since $min\left\{c_j - \pi^T \mathbf{a_j}\right\} = c_j - max\left\{\pi^T \mathbf{a_j}\right\}$ the column comes from solving the restrict knapsack problem:

$$\begin{aligned} &maximize \sum_{i=1}^{m} \pi_i \alpha_{ij} \\ &subject\ to: \sum_{i=1}^{m} l_i \alpha_{ij} \leq L \\ &0 \leq \alpha_{ij} \leq d_i, \text{ and integers } \quad i = 1, \ldots, m.\ j \in J \end{aligned} \tag{4}$$

Techniques for solving the knapsack problem are found in Pisinger (1993) and Martello and Toth (1990). The method used in our work was that implicit enumeration suggested by Gilmore and Gomory (1963), based on depth search first, enabling through the use of bounding that the worst solutions be discarded without losing the optimal solution. For $i = 1, \ldots, m$, v_i is the utility value of item-i. The complete algorithm is shown below.

Table 1 Cutting problem

Item	Length	Demand
1	11	7
2	5	9
3	3	4

Table 2 Relaxed solution

Freq	Padrões		
2.00	1	0	2
5.00	1	1	0
1.33	0	3	0

To better understand the Knapsack Problem Algorithm we present a small example of the one-dimensional cutting stock problem, with three items. The size of the object in stock is $L = 17$. The data are shown in the Table 1. With each iteration of the Simplex Column Generation, the steps **P.2.** and **P.3.** determine the simplex multiplier vector $\pi = (\pi_1, \pi_2, \pi_3)$, π_i utility value of item-i, $i = 1, 2, 3$. The solution to the relaxed problem is shown in Table 2. We will now detail the first iteration, whose vector $\pi = (1, 0.3333, 0.25)$ is passed as utility vector of the knapsack, that is, $v_1 = 1$; $v_2 = 0.333$; $v_3 = 0.25$. With these data the problem 4 becomes:

$$\begin{aligned} &maximize\ g(\mathbf{y}) = 1.00\alpha_{1j} + 0.33\alpha_{2j} + 0.25\alpha_{3j} \\ &subject\ to: 11\alpha_{1j} + 5\alpha_{2j} + 3\alpha_{3j}.\ \ j \in J \\ &0 \leq \alpha_{1j} \leq 7; \qquad 0 \leq \alpha_{2j} \leq 9; \qquad 0 \leq \alpha_{3j} \leq 4 \end{aligned} \tag{5}$$

In **P.1.** Now defining $\pi_i = v_i/l_i$, the most valuable items are ordered by length unit. Suppose $\pi_1 \geq \pi_2 \geq \pi_3$, so π is now given by $\pi = (0.09; 0.08; 0.06)$. The initial solution according to **P.2.** is such that $y_1 = min\left\{\left\lfloor \frac{17}{11} \right\rfloor, 7\right\} = 1$, $y_2 = min\left\{\left\lfloor \frac{17-11}{3} \right\rfloor, 4\right\} = 2$, $y_3 = min\left\{\left\lfloor \frac{17-11-6}{5} \right\rfloor, 9\right\} = 0$, so $\mathbf{y} = (y_1, y_2, y_3) = (1, 2, 0)$. According **P.3.** $g(\mathbf{y}) = \sum_{i=1}^{3} v_i y_i = 1.5 = \underline{G}$ and $y^* = \mathbf{y}$. In **P.4.** $k = 2$ and $\overline{G} = v_1 y_1 + v_2(y_2 - 1) + \frac{v_3}{l_3}(L - l_1 y_1 - l_2(y_2 - 1) = 1.33$. Since $\overline{G} \leq \underline{G}$ in **P.5.** make $y_2 = 0$ turning $\mathbf{y} = (1, 0, 0)$ coming back to **P.4.** and repeating the process one has $\overline{G} = 0$ and again $\overline{G} \leq \underline{G}$, now k=1 then one makes $y_1 = 0$ which turns $\mathbf{y} = (0, 0, 0)$ and therefore the current solution saved in y^*, $\mathbf{y} = (1, 2, 0)$ which in the original problem corresponds to $\mathbf{y} = (1, 0, 2)$ it is optimal and it is exactly the first pattern in the solution of the relaxed problem in Table 2. Repeating the process, the other patterns of the solution are determined.

Knapsack Problem Algorithm

BEGIN

P.1. Sort the most valuable items by length unit

Do: $STOP = False$ and $IT = 0$.

{ The logic variable $STOP$ will be $False$ until the optimal solution is obtained and IT indicates the number of the current iteration }.

Define: $\pi_i = {}^{v_i}/_{l_i}$, $i = 1, \ldots, m$, re-sort the variables by the non-increasing order. Suppose that: $\pi_1 \geq \pi_2 \geq \cdots \geq \pi_m$

P.2. Determine the initial solution $\mathbf{y} = (y_1, \ldots, y_m)$, by searching in depth first such that

$$y_1 = min\left\{\left\lfloor \frac{L}{l_1} \right\rfloor, d_1\right\}$$

$$y_2 = min\left\{\left\lfloor \frac{L - y_1 l_1}{l_2} \right\rfloor, d_2\right\}$$

...

$$y_k = min\left\{\left\lfloor \frac{L - \sum_{i=1}^{k-1} l_i y_i}{l_k} \right\rfloor, d_k\right\}, k = 2, \ldots, m.$$

P.3. Evaluate the current solution and save the best one.

