The generating of the cutting-covering receipts using Euclid's algorithm

IACOB PAUL, MARINESCU DANIELA, BAICOIANU ALEXANDRA
Department of Computer Science
Transilvania University of Brașov
500091 Brașov, Str Iuliu Maniu No 50
ROMANIA

Abstract: - We are dealing with a cutting-covering problem defined by us in [4, 5, 6]. We try to model a practical problem, to cover a rectangular room with a material. 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. In the papers [4, 5, 7, 8] we gave the cutting and covering algorithms with a high complexity. In this paper we improve the complexity by a Greedy variant. The proof of the algorithm’s complexity is based on Euclid’s algorithm and Lamé’s theorem.

Key-Words: - Cutting and covering, Euclid’s algorithm, complexity, Greedy method.

1 Introduction
The number of publications in the area of Cutting and Covering (Packing) has increased considerably over the last two decades. The typology of Cutting-Covering problems introduced by Dyckhoff [1] and Sweeney [13] initially provided an excellent instrument for the organization and categorization of existing and new literature.

These problems are in fact NP-complete problems. However, over the years also some deficiencies of this typology have become evident, which created problems in dealing with recent developments and prevented it from being accepted more generally.

If we know the dimensions of the pieces then we are dealing with a classical cutting-stock problem, which can be modeled as a mixed 0-1 programming problem [10]. There are also heuristic models as [2, 12].

Now, if the dimensions of pieces used for covering are unknown then the problem is more complicated. This is the cutting-covering problem defined by us in [4,5], for which we presented algorithms for generating the admissible solutions in [4, 5, 7, 8]. Our objective is to present one polynomial algorithm based on Euclid’s algorithm for compute the greatest common devisor.

2 Problem Formulation
The practical problem we are trying to model is: we have to cover a rectangular room with a cover (linoleum 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 cut the remainder of material (that does not cover the floor), we obtain a new piece of cover and a new piece of floor that have the same properties described by condition c1.

3 Solution
The algorithm we propose is based on the following two properties:
1. If we put the cover on the floor overlapping two adjacent sides of the cover over two adjacent sides of the floor and we cut the remainder of material (that does not cover the floor), we obtain a new piece of cover and a new piece of floor that have the same properties described by condition c1.
2. The cover may be put on the floor in two directions (see the example from fig 1 and 2)

Fig. 1
F

After we laid out the material in one of the directions we face the same issue regarding dimensions smaller than a or b or x or y.

Algorithm 1.

The algorithm (proposed in [5]) is recursively generating a binary tree: if the initial problem is to cover the rectangle of dimensions a and b with a cover of dimensions x and y then it is the root T (a, b, x, y); putting x on a we obtain the right sub-tree with the root T(a-x, b, x, y-b) and putting x on b we obtain the left sub-tree with the root T(a, b-x, x, y-a).

\[ T (a, b, x, y) \]

\[ T (a', b', x', y') \]

\[ T (a'', b'', x'', y'') \]

where a' = a, b' = b - x, x' = x and y' = y - a, if b > x or a' = a - x, b' = b, x' = x - b and y' = y if x > b and a'' = a - x, b'' = b - y, x'' = x - a, y'' = y if a > x

It is obvious that if a = x or b = x we have already a receipt of cutting covering.

As it was shown in [5], 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. We can detect some situations when growing the y we can cover the initial rectangle; between them some are Pareto optimum points.

Let's assume that: \( R_0 = T(a-x, b, x, y-b) \) and \( R_1 = T(a, b-x, x, y-a) \). The zero index means that the material was laid out on the direction of b while the one index value means that material was laid out in the direction of a.

**Theorem 1:** If \( a > x \) and \( b > x \) then \( R_{01} = R_{10} \).

**Proof:**

\[ R_{01} = T(a-x, b-x, x, y-b-a) = T(a-x, b-x, x, y-a-b) = R_{10}. \]

\[ \square \]

**Consequence:** If \( a > x \) and \( b > x \) then it doesn't matter in which of the directions we laid out the material. Now we are able to give a few cutting and covering receipts on \( O(q_a + q_b) \), where \( a = x*q_a + r_a \) si \( b = x*q_b + r_b \).

