Mini Max Wallpaper

Antonio Marigonda, Dragomir Aleksov, Jan Idziak,
Krasimir Georgiev, Mariusz Kozlowski, Mikhail Krastanov,
Milena Veneva, Monika Sikora, Slav Angelov

Executive summary

Problem: Mini Max company formulated a problem for automatic calculation the number of wallpaper rolls necessary for decorating a room with wallpapers. The final goal is the development of a web-based calculator open for use to both Mini Max staff and the general public.

Proposed solution: We propose an approach for reducing the studied problem to the one-dimensional cutting-stock problem. We show this in details for the case of plain wallpapers as well as for the case of patterned wallpapers with straight match. The case of patterned wallpapers with offset match can be considered similarly.

The one-dimensional cutting-stock problem can be formulated as a linear integer programming problem. The approach proposed by P. C. Gilmore and R. E. Gomory in 1961 can be used for solving this problem for the case when the number of variables is large. In our opinion, the dynamic programming approach or the approach based on generating of all possible cuttings is more appropriate for the case of separate room for which a relatively small number of wallpapers' strips is needed.

We develop an approach for calculating the needed number of wallpapers for relatively small problems, create an algorithm in a suitable graphical interface and make different tests. The tests show the efficiency of the proposed approach compared with the existent (available) wallpapers' calculators.

1. Introduction

For 20 years now Mini Max Ltd company distributes in Bulgaria high quality wallcoverings manufactured by leading European companies. In its showroom can be found a wide variety of designs, original patterns and stylish combinations appropriate for both residential and commercial environments. A professional

team assists clients in selecting the right wallpapers. But there is a problem: how to calculate the necessarily amount of rolls.

Everyone can find different types of wallpapers' calculators on the Internet (cf., for example, the following WEB addresses

http://www.diy.com/diy/jsp/bq/templates/content$_1$ookup.jsp?content
= /content/knowledge/calculators/wallpaper/wallpaper.jsp
http://www.tangletree − interiors.co.uk/wallpaper − calculator/
http://www.praktiker.bg/praktiker − international/html/bg$_B$G/
161211/index.html
http://www.wallpaperdirect.com/wallpaper − calculator.php?)

requiring usually the sizes of the walls (and, in some cases, the sizes of a door and/or windows).

Simple tests convinced us that these calculators are not appropriate for the practical needs of clients. And this is our main motivation for studying this problem.

2. Problem statement

Now we shall state the problem in full details.

2.1. Data of the problem

The client will provide:

- number of walls to be decorated;
- width and height of each wall;
- sizes of door and windows present in each wall (if any).

The wallpaper rolls are characterized by:

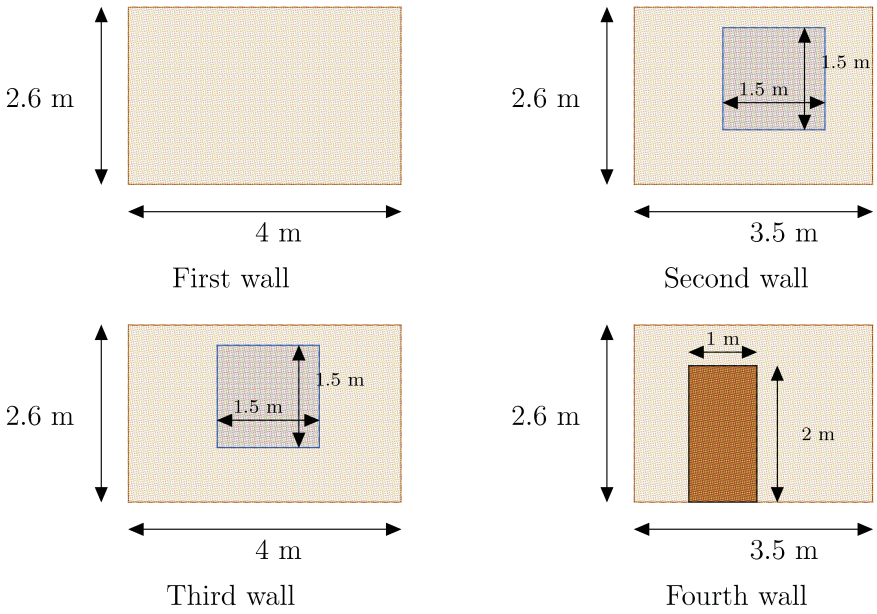
- width and length of the roll;
- design of the roll (plain, straight pattern, offset pattern).
- for each roll with straight pattern, it is specified the height of the pattern.
- for each roll with offset pattern, it is specified the height of the pattern and the offset.

2.2. Restrictions

($R1$) On each wall can be used only one kind of roll.
($R2$) Each wall should be fully covered with vertical stripes, without overlapping.
($R3$) It is forbidden to join two stripes along horizontal in horizontal line.
($R4$) In the case of straight pattern, two consecutive stripes must match their pattern.
($R5$) In the case of offset pattern, the patterns between two consecutive stripes must have an offset specified by the offset parameter of the roll.

3. Basic example

We assume to have to decorate a standard room with two windows and one door, using rolls of length 10 m, width 0.7 m, and straight pattern of height 0.64 m as in the following basic example:



4. Room analysis

Now we will begin our analysis of the room. Assume that the room has d walls, we start considering a matrix $\tilde{W}$ whose i-th row contain the width w_i and

the height h_i of the i-th wall of the room.

$$\tilde{W} := \begin{pmatrix} w_1 & h_1 \\ \vdots & \vdots \\ w_d & h_d \end{pmatrix}.$$

In the basic example, the matrix is:

$$\tilde{W} := \begin{pmatrix} 4 & 2.6 \\ 3.5 & 2.6 \\ 4 & 2.6 \\ 3.5 & 2.6 \end{pmatrix}.$$

We have to take into account the following further technical restriction:

(T_1) When cutting the wallpaper to size for the required wall height, it is necessary to take into account a 10 cm safety margin, which is used in the case the ceiling or floor is not completely straight.

So we will consider all the walls as 10 cm longer than in the given data, forming the new matrix:

$$W := \begin{pmatrix} w_1 & h_1 + 0.1 \\ \vdots & \vdots \\ w_d & h_d + 0.1 \end{pmatrix},$$

in our example:

$$W := \begin{pmatrix} 4 & 2.7 \\ 3.5 & 2.7 \\ 4 & 2.7 \\ 3.5 & 2.7 \end{pmatrix}.$$

4.1. Plain wall

We will call *plain walls* the walls of the room in which there are neither windows, nor doors. The situation is the one depicted in the First Wall of the basic example. This is the most simple situation.

In the straight pattern case, since we have to match the pattern along consecutive stripes, we will impose that all the strips should begin with the beginning of a pattern.

The number of patterns required on each strip with patterns of height h_P in order to fill the height of the i-th wall is:

$$\ell^i_{\max} := \left\lceil \frac{h_i + 0.1}{h_P} \right\rceil,$$

where we set:

$$\lceil x \rceil = \min\{n \in \mathbb{N} : \, x \leq n\},$$
$$\lfloor x \rfloor = \max\{n \in \mathbb{N} : \, x \geq n\}.$$

In our basic example, we have that $h_P = 0.64$, $h_i = 2.6$, $i = 1, \ldots, 4$, whence

$$\ell^i_{\max} := \left\lceil \frac{2.7}{0.64} \right\rceil = 5, \qquad i = 1, \ldots, 4.$$