Determine : $g(\mathbf{y}) = \sum_{i=1}^{m} v_i y_i$.

If $\underline{G} < g(\mathbf{y})$ then do: $\underline{G} = g(\mathbf{y})$ (initially $\underline{G} = 0$)

G(X) is the optimal value of the knapsack. ($X \leq L$)

Save the corresponding solution $y^* = \mathbf{y}$.

P.4. Test the optimality and calculate the upper bound.

Determine k, the highest index such that $y_k \neq 0$.

If $\mathbf{y} = 0$ STOP, the solution saved in y^* is optimal.

Otherwise, calculate: (if $k = m$, do $v_{m+1} = 0$ and $l_{m+1} = 1$.)

$$\overline{G} = v_1 y_1 + v_2 y_2 + \cdots + v_k(y_k - 1) + \frac{v_{k+1}}{l_{k+1}}(L - l_1 y_1 - l_2 y_2 - \cdots - l_k(y_k - 1))$$

P.5. Backtracking

5.1. Long return. If $\overline{G}(\mathbf{y}) \leq \underline{G}$, then do: $y_k = 0$ and come back to **P.4.**

5.2. Return to the previous node and new search in depth.

If $\overline{G}(\mathbf{y}) > \underline{G}$ then do $y_k \leftarrow y_k - 1$ and define the new solution $\mathbf{y}$:

$$y_j = min\left\{\left\lfloor \frac{L - \sum_{i=1}^{j-1} l_i y_i}{l_j} \right\rfloor, d_j\right\}, j = k+1, \ldots, m \text{ and come back to } \textbf{P.3.}$$

END.

3 Heuristics of rounding solutions

Be $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ the optimal solution of the relaxed problem 1. Let's admit that at least one component of $\mathbf{x}$ not integer, otherwise the integer solution for the problem would already have been found. There are two possibilities in this case to determine an integer solution: one either round the components of vector $\mathbf{x}$ for the inferior integer or for the superior integer. In the first case, the found solution is not

Simplex Column Generation

BEGIN
P.1. Determine the initial basic matrix **B**.
Do: $STOP = False$ and $IT = 0$.
{The logic variable $STOP$ will be False until the optimal solution be obtained and IT indicates the number of the current iteration}.
While $(STOP = False)$ Do:
P.2. Determine the current variable basic solution: $\mathbf{x_B} = \mathbf{B}^{-1}\mathbf{d}$.
P.3. Determine the simplex multiplying vector: $\mathbf{B}^T\pi = \mathbf{c_B}$.
P.4. Solve the Knapsack Problem 4. Suppose that
$\mathbf{a} = (\alpha_{1j}, \alpha_{2j}, \ldots, \alpha_{mj})\ j \in J$, is the optimal solution
P.5. Optimality Test.
If $(1 - g(\mathbf{a}) \geq 0)$ **then** $STOP = Truth$
{The solution is optimal and it was obtained in IT iterations}.
Otherwise (determine the simplex direction: $\mathbf{y} = \mathbf{B}^{-1}\mathbf{a}$).
If $(\mathbf{y} < 0)$ **then** $(STOP = Truth)$. {The solution is unlimited}.
Otherwise (determine the size of the step):
Find l so that: $\frac{x_l}{y_l} = min\left\{\frac{x_i}{y_i} \mid y_i > 0,\ i = 1, \ldots, m\right\}$
P.6. Update of the basis.
Replace the l-nth column of matrix **B** by **a**.
do: $IT = IT + 1$.
End of the While
END.

always viable because the resulting rounding from the non-integer components of **x** does not necessarily supply the demand of all items, while in the second case more items are produced. In both cases, it is necessary to solve at least one residual problem. Be $x^* = (\lceil x_1 \rceil, \lceil x_2 \rceil, \ldots, \lceil x_n \rceil)$the rounding of the components of vector **x** for the superior integer with objective function corresponding to $f(x^*) = \sum_{j=1}^{n} \lceil x_j \rceil$ and $f(\mathbf{x}) = \sum_{j=1}^{n} x_j$ the objective function of the relaxed solution. An lower bound (LB) for the objective function of problem 1 is given by $LB = \lceil f(\mathbf{x}) \rceil$. We call $integrality\ gap$ the difference $\Delta(\mathbf{x}) = f(x^*) - f(\mathbf{x})$. According to Marcotte (1985) and Scheithauer and Terno (1995) if $\Delta(\mathbf{x}) < 1$ the (1CSP) 1 has IRUP (Integer Round-Up Property), that is, the integrality gap, the difference between the optimal value of the entire solution and the optimal value of the relaxed solution rounded to the nearest upper integer is less than or equal to 1 (one). If $f(\mathbf{x}) = LB$ the solution is optimal.

3.1 Residual heuristics

We call residual heuristics each procedure that starts a search for an integer solution for the (1CSP) by means of rounding techniques applied to the relaxed solution.