Now we have the Part 1 of the algorithm.

**Part 1.** Construct the set of the receipts:

<table>
<thead>
<tr>
<th>Nr of receipt</th>
<th>Receipt</th>
<th>Nr of pieces</th>
<th>Waste</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( R_{00} \ldots 0 ) ( q_a+1 ) times</td>
<td>( q_a+1 )</td>
<td>( b*(x-r_a) )</td>
</tr>
<tr>
<td>2</td>
<td>( R_{00} \ldots 0010 ) ( q_a ) times</td>
<td>( q_a+2 )</td>
<td>( (b-x)*(x-r_a) )</td>
</tr>
<tr>
<td>3</td>
<td>( R_{00} \ldots 0110 ) ( q_b ) times</td>
<td>( q_a+3 )</td>
<td>( (b-2x)*(x-r_a) )</td>
</tr>
<tr>
<td>...</td>
<td>( R_{00} \ldots 011111 \ldots 1100 ) ( q_a ) times ( q_b ) times</td>
<td>( q_a+q_b+1 )</td>
<td>( r_b*(x-r_a) )</td>
</tr>
<tr>
<td>( q_b+2 )</td>
<td>( R_{11111111} \ldots 10 ) ( q_b+1 ) times ( q_b+1 ) times</td>
<td>( q_b+1 )</td>
<td>( a*(x-r_b) )</td>
</tr>
<tr>
<td>( q_b+3 )</td>
<td>( R_{00} \ldots 0010 ) ( q_b ) times</td>
<td>( q_b+2 )</td>
<td>( q_b+2 )</td>
</tr>
<tr>
<td>( q_b+4 )</td>
<td>( R_{00} \ldots 0110 ) ( q_b ) times</td>
<td>( q_b+3 )</td>
<td>( (a-2x)*(x-r_b) )</td>
</tr>
<tr>
<td>...</td>
<td>( R_{00} \ldots 011111 \ldots 1100 ) ( q_a ) times ( q_b ) times</td>
<td>( q_a+q_b+1 )</td>
<td>( r_a*(x-r_b) )</td>
</tr>
</tbody>
</table>

It is obvious that \( q_a \) and \( q_b \) receipts have the same number of pieces, meaning that we will choose the one corresponding with min \((r_a, r_b)\). We will proceed in the same manner with the other receipts having the same number of pieces, choosing the one with minimal waste.

Let for example: \( a = 27, b = 16, x = 10 \)

<table>
<thead>
<tr>
<th>Nr</th>
<th>Receipt</th>
<th>Nr</th>
<th>Waste</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( R_{00} \ldots 001111 \ldots 1100 ) ( q_a ) times ( q_b ) times</td>
<td>( q_a+q_b+1 )</td>
<td>( r_a*(x-r_b) )</td>
</tr>
</tbody>
</table>
Receipt of pieces

<table>
<thead>
<tr>
<th>Nr of receipt</th>
<th>Receipt</th>
<th>Nr of pieces</th>
<th>Waste</th>
<th>Figure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R_000_0</td>
<td>3</td>
<td>16*3=48</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>R_001_0</td>
<td>4</td>
<td>6*3=18</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>R_11</td>
<td>2</td>
<td>6*27=162</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>R_101</td>
<td>3</td>
<td>6*17=102</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>R_100_1</td>
<td>4</td>
<td>6*7=42</td>
<td></td>
</tr>
</tbody>
</table>

Taking only the receipts having the same number of pieces and minimal losses we obtain

But applying the first algorithm a better solution can be obtained

\[
\begin{align*}
\text{We will take the algorithm from [6] and modify it in the following way:} \\
\text{We choose from table the optimal decomposition of } \frac{a}{x} \text{ and } \frac{b}{x}. \text{ For each fraction of the decomposition the initial algorithm is applied, but the tree is built in one direction, on the left for the decomposition of } \frac{a}{x}, \text{ respectively on the right for } \frac{b}{x}. \\
\text{By applying it to our example we get:} \\
\frac{a}{x} = \frac{7}{10} - \frac{1}{2} + \frac{1}{5} \quad \text{and} \quad \frac{b}{x} = \frac{6}{10} - \frac{3}{5} \\
\text{The corresponding receipts will be:}
\end{align*}
\]

