An. Şt. Univ. Ovidius Constanţa Vol. 11(2), 2003, 75–86

# COVERING WITH RECTANGULAR PIECES

**Paul Iacob, Daniela Marinescu and Cristina Luca**

**Abstract**

The practical problem we are trying to model is: we have to cover a room with a material(carpet or linolleum). This material is in a roll of fixed width. We want to cover the room with a minimum number of pieces and to waste a minimum amount of material.

We present in this paper a synthesis of our works. We can easily see that the two criteria, the minimum loss of material and the minimumnumber of pieces, are conflicting. In the first part we present two algorithms for finding the minimum number of pieces when the two areas are equal and to find minimum Pareto optimal points with a diagram when the room is rectangular and "L" shape.

Second we study the property of the covering models. It is shown that all the covering models which contains at most 4 components are with guillotine restrictions and for 5 and 6 components there is only one type of model without guillotine restrictions. Using a two-dimensional language, we find some symmetrical cutting model.

## INTRODUCTION

The practical problem we are trying to model is: we have to cover a rectangular room with a material (linolleum or carpet). This material is in a roll of fixed width and by cutting it with a guillotine we get another rectangle. We want to cover the room with a minimum number of pieces and to waste a minimum amount of material (the lost represents the difference between area of the bought material and the area of the room). We will construct two algorithms for finding the minimum number of pieces when the two areas are equal in the two cases: the rectangular room and the "L" shaped room. From these algorithms we can obtain also minimum Pareto solutions. Then we study the property of the covering models and we give an algorithm for finding

Key Words: Multiple Criteria Optimization, Algorithms, Database Systems.

symmetrical model of cutting. We present also some possible applications and extensions.

## BACKGROUND

## THE COVERING OF A RECTANGULAR ROOM

First for the rectangular case, suppose that we have to cover a room of 2m x 6m with a material of 3 m width. Then we can make the cover with one piece of 3m x 6m loosing 18m - 12m = 6m, or with two pieces of 3m x 2m and no loss. Customers, depending on money and taste, will choose between the two extremes: one piece (when it is possible) or any number of pieces with no loss.

We want to know how the material with dimensions $x$ and $y$ can be cut to cover a room of dimensions $a$ and $b$ with the condition (c1):

$$a * b = x * y$$

It is clear that if $x = a$ (or $x = b$) we have equal rectangles and there is no cutting problem. Therefore we will assume that we are in the situation:

$$x < a < b < y.$$

If $x > a$ we can change $x$ with $y$.

Practically we may consider only the case where $x$, $a$, $b$, $y$ are integers.

Let $m = g.c.d(a, b, x, y)$. We may consider the cover divided in squares of dimension $m$ and any coverage of the room will be a permutation of these squares. This is a finite number but unacceptably great.

The algorithm we propose is based on the following two properties:

- If we cover a part of the floor with a material, starting fron N-W corner, we obtain a new piece of material and a new piece of floor. This is a new problem, that has the same properties described by condition c1.

- The cover may be put on the floor in two directions (see in the example below figures 1 and 2).

**Example 1**

$a = 6$ $b = 10$ $x = 4$ $y = 15$ $a * b = x * y = 60$

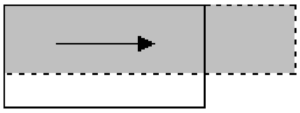


Figure 1

Now we have to cover a piece of floor of 2m x 10m with a coverage of 4m x 5m.

If we put the cover in the other direction we will have following case:

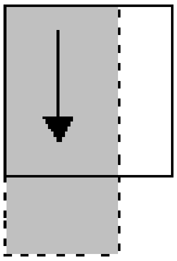


Figure 2

We still have to cover an area of 6m x 6m with a material of 4m x 9m.

## THE ALGORITHM

The algorithm we propose is recursively generating a binary tree:

If the initial problem is to cover the rectangle of dimensions $a$ and $b$ with a material of dimensions $x$ and $y$ then it is the root $T(a, b, x, y)$; putting $x$ on $a$ we obtain the left subtree with the root $T(a - x, b, x, y - b)$ and putting $x$ on $b$ we obtain the right subtree with the root $T(a, b - x, x, y - a)$.