Since in such cases the original demand is not completely supplied one or more residual problems are solved with the remaining demand. One of the residual heuristics often used in the literature was proposed by Wascher and Gau (1996) and it consists in rounding the components of vector $\mathbf{x}$ downwards, $\overline{x} = (\lfloor x_1 \rfloor, \lfloor x_2 \rfloor, \ldots, \lfloor x_n \rfloor)$ the corresponding objective function being given by $f(\overline{x}) = \sum_{j=1}^{n} \lfloor x_j \rfloor$ and in solving the residual problem, which corresponds to solving problem 1 by replacing the demand vector d by the residual demand $\mathbf{r} = (r_1, \ldots, r_i, \ldots, r_m)^T$ com $r_i = d_i - \sum_{j=1}^{n} a_{ij} \lfloor x_j \rfloor$, $i = 1, \ldots, m$. In case of another fractional solution, rounding for the inferior integer is again carried out, generating a new residual demand. One repeats the process until the rounding generates null frequencies. If there still remain items with non-null demands, one solves a final residual problem with any heuristic that furnishes an integer solution for the problem.

Another residual approach proposed by Poldi and Arenales (2009), at each iteration solves a relaxed cutting stock problem, sorts the solution vector $\mathbf{x}$ in decreasing order, rounds the first frequency for the superior integer and tests the feasibility of the first pattern of the solution in order to verify that there were no excesses in the production of any item, otherwise, the frequency is reduced one unit until excesses be eliminated and one proceeds to the next frequency pattern. When the last generated cutting pattern is examined, one updates the demand and the stock. The residual problem is solved and the rounding procedure is repeated until all the demand is supplied. Since every time a relaxed problem is solved at least one cutting pattern is accepted to the solution, the *Greedy Rounding Heuristic*-GRH guarantees that every demand be supplied in a finite number of iterations. Two other versions of the heuristic, GRH2 and GRH3 sort the solution vector $\mathbf{x}$ with priority for the patterns with the least waste and for patterns in which the solution vector $\mathbf{x}$ presents the largest fractional part respectively.

Greedy Rounding Heuristic

P.1. Do $\mathbf{d}^0 = \mathbf{d}$, k= 0

P.2. While $\mathbf{d}^k \neq 0$ do: solve the relaxed problem 1

Obtain $\mathbf{x}^k = (x_1^k, \ldots, x_m^k)$ and the basic matrix $\mathbf{B}^k = [\mathbf{a}_1^k, \ldots, \mathbf{a}_m^k]$

$\mathbf{a}_i^k$, $i = 1, \ldots, m$, are the patterns for the solution of problem-k

2.1: Order the patterns so that: $x_1^k \geq \cdots \geq x_m^k$

Determine a feasible solution:

2.2: If $(\lceil x_1^k \rceil, 0, \ldots, 0)$ is feasible do $y_1^k = \lceil x_1^k \rceil$

Otherwise $y_1^k = \lceil x_1^k \rceil - 1$

For $i = 2, \ldots, m$ do $y_i^k = \lceil x_i^k \rceil$; $\mathbf{y^k} = (y_1^k, \ldots, y_{i-1}^k, y_i^k, 0, \ldots 0)$

2.3: While $\mathbf{y}^k$ is no a feasible solution do $y_i^k = y_i^k - 1$

$\mathbf{y}^k$ is no feasible solution if $(\mathbf{B}^k \mathbf{y}^k > \mathbf{d}^k)$

2.4: Keep $\mathbf{B}^k$ e $\mathbf{y}^k$

2.5: do: $\mathbf{d}^{k+1} = \mathbf{d}^k - \mathbf{B}^k \mathbf{y}^k$ and $k = k + 1$.

3.2 Constructive heuristics

Another way to determine an integer solution for a one-dimensional cutting problem consists in constructing a good cutting pattern and use it as many times as possible (see Hinxman 1980), without excess in the production of any item. These are the so-called constructive heuristics. At each iteration of such procedure, the demand of the items is updated and the process is repeated until it is fully supplied, generating at last an integer solution for the problem.

3.2.1 FFD heuristic

FFD Heuristic consists in putting the largest item in the pattern the highest possible number of times without excess in the demand. If the selected item does not fit the pattern anymore, the second largest item is selected and so forth. The complete algorithm is presented below:

FFD Heuristic

BEGIN

P.1. Sort the items in decreasing order of size.

Suppose $l_1 \geq l_2 \geq \cdots \geq l_m$

P.2. Be r_i the residual demand of the item $i \in I$.

$I = \{1, \ldots, m\}$. {set of indexes of the items}

At first: $r_i = d_i, \forall i \in I$.

Do $k = 1$ {First cutting pattern }

STOP=False {logic variable tht indicates a non-null demand}

While STOP=False

P.3. Do: $Rest = L$ and $\alpha_{ik} = 0, \forall i \in I$;

Be $i = 1$ {start by putting the first item in the pattern }

While ($i \leq m$ and $Rest \geq l_i$) Do:

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

(α_{ik} is the quantity of items type i in pattern k)

Do: $Rest = Rest - (\alpha_{ik} l_i)$

$r_i = r_i - \alpha_{ik}$; $i = i + 1$

End of While

P.4. Determine the frequency of pattern k: $x_k = \left\{ \begin{array}{l} \min \left\lfloor \frac{d_i}{\alpha_{ik}} \right\rfloor, \\ \forall i \in I; \alpha_{ik} > 0 \end{array} \right\}$

P.5. (Stop Criterion)

If $r_i = 0, \ \forall i \in I$ **then** STOP = Truth.

Otherwise Do $k = k + 1$ and come back to **P.3**

End of While.

END

3.2.2 Greedy Heuristic

Greedy Heuristic follows the same idea as FFD Heuristic, except for step **P.3.** Instead of giving priority to the largest items to build the pattern, this is obtained by solving the restrict knapsack problem 4, making $\pi_i = l_i$, in the objective function for $i = 1, \ldots, m$.

3.2.3 Modified Greedy Heuristic

Until the submission of this article, we did not find in the literature any works that propose any modification in the methodology of the pattern of Greedy Heuristic for the CSP. Cherri et al. (2009) modify the FFD and Greedy Heuristic without modifying the methodology of putting in the pattern the largest item in the list, the difference is that at each iteration this list is shorter since the pattern is only accepted if it is within the bounding acceptable loss, otherwise an item in the pattern (the largest one) is withdrawn, the pattern is generated with FFD or Greedy Heuristic and the loss is assessed, if the pattern is not accepted one withdraws the largest item and so forth. This

Table 3 Example: cutting problem

Item	Length	Demand
1	765	58
2	633	88
3	595	64
4	571	66
5	545	122
6	337	159
7	332	138
8	87	125
9	77	112
10	73	68

Table 4 Relaxed problem solution

Freq	Pattern									
58.00	1	0	0	0	0	0	0	0	0	0
29.92	0	1	0	0	0	0	0	0	0	0
30.67	0	0	0	0	1	1	0	1	0	0
66.00	0	0	0	1	0	1	0	1	0	0
62.33	0	0	0	0	1	1	0	0	1	0
0.667	0	0	0	0	1	0	0	0	0	6
28.33	0	0	0	0	1	0	1	1	0	0
64.00	0	0	1	0	0	0	1	0	0	1
12.42	0	1	0	0	0	0	0	0	4	0
45.67	0	1	0	0	0	0	1	0	0	0

Table 5 GRH solution

Freq	Pattern									
66	0	0	0	1	0	1	0	1	0	0
64	0	0	1	0	0	0	1	0	0	1
63	0	0	0	0	1	1	0	0	1	0
58	1	0	0	0	0	0	0	0	0	0
46	0	1	0	0	0	0	1	0	0	0
30	0	0	0	0	1	1	0	1	0	0
30	0	1	0	0	0	0	0	0	0	0
28	0	0	0	0	1	0	1	1	0	0
12	0	1	0	0	0	0	0	0	4	0
1	0	0	0	0	1	0	0	1	1	3
1	0	0	0	0	0	0	0	0	0	1

Table 6 Greedy solution

Freq	Pattern									
64	0	0	1	0	0	0	1	0	0	1
2	0	0	0	0	0	0	1	6	0	2
22	0	0	0	0	0	0	1	5	3	0
1	0	0	0	0	0	1	1	2	2	0
14	1	0	0	0	0	0	0	0	3	0
16	0	0	0	0	0	0	3	0	0	0
1	0	0	0	1	0	1	0	1	0	0
2	0	0	0	1	0	1	0	0	1	0
88	0	1	0	0	0	1	0	0	0	0
63	0	0	0	1	0	1	0	0	0	0
4	0	0	0	0	1	1	0	0	0	0
1	0	0	0	0	1	0	1	0	0	0
44	1	0	0	0	0	0	0	0	0	0
117	0	0	0	0	1	0	0	0	0	0

procedure is repeated until the loss or rest is within the bounding defined as acceptable or the demand is totally supplied. Ongkunaruk (2005) presents a modification in the FFD Heuristic in order to solve the Bin Packing Problem—BPP, but the idea is the same contained in Cherri et al. (2009). The modification proposal that present in this work differs from the Greedy Heuristic in the way the items are arranged in the decreasing order. If there are more pair than odd numbers these are ordered first and the same happens in the opposite case.

Following is an example of the cutting problem with 10 types of items ($m = 10$). The length and the demand of each item are shown on Table 3. The objects in stock have lengths 1000 and 1001. Variation in the lengths of the objects in stock also appear in the works of Poldi and Arenales (2009) and Belov and Scheithauer (2002). On Tables 4, 5, 6 and 7 one sees the patterns and frequencies of the solutions obtained by generating

Modified Greedy Heuristic

BEGIN

P.1. Identify the sets of s-itens l_i so that $l_i = 2w_i$
and $t = (m - s)$-items so that $l_i = 2w_i + 1$; $w_i \in N$,
$i = 1, \ldots, m$. Sort them in decreasing order.
Suppose $l_1 > l_2 > \cdots > l_s$ and $l_{s+1} > l_{s+2} > \cdots > l_m$

1.1. If $(s \geq t)$ start by putting the items li in the pattern l_i
so that $l_i = 2w_i$, $i = 1, \ldots, s$; with $L = 2 \cdot W; W \in N$

1.2. Be r_i residual demand of item-i.
At first $r_i = d_i$. $i = 1, \ldots, m$
Do $k = 1$ {first cutting pattern }
STOP=False {logic variable of the non-null demand}
While STOP=False

