Revista
Torreón Universitario
www.faremcarazo.unan.edu.ni - ISSN 2313-7215 (electronic edition)
Year 4 | No. 11 | p. 91 - p. 102 | october 2015 - january 2016

# Design of a heuristic to solve the problem of Two-dimensional Rectangular Cut by the Method of Guillotine

**Jairo José Flores Morales / Jazcar Bravo Rivas**
Researchers Teachers
UNAN-MANAGUA, FAREM-CHONTALES
*jairofmdjmix@yahoo.com / jbravo@unan.edu.ni*

**Michel Roberto Traña Tablada**
Programmer analyst
UNAN-MANAGUA, FAREM-CHONTALES
*mtraatabladaa94@gmail.com*

**Keywords:** *Cutting problem, heuristic, guillotine method, algorithm and programming.*

## ABSTRACT

This paper focuses on the development of a heuristic to solve the problem efficiently, of two-dimensional cutting of plates using the method of the guillotine, offering a cutting plan that minimizes the number of plates to be used; therefore, it can satisfy the demand for each type of piece. Such heuristic has been developed in two phases, the first obtained an initial solution and in the second solution the first solution is improved. This heuristic was tested by means of an instance that allows you to see the improved solution algorithm with three conditions: length, width and demand. Heuristics was worked in C++ as part of a work order module PhD in applied mathematics, which seeks to resolve many applications of our own field of study.

## INTRODUCTION

In many problems of our daily life related with industrial processes it arises some problems that operative investigation encompass as problems of cutting and packaging. On one hand, many manufacturing processes produce panels or sheets of large wood, metal, paper, plastic or glass which are subsequently cut to be into smaller pieces to fit the needs of customers. These problems, from the point of view of complexity, continue attracting the interest of researchers who keep developing accurate and heuristic algorithms with increasing efficiency.

The design of the pieces constitutes the pattern in the cutting mold. In general, pattern are to be octagonal. The pattern of the guillotine seeks to cut sheets into rectangles of specified sizes, using only the cuttings being all the way of each sheet (Morabito and Morales, 1998). Using guillotinables cuts means that all cuts must be made from one side of the rectangular material to the other and also it must be parallel to one axis (Figure 1 (a)) patterns. The complexity of guillotinable patterns depends on the number of changes in cutting direction (stages) and the number of parallel cuts per stage (Armas, 2011).

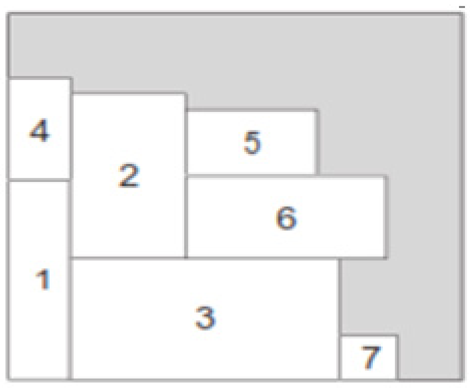


*Figure 1 (a) Guillotine Cut.*

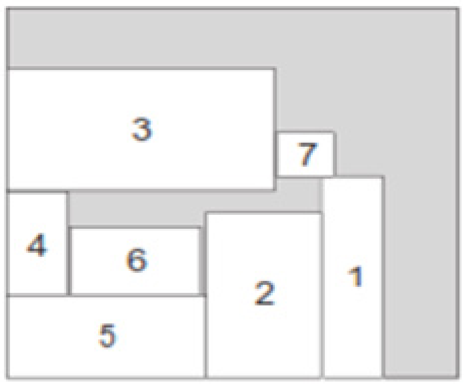


*Figure 1 (b) No guillotine Cut*

It is worth mentioning that a problem of packaging is made of by a set of rectangular pieces or plates with known length and it is wanted to distribute them in plates of shorter length and size than the piece that will be used to make rectangular cuts, so the wasted area is minimized. An alternative way of interpreting the problem assumes that you have a rectangular object from which you have to get a set of parts making cuts perpendicular to the axes.

It should be considered that, the length, the width and the number of the plates that we count on are known, intending therefore, to use them, in those pieces which maximize the objective function considered. The length and width of the board is known. It tries, therefore, to distribute in the object distribute those pieces that optimize the objective function considered. This situation is addressed in this article.