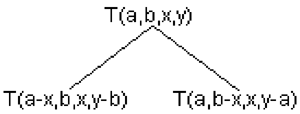



Figure 3

As it is showed above, the algorithm finishes after a finite number of steps. The cutting design with the smallest number of pieces will be the shortest way from the root to a leaf.

More details can be found in [1] and [3].

Theoretical results for this algorithm can be found in [2].

We can detect some situations when growing the number of pieces for covering the initial rectangle; between them some are Pareto optimum points.

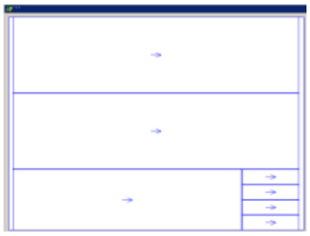
If $nr$ is the number of pieces and $lo$ is the lost material then we say that $(nr, lo)$ is a optimum Pareto point if there are no points with smaler number of pieces and with smaller lost of material. Equivalent: if we want a smaller number of pieces we must increase the lose and if we want to decrease the lose we must increase the number of pieces. More details on this definition can be found in [10].

Our program is written in Borland DELPHI 5.0 and detects such points like in the example below where the results are coming from the program's execution.

**Example 2**

Let the input data $a = 8$, $b = 7$, $x = 2.5$, $y = 22.40$, $Error = 0.01$, $No.levels = 100$, where $y$ is the minimum length of the needed material for covering the room with $nr$ pieces. The program's results are:

| No.pieces | Loses | y |
|---|---|---|
| 3 | 4.00 | 24.00 |
| 4 | 2.75 | 23.50 |
| 5 | 1.50 | 23.00 |
| 6 | 0.25 | 22.50 |
| 7 | 0.00 | 22.40 |



where the image ilustrate the covering with 7 material pieces.

## THE COVERING OF AN "L"-SHAPED ROOM

We will arrange the material from the corners $(a, e)$, and $(b, f)$; in the both situations the material may have two directions symbolizing the direction of $y$ by "$\rightarrow$". Therefore, we will construct a quaternary tree.

In the algorithm we will also use the notations $s$ who has two value: 0 if the surface is an "L" and the material is a rectangle or 1 if the surface is a rectangle and the material is a "L"; $re$ is the rest of the material and $l$ is the level's number.

An "L" form will be describes by four dimension: $a$, $b$, $c$, $d$ like in the figure 4. The dimensions $e$, $f$, $A$, $y$ are calculated by the formulas: $e = b - d$; $f = a - c$; $A = a * e + d * f = b * f + c * e$; $y = A/x$, where $x$ is the material's width and $y$ is the material's length.

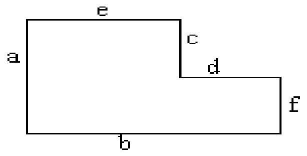
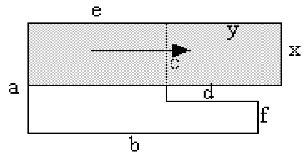


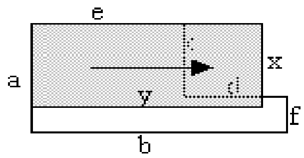

Figure 5

We will show just the configurations for the first situation, when $p = 0$ - the material is directed put from the corner $(a, e)$.

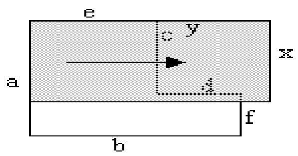
**Case I:** $x \leq c$ After the material is arranged, it remains to cover a surface "L" with a rectangle. The new dimensions are calculate after the formulas: $a' = a - x$, $b' = b$, $s' = 0$, $c' = c - x$, $d' = d$, $re' = -1$, $x' = x$, $y' = y - e$



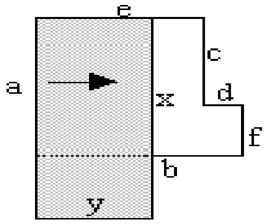

**Case II:** $c < x \leq a$ **and** $e < y$
After the material is arranged, it remains to cover a surface "L" with a rectangle. The new dimensions are calculate after the formulas: $a' = f$, $b' = b$, $s' = 0$, $c' = x - c$, $d' = y$, $re' = -1$, $x' = c$, $y' = y - e$



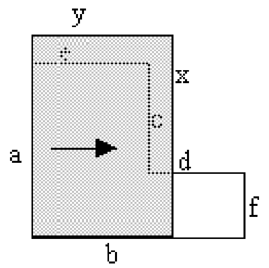

**Case III:** $c < x < a$ **and** $b \leq y$
After the material is arranged, it remains to cover a rectangle surface with an "L". But, in algorithm, we will consider that we cover also an "L" with a rectangle. The new dimensions are calculate after the formulas: $a' = x$, $b' = y - e$, $s' = 1$, $c' = x - c$, $d' = d$, $re = -1$, $x' = a - x$, $y' = b$



**Case IV:** $x > a$ **and** $y \leq e$
After the material is arranged, it remains to cover a surface "L" with a rectangle. The new dimensions are calculate after the formulas:
$a' = a$, $b' = b - y$, $s' = 0$, $c' = c$, $d' = d$, $re' = x * (b - y)$, $x' = x - a$, $y' = y$




**Case V:** $x > a$ **and** $y > e$
In this case, if we put the rectangle also in the corner (a, e) we obtain two remains of material. To avoid this we put the rectangle in the corner (a, b). After the material is placed, it remains to cover a rectangle with a "L". The new dimensions are calculate after the formulas: $a' = x - f$, $b' = y$, $s' = 1$, $c' = c$, $d' = e$, $re' = x * (b - y)$, $x' = f$, $y' = b - y$




*Remark 1:* It is easy to observe that if $x = c$ or $b = y$ or $x = a$ or $y = e$ it remains to cover a rectangle surface with a rectangle. This situation was presented in the first part.

*Remark 2:* The rest of the material *(re)* may have the value 0 just in the cases at the first remark.

The placements for $p = 1$ or $p = 2$ or $p = 3$ contain the same sub cases but with little modifies as follows:

The placement for $p = 1$ is similar with $p = 0$ but in the formulas it must be change the $x$ with $y$ and vice-versa.

The placement for $p = 2$ is similar with $p = 1$ but in the formulas it must be change the $a$ with $b$ and vice-versa and the $c$ with $d$ and vice-versa.

The placement for $p = 3$ is similar with $p = 2$ but in the formulas it must be change the $x$ with $y$ and vice-versa.

*Remark 3:* We will consider that the dates are integer numbers (in concordance with the reality). Let $m = lcm\ (a,\ b,\ c,\ d,\ x)$. Dividing the surface and the material in squares with side $m$, the solution of the covering without loses find out in the permutation of the material squares. The number of squares being a finite number. It results that after a finite number of permutations the algorithm will have to match two rectangles with the same dimensions.

The following algorithm creates a quaternary tree level by level. Each node has four descendents for each placement $p = 0$, $p = 1$, $p = 2$, respectively

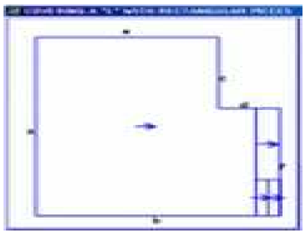
$p = 3$. The algorithm stops when $re = 0$ or the user stops it. For halve of the necessary memory we retain just the last level of the tree using a stack *ST1* and create the next level in another stack *ST2*. For each level it retains in a list solution the node which has the $re > 0$ minim. The solution is construct by restoring the way from the root to the respective node using the level's number and the order's number in the level of the node. For each node we retain the value $a$, $b$, $c$, $d$ (for the "L") and $x$, $y$ (for the rectangle), $l$ (the level's number), $ind$ (the order's number in the level), $s$ (has two value: 0 if the surface is an "L" and the material is a rectangle or 1 if the surface is a rectangle and the material is an "L")¸$re$ (the rest of the material), $p$ (the style of the co! vering).

The application was implemented in Borland DELPHI 5.0. The execution of programme is illustrate in [4].