P.2. Do: Rest=L; $\alpha_{ik} = 0$
Be i=1 {Put the first item in the pattern, $i = 1, \ldots, m$}
While$(i \leq s \; or \; (s < i \leq m) \; and \; Rest \geq l_i)$ do
$\alpha_{ik} = \min \left\{ \left\lfloor \frac{Rest}{l_i} \right\rfloor, r_i \right\}$
Do: Rest= Rest-α_{ik}
$r_i = r_i - \alpha_{ik}; i = i + 1$
End of While

P.3. Determine the frequency of pattern k
$x_k = \left\{ \min \left\lfloor \frac{d_i}{\alpha_{ik}} \right\rfloor, \forall i \in I; \; \alpha_{ik} > 0 \right\}$ Keep x_k and and pattern-k

P.4. (Stop Criterium)
If $r_i = 0 \; \forall i \in I$ **Then** STOP=Truth.
Otherwise Do $k = k + 1$ and come back to **1.2**
End of While.

Otherwise (start by putting the items l_i so that $l_i = 2w_i + 1$)
$i = (s + 1), (s + 2), \ldots, m$ with $L = 2 \cdot W + 1; W \in N$
and come back to step **P.2**

END

columns and by the GRH, Greedy and Modified Greedy Heuristic respectively. The tables show that while the modified heuristic obtained a solution with the same 398 objects of the relaxed solution and 12 patterns, 02 more than that, the greedy heuristic produced 41 objects and 4 extra patterns. The GRH Heuristic obtained a solution very close to the relaxed solution with 01 object and 01 extra pattern. The only one to have obtained an optimal solution in this case was the Modified Greedy Heuristic (MGH) because according to the exposition on Sect. 3 $f(\mathbf{x}) = 398 = LB$ and the $integrality \; gap \; \Delta(\mathbf{x}) = 0$.

Table 7 Modified greedy solution

Freq	Pattern									
37	1	0	0	0	0	0	0	0	3	0
21	1	0	0	0	0	0	0	2	0	0
20	0	1	0	0	0	0	0	4	0	0
1	0	1	0	0	0	0	0	3	1	0
13	0	1	0	0	0	0	0	0	0	5
1	0	1	0	0	0	0	0	0	0	3
53	0	1	0	0	0	1	0	0	0	0
64	0	0	1	0	0	1	0	0	0	0
42	0	0	0	1	0	1	0	0	0	0
24	0	0	0	1	0	0	1	0	0	0
114	0	0	0	0	1	0	1	0	0	0
8	0	0	0	0	1	0	0	0	0	0

Table 8 Test problems used

Class	m	v_1	v_2	$\overline{\mathbf{d}}$
1	10	0.01	0.2	10
2	10	0.01	0.2	100
3	20	0.01	0.2	10
4	20	0.01	0.2	100
5	40	0.01	0.2	10
6	40	0.01	0.2	100
7	10	0.01	0.8	10
8	10	0.01	0.8	100
9	20	0.01	0.8	10
10	20	0.01	0.8	100
11	40	0.01	0.8	10
12	40	0.01	0.8	100
13	10	0.2	0.8	10
14	10	0.2	0.8	100
15	20	0.2	0.8	10
16	20	0.2	0.8	100
17	40	0.2	0.8	10
18	40	0.2	0.8	100

4 Computational experiments

For the computing experiments, the test problems were obtained by Cutgen1, a random generator of cutting problems of one-dimensional stock developed by Gau and Wascher (1995). We considered 18 classes of problems each one with 100 instances and we adopted in Cutgen1 the seed 1994. The classes are divided according to the quantity of items m, the average of the items demands $\overline{\mathbf{d}}$, and different combinations

Table 9 Cut objects average

Classe	Constructive			Residual				
	FFD	Greedy	MGH	FFD	Greedy	BPP	GRH	MGH
1	11.61	**11.05**	11.67	11.54	11.48	11.48	11.50	11.57
2	111.68	110.72	112.45	110.28	**110.23**	110.25	110.31	110.28
3	**22.07**	22.20	22.26	22.26	22.26	22.26	22.15	22.26
4	217.70	216.16	217.89	225.97	215.91	215.93	**215.91**	215.94
5	43.13	**42.95**	43.09	42.99	**42.95**	43.00	42.96	**42.95**
6	426.53	424.71	426.39	424.74	424.72	424.68	424.86	**424.66**
7	50.35	53.30	52.46	50.26	50.26	50.27	50.53	**50.25**
8	501.15	532.53	523.62	499.62	499.62	499.63	500.65	**499.61**
9	94.06	100.49	97.68	93.73	93.70	93.70	94.38	**93.67**
10	936.30	1003.03	974.76	932.32	932.35	932.35	936.87	**932.29**
11	177.56	191.75	183.66	176.96	177.41	177.41	177.89	**177.00**
12	1770.26	1916.53	1836.03	**1763.53**	1766.40	1766.40	1771.15	1770.79
13	63.59	65.93	65.39	**63.46**	**63.46**	63.47	63.50	**63.46**
14	633.70	658.53	653.23	632.37	632.36	632.36	632.42	**632.35**
15	119.93	125.55	122.13	119.62	119.63	119.63	119.68	**119.61**
16	1195.29	1254.7	1218.76	1192.02	**1192.01**	1192.02	1192.08	**1192.01**
17	225.30	244.24	230.37	224.89	224.90	224.89	224.95	**224.86**
18	2247.42	2435.96	2298.88	**2242.58**	2242.63	2242.64	2242.70	2242.67

of values v_1 and v_2 for determining the length of the items that are generated in the interval $[v_1L, v_2L]$, as shown on Table 8. In the carried out computing experiments, the considered size of the stock object was $L = 1000$ for all implemented heuristics. For Modified Greedy Heuristic we used two sizes of objects, because for every problem one decision is taken to cut the items out of the object of length $L = 1000$ or $L = 1001$ inasmuch we have even or odd length respectively, a variation of 0.1% longer than the object used by the other heuristics. For FFD, Greedy, GRH, BPP and Modified Greedy Heuristic the Simplex Method of Column Generation and the knapsack algorithm were implemented in C_{++} and all the computing tests for this work were carried out in an Intel Core i5-2450M computer with 4GB memory (Ram).

5 Results analysis

On Table 9 one presents the results of the computing tests carried out with the eight tested heuristics: three constructive ones: FFD, Greedy, and Modified Greedy (MGH); and five residual ones: FFD, Greedy, Bin Packing (BPP), GRH and Modified Greedy. Each column of Table 9 provides the average of the number of cut objects from the 100 problems of each of the 18 classes that were used. The values in bold in the rows of Table 9 indicate the heuristics that presented the lowest average of cut objects. In this case we point out the optimal performance of the residual Modified Greedy

Table 10 Patterns average

Class	Constructive			**Residual**				
	FFD	Greedy	MGH	FFD	Greedy	BPP	GRH	MGH
1	9.50	9.07	9.76	9.06	8.80	8.80	**7.77**	9.07
2	16.8	15.67	16.90	13.92	13.4	13.42	**12.80**	13.94
3	18.32	17.77	16.43	16.48	**16.31**	22.26	17.23	16.43
4	32.36	30.54	32.02	26.08	25.80	25.82	**24.79**	26.02
5	34.50	33.78	34.37	30.88	30.79	30.79	**28.55**	30.93
6	62.05	59.11	60.62	49.63	49.30	49.33	**46.98**	49.56
7	10.89	11.08	11.12	9.78	**9.75**	**9.75**	10.10	9.87
8	12.12	12.34	12.46	**10.41**	10.43	10.45	11.22	10.58
9	20.94	21.21	21.26	19.38	19.50	19.50	**19.13**	19.67
10	23.50	23.51	23.82	**20.82**	20.90	20.90	21.77	20.97
11	39.84	40.36	40.55	38.09	38.58	38.58	**36.58**	38.30
12	44.65	44.36	44.23	42.00	**41.94**	**41.94**	43.57	45.81
13	10.55	10.55	10.75	10.21	10.22	10.23	**10.19**	10.21
14	11.05	11.10	11.27	10.79	**10.77**	**10.77**	10.78	10.81
15	20.64	20.65	20.79	20.10	20.11	20.11	**19.82**	20.09
16	21.72	21.66	21.83	21.07	21.03	21.04	**20.88**	21.12
17	39.36	39.38	39.73	38.50	38.63	38.60	**37.87**	38.54
18	41.58	41.38	41.95	41.04	41.09	41.09	**40.41**	41.17

Heuristic (MGH), which got a better performance than the others in 12 out of the 18 tested classes. The constructive heuristics obtained better performances in only 02 out of the 18 classes, thus confirming the tendency that the best solutions in this case be gotten by the residual heuristics, because the solution of the relaxed problem besides a technique of rounding solutions are demonstrably more efficient than building one pattern after another and use it exhaustively, as do the constructive heuristics.

The Table 10 provides the average of the number of patterns for each of the 18 tested classes. The values in bold in the rows indicate the heuristics with the lowest average of patterns used. We did not use any efforts in order to reduce the number of patterns as the methods proposed by Foerster and Wascher (1999) and Yanasse et al. (2011) which were used for example in the works by Yanasse and Limeira (2006), Cerqueira and Yanasse (2009) and Cui et al. (2008) among others. Even though that was not an objective of our work, by the data in the table we notice that the averages of the patterns used by the residual heuristics kept themselves very close to one another, and much better than the average of the constructive heuristics, something expected as we explained earlier. The residual GRH got the best results for the average of the objects, surpassing the others in 12 of the 18 classes. There were no significant variations in the average time of the heuristics as shown in Table 11.

The Table 12 shows the average rate of variation of objects and patterns between the constructive Modified Greedy Heuristic and Greedy Heuristic. The positive and negative values indicate respectively the upward and downward average of objects and

Table 11 Average time

Class	Constructive			Residual				
	FFD	Greedy	MGH	FFD	Greedy	BPP	GRH	MGH
1	0.0301	0.0010	0.0115	0.0016	0.0035	0.0036	0.0228	0.0024
2	0.0302	0.0010	0.0114	0.0015	0.0036	0.0036	0.0254	0.0023
3	0.0291	0.0019	0.0046	0.0037	0.0057	0.0059	0.0397	0.0046
4	0.0309	0.0012	0.0117	0.0028	0.0049	0.0051	0.3236	0.0037
5	0.0302	0.0014	0.0124	0.0117	0.0142	0.0142	0.0559	0.0125
6	0.0354	0.0018	0.0129	0.0125	0.0150	0.0151	0.1168	0.0133
7	0.0302	0.0010	0.0119	0.0008	0.0029	0.0027	0.0132	0.0015
8	0.0306	0.0010	0.0123	0.0007	0.0027	0.0027	0.0135	0.0014
9	0.0293	0.0013	0.0119	0.0017	0.0039	0.0039	0.0165	0.0028
10	0.0400	0.0016	0.0125	0.0017	0.0039	0.0037	0.0168	0.0026
11	0.0300	0.0041	0.0119	0.0118	0.0141	0.0143	0.0271	0.0127
12	0.1114	0.0153	0.0126	0.0107	0.0132	0.0134	0.0341	0.0682
13	0.2965	0.0010	0.0115	0.0007	0.0027	0.0025	0.0108	0.0014
14	0.0326	0.0011	0.0114	0.0006	0.0027	0.0025	0.0115	0.0013
15	0.0299	0.0012	0.0120	0.0015	0.0036	0.0032	0.0132	0.0023
16	0.0495	0.0015	0.0121	0.0014	0.0034	0.0032	0.0135	0.0023
17	0.0314	0.0036	0.0123	0.0112	0.0137	0.0130	0.0208	0.0123
18	0.1672	0.0075	0.0126	0.0107	0.0129	0.0126	0.0214	0.0118

Table 12 Greedy X modified Greedy Heuristic

Class	1	2	3	4	5	6	7	8	9
Object	0.62	1.73	0.06	1.73	0.14	1.68	− 0.84	− 8.91	− 2.82
Pattern	0.69	1.23	− 1.34	1.48	0.59	1.51	0.04	0.12	0.05
Class	10	11	12	13	14	15	16	17	18
Object	− 28.27	− 8.09	− 80.50	− 0.54	− 5.30	− 3.42	− 35.94	− 13.87	− 137.08
Pattern	0.31	0.19	− 0.13	0.20	0.17	0.14	0.17	0.35	0.57

patterns used by MGH as compared with Greedy Heuristic. One notices by the data in the table that in 12 of the 18 tested classes, MGH reduces enough the average of objects cut by Greedy Heuristic, virtually keeping the patterns average stable, reducing in 80.5 the average of the objects and 0.13 the average of the patterns in class 12. We calculated the average pattern deviation of objects in the solution of the Greedy Heuristic (G-XXX) and Modified Greedy Heuristic (M-XXX) in relation to other tested heuristics, which consists in calculating the difference of the average of objects among the generated solutions.

The results on Table 13 show how much each heuristic was better than the others, producing downward objects, if the values are negative, or upward if they are positive. As pointed out earlier we have not proposed as an objective to reduce patterns, even

Table 13 Modified Greedy performance

Class	Constructive		Residual							
	G-FFD	M-FFD	G-FFD	M-FFD	G-Greedy	M-Greedy	G-BPP	M-BBP	G-GRH	M-GRH
1	− 0.56	0.07	− 0.49	0.13	− 0.43	0.19	− 0.43	0.19	− 0.45	0.17
2	− 0.96	0.77	0.44	2.17	0.49	2.22	0.47	2.20	0.41	2.14
3	0.13	0.19	− 0.06	0.00	− 0.06	0.00	− 0.06	0.00	0.05	0.11
4	− 1.54	0.19	− 9.81	− 8.08	0.2	1.98	0.23	1.96	0.11	1.84
5	− 0.18	− 0.04	− 0.04	0.10	0.00	1.73	− 0.05	0.09	− 0.01	0.13
6	− 1.82	− 0.14	− 0.03	1.65	− 0.01	1.67	0.03	1.71	− 0.15	1.53
7	2.95	2.11	3.04	2.20	3.04	2.20	3.03	2.19	2.77	1.93
8	31.38	22.47	32.91	24.00	32.91	24.00	32.90	23.99	31.88	22.97
9	6.43	3.62	6.76	3.95	6.79	3.98	6.79	3.98	6.11	3.30
10	66.73	38.46	70.98	42.44	70.95	42.41	70.95	42.41	66.43	37.89
11	14.19	6.10	14.79	6.70	14.34	6.25	14.34	6.25	13.86	5.77
12	146.27	65.77	153	72.50	150.13	69.63	150.13	69.63	145.38	64.88
13	2.34	1.80	2.47	1.93	2.47	1.93	2.46	1.92	2.43	1.89
14	24.83	19.53	26.16	20.86	26.17	20.87	26.17	20.87	26.11	20.81
15	5.62	2.50	5.93	2.51	5.92	2.50	5.92	2.50	5.87	2.45
16	59.41	23.47	62.68	26.74	62.69	26.75	62.68	26.74	62.62	26.68
17	18.94	5.07	19.35	5.51	19.34	5.47	19.35	5.48	19.29	5.42
18	188.54	51.46	193.38	56.30	193.33	56.25	193.32	56.24	193.26	56.18

though we consider excellent the performance of MGH in relation to the average of patterns in the solution. In class 18, for instance, with a small rise of 0.57 in the average of patterns one cuts 137.08 objects fewer than Greedy Heuristic, and the few classes in which MGH does not reduce the average of objects and patterns, the upward variation rate is very low, in class 5 for instance, 0.14 and 0.59 respectively (Table 11).