## PROBLEM DESCRIPTION

Some problems to be solved by the companies dedicated to cutting rectangular pieces into smaller pieces, according to the specifications of their customers, are precisely the types of cuts to be made, in which the gain is maximized, minimize costs and without forgetting the reduction of waste or surplus in each cut sheet.

This situation causes the search of heuristics able to meet customer expectations and therefore the owner of the company.

If it takes as a reference the need to meet the demand for a builder of metal banners urgently requesting the following specifications.

(See *Table 1. Considerations of the metal parts required.* on next page)

Where the length and width are in centimeters (cm) and demand represents the amount of parts they require. If to complete the demand, the company will use sheets with the following dimensions: (See *Table 2. The length of the plate where the rectangular pieces will be taken from*. on next page)

Table 1. Considerations of the metal parts required

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Large** | **121** | **254** | **261** | **451** | **124** | **351** | **351** | **185** | **319** | **261** | **451** | **219** |
| **Width** | 354 | 321 | 308 | 187 | 237 | 402 | 165 | 201 | 201 | 308 | 187 | 315 |
| **Demand** | 3 | 2 | 4 | 3 | 7 | 3 | 3 | 4 | 1 | 4 | 3 | 4 |
| **Large** | **386** | **183** | **212** | **354** | **250** | **222** | **309** | **273** | **325** | **344** | **380** | **301** |
| **Width** | 304 | 329 | 214 | 254 | 149 | 237 | 227 | 209 | 206 | 421 | 497 | 228 |
| **Demand** | 3 | 2 | 5 | 3 | 4 | 3 | 34 | 4 | 2 | 4 | 2 | 2 |
| **Large** | **261** | **194** | **175** | **256** | **183** | **282** | **326** | **268** | **384** | **248** | **295** | **315** |
| **Width** | 409 | 206 | 300 | 244 | 342 | 231 | 147 | 237 | 362 | 363 | 368 | 408 |
| **Demand** | 2 | 3 | 2 | 4 | 2 | 1 | 2 | 2 | 2 | 1 | 3 | 2 |
| **Large** | **411** | **189** | **261** | **181** | **256** | **315** | **186** | **167** | **500** | **313** | **492** | **251** |
| **Width** | 246 | 576 | 314 | 382 | 367 | 464 | 561 | 354 | 267 | 471 | 321 | 260 |
| **Demand** | 3 | 3 | 3 | 2 | 1 | 3 | 2 | 2 | 2 | 1 | 2 | 3 |
| **Large** | **282** | **429** | **198** | **350** | | | | | | | | |
| **Width** | 364 | 264 | 254 | 241 | | | | | | | | |
| **Demand** | 1 | 2 | 1 | 3 | | | | | | | | |

Table 2. The length of the plate where the rectangular pieces will be taken from

| Large | Width | Cost per Sheet |
|---|---|---|
| 977 | 953 | 50 |

To make the required cuts it is necessary to ask, how many sheets of 977 cm x 953 cm will I need? what will be the cost for all sheets used? it is useful guillotine cutting in this case? Do I guarantee the optimization of the material? is it possible to improve the cuts by entering new generic codes be improved? Next, it is explained the development of heuristic to determine the number of pieces to be cut, for that, all parts with their length, width and the requested action.

## MATERIALS AND METHODS

This heuristic procedure solves the problem of optimization Cutting Stock Problem (CSP) through an intuitive approach, in which the problem structure is used smartly to get a good solution. It should be highlighted that the heuristic methods are limited to provide a good but not necessarily optimal solution and this is precisely what this article is about.

From the analysis and the requirements of the problem it arises the following:

$R = (L, W)$ Rectangular sheet from large and width

$S= \{(l_1 ,w_1), \ldots (l_i ,w_i) ,\ldots ,(l_m ,w_m)\}$ Set of *m* smaller rectangular pieces.

**Some restrictions:**

$0 \leq x_i \leq d_i$ The number of cut pieces must be greater or equal to zero and less than or equal to demand.

(Some $d_i < \lfloor L / l_i \rfloor \lfloor W / w_i \rfloor$) Delimitation of the problem, a required demand is provided.

$(v_i = l_i w_i, \forall i)$ Value of the parts if there is no weight.

$b1, b2, \ldots\ldots, bm$ Copy of each rectangle (cardinality)

The objective is to build a pattern of valid court to $R$, which maximizes the sum of the value of the cut pieces using no more than $b_1$ copy of each rectangle $S_i$ in the pattern, this pattern must meet:

- All dimensions $(l_i, w_i)$ to all $i=1, 2, \ldots, m$ are whole along the axes $x$ and $y$.
- The orientation of parts is considered fixed, so $[(l_i, w_i) \neq (w_i, l_i)]$.
- The cuts are orthogonal with valid sequence cutting guillotine.

## CONSTRUCTIVE ALGORITHM

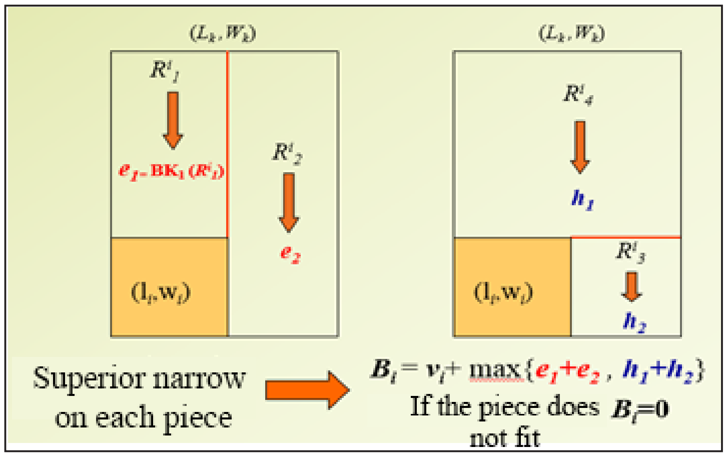


*Figure 2. Greedy algorithm proposed by Martello and Toth (1990)*

The problem is solved through narrowed variables:

$$\text{Upper dimension: } BK_1(R) = Max \sum_{i \in S^*} v_i x_i \quad \text{where } S^* = \{i / l_i \leq L, w_i \leq W\}$$

$$\text{s.t. } \sum_{i \in S^*} s_i x_i \leq LW$$

$$0 \leq x_i \leq \min\{d_i - n_i, \lfloor L / l_i \rfloor \lfloor W / w_i \rfloor\}, i=1, \ldots, m$$

$$B_j = \max\{B_i, i=1 \ldots m\}$$

If $Bj=0$, the considered rectangle is waste.

$$\text{Upper dimension: } BK_2(R) = Max \sum_{i \in S^*} v_i x_i \quad \text{where } S^* = \{i / l_i \leq L, w_i \leq W\}$$

$$\text{s.t. } \sum_{i \in S^*} s_i x_i \leq LW$$

$$0 \leq xi \leq min\ \{d_i - n_i,\ max\{ \lfloor L_1 / l_i \rfloor \lfloor W_1 / w_i \rfloor + \lfloor L_2 / l_i \rfloor \lfloor W_2 / w_i \rfloor, \lfloor L_3 / l_i \rfloor \lfloor W_3 / w_i \rfloor + \lfloor L_4 / l_i \rfloor \lfloor W_4 / w_i \rfloor \}\}$$

$$i=2,\dots,m$$

Where $R_j = (L_j, W_j),\ j=1,2,3,4$

Considering the above, Microsoft Visual Studio is used in its 2012 version, to develop generic codes that solve the CSP, first characterizing the problem to its later analysis phase where pseudo code are made.

The requirement of Table 1, will be the input of the heuristic, which starts with an algorithm to place the pieces respecting the length and width, however when unable to continue to perform the process in the first film, it will do in the second one and so on until all the pieces placed in some sheet. The total number of plates will be the response of the algorithm.

**Stage 1. Bookshelves construction**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Threading.Tasks;
using c = System.Console;

namespace Problema_de_Corte // Nombre general del programa
{
class Program   // Lenguaje orientado a objetos, clase o plantilla
  {
    static int[] Bi = new int[60];
    static int[] Hi = new int[60];
    static int[] Vi = new int[60];          Global variables, vector of the whole
    static int[] Di = new int[60];          type, with maximum cut of 60.
    static int Bl; // Base de Área General  Parallel vectors.
    static int Hl; // Altura Lámina
    static int i = 0; // Variable para controlar la cantidad de cortes leídos
static void Main (string [ ] args)   // Función principal
{
```

## Stage 2. Development

```
// ............. variables primarias....................//
var lector = new StreamReader(File.OpenRead(@"C:\BD.csv")); //variable para leer del archivo

// Lectura de los datos desde el archivo hacia los vectores
// lectura de datos generales
c.WriteLine("ingrese la Base de la lámina");
Bl = int.Parse(c.ReadLine());
c.WriteLine("ingrese la Altura de la lámina");
Hl = int.Parse(c.ReadLine());
```

Enter input of the base and high in the console.

```
while (!lector.EndOfStream) // Mientras no sea el final del archivo (Estructura iterativa)
 {
   var line = lector.ReadLine();  // lee línea a línea
   varvalues = line.Split(';');       // separa los valores en lista
   Bi[i] = int.Parse(values[0]);   // vector para base de los cortes
   Hi[i] = int.Parse(values[1]);  // vector para altura de cortes
   Vi[i] = int.Parse(values[2]);   // vector para valor de cortes
   Di[i] = int.Parse(values[3]);   // vector para la demanda de cortes
   i++;
  }
```

To read the document CSV and do iterations.

## Stage 3. Iteration cycle (j) and cutting (i)

**Process to determine the cuts both horizontals and verticals**

```
for (int j = 0; j < i; j++) {
   //.............. vertical y horizontal .................... //
      c.BackgroundColor = ConsoleColor.Blue;
      c.WriteLine("****SEGMENTOS DE CORTES ({0},{1})****", Bi[j], Hi[j]);
      c.BackgroundColor = ConsoleColor.Black;
       int H1 = Hl - Hi[j];
       c.ForegroundColor = ConsoleColor.Cyan; c.WriteLine("H1 ({0},{1})", Bl, H1);
       c.ForegroundColor = ConsoleColor.White;
// llamar procedimiento de calculos para ésta área. pasar como parametros Bl, H1 y arreglos
       Evaluar(Bl, H1);
       int H2 = Bl - Bi[j];
       c.ForegroundColor = ConsoleColor.Cyan; c.WriteLine("H2 ({0},{1})", H2, Hi[j]);
```

```
        c.ForegroundColor = ConsoleColor.White;
        Evaluar(H2, Hi[j]);
        int V3 = Hl - Hi[j];
        c.ForegroundColor = ConsoleColor.Cyan; c.WriteLine("V1 ({0},{1})", Bi[j], V3);
        c.ForegroundColor = ConsoleColor.White;
        Evaluar(Bi[j], V3);
        int V4 = Bl - Bi[j];
       c.ForegroundColor = ConsoleColor.Cyan; c.WriteLine("V2 ({0},{1})", V4, Hl);
       c.ForegroundColor = ConsoleColor.White;
       Evaluar (V4, Hl);
  }
```

**Stage 4. Restrictions**

```
    Console.ReadKey();
        }
    static void Evaluar(int B, int H) {
    int[] tempAr = new int[i];
    int AreaTemporal = B * H;
    int CostoMaximo = 0;
    int CantidadMinima = 0;
    for (int j = 0; j < i; j++) {
      if (B >= Bi[j] && H >= Hi[j]) {
        if (Di[j] > ((B / Bi[j]) * (H / Hi[j])))
        { tempAr[j] = (B / Bi[j]) * (H / Hi[j]);}
        else
         { tempAr[j] = Di[j];}
     } }
    for (int j = 0; j < i; j++) {
     if (B >= Bi[j] && H >= Hi[j]) {
     c.WriteLine("El mínimo de ("+ Bi[j] + "," + Hi[j] + ") es "+ tempAr[j] + "Z = "+ (tempAr[j] * Vi[j]));
        if ((AreaTemporal / (Bi[j] * Hi[j])) > Di[j])
        { CantidadMinima = Di[j];}
          else { CantidadMinima = (AreaTemporal / (Bi[j] * Hi[j])); }
         if (AreaTemporal> 0)
        { CortesSucesivos(Bi[j], Hi[j], tempAr[j], Vi[j], CostoMaximo, AreaTemporal, CantidadMinima);
```

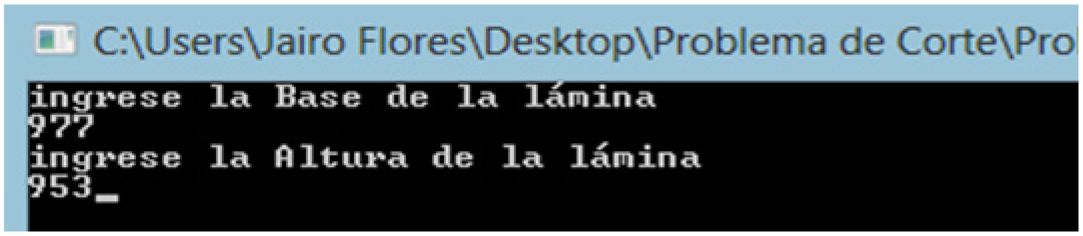
```
if (tempAr[j] >CantidadMinima){                                      Improvement
  AreaTemporal = AreaTemporal - (Bi[j] * Hi[j] * CantidadMinima);     pseudo code.
  CostoMaximo = CostoMaximo + (CantidadMinima * Vi[j]); }
    else {
      AreaTemporal = AreaTemporal - (Bi[j] * Hi[j] * tempAr[j]);
      CostoMaximo = CostoMaximo + (tempAr[j] * Vi[j]);}
         } }
    }
c.ForegroundColor = ConsoleColor.Red;
c.WriteLine("Función Objetivo tomando primero (" + B + ", " + H + "): " + CostoMaximo);
c.ForegroundColor = ConsoleColor.White;
}


static void CortesSucesivos(int B, int H, int C1, intCostAr, int CosteMaximo, intAreaMax, int C2) {
  intTempZ = (AreaMax);
  if (TempZ>= 0)
   { int minC = 0;
     if (C1 > C2){ minC = C2; }
       else { minC = C1; }
```

**Stage 5. Answers to get**

```
TempZ = (AreaMax - (minC * B * H));
c.WriteLine("X1 <= min {" + C1 + ", [" + AreaMax + " / " + (B * H) + "]} = " + minC);
c.WriteLine("Z = " + (CosteMaximo) + " + (" + minC + ")(" + CostAr + ") = " + (CosteMaximo +
(minC * CostAr)));
c.WriteLine("Área = " + AreaMax + " - (" + minC + ")(" + (B * H) + ") = " + TempZ);
c.WriteLine();
        } }  } }
```

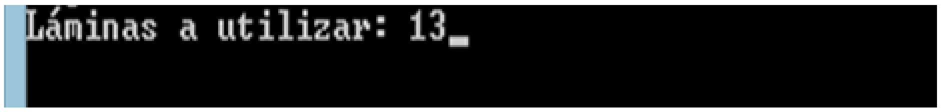
**Console Enter**



```
ingrese la Base de la lámina
977
ingrese la Altura de la lámina
953_
```

*Figure 3. Input of base and height of the sheet on the console.*

**Obtained answer**



```
Láminas a utilizar: 13_
```

*Figure 4. Answer obtained*

**Improved Phase**

The improvement phase allows knowing the necessary plates needed to meet customer`s demand. Thirteen metal plates of 977cm x 953cm will be used. If the cost per unit is $50, then you will have a total cost of $650 for 13 rectangular sheets to be used.

On the other hand there are respective iterations between each $(l_i, w_i)$ of the set $S_i$ of the set requirements, allowing to decide if the convenient initial cut is whether the horizontal or vertical.

The aesthetics of the console is also improved.

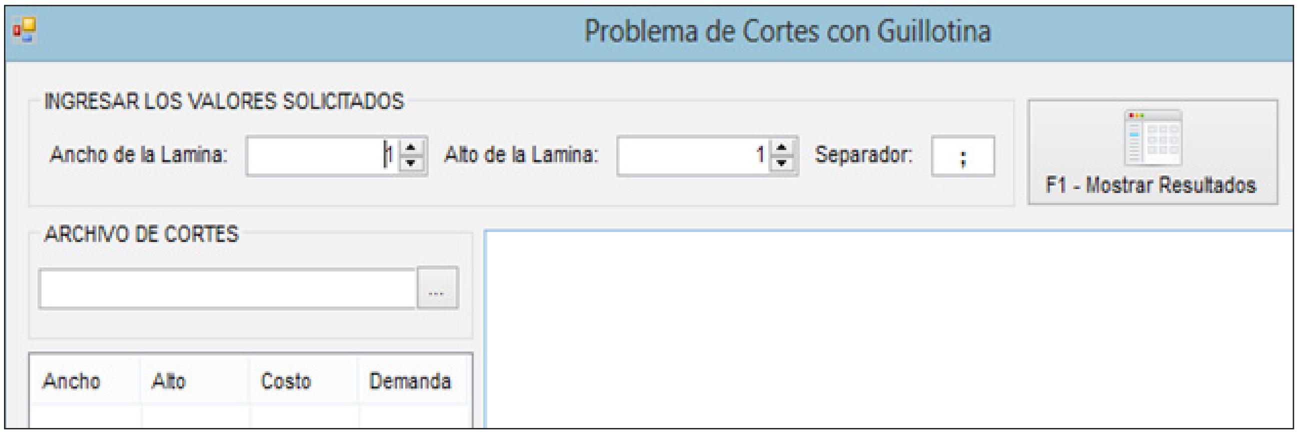
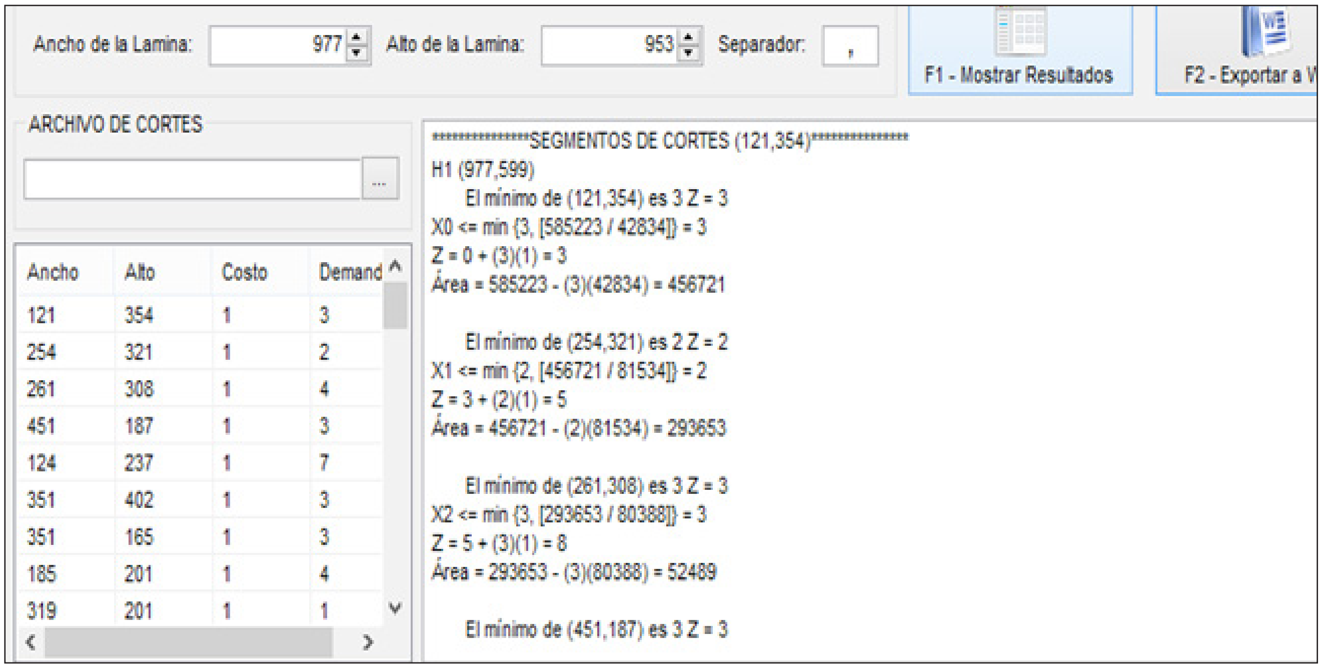


*Figure 5. Improved console input.*

This improvement allows us to obtain the accurate cuts of the initial problems reflected in Table 1, giving the order to open the cut documents that is in the CSV format.



| Ancho | Alto | Costo | Demand |
|---|---|---|---|
| 121 | 354 | 1 | 3 |
| 254 | 321 | 1 | 2 |
| 261 | 308 | 1 | 4 |
| 451 | 187 | 1 | 3 |
| 124 | 237 | 1 | 7 |
| 351 | 402 | 1 | 3 |
| 351 | 165 | 1 | 3 |
| 185 | 201 | 1 | 4 |
| 319 | 201 | 1 | 1 |

```
***************SEGMENTOS DE CORTES (121,354)***************
H1 (977,599)
    El mínimo de (121,354) es 3 Z = 3
X0 <= min {3, [585223 / 42834]} = 3
Z = 0 + (3)(1) = 3
Área = 585223 - (3)(42834) = 456721

    El mínimo de (254,321) es 2 Z = 2
X1 <= min {2, [456721 / 81534]} = 2
Z = 3 + (2)(1) = 5
Área = 456721 - (2)(81534) = 293653

    El mínimo de (261,308) es 3 Z = 3
X2 <= min {3, [293653 / 80388]} = 3
Z = 5 + (3)(1) = 8
Área = 293653 - (3)(80388) = 52489

    El mínimo de (451,187) es 3 Z = 3
```

*Figure 6. Response to the problem offered cutting input 1, time required 32 seconds.*

## COMPUTATIONAL RESULTS

The answer given computationally is better analyzed by copying it to a Word file, where it is appreciated for example, that taking the first cut (121.354) in the sheet (977.953) must be done vertically to get a better optimization of the area of the sheet than if it is done horizontally.

*Table 3. Computacional Responses*

| $l_i, w_i$ | Amount achieved in the remaining area $H_1$ | Cost |
|---|---|---|
| 121,354 | 3 | Because the input does not have cost it can be use $v_i = l_i w_i$ but it was decided to use the 1 value, it does not affect what the problem ask, as well as, it would work perfectly if it gives the cost for each $l_i, w_i$ getting the objective function. |
| 254,321 | 2 | |
| 261,308 | 3 | |
| 124,237 | 1 | |
| The rest | 0 | |
| **Objective Function** | **9** | |

| $l_i, w_i$ | Amount achieved in the remaining area $H_2$ |
|---|---|
| 121,354 | 3 |
| 254,321 | 1 |
| The rest | 0 |
| **Objective Function** | **4** |

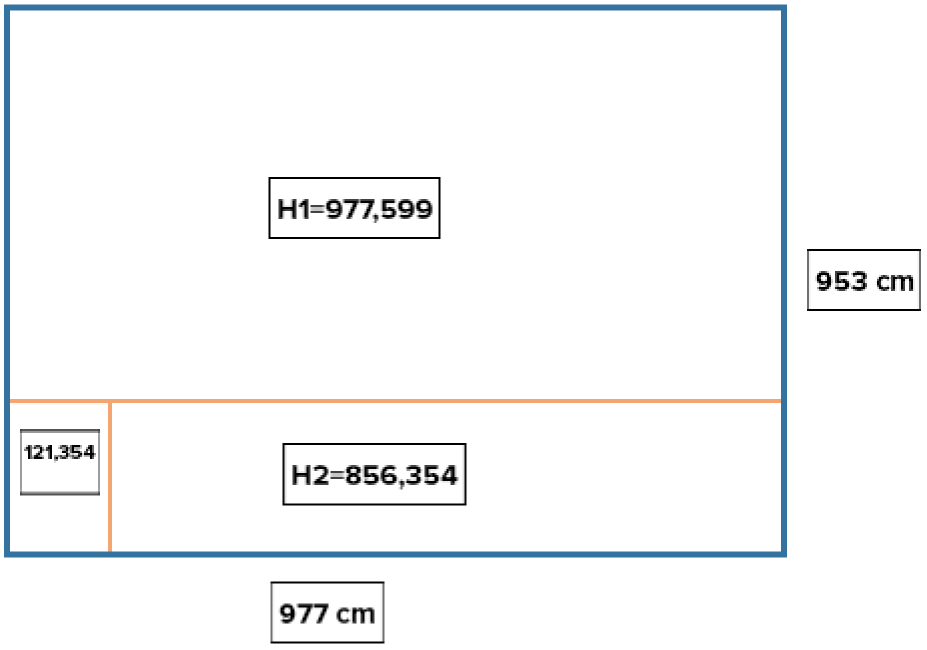


*Figure 7. Cut sheet horizontally.*

*Table 4. Computacional Responses*

| $l_i, w_i$ | Amount fitting in the remaining area $V_1$ |
|---|---|
| 121,354 | 1 |
| The rest | 0 |
| **Objetive function** | **1** |

| $l_i, w_i$ | Amount fitting in the remaining area $V_2$ |
|---|---|
| 121,354 | 3 |
| 254,321 | 2 |
| 261,308 | 4 |
| 451,187 | 2 |
| 124,237 | 1 |
| The rest | 0 |
| **Objetive function** | **12** |

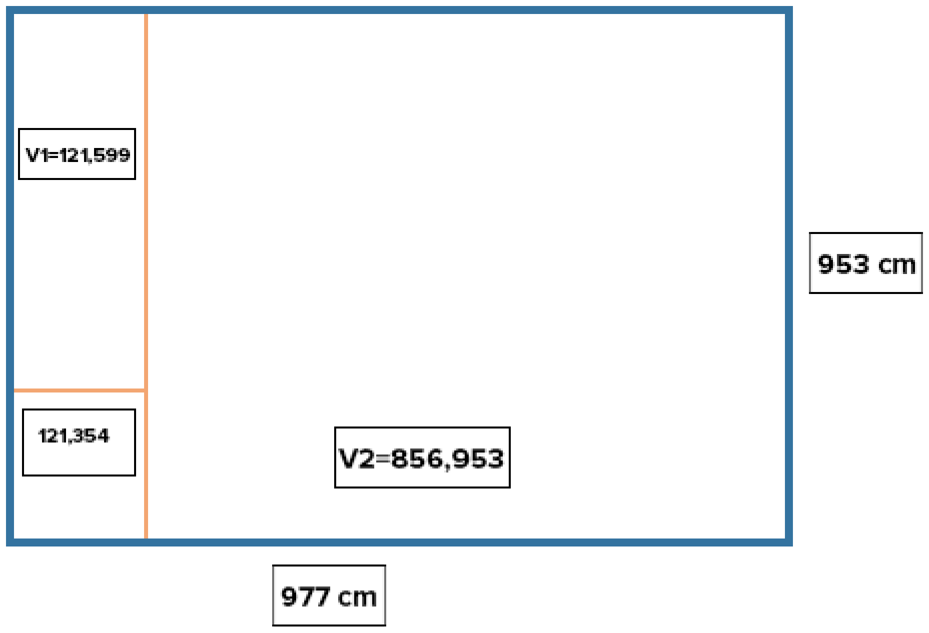


*Figure 8. Sheet cut vertically.*

These answers provide a clear picture of the orthogonal cuts needed in the main sheet to ensure greater optimization area thereof. It also provides the different solutions by each sheet wanted in the problem.