<table>
<thead>
<tr>
<th>Nr of Receipt</th>
<th>Receipt</th>
<th>Nr of pieces</th>
<th>Waste</th>
<th>Figure</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>R_4</td>
<td>2+2+5=9</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>R_5</td>
<td>1+4=5</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

In the cases of \(a < x\) and \(b < x\) we use the algorithm described in [8]. But this algorithm can also be furthermore optimized. Let us take the following situation when \(\frac{a}{x} = \frac{2}{3}\), then the solution from [6] is to divide the cover and the surface into \(2*3=6\) pieces. The receipt is obvious.

Only the receipt number 5 remains because for the same waste (in our case 0) results in fewer pieces.
Theorem 2: The construction in one direction of the cutting-covering receipt is a geometrical construction of Euclid’s algorithm for the numbers \(a\) and \(x\) where the fraction \(\frac{a}{x}\) is irreducible and \(a < x\).

Proof: Let us assume the situation of \(T(a, b, x)\).
Computing \(y = \frac{a \cdot b}{x}\) the length of the cover material, and \(b' = y \cdot \frac{b}{a} = \frac{b}{x}\). The surface to cover and the material are split into rectangles with dimensions 1 on the direction of \(a\) and \(b'\) on the direction of \(b\) (the material will be laid out with \(x\) on \(a\)).

The obtained rectangles are aligned in the same direction on the surface to be covered and on the covering material. We can cut out these rectangles from the covering material and lay them onto the surface to be covered without any rotation which would lead us to a receipt of \(a \cdot x\) pieces and 0 waste according with [6].

Another solution would be the construction of a binary tree only on the right side:

\[
T(a, b, x) \\
T(a, b-y, x-a) \\
T(a, b-2y, x-2a) \\
\ldots \\
T(a, b-q_1y, r_1) \quad \text{where } x = a \cdot q_1 + r_1 \\
T(a-r_1, b', r_1) \\
T(a-2r_1, b', r_1) \\
\ldots \\
T(r_{n-1}, 0', r_n)
\]

The successive dimensions of the covering material and the surface to be covered are: \(x, a, r_1, r_2, \ldots, r_n, 0\), the remainders of the divisions of Euclid’s algorithm.

The number of pieces \(S = \sum_{i=1}^{n} q_i\) will be the sum of the successive quotients from the same Euclid’s algorithm.

□

Consequence: If \(\frac{a}{x}\) is irreducible and \(a < x\) then

\[
S = \sum_{i=1}^{n} q_i \leq a \cdot x + 1 - a^2.
\]

Algorithm 2.

Part 1. Use part1 from the algorithm 1.
Part 2. Look in the table described in [7] to find the optimal decomposition of the fraction \(\frac{a}{x}\).
Part 3. Apply Algorithm 1 only on the right side of the binary tree for each fraction.

We are now able to conclude the complexity of the new algorithm based on the number of divisions in Euclid’s algorithm.

Lamé [11] had found that the complexity is less than \(5 \times \) (the number of \(a\)’s digits).

So the complexity of the new algorithm is

\[
O(5 \times \text{Max}(\text{NumberOfDigits}(r_a), \text{NumberOfDigits}(r_b))).
\]

4 Conclusion

Cutting and covering problems have many applications in production processes in paper [9], glass, metal and timber cutting industries. There are also unconventional applications like [3], where covering model is used in the formalization of pattern recognition problems.

Our cutting-covering problem is also important for the situations in which we don’t know the dimensions of the pieces for cutting and covering. For this kind of problems it is possible to use the algorithm 2 which is polynomial and it uses a small amount of memory for intermediate data storage. It follows that it is faster even for big integers as input data.

Acknowledgements: The research was supported by the Transilvania University of Brasov and in the case of the first two authors, it was also supported by the
References:


