Report CORO Project

Resolution of the Cutting Stock problem with Column Generation

Marine Astruc, Léa Darbot & Ramón Daniel Regueiro Espiño

2021/2022

École Nationale Supérieure d'Informatique pour l'Industrie et l'Entreprise

Contents

1 Introduction

The aim of our project is to solve a cutting stock problem using the column generation method. For this, we start introducing the problem: its main idea, its origin and the concrete case conceived. After, we'll introduce the method that we are going to use, a linear programming called the column generation method.

1.1 The Cutting Stock Problem

The one-dimensional cutting stock problem (CSP) is a classical problem in operational research. The aim of its resolution is to minimize the cost of filling an order, from certain stock lengths of given cost, for specified number of lengths of a certain material.

For example, we can suppose that a company produces steel bars of $L = 100m$. And, afterwards, it will cut each bar according to the customer's request.

l_i (meters)	22	42	52	53	78
number of pieces	45	38	25	11	12

Table 1: Steel bars order done by a customer.

A concrete case of a client's demand could be the one collected in the Table 1. We can notice that the length of each piece of the demand cannot be superior to the length of each steel bar produced by the company.

The company should try to minimize the number of bars used to fill the demand in order to minimize its costs. For example, if one bar is used to supply one piece of $78m$ the same bar could be used to supply a piece of $22m$. This will reduce the total number of bars used by the company. Otherwise, this minimize the total waste of one bar, as we went from throwing away $22m$ of steel to nothing.

We consider a list of m orders of b_j pieces, $j \in \{1, \cdots, m\}$. We are going to denote P the total number of possible patterns and x_i the number of times that pattern i is used. So, one way of modelling the problem, introduced in [Gilmore and Gomory, 1961], is

$$\begin{aligned} &\min \sum_{i=1}^{P} c_i x_i \\ \text{s. t.} \quad &\sum_{i=1}^{P} a_{j,i} x_i \geq b_i \quad \forall j = 1, \cdots, m, \\ &x_i \geq 0 \text{ and integer} \quad \forall i = 1, \cdots, P\ , \end{aligned} \tag{1}$$

where P is the number of patterns, c_i is the cost of the pattern i and $a_{j,i}$ is the time that order i appears in pattern j.

We want to highlight that this is not the unique way to model the problem, for example, we can consider as objective function the waste of the bars. In addition, the way described in equation (1) is not the most intuitive or the first one in appear. The first identification of the problem was done in 1939 by Kantorovich and its formulation can be seen in [Kantorovich, 1960].

On the other hand, as we can deduce from the equation (1), the problem is described in a linear way and all the variables are integers. Hence, the one-dimensional CSP is a case of integer linear programming.

We may ask ourselves how this problem can be solved. For this case, we are going to use the algorithm called Column Generation. However, we would like to highlight that, as it is collected in [Garey and Johnson, 1979], the CSP is a case of a NP-hard problem. The fact of being NP-hard implies that it cannot be considered, as far as we known, as a problem that can be solved in a polynomial time. So, the algorithm used would require a huge computer capacity if the CSP wanted to be solve is big.

1.2 Column Generation

Column generation is an algorithm used to solve linear programming problems by creating two different problems, the master problem and the restricted master problem.

- The master problem is the original column-wise formulation of the problem with only a subset of variables being considered.
- The restricted master problem is a new problem created to identify a new promising variable. The objective function of the sub-problem is the reduced cost of the new variable with respect to the current dual variables, and the constraints require that the variable obeys the naturally occurring constraints.

When the master problem is solved, we are able to obtain dual prices for each of the constraints in the master problem. This information is then utilized in the objective function of the restricted master problem. Then, the restricted master problem is solved. If the objective value of the restricted master problem is negative, a variable with negative reduced cost has been identified. The variable with negative reduced cost will help to reduce the cost. So, it is added to the master problem, and the master problem is re-solved.

Re-solving the master problem will generate a new set of dual values, and the process is repeated until no negative reduced cost variables are identified. Finally, the restricted

master problem returns a solution with non-negative reduced cost, this implies that there isn't any column that will improve the obtained solution. So, we can conclude that the solution to the master problem is the optimal solution.

Algorithm Column generation flowchart

Data: A, L, l, n

Initial Problem
Restricted Master Problem (RMP)
Solve Dual of RMP
Pass Dual multipliers to subproblem and solve
while reduces cost < 0 **do**
  Add column to RMP
  Restricted Master Problem (RMP)
  Solve Dual of RMP
  Pass Dual multipliers to subproblem and solve
end while

Result: Solution of the most recent Primal

Where A is the initial matrix, L is the length of the bar, l is the list of the different sizes of the bar asked by the customers and n the list of the number wanted of each length.

2 Description of our algorithm

In this section we'll explain the different parts of our built algorithm. We decided to split it in three different parts with different tasks: initializing the problem (init_master), doing the different iterations (resolution, auxiliary_problem) and printing the conclusion (conclusion).

• **Initialisation algorithm**:

In the first place, we consider the initialisation of the problem. In this case, this part includes the construction of the RMP1 problem, solving it and obtaining the first solutions for the dual problem.

Algorithm Initialisation of the problem

Data: A, L, l, n

$res \leftarrow$ optimisation of the RMP1 problem

$dual\_coeff \leftarrow$ optimisation of the RMP1 dual problem

Result: $res, dual\_coeff$

• **The iterations**:

Once we have our first problem we can consider a while loop that keeps solving the next auxiliary problem and the next restricted master problem until the reduced cost is non-strictement negative.

Algorithm Resolution

Data: A, L, l, n

while reduced cost < 0 **do**

Add column to RMP with auxiliary_problem function

Calculate reduced cost

end while

Result: res, A

In addition, inside this part we create a function that solves the auxiliary problem.

• **Conclusion**:

Algorithm Auxiliary_problem

Data: A, L, l, n

Solve the auxiliary problem

Result: x

Algorithm Conclusion

Data: A, L, l, n

Print the final result

Result: res, A

To conclude, once that the iterations part has ended, we print the final result obtained using this algorithm.

3 The resolution

For the resolution of both proposed exercises we utilise the modeling language [Julia JuMP]. We created a program to solve the general case, showing the results in a *.txt* file. The code of the program and its execution for both cases can be seen in the Appendix A.

We are going to round every number to two decimals in order to simplify its expression. Although, our algorithm considers and shows it in the results file with a bigger number.

3.1 Exercise 1

The first proposed exercise suppose that the length of the bars is 10 m. In this case, the following demand was made

l_i (meters)	2	3	4
number of pieces	6	4	2

Table 2: Steel bars order to satisfy.

In order to find an initial solution for the demand of the Table 2, we compute how many pieces of each length fits in one bar.

So, our initial matrix A is

$$A = \begin{pmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 2 \end{pmatrix}.$$

Then, we build the first problem of the algorithm, RMP1. For this, we consider a minimization linear programming problem with the same number of variables than the number of columns of A, all with coefficients one in the loss function. Also, we consider the column $[6, 4, 2]$ as the right part of the constraints and A as the constraints matrix.

Then we can consider the RMP1 problem as

$$\begin{aligned} \min\ & \lambda_1 + \lambda_2 + \lambda_3 \\ \text{subject to: } & 5\lambda_1 \geq 6, \\ & 3\lambda_2 \geq 4, \\ & 2\lambda_3 \geq 2, \\ & \lambda_1, \lambda_2, \lambda_3 \geq 0. \end{aligned} \tag{2}$$

The solution of the problem shown in the equation (2) is $\lambda_1 = 1.2$, $\lambda_2 = 1.33$ and $\lambda_3 = 1$.

Once we know the solution for this problem, using the complementary slackness theorem, we can calculate the values for each dual variable, obtainining $\pi_1 = 0.2$, $\pi_2 = 0.33$ and $\pi_3 = 0.5$.

These values allow us to formulate the problem AP1, where the matrix of constraints is a one row matrix obtained from the length of the client's demand that it was collected in the Table 2. Also, the right part of the constraint is the total length of each bar, L=10.

$$\begin{aligned} \max\ & 0.2x_1 + 0.33x_2 + 0.5x_3 \\ \text{subject to: } & 2x_1 + 3x_2 + 4x_3 \leq 10, \\ & x_1, x_2, x_3 \geq 0 \text{ and integer.} \end{aligned} \tag{3}$$

The solution of the problem described in the equation (3) is $x_1 = 1$, $x_2 = 0$ and $x_3 = 2$. Due to the fact that the reduced cost is $-0.2 < 0$, we add a new column formed by the optimal solution of AP1.

This addition implies a new problem with four variables called RMP2,

$$\begin{aligned} \min\ & \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 \\ \text{subject to: } & 5\lambda_1 + \lambda_4 \geq 6, \\ & 3\lambda_2 \geq 4, \\ & 2\lambda_3 + 2 + \lambda_4 \geq 2, \\ & \lambda_1, \lambda_2, \lambda_3, \lambda_4 \geq 0. \end{aligned} \tag{4}$$

We can notice that it's the same problem as RMP1 but adding a new variable with coefficient 1 to the loss function and adding the column a_4 to the matrix A.

Solving the problem described in the equation (4), we obtain $\lambda_1 = 1$, $\lambda_2 = 1.33$, $\lambda_3 = 0$ and $\lambda_4 = 1.0$. Also, the values for the dual variables are $\pi_1 = 0.2$, $\pi_2 = 0.33$ and $\pi_3 = 0.4$. This allows us to formulate the AP2 problem

$$\begin{aligned} \max\ & 0.2x_1 + 0.33x_2 + 0.4x_3 \\ \text{subject to: } & 2x_1 + 3x_2 + 4x_3 \leq 10, \\ & x_1, \cdots, x_3 \geq 0 \text{ and integer.} \end{aligned} \tag{5}$$

The solution of the problem shown in the equation (5) is $x_1 = 0$, $x_2 = 2$ and $x_3 = 1$. Hence, the reduced cost is $-0.07 < 0$. So, we consider the column obtained from the values of the optimal solution of the problem AP2, $a_5 = [0, 2, 1]$. If we add it to the problem RPM2, we obtain the RMP3 problem

$$\begin{aligned} \min\ & \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 \\ \text{subject to: } & 5\lambda_1 + \lambda_4 \geq 6, \\ & 3\lambda_2 + 2\lambda_5 \geq 4, \\ & 2\lambda_3 + 2\lambda_4 + \lambda_5 \geq 2, \\ & \lambda_1, \cdots, \lambda_5 \geq 0. \end{aligned} \tag{6}$$

The solution of the problem collected in the equation (6) is $\lambda_1 = 1.2$, $\lambda_2 = \lambda_3 = \lambda_4 = 0$ and $\lambda_5 = 2$. We can calculate the dual variables which are $\pi_1 = 0.2$, $\pi_2 = 0.3$ and $\pi_3 = 0.4$. These values allow us to write the AP3 problem

$$\begin{aligned} \max\ & 0.2x_1 + 0.3x_2 + 0.4x_3 \\ \text{subject to: } & 2x_1 + 3x_2 + 4x_3 \leq 10, \\ & x_1, \cdots, x_3 \geq 0 \text{ and integer.} \end{aligned} \tag{7}$$

The solution of the problem described in the equation (7) is $x_1 = 0$ and $x_2 = 2$ and $x_3 = 1$. So, the reduced cost is 0, non-negative.

In this case, we don't have any new column to add and the optimal solution is the one of the RMP3 problem, collected in the equation (6).

If we round up the solution of this problem, we have that the company should do to satisfy the client's demands are cutting 2 bars in pattern 1 (five pieces of 2m) and cutting 2 bars in pattern 2 (two pieces of 3 m) and a bar in pattern 3 (a piece of 4 m).

This solution gives the company 10 pieces of 2 m, 4 pieces of 3 m and 2 pieces of 4 m.

This solution requires 4 bars and it satisfies the client's demands. We notice that the number of pieces of 2 m is higher than the client's demands. This issue comes because we need an extra bar for the missing piece of 2m.

3.2 Exercice 2

Now, we are going to suppose the introductory example of Section 1.1 with the client's demand of the Table 1.

In this case, our initial matrix is

$$A = \begin{pmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

So, the RMP1 problem is

$$\begin{aligned} \min\ & \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 \\ \text{subject to: } & 4\lambda_1 \geq 45, \\ & 2\lambda_2 \geq 38, \\ & \lambda_3 \geq 25, \\ & \lambda_4 \geq 11, \\ & \lambda_5 \geq 12, \\ & \lambda_1, \cdots, \lambda_5 \geq 0. \end{aligned} \tag{8}$$

We solve the equation (8), if we want to do it by hand we can do it using the Simplex method, and we obtain that its solution is $\lambda_1 = 11.25, \lambda_2 = 19, \lambda_3 = 25, \lambda_4 = 11$ and $\lambda_5 = 12$. In this case, using the complementary slackness theorem, we find that the values for the dual variables are $\pi_1 = 0.25, \pi_2 = 0.5, \pi_3 = 1, \pi_4 = 1$ and $\pi_5 = 1$.

Next, we build the first auxiliary problem, AP1. Its resolution will indicate if there exists a new column with a negative reduced cost. For this problem, the loss function will have as coefficients for the variables the values of the dual variables of RMP1 solution. So, the AP1 problem obtained is

$$\begin{aligned} \max\ & 0.25x_1 + 0.5x_2 + x_3 + x_4 + x_5 \\ \text{subject to: } & 22x_1 + 42x_2 + 52x_3 + 53x_4 + 78x_5 \leq 100, \\ & x_1, \cdots, x_5 \geq 0 \text{ and integer.} \end{aligned} \tag{9}$$

If we solve the equation (9), we obtain $x_1 = x_3 = x_5 = 0$ and $x_2 = x_4 = 1$. It's noticeable that this is a knapsack problem, a classical Integer Linear Programming problem, concretely the unbound case. A deep introduction to this problem can be found in [Martello and Toth, 1990].

So, if we add a new variable, λ_6, to RMP1 with adding to the matrix A the column generated by the solution of AP1,$a_6 = [0, 1, 0, 1, 0]$, we obtain the RMP2 problem:

$$
\begin{aligned}
&\min \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 + \lambda_6 \\
\text{subject to: } &4\lambda_1 \geq 45, \\
&2\lambda_2 + \lambda_6 \geq 38, \\
&\lambda_3 \geq 25, \\
&\lambda_4 + \lambda_6 \geq 11, \\
&\lambda_5 \geq 12, \\
&\lambda_1, \cdots, \lambda_6 \geq 0.
\end{aligned} \tag{10}
$$

Like for the RMP1 problem, we solve the equation (10) and we obtain $\lambda_1 = 11.25, \lambda_2 = 13.5, \lambda_3 = 25, \lambda_4 = 0$, $\lambda_5 = 12$ and $\lambda_6 = 11$. We notice the similitude of the solution to the one of the equation (8). This similarity is due to the fact that the constraints affecting λ_1, λ_3 and λ_5 weren't modified when we added λ_6.

The dual variables of this problem are $\pi_1 = 0.25$, $\pi_2 = \pi_4 = 0.5$ and $\pi_3 = \pi_5 = 1.0$.

So, we obtain the AP2 problem:

$$
\begin{aligned}
&\max\ 0.25x_1 + 0.5x_2 + x_3 + 0.5x_4 + x_5 \\
&\text{subject to: } 22x_1 + 42x_2 + 52x_3 + 53x_4 + 78x_5 \leq 100, \\
&x_1, \cdots, x_5 \geq 0 \text{ and integer.}
\end{aligned} \tag{11}
$$

The solution of the problem of the equation(11) is $x_1 = x_4 = x_5 = 0$ and $x_2 = x_3 = 1$. In addition, the reduced cost is $-0.5 < 0$. So, we have a new column $a_7 = [0, 1, 1, 0, 0]$.

Also, we obtain the problem RMP3

$$
\begin{aligned}
&\min \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 + \lambda_6 + \lambda_7 \\
\text{subject to: } &4\lambda_1 \geq 45, \\
&2\lambda_2 + \lambda_6 + \lambda_7 \geq 38, \\
&\lambda_3 + \lambda_7 \geq 25, \\
&\lambda_4 + \lambda_6 \geq 11, \\
&\lambda_5 \geq 12, \\
&\lambda_1, \cdots, \lambda_7 \geq 0.
\end{aligned} \tag{12}
$$

Solving the problem described in the equation (12), we obtain $\lambda_1 = 11.25$, $\lambda_2 = 1$, $\lambda_3 = \lambda_4 = 0$, $\lambda_5 = 12$, $\lambda_6 = 11$ and $\lambda_7 = 25$. Also, the values for the dual variables are $\pi_1 = 0.25$, $\pi_2 = \pi_3 = \pi_4 = 0.5$ and $\pi_5 = 1$. This allows us to formulate the AP3 problem

$$\begin{aligned} \max\ & 0.25x_1 + 0.5x_2 + 0.5x_3 + 0.5x_4 + x_5 \\ \text{subject to: } & 22x_1 + 42x_2 + 52x_3 + 53x_4 + 78x_5 \leq 100, \\ & x_1, \cdots, x_5 \geq 0 \text{ and integer.} \end{aligned} \tag{13}$$

The solution of the problem shown in the equation (13) is $x_1 = x_5 = 1$ and $x_2 = x_3 = x_4 = 0$. Hence, the reduced cost is $-0.25 < 0$. So, we consider the column $a_8 = [1, 0, 0, 0, 1]$ and we add it to the problem RPM3, obtaining the RMP4 problem

$$\begin{aligned} \min\ & \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 + \lambda_6 + \lambda_7 + \lambda_8 \\ \text{subject to: } & 4\lambda_1 + \lambda_8 \geq 45, \\ & 2\lambda_2 + \lambda_6 + \lambda_7 \geq 38, \\ & \lambda_3 + \lambda_7 \geq 25, \\ & \lambda_4 + \lambda_6 \geq 11, \\ & \lambda_5 + \lambda_8 \geq 12, \\ & \lambda_1, \cdots, \lambda_7 \geq 0. \end{aligned} \tag{14}$$

The solution to the problem collected in the equation (14) is $\lambda_1 = 8.25$, $\lambda_2 = 1$, $\lambda_3 = \lambda_4 = \lambda_5 = 0$, $\lambda_6 = 11$, $\lambda_7 = 25$ and $\lambda_8 = 12$. We can calculate the dual variables which are $\pi_1 = 0.25$, $\pi_2 = \pi_3 = \pi_4 = 0.5$ and $\pi_5 = 0.75$. These values allow us to write the AP4 problem

$$\begin{aligned} \max\ & 0.25x_1 + 0.5x_2 + 0.5x_3 + 0.5x_4 + 0.75x_5 \\ \text{subject to: } & 22x_1 + 42x_2 + 52x_3 + 53x_4 + 78x_5 \leq 100, \\ & x_1, \cdots, x_5 \geq 0 \text{ and integer.} \end{aligned} \tag{15}$$

The solution of the problem described in the equation (15) is $x_1 = 4$ and $x_2 = x_3 = x_4 = x_5 = 0$. So, the reduced cost is 0, non-negative.

In this case, we don't have any new column to add and the optimal solution is the one of the RMP4 problem, collected in the equation (14).

If we round up the solution of this problem, we have that the company should do to satisfy the client's demands are cutting 9 bars in pattern 1 (four pieces of 22 m), cutting one bar in pattern 2 (two pieces of 42 m), cutting 11 bars in pattern 6 (a piece of 42 m and a piece of 53 m), cutting 25 bars in pattern 7 (a piece of 42 m and a piece of 52 m) and cutting 12 bars in pattern 8 (a piece of 22 m and 78 m).

This solution gives the company 48 pieces of 22 m, 38 pieces of 42 m, 25 pieces of 52 m, 11 pieces of 53 m and 12 pieces of 78 m.

This solution requires 58 bars and it satisfies the client's demands. We notice that the number of pieces of 22 m is higher than the client's demands. This issue comes because we need an extra bar for the missing piece of 22m.

4 Conclusions

We can conclude that our developed program can solve diverse one-dimensional cutting stock problems using the column generation algorithm. In addition, it describes how we should cut each bar of the considered length.

On the other hand, we could consider as an improvement of this problem the study for a higher dimension case of the CSP problem, for example, if we consider two dimensions instead of one.

A Code for the program with both exercises resolution

```
    using JuMP, LinearAlgebra, GLPK, DelimitedFiles

function init_A(N,L,l,n)
    A = zeros((N,N))
    for i in 1:N
        A[i,i]=min(floor(Int,L/l[i]),round(Int,n[i]))
    end
    return(A)
end

function print_matrix(f,M)
    nrow = size(M)[1]
    for i in 1:nrow
        println(f,M[i,:])
    end
end

function init_master(A,L,l,n,f,i)
    N = length(l)
    ncols = size(A)[2]
    model = Model(GLPK.Optimizer)
    @variable(model, lambda[1:ncols]>=0)
    @constraint(model, c[j=1:N], sum(A[j,p]*lambda[p] for p in 1:ncols)>=n[j])
    @objective(model, Min, sum(lambda[p] for p in 1:ncols))
    println(f, "RMP $i\n")
    println(f, model)
    optimize!(model)
    println(f, "The solution is:")
    res = zeros(ncols)
for i in 1:ncols
        res[i] = JuMP.value(lambda[i])
println(f, "lambda[", i, "] = ", string(JuMP.value(lambda[i])))
end
    println(f,"\nThe value of the dual variables are:")
    dual_coeff = zeros(N)
    for i in 1:length(l)
        dual_coeff[i]=dual(c[i])
        println(f, "pi[", i, "] = ", dual_coeff[i])
    end
```

```
    println(f)
    return res,dual_coeff
end

function auxiliary_problem(mp,l,L,f,i)
    N = length(l)
    ncols = length(l)
    println(f, "AP $i\n")
    model = Model(GLPK.Optimizer)
    @variable(model, x[1:ncols]>=0, Int)
    @constraint(model, sum(l[p]*x[p] for p in 1:ncols)<=L)
    @objective(model, Max, sum(mp[p]*x[p] for p in 1:ncols))
    println(f, model)
    optimize!(model)
    x_values = zeros(ncols)
    println(f, "The solution is:")
for i in 1:ncols
        x_values[i]=value(x[i])
println(f, "x[", i, "] = ", string(JuMP.value(x[i])))
end
    println(f)
    return x_values
end

function resolution(A,L,l,n,f)
    i=1
    reduced_cost = -1
    new_column = 0
    res = 0
    while (reduced_cost<0)
        res, mp = init_master(A,L,l,n,f,i)
        new_column = auxiliary_problem(mp,l,L,f,i)
        reduced_cost = 1 - sum(new_column[p]*mp[p] for p in 1:length(new_column))
        println(f, "Hence, the reduced cost is $reduced_cost\n")
        if reduced_cost < 0
            A = hcat(A,new_column)
            println(f,"So we have a new column a$(size(A)[2])")
            println(f,new_column)
            println(f)
        end
        if reduced_cost >= 0
```

```
            println(f,"Thus we do not have a new column
and the optimal solution was found at RMP $i\n")
        end
        i+=1
    end
    return res,A
end

function presentation_exercice(N,L,l,n,A,f)
    println(f, "A company produces steel bars with L = $L m and cuts the bars
for the costumers according to their necessities.
Now, the company has to satisfy the following demand:\n")
    println(f, "l_i | number of pieces needed")
    for i in 1:N
        println(f, "$(l[i])  | $(n[i])")
    end
    println(f)
    println(f, "The costs c_p are all equal to one.
We want to minimize the number of steel bars we need to cut
in order to satisfy the demand.\n")
    println(f,"In order to find an initial solution,
we compute how many pieces of each length fits in one bar.\n")
    println(f, "So our matrix A at the first iteration is:")
    print_matrix(f,A)
    println(f)
end

function conclusion_exercice(A,l,res,f)
    nrow = size(A)[1]
    for i in 1:length(res)
        res[i]=round(ceil(res[i]), digits=0)
    end
    println(f,"Rounding the solution of the ultimate RMP we have the following.\n")
    for i in 1:length(res)
        if res[i]>0
            println(f, "- cut $(res[i]) entire bar(s) in pattern $i, which means")
            for j in 1:(size(A)[1])
                if A[j,i]>0
                    println(f, "    $(A[j,i]) piece(s) of $(l[j]) m")
                end
            end
```

```
        end
    end
    println(f,"\nThis solution gives us :")
    prod = A*res
    for i in 1:nrow
        println(f,"- $(prod[i]) piece(s) of $(l[i])m")
    end
end

function ex1()
    L=10
    l=[2,3,4]
    n=[6,4,2]
    N=length(l)
    A = init_A(N,L,l,n)
    f = open("output_ex1.txt", "w")
    println(f,"EX1\n")
    presentation_exercice(N,L,l,n,A,f)
    res, A = resolution(A,L,l,n,f)
    conclusion_exercice(A,l,res,f)
    close(f)
end

ex1()

function ex2()
    L=100
    l=[22,42,52,53,78]
    n=[45,38,25,11,12]
    N=length(l)
    A = init_A(N,L,l,n)
    f = open("output_ex2.txt", "w")
    println(f,"EX2\n")
    presentation_exercice(N,L,l,n,A,f)
    res, A = resolution(A,L,l,n,f)
    conclusion_exercice(A,l,res,f)
    close(f)
end

ex2()
```

References

[Julia JuMP] Bezanson, J., Edelman, A., Karpinski, S. and Shah, V. B. (2017) Julia: A fresh approach to numerical computing. *SIAM review, 59* (1).

[Garey and Johnson, 1979] Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freemann.

[Gilmore and Gomory, 1961] Gilmore, P. C. and Gomory, R. E. (1961). A linear programming approach to the cutting-stock problem, *Operations research, 9* (6).

[Kantorovich, 1960] Kantorovich, L. V. (1960). Mathematical methods of organizing and planning production. *Management science, 6* (4).

[Martello and Toth, 1990] Martello, S. and Toth, P. (1990). *Knapsack Problems: Algorithms and Computer Implementations*, Wiley.