## CONCLUSIONS

This work provides a heuristic able to solve dimensional cutting troubleshoot, generating responses that facilitate the accommodation of the required rectangles characteristic of required demand.

On the other hand, the codes in C are provided, for enthusiasts who study optimization problems such as that found in this article are provided.

The algorithm allows the selection of a cutting direction (horizontal vs. vertical), to try to accommodate additional parts, it also offers the costs generated by each cut done. Results can be improved using other types of coding, or better yet, by making the program offer images of the cuts by guillotine in each selected sheet, as specialized software does such as Smart2DCutting, Cut Login, CutMaster2D, Cut Optimization pro and Cutting3.

It is considered that the objective of the work is accomplished, which was to "develop a computer program that provides an optimal solution to Cutting Stock Problem by the instance offered" and most importantly, to understand the range of problems that can be solved through operations research.

## REFERENCES

ARMAS, J. (2011) *Problemas de corte: métodos exactos y aproximados para formulaciones mono y multi-objetivo.* Serie tesis Doctorales. San Cristóbal de la Laguna: SPUDL.

MARTELLO, S. Y TOTH, P. (1990) *Knapsack Problems: Algorithms and Computer Implementations.* Wiley: New York.

MORABITO, R. Y MORALES, A. (1999) *Errata 'A simple and effective recursive procedure for the manufacturer's pallet loading problem.* Journal of the Operational Research Society, 50:876.

MORENO, F. Y JIMENEZ, J. (2001) *Una aproximación al problema del corte en 2 dimensiones con el algoritmo de recocido simulado.* Congreso Nacional de Estadística e Investigación Operativa. Úbeda.

PARREÑO, F. Y ÁLVAREZ, O. (2004) *Algoritmos heurísticos y exactos para problemas de corte no guillotina en dos dimensiones.* Universidad de Valencia. Disponible en: http://goo.gl/Iw4MOr

TEODORO, A. (2003). *Un problema de corte Bidimensional utilizando un método de columnas.* Tesis de maestría. Instituto de computación, Universidad Estatal de Campinas. São Paulo.