**Example 3**

For the initial dimensions: $a = 50$, $b = 40$, $c = 20$, $d = 10$, $x = 50$ the programme calculate the dimensions: $e = 30$, $f = 30$, $y = 36$. If $Error = 0.01$ and $No.levels = 10$ than we obtain:

| No.pieces | Loses | y |
|---|---|---|
| 1 | 35000.00 | 4.00 |
| 2 | 10000.00 | 24.00 |
| 3 | 100.00 | 2.00 |
| 4 | 0.00 | 0.00 |



where the image ilustrates the covering with 4 material pieces.

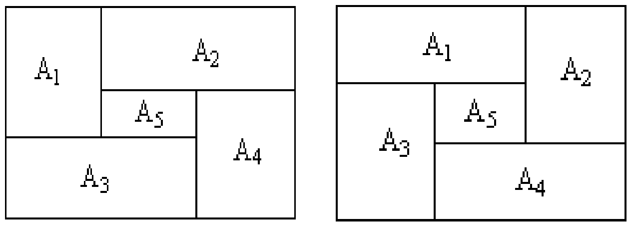
**We can see that while *No. pieces* increases, *Loses* decrease.**

Details for "L" shaped room can be found in [4].

A covering model has guillotine restrictions if in a cutting process of the rectangles $A1$, $A2$, ... ,$An$ from support $A$ is possible to use a guillotine technologies, that mean in every moment of cutting-stock process, a cut is possible from a border to another border parallel with another border of the remains support.

## MODELS WITH OR WITHOUT GUILLOTINE RESTRICTIONS

In [1] we described the covering models depending of the number of pieces, $n$. For $n \in 1, 2, 3, 4$ we have just cutting models with guillotine restrictions. In the case $n = 5$ we obtain also a model without guillotine restrictions like in the below image.




The cases for $n > 5$ are similar with $n < 5$ or $n = 5$, considering a piece $A_i$ like a covering model.

From this it follows a technology for covering a rectangle with rectangular pieces, depending from the dimensions of the pieces.

Demonstrations for this part of the paper are in [2] and [4].

## THE GENERATION OF SYMMETRIC COVERING MODEL

Let us consider a rectangular support plate $A$ which is cover by $k$ rectangular plate $A_i$ for $i = 1, 2, ..., k$.

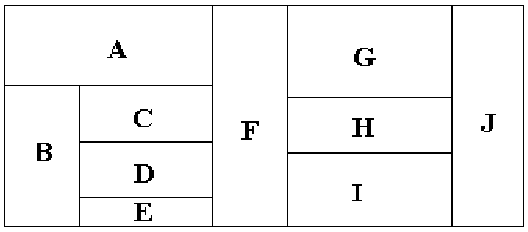
A *covering model* $M$ of the support plate $A$ by the $k$ plates $A_i$ is a possible arrangement of the components $A_i$ on the support $A$.

Note that a covering model is similar with a cutting-stock model from [5].

The *covering model* is *with guillotine restrictions* when, in every moment of the covering process, a cut is possible from a border to another border parallel with another border of the support.

In our paper [6] we have demonstrate that a covering model with guillotine restrictions may be represented by an arithmetic expression with two binary operations: row-concatenation "=" and column-concatenation "—" and with $k$ operands $A_1$, $A_2$, ..., $A_k$. We note that the concatenation defined by us in [5] is not commutative. For every arithmetic expression there is a syntactic tree, a binary tree, in which each node represents an operation ("=" or "—") and the children of the node represent the arguments of the operation.

So the cutting model with guillotine restrictions for the below figure:




can be represented by the following arithmetic expression:

$$(A = (B|(C = D = E)))|F|(G = H = I)|J$$

Let us return to our problem: we have a covering model with guillotine restrictions and we try to generate, if it is possible, another model which is symmetric:

( a ) Relatively to an horizontal axe;

( b ) Relatively to a vertical axe;

( c ) Relatively to a central point.

In [5] we presented an algorithm for the generation of all models with guillotine restrictions.

By using a crossing method of binary trees and by changing the order of the children nodes for an interior nod, we presented, in [7], an algorithm for the generation of all symmetric equivalent models starting from an initial model.

Now we define a modification of this algorithm by using only the row-concatenation nodes in the case (a), the column-concatenation nodes in case (b) or using all nodes in the case (c).

In this way we can obtain a symmetric model of type (a), (b), or (c) starting from an initial model.