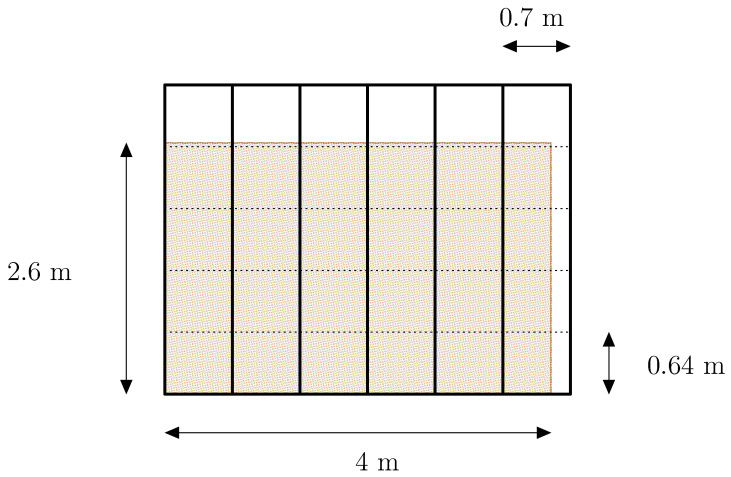
The number of strips of width w_P needed to fill the i-th wall is given by

$$n_i := \left\lceil \frac{w_i}{w_P} \right\rceil.$$

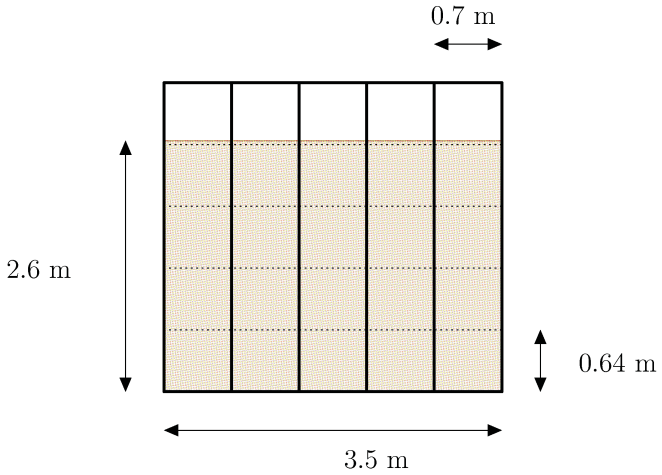
In this computation it **does not matter** if the wall contains windows or door.

In our basic example, we have $w_P = 0.7$ m, whence:

$$n_1 = n_3 = \left\lceil \frac{4}{0.7} \right\rceil = 6,$$
$$n_2 = n_4 = \left\lceil \frac{3.5}{0.7} \right\rceil = 5.$$



The First and Third wall of the basic example:
6 strips containing 5 patterns are needed.

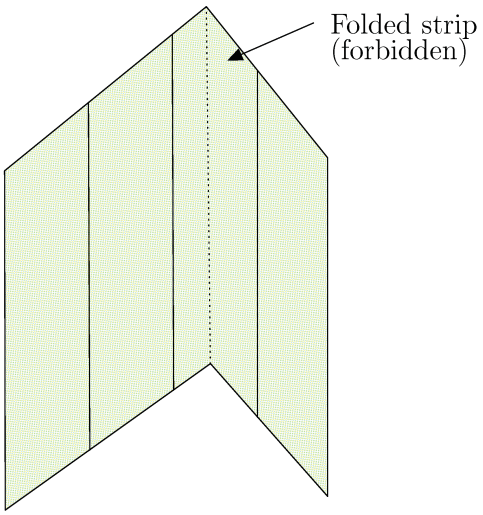


The Second and Fourth wall of the basic example:
5 strips containing 5 patterns are needed.

The part exceeding the measure of the wall is considered *wasted*. In particular we are assuming that the following condition holds:

(G_1) The part of the strip that exceeds the width of the wall cannot be used to another wall.

In other words, (G_1) forbids us to *fold* a strip around a corner.



Assumption (G_1) forbids to fold strips around corners.

If we *do not* assume that (G_1) holds, i.e. we *allow* folding, the problem is reduced to the situation in which we have a *single* wall.