6 Conclusion and future works

In this work, we presented the Modified Greedy Heuristic, a change of Greedy Heuristic, which instead of ordering in decreasing order of the size of all the items, it orders first the pair items or the odd ones in case there are more items in the problem, of pair and odd size respectively. The results of the computing tests carried out showed that the solutions generated by the MGH have the average of cut objects lower than the Greedy Heuristic. They were also assessed in other constructive and residual heuristics in the literature for the problem of cutting in the one-dimensional stock, the residual MGH being the one that got the best performance as for the proposed objective of minimizing the number of objects in the solution, as shown on Table 9. The average of patterns was also compared and GRH obtained the best performance, but the residual MGH got results very close to the other tested residual heuristics, since these obtain much better solutions than the constructive heuristics, as we explained earlier. For

future works, we intend to carry out other computing simulations with MGH, assigning different pairs of values with the size of the pair and odd object and consider the problem with different sizes of the stocked object.

Acknowledgements The authors thank Fapesb, Capes and CNPq for the financial support. The authors are also indebted to the reviewers for their valuable and helpful suggestions that improved the presentation of this paper.

Compliance with ethical standards

Conflicts of interest The authors declare that the research presented in this manuscript is in compliance with the rules of the institution's Research Ethics Committee, and since it does not involve human beings, informed consent does not apply. The authors also declare that they have no personal, commercial, academic, political or financial conflicts of interest for the submission of this manuscript.

References

Barnhart C, Johnson EL, Nenhauser GL, Vance PH (1994) Solving binary cutting stock problems by colum generation and branch and bound. Comput Optim Appl 3:111–130

Bazaraa MS, Jarvis JJ, Sherali HD (1990) Linear programming and network flows, 2nd edn. Wiley, New York

Belov G, Scheithauer G (2002) Decomposition approaches for solving the integer one-dimensional cutting stock problem with different types of standard lengths. Eur J Oper Res 141:295–312

Cerqueira GRL, Yanasse HH (2009) A pattern reduction procedure in a one-dimensional cutting stock problem by grouping items according to their demands. J Comput Interdiscip Sci 2:159–164

Cherri AC, Arenales MN, Yanasse HH (2009) The one-dimensional cutting stock problem with usable leftover: a heuristic approach. Eur J Oper Res 196:897–908

Cui Y, Yang Y, Yu P (2008) A heuristic for the one-dimensional cutting stock problem with pattern reduction. Eng Manuf 222:677–685

Cui Y (2005) A cutting stock problem and its solution in the manufacturing industry of large electric generators. Comput Oper Res 32:1709–1721

Farley AA (1990) The cutting stock problem in the canvas industry. Eur J Oper Res 44:247–255

Foerster H, Wascher G (1999) Pattern reduction in one-dimensional cutting stock problems. Int J Prod Res 38:1657–676

Gau T, Wascher G (1995) Cutgen1: a problem generator for one-dimensional cutting stock problem. Eur J Oper Res 84:572–579

Gilmore P, Gomory R (1961) A linear programming approach to the cutting stock problem. Oper Res 9:849–859

Gilmore P, Gomory R (1963) A linear programming approach to the cutting stock problem-part II. Oper Res 6:863–888

Hinxman AI (1980) The trim-loss and assortment problems: a survey. Eur J Oper Res 5:8–18

Johnson MP, Rennick C, Zak E (1997) Skiving addition to the cutting stock problem in the paper industry. Soc Ind Appl Math 39:472–483

Marcotte O (1985) The cutting stock problem and integer rounding. Math Program 33:82–92

Martello S, Toth P (1990) Knapsack problems: algorithms and computer implementations. Wiley, New York

McDiarmid C (1999) Pattern minimisation in cutting stock problems. Discrete Appl Math 98:121–130

Ongkunaruk P (2005) Asymptotic worst-case analyses for the open bin packing problem. Faculty of the Virginia Polytechnic Institute

Pisinger D (1993) Algorithms for knapsack problems. Thesis-DIKU, University of Copenhagen, Copenhagen

Poldi KC, Arenales MN (2009) Heuristics for the one-dimensional cutting stock problem with limited multiple stock lengths. Comput Oper Res 36:2074–2081

Scheithauer G, Terno J (1995) The modified integer round-up property of the one-dimensional cutting stock problem. Eur J Oper Res 84:562–571
Stadtler H (1990) A one-dimensional cutting stock problem in the aluminium industry and its solution. Eur J Oper Res 44:209–223
Wascher G (1990) An lp-based approach to cutting stock problems with multiple objectives. Eur J Oper Res 44:175–184
Wascher G, Gau T (1996) Heuristics for the integer one-dimensional cutting stock problem: a computational study. OR Spektrum 18:131–144
Yanasse HH, Poldi KC, Cerqueira GRL (2011) Modified KOMBI to reduce the different patterns in cutting stock problems. International Federation of Operational Research Societies Melbourne, Australia
Yanasse HH, Limeira MS (2006) A hybrid heuristic to reduce the number of different patterns in cutting stock problems. Comput Oper Res 33:2744–2756