Now is very easy to verify if a model is symmetrical by comparing the initial formula, that represent the initial covering model, and the symmetrical formula, that represent the symmetrical covering model. If this two formulas are identical then the initial covering model is symmetrical.

## THE ALGORITHM

***Input:*** A covering model with guillotine restrictions, in a form of a set of rectangular plates $A_i$ for $i = 1, 2, ..., k$, that cover completely a rectangular room.

***Output:*** A set of covering model $V$ that is symmetrical in sense (a), respectively (b) or (c).

**Step 1.** Generate the set $S$ of covering model with guillotine restrictions.

**Step 2.** Put $V = \phi$;
For every $M$ from $S$ do: generate the symmetric covering model $M'$ in sense (a), respectively (b) or (c). Let $F$ and $F'$ the formulas that represent $M$ and $M'$. If $F = F'$ then $V = VUM$.

**Step 3.** If $V = \phi$ then "there is no symmetric model"
else Output $V$.

**Stop.**

## APPLICATIONS

Beside the application we started this paper, we propose to apply this algorithm to cover (to protect) a sport hall for changing the initial destination ( for a rock concert or a exposition).

Of course there are situations when the cover contains a model repeatable in a direction or in another; we can adapt the algorithm (going only on the left or on the right branch of the tree in figure 3) to solve this problem.

Beside the application we started this paper, we propose to apply this algorithm to construct the above insulation of a flat.

Of course there are situations when the cover contains a model repeatable in a direction or in the other; we can adapt the algorithm to solve this problem.

If the room has other form (like "Z" for example) a similar algorithm can be build. We are working on.

We can take different values of $x$ from a database and search for the best solution.

## References


[1] Iacob P., Marinescu D., Luca C. *"Covering a rectangle with rectangular pieces"*, Proceedings - Logistics from a to W: Strategies and Applications, Thessaloniki, Greece, October 18-20, 2001; pp.506-513.

[2] Iacob P., Marinescu D., *"Upon a cutting problem with two optimization criteria"*, Conference on analysis, functional, equations, approximation and convexity Cluj-Napoca, October 15-16, 1999;pp.84-88.

[3] Iacob P., Marinescu D., Luca C., *"A rectangular covering problem"*, Second international conference on Symmetry and Antisymmetry in mathematics, Formal Languages and Computer Science, Brasov, June, 2000, pp.165-174.

[4] Iacob P., Marinescu D., Luca C.,*""L" shape room"*, unpublished paper, 2003.

[5] Marinescu D. - *"Un limbaj pictural pentru un model de croire dreptunghiulara cu restrictii de tip ghilotina"* - Lucrarile Primului Colocviu National de Limbaje, Logica si Lingvistica Matematica - Brasov, 1986, pp. 117 - 125.

[6] Marinescu D. - *"A s-picture language for a cutting-stock model with guillotine restrictions"* - Bulletin of the Transilvania University of Brasov - seria C, Vol XXXIII, 1991 pp 39-45.

[7] Marinescu D., Ciurea E., Tabirca T. *" An Algorithm for the generation of the Symmetric Cutting-Stock Model with Guillotine Restrictions"*, Proceedings of the International Conference "Symmetry and Antisymmetry in Mathematics, Formal Languages and Computer Science", Brasov, 1996, pp 67-70.

[8] Marinescu D., *"Proprieties des matrices des graphes attaches a un model de couture"*, Bulletin of the Transylvania University of Brasov, vol. XXXIV - 1992.

[9] Marinescu D., *"An array language for a two-dimensional cutting-stock problem"*, The third colloquium on logic, language, mathematics linguistics, Brasov 1991

[10] Geofrion A., *Solving bi-criterion Mathematical Programs*, in "Operation Research" 15, no. 1, 39-54, 1967.

Department of Computer Sciences
"Transilvania" University of Brasov, Brasov, Romania
E-mail: iacob.paul@xnet.ro

Department of Computer Sciences
"Transilvania" University of Brasov, Brasov, Romania
E-mail: mdaniela@unitbv.ro

Department of Computer Sciences
"Transilvania" University of Brasov, Brasov, Romania
E-mail: c.luca@info.unitbv.ro