4.2. Managing obstacles

If a window or a door is present on a wall, then we say that there is an *obstacle*. If the obstacle is sufficiently large, we may spare a certain amount of wallpaper. However, since the exact knowledge of the position of obstacles in the wall is unknown (we have information only about the size of each obstacle), we will consider the worst case scenario, i.e. we imagine that the obstacles are always placed in the position that minimize the spare of wallpaper.

All the information on obstacles is encoded in the following matrix:

$$\mathcal{O} := \begin{pmatrix} o^1_{wl} & o^1_w & o^1_h \\ \vdots & & \vdots \\ o^k_{wl} & o^k_w & o^k_h \end{pmatrix},$$

where o^j_{wl} denotes the index of the wall containing the j-th obstacle, and o^k_w, o^k_h are its width and height, respectively. k is the total number of obstacles.

In our basic example, we have that:

$$\mathcal{O} := \begin{pmatrix} 2 & 1.5 & 1.5 \\ 3 & 1.5 & 1.5 \\ 4 & 1 & 2 \end{pmatrix}.$$

Consider now a generic obstacle of width o_w and o_h on a wall. We want to compute the minimum number of strips that will be fully horizontally cut by the presence of the obstacle. We will call them the strips *blocked* by the obstacle.

The maximum number of strips that can be entirely contained in the obstacle are $\left\lfloor \frac{o_w}{w_P} \right\rfloor$. However, if we perform a slight horizontal translation to the right, we have that the first strip now is not fully horizontal cut, and if the translation was sufficiently small, we did not cut in full the last one in which the translated obstacle is present. So the number of strips blocked by an obstacle of width o_w is

$$b(o_w) = \max\left\{0, \left\lfloor \frac{o_w}{w_P} \right\rfloor - 1\right\}.$$

The strips that encounter an obstacle, but are not blocked by it in the above sense, will be considered as the one not encountering the obstacle *at all*. Accordingly, every obstacle whose width is strictly less that $2w_P$ can be actually *neglected*.

In our example, the number of strips blocked by each window is

$$b(1.5) = \max\left\{0, \left\lfloor \frac{1.5}{0.7} \right\rfloor - 1\right\} = 1,$$

while the door is negligible since $b(1) = 0$.

The same argument is used to find the minimum number of patterns fully contained in an obstacle: indeed if a pattern of the strip is completely contained into an obstacle, we may cut the strip in order to *save* it. The worst case possible is the one in which the number of patterns that can be saved is minimal.

So the number of patterns that can be saved due to presence of an obstacle of height o_h is

$$s(o_h) = \max\left\{0, \left\lfloor \frac{o_h}{h_p} \right\rfloor - 1\right\}.$$

The stripes that are blocked by a thin obstacle that does not allow any sparing of patterns, will be considered as the one not encountering the obstacle *at all*. Accordingly, every obstacle whose height is strictly less that $2h_p$ can be actually *neglected*.

In our example, we can spare from each window

$$s(1.5) = \max\left\{0, \left\lfloor \frac{1.5}{0.64} \right\rfloor - 1\right\} = 1$$

pattern.

If in a wall there are more than one obstacle (i.e. many windows, or doors), we make the following assumption (*non interacting obstacles*):

(G_2) The sets of strips blocked by each obstacle are pairwise disjoint.

If assumption (G_2) is not fulfilled, we start to ignore some obstacles until we fulfill (G_2) with the remaining ones.

Under assumption (G_2), we are able for each wall to construct a list containing the number of strips and the number of pattern for each strips allowing us to cover it. Assume to have to cover the wall indexed by i, then we consider the rows of the matrix $\mathcal{O}$ whose first element is i:

$$J_i = \{j : \text{ the first element of the } j\text{-th row of } \mathcal{O} \text{ is } i\}.$$

If $j \in J_i$, j-th obstacle will allow us to spare $s(o_h^j)$ patterns for each of the $b(o_w^j)$ stripes that blocks. So in that wall we can use $b(o_w^j)$ stripes with $\ell^i_{\max} - s(o_h^j)$ patterns to cover the area where the obstacle is present. We stress on the fact

that this is the worst case scenario, indeed if the obstacle is well-placed, the spare of wallpaper can be much greater.

We partition J_i in the following way. For every $0 \leq k \leq \ell^i_{\max} - 1$ we have the (possibly empty) sets:

$$J_i^k := \{j \in J_i : \ell^i_{\max} - s(o_h^j) = k\},$$

Given $0 \leq k \leq \ell^i_{\max} - 1$, we will define

$$n_{i,k} = \begin{cases} \displaystyle\sum_{j \in J_i^k} b(o_w^j), & \text{if } J_i^k \neq \emptyset, \\ \\ 0, & \text{if } J_i^k = \emptyset, \end{cases}$$

$$n_{i,\ell^i_{\max}} = n_i - \sum_{k=0}^{\ell^i_{\max}-1} n_{i,k},$$

$$n_{i,m} = 0, \text{ if } p > \ell^i_{\max}.$$

Finally, we have that to cover the i-th wall we need $n_{i,k}$ stripes of length k, $k = 1, \ldots, \ell^i_{\max}$. Summing up on the indexes of the walls, we end up with the following matrix:

$$\mathcal{N} := \begin{pmatrix} \displaystyle\sum_{i=1}^{d} n_{i,1} & 1 \\ \vdots & \vdots \\ \displaystyle\sum_{i=1}^{d} n_{i,\ell_\infty} & \ell_\infty \end{pmatrix},$$

where

$$\ell_\infty := \max_{i=1,\ldots,d} \ell^i_{\max}.$$

This matrix encodes the total number of strips for each length that are needed in order to cover the room.

In our basic example, we have $\ell_\infty = 5$, $J_1 = J_4 = \emptyset$, $J_2 = J_2^1 = \{1\}$, $J_3 = J_3^1 = \{2\}$, $n_{1,5} = 6$, $n_{2,4} = n_{3,4} = 1$, $n_{2,5} = 4$, $n_{3,5} = 5$, $n_{4,5} = 5$, in all the other

cases we have $n_{p,q} = 0$. So

$$\mathcal{N} := \begin{pmatrix} 0 & 1 \\ 0 & 2 \\ 0 & 3 \\ 2 & 4 \\ 20 & 5 \end{pmatrix},$$

which means that to cover the walls of the room we need 2 stripes of length 4 and 20 stripes of length 5.

5. The one-dimensional cutting-stock problem

The one-dimensional cutting-stock problem can be formulated as follows: We assume that an unlimited number of wallpapers' rolls of length L is given. We need a given number b_i of pieces of length l_i with $l_i \leq L$ for $i = 1, 2, \ldots, m$. We say that the vector $(a_1, a_2, \ldots, a_m)$ whose components are nonnegative integers is an admissible cut if the following inequality holds true:

$$a_1 l_1 + a_2 l_2 + \cdots + a_m l_m \leq L.$$

Let us assume that n is the maximum number of all admissible cuts

$$a_j = (a_{1j}, a_{2j}, \ldots, a_{mj})^T,\ j = 1, 2, \ldots, n,$$

where by a^T is denoted the transposed vector of the vector a. If we denote by x_j, $j = 1, 2, \ldots, n$, the number of rolls needed to be cut using the admissible cut a_j, our problem consists in minimizing the sum

$$x_1 + x_2 + \cdots + x_n$$

in such a way that the needed amount of pieces of strips of wallpapers with fixed length to be satisfied.

In this way we can formalize the one-dimensional cutting-stock problem by considering the following linear integer programming problem (P):

$$x_1 + x_2 + \cdots + x_n \ \rightarrow\ \min$$

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \geq b_1 \\ &a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \geq b_2 \\ &\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ &a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \geq b_m \end{aligned}$$

x_j is a nonnegative integer
for each index $j = 1, \ldots, n$.

5.1. An approach proposed by P. C. Gilmore and R. E. Gomory

To solve the problem (P) we can apply the approach proposed by P. C. Gilmore and R. E. Gomory in [2] and [3] (cf. also, for example, [1] and [4]). In our case, this approach consists in solving the linear problem (P_0)

$$x_1 + x_2 + \cdots + x_n \ \rightarrow \ \min$$

$$\begin{array}{l} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \end{array}$$

$$x_j \geq 0, \ j = 1, \ldots, n.$$

Without loss of generality we may write this problem in the following form

$$< e, y > + \sum_{i \in N} z_i \ \rightarrow \ \min$$

$$\begin{array}{c} By + \sum\limits_{j \in N} a_j z_j = b \\ y_j \geq 0, \ z_k \geq 0, \end{array}$$

where B is the basic matrix, e is a vector of length m with components equal to 1 and $< \cdot, \cdot >$ denotes the standard scalar product in R^m. When we start to solve this problem we set

$$B = \begin{pmatrix} \lfloor L/l_1 \rfloor & 0 & 0 & \ldots & 0 \\ 0 & \lfloor L/l_2 \rfloor & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & \lfloor L/l_m \rfloor \end{pmatrix}.$$

Let us underline that the columns a_j, $j \in N$, are unknown. Then from the presentation

$$y = B^{-1}b - \sum_{j \in N} B^{-1}a_j z_j.$$

we can replace each basic variable in the target function and to obtain that the coefficient in front of the j-th nonbasic variable x_j in the target function is $c_j = 1 - e^T B^{-1} a_j$, $j \in N$. If all coefficients c_j, $j \in N$, are nonnegative, then this will be the solution of the problem (P_0).

But the columns a_j are in fact unknown. In order to understand if there exists an admissible cut a for which the coefficient $1-e^T B^{-1}a < 0$, the following integer knapsack problem (K_0) is considered:

$$p_1 a_1 + p_2 a_2 + \cdots + p_m a_m \to \max$$

$$l_1 a_1 + l_2 a_2 + \cdots + l_m a_m \le L$$

a_j is a nonnegative integer
for each index $j = 1, \ldots, m$.

where $p^T = e^T B^{-1}$.

Let $\bar{a}$ denotes its solution. If $p^T \bar{a} > 1$, then $\bar{a}$ is admissible and the cut should be added as a new column to the matrix of the considered linear problem (P_0) and to repeat the same procedure. If $p^T \bar{a} \le 1$, we are sure that there is no admissible cut that has to be taken into account for solving the problem (P_0).

5.2. "A brute force algorithm"

Data of the algorithm: We have a matrix $\mathcal{N}_{m\times 2}$, where the first element of i-th row N_{i1} denotes the needed number of strips containing the number of patterns specified by the second element N_{i2}.

Steps of the algorithm

Step 1: We calculate the vector $w = (w_1, w_2, \ldots, w_m)$ with

$$w_i = \min\left(\left\lfloor \frac{L}{l_i} \right\rfloor, N_{i1}\right)$$

which estimates the number of strips that is reasonable to be obtained from a roll of wallpapers. Here L denotes the number of patterns contained in a roll and l_i, $i = 1, \ldots, m$ - the number of patterns contained in each needed strip.

Step 2: We form a set of all possible different vectors $v = (v_1, v_2, \ldots, v_m)$ such that each component v_i belongs to the set $\{0, 1, \ldots, w_i\}$. Each element of this set represents an abstract cut of a roll. Then we form the matrix A whose columns are the elements of this set.

Step 3: From the matrix A we form the matrix B by eliminating the zero vector as well as all the columns $v = (v_1, v_2, \ldots, v_m)$ of the matrix A satisfying the inequality

$$v_1 w_1 + v_1 w_1 + \cdots + v_m w_m > L.$$

The columns of the matrix B represent all admissible cuts of a roll.

Step 4: From the matrix B we form the matrix C by eliminating all the columns $v = (v_1, v_2, \ldots, v_m)$ of the matrix B satisfying the following: $m-1$ components of the vector v are zeros and there exists another column $\bar{v}$ of B also with $m-1$ zeros on the same places as v such that the nonzero component of v is strictly less that the same component of $\bar{v}$. The matrix C is a subset of admissible cuts of a roll that are more efficient than the columns of B, i.e. if we can fulfill some needs with the cuts from B we can do the same needs with the cuts from C.

Step 5: We form the augmented matrix

$$D = [B : C : \cdots : C]$$

where the number of copies of the matrix C is equal to

$$M = \max\left\{\left\lceil \frac{N_{i1}}{\lfloor L/l_i \rfloor} \right\rceil\right\} - 1.$$

The number $M+1$ gives us an upper bound of the needed number of cuts to realize each kind of strip number separately. So the total cuts needed to realize all requirements is bounded from above by the number of columns of the matrix B $+ M *$ (number of columns of C). We have to pay your attention that replacing the matrix B with C we can not ensure to fulfill the needed requirements as equalities.

Step 6: So we formulate the linear algebraic system $Dx = b$ with b equal to the first column of the matrix $\mathcal{N}$, where the components of the unknown vector x take values only 0 and 1. We are minimizing the sum of the components of the vector x by using a standard procedure from Matlab. The minimum of the sum of the components gives us the minimum rolls needed to fulfill the requirements. The columns corresponding to the nonzero components of x determine the optimal way of cuts of each roll.

5.3. The recursive solution

Our starting point is the two-column matrix $\mathcal{N}$ encoding the number of strips needed for each length, together with the number of patterns in a roll L. Assume that $\mathcal{N}$ has m rows. We can reduce $\mathcal{N}$ to the case in which the first element of each row is nonzero. Define E_i to be the $m \times 2$ matrix with 1 in the $(m, 1)$ entry and 0 elsewhere.

We define now the following recursive function
$\mathrm{Roll} : \mathrm{Mat}_{m,2}(\mathbb{N}) \times \{1, \ldots, m\}$ by setting:

- $\mathrm{Roll}(M, p) = 1$ if the first column of M is the zero vector
- If this is not the case, we form the m-dimensional vector $c = (c_1, \ldots, c_m)$ as follows:

$$c_j = \begin{cases} \mathrm{Roll}(M - E_j, p - M_{j2}), & \text{if } M_{i1} > 0,\ p \geq M_{j2} \\ \\ +\infty \text{ otherwise,} \end{cases}$$

and set

$$\mathrm{Roll}(M, p) = \begin{cases} 1 + \mathrm{Roll}(M, L), & \text{if } c_j = +\infty \text{ for every } j = 1, \ldots, m, \\ \\ \min\limits_{j=1,\ldots,m} c_j, & \text{otherwise.} \end{cases}$$

This recursion procedure can be improved also keeping track of the decision made at each step, hence providing not only the optimal number of rolls, but also the optimal way of cutting. Unfortunately, the time needed to have a solution with this procedure is considerably sensitive with respect to the size of the matrix $\mathcal{N}$ and also to the size of the entries.

References

[1] A. C. Dikili, B. Barlas, *A Generalized Approach to the Solution of One-Dimensional Stock-Cutting Problem for small Shipiyards*, Journal of Marine Science and Technology, Vol. 19, No. 4, pp. 368–376, 2011.

[2] P. C. Gilmore and R. E. Gomory, *A Linear Programming Approach to the Cutting-Stock Problem*, Operations Research, Vol. 9, No. 6 (Nov.–Dec., 1961), pp. 849–859, 1961.

[3] P. C. Gilmore and R. E. Gomory, *A Linear Programming Approach to the Cutting-Stock Problem – Part II*, Operations Research, Vol. 11, No. 6 (Nov.–Dec., 1963), pp. 863–888, 1963.

[4] R. W. Haessler, P. E. Sweeny, *Cutting-Stock Problems and Solution procedures*, Europenian Journal for Operations Research, Vol. 54, pp. 141–150, 1991.