Revista
Torreón Universitario
www.faremcarazo.unan.edu.ni - ISSN 2313-7215 (versión electrónica)
Año 4 | No. 11 | pág. 16 - pág. 27 | octubre 2015 - enero 2016

Diseño de una heurística para resolver el problema de Corte Bidimensional Rectangular por el Método de Guillotina

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

Michel Roberto Traña Tablada
Analista Programador
UNAN-MANAGUA, FAREM-CHONTALES
mtraatabladaa94@gmail.com

Palabras clave: *Problema de corte, heurística, método de guillotina, algoritmo, programación.*

RESUMEN

El presente trabajo se enfoca en el desarrollo de una heurística que resuelva eficientemente el problema de corte bidimensional de placas aplicando el método de la guillotina, ofreciendo un plan de corte que minimice el número de placas a utilizar, de tal forma que satisfaga la demanda por cada tipo de pieza. Dicha heurística ha sido elaborada en dos fases, la primera obtiene una solución inicial, y en la segunda se mejora la solución obtenida en la primera. Esta heurística fue probada por medio de una instancia que permite ver la solución mejorada del algoritmo con tres condiciones: largo, ancho y demanda. La heurística fue trabajada en C++ como parte de un trabajo de fin de módulo del Doctorado en Matemática Aplicada, la cual busca resolver diversas aplicaciones propias de nuestro campo de estudio.

INTRODUCCIÓN

En muchas aplicaciones de nuestra vida relacionadas con los procesos industriales surgen problemas que la Investigación Operativa engloba bajo el título de Problemas de Corte y Empaquetamiento. Por una parte, muchos procesos de fabricación producen tableros o láminas de grandes dimensiones de madera, metal, papel, plástico o vidrio que luego han de ser cortados en piezas más pequeñas para ajustarse a las necesidades de los clientes. Estos problemas, desde el punto de vista de la complejidad, siguen atrayendo el interés de los investigadores que continúan desarrollando algoritmos exactos y heurísticos cada vez con mayor eficiencia.

La configuración de las piezas constituye el patrón en el tablero en el que se cortan. En general, los patrones serán ortogonales. El patrón de guillotina busca cortar láminas en rectángulos de

tamaños especificados, utilizando sólo los cortes que siguen todo el camino a través de cada hoja (Morabito y Morales, 1998). El uso de patrones guillotinables significa que cualquier corte debe realizarse desde un lado del material rectangular hasta el otro y además, debe ser paralelo a uno de los ejes (Figura 1 (a)). La complejidad de los patrones guillotinables depende del número de cambios de dirección de corte (etapas) y del número de cortes paralelos por etapa (Armas, 2011)

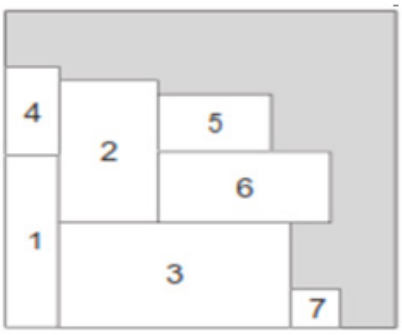

Figura 1(a) Corte guillotina.

Figura 1(b) Corte no guillotina.

Cabe mencionar, que en un problema de empaquetado de rectángulos se dispone de un conjunto de piezas rectangulares o placas con longitudes conocidas y se desean distribuir estas en placas de menor longitud y tamaño que la pieza que será utilizada para realizar los cortes rectangulares, de forma que se minimice el área desperdiciada. Una forma alternativa de interpretar el problema supone que se dispone de un objeto rectangular desde el que hay que obtener un conjunto de piezas realizando cortes perpendiculares a los ejes.

Se debe considerar que tanto el largo, el ancho de la placa y el número de ellas que contamos son conocidos, pretendiendo por tanto, distribuir con ellas, aquellas piezas que optimicen la función objetiva considerada. El largo y el ancho de la paleta son conocidos. Se pretende, por tanto, distribuir en el objeto aquellas piezas que optimicen la función objetiva considerada. Esta situación es la que se aborda en este artículo.

DESCRIPCIÓN DEL PROBLEMA

Algunos problemas que deben resolver las empresas que se dedican al negocio de cortar piezas rectangulares en piezas más pequeñas, acorde a las especificaciones de sus clientes, son precisamente los tipos de cortes a realizar, en la cual se maximice la ganancia, minimice los costos y sin olvidar la reducción de los desperdicios o sobrantes en cada lámina cortada.

Esta situación, provoca la búsqueda de heurísticas capaces de cumplir con las expectativas del cliente y por ende, del dueño de la empresa.

Si se toma como referencia la necesidad de suplir la demanda de un constructor de anuncios publicitarios metálicos que solicita con urgencia las siguientes especificaciones:

(Ver *Tabla 1. Consideraciones de las piezas de metal requeridas.* en página siguiente)

En donde el largo y el ancho están en centímetros (cm) y la demanda representa las cantidades de piezas que solicitan. Si para suplir la demanda, la empresa utilizará láminas con las siguientes dimensiones. (Ver *Tabla 2. Longitud de la placa de la que se sacarán las piezas rectangulares.* en página siguente)

Tabla 1. Consideraciones de las piezas de metal requeridas

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

Tabla 2. Longitud de la placa de la que se sacarán las piezas rectangulares

Largo	Ancho	Costo por lámina
977	953	50

Para efectuar los cortes requeridos se hace necesario preguntar primero, ¿cuántas láminas de 977 cm x 953 cm voy a necesitar?, ¿cuál será el costo por todas las láminas utilizadas?, ¿es útil el corte por guillotina en este caso?, ¿garantizo la optimización del material?, ¿se puede mejorar los cortes ingresando nuevos códigos genéricos? A continuación se explica el desarrollo de la heurística para determinar el número de piezas a cortar, para eso se considerarán todas las piezas con su largo, ancho y la demanda solicitada.

MATERIALES Y MÉTODOS

Este procedimiento heurístico resuelve el problema de optimización Cutting Stock Problem (CSP) mediante una aproximación intuitiva, en la que la estructura del problema se utiliza de forma inteligente para obtener una buena solución. Se debe hacer énfasis, que los métodos heurísticos se limitan a proporcionar una buena solución no necesariamente óptima y precisamente esto es lo que se trata en este artículo.

A partir del análisis y de los requerimientos que forman el problema se plantea lo siguiente:

$R = (L, W)$ Lámina rectangular de largo L y de ancho W.

$S = \{(l_1, w_1), \ldots (l_i, w_i), \ldots, (l_m, w_m)\}$ Conjunto de *m* piezas rectangulares más pequeñas.

Algunas restricciones:

$0 \leq x_i \leq d_i$ El número de piezas cortadas debe ser mayor o igual a cero y menor o igual a la demanda.

(Algún $d_i < \lfloor L / l_i \rfloor \lfloor W / w_i \rfloor$) Acotamiento del problema, pues se brinda una demanda requerida.

$(v_i = l_i w_i, \forall i)$ valor de las piezas si no hay peso.

$b1, b2, \ldots\ldots, bm$ Réplicas de cada rectángulo (cardinalidad)

El objetivo es construir un patrón de corte válido para R, que maximice la suma del valor de las piezas cortadas usando no más que b_1 réplicas de cada rectángulo S_i en el patrón, este patrón debe cumplir con:

- Todas las dimensiones (l_i, w_i) para todo $i=1, 2, . , m$ son enteros a lo largo del eje x y el eje y.
- La orientación de las piezas se considera fija, entonces $[(l_i, w_i) \neq (w_i, l_i)]$.
- Los cortes serán ortogonales, con secuencia válida de corte por guillotina.

ALGORITMO CONSTRUCTIVO

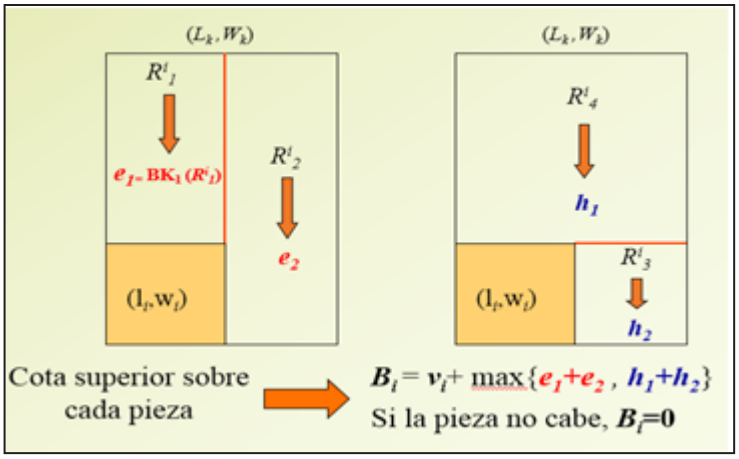

Figura 2. Algoritmo Greedy propuesto por Martello y Toth (1990)

El problema se resuelve *con variables acotadas:*

$$\textit{Cota superior: } BK_1(R) = Max \sum_{i \in S^*} v_i x_i \quad \textit{donde } S^* = \{i / l_i \leq L, w_i \leq W\}$$

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

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

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

Si $B_j=0$, el rectángulo considerado es desperdicio

$$\textit{Cota superior: } BK_2(R) = Max \sum_{i \in S^*} v_i x_i \quad \textit{donde } S^* = \{i / l_i \leq L, w_i \leq W\}$$

$$s.t. \sum s_i x_i \leq LW$$

$$i \in S^*$$

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

$$i=2,\ldots,m$$

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

Teniendo en cuenta lo antes señalado, se utiliza Microsoft Visual Studio en su versión 2012, para elaborar los códigos genéricos que solucionen el CSP, caracterizando primeramente el problema para su posterior fase de análisis en donde se elaboran los pseudocódigos.

Los requerimientos de la Tabla 1, será el imput de la heurística, la cual iniciará con un algoritmo que coloque las piezas respetando el largo y el ancho, considerando que cuando no pueda seguir realizando el proceso en la primera lámina, lo hará en la segunda y así hasta haber colocado todas las piezas en alguna lámina. El número total de placas será la respuesta del algoritmo.

Fase 1. Construcción de librerías

```
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];
  static int[] Di = new int[60];
  static int Bl; //Base de Área General
  static int Hl; // Altura Lámina
  static int i = 0; // Variable para controlar la cantidad de cortes leídos
static void Main (string [ ] args)   // Función principal
{
```

Variables globales, vectores de tipo enteros, con máximo corte de 60, Vectores en paralelo

Fase 2. Desarrollo

```
// ..... 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());
```

Entrada del ingreso de la base y altura en la consola.

```
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++;
 }
```

Para mandar a leer el archivo CSV y hacer iteraciones

Fase 3. Ciclo de Iteraciones (j) y cortes (i)

Proceso para determinar los cortes horizontales y verticales

```
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);
    }
```

Fase 4. Restricciones

```
    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);
```

Pseudocódigo de mejora.

```
if (tempAr[j] >CantidadMinima){
  AreaTemporal = AreaTemporal - (Bi[j] * Hi[j] * CantidadMinima);
  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; }
```

Fase 5. Respuestas a obtener

```
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();
        } }  } }
```

Entrada de la consola

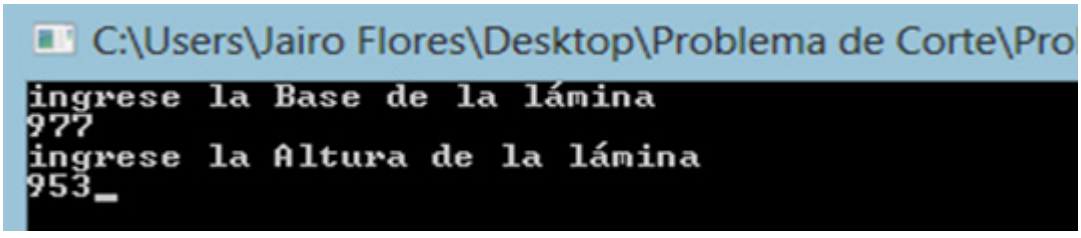

Figura 3. Ingreso de la base y altura de la lámina en la consola.

Respuesta obtenida

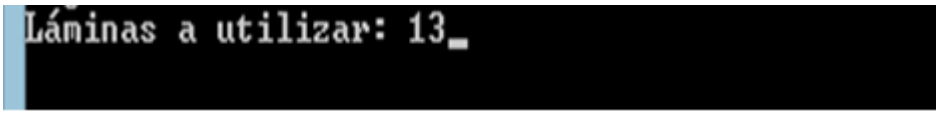

Figura 4. Respuesta obtenida

Fase Mejorada

La fase de mejora permite conocer las placas necesarias que se necesitan para cumplir la demanda del cliente. Se utilizarán 13 placas metálicas de 977cm x 953cm. Si el costo por unidad es de $50, entonces tendrá un costo total de $650 por las 13 láminas rectangulares a utilizar.

Por otro lado hace iteraciones respectivas entre cada (l_i, w_i) del conjunto S_i de los requerimientos establecidos, permitiendo decidir si el corte inicial que conviene es el horizontal o vertical.

También se mejora la estética de la consola:

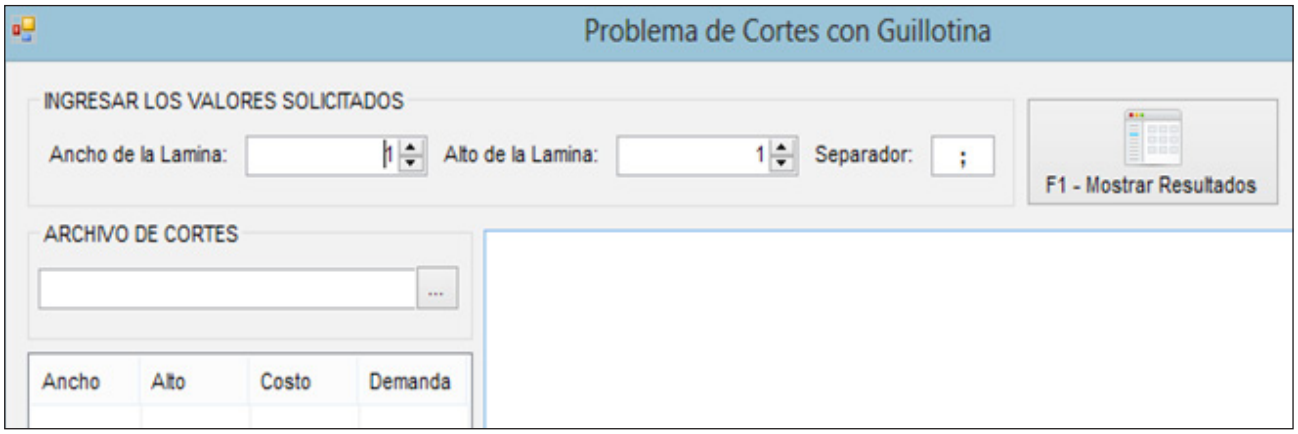

Figura 5. Mejora de la consola de entrada

Esta mejora permite obtener los cortes precisos del imput inicial del problema reflejado en la Tabla 1, mandando a abrir el archivo de cortes que está en formato CSV.

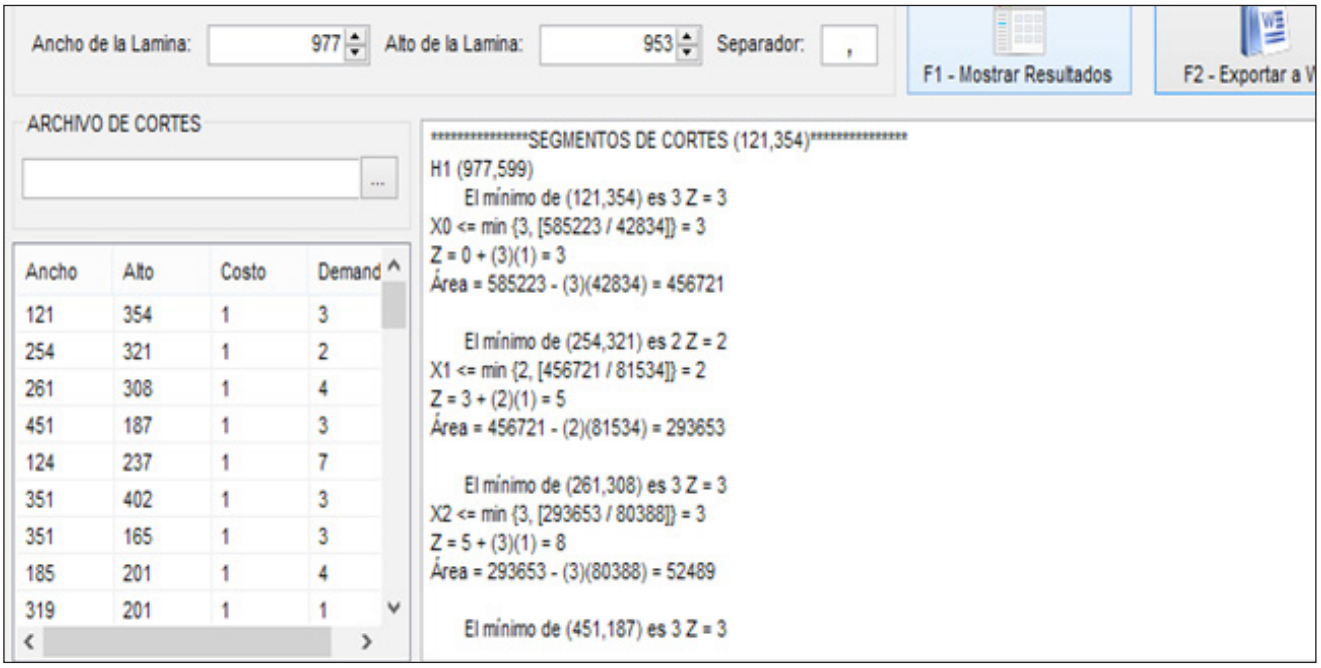

Figura 6. Respuesta ofrecida al problema de corte del imput 1, tiempo requerido 32 segundos.

RESULTADOS COMPUTACIONALES

La respuesta ofrecida computacionalmente, se analiza mejor exportándolas a un archivo word, en donde se aprecia por ejemplo, que tomando el primer corte (121,354) en la lámina de (977,953) se debe hacer en forma vertical pues se obtiene mejor optimización del área de la lámina en relación si se hace en forma horizontal.

Tabla 3. Respuestas computacionales.

l_i, w_i	Cantidad que alcanzan en el área sobrante H_1	Costo
121,354	3	Debido a que el imput no posee costos, se puede usar $v_i = l_i w_i$ pero se ha decidido usar el valor de 1, pues no afecta lo que pide el problema, además funciona a la perfección si dieran el costo por cada l_i, w_i obteniendo así la función objetivo.
254,321	2	
261,308	3	
124,237	1	
El resto	0	
Función objetivo	**9**	

l_i, w_i	Cantidad que alcanzan en el área sobrante H_2
121,354	3
254,321	1
El resto	0
Función objetivo	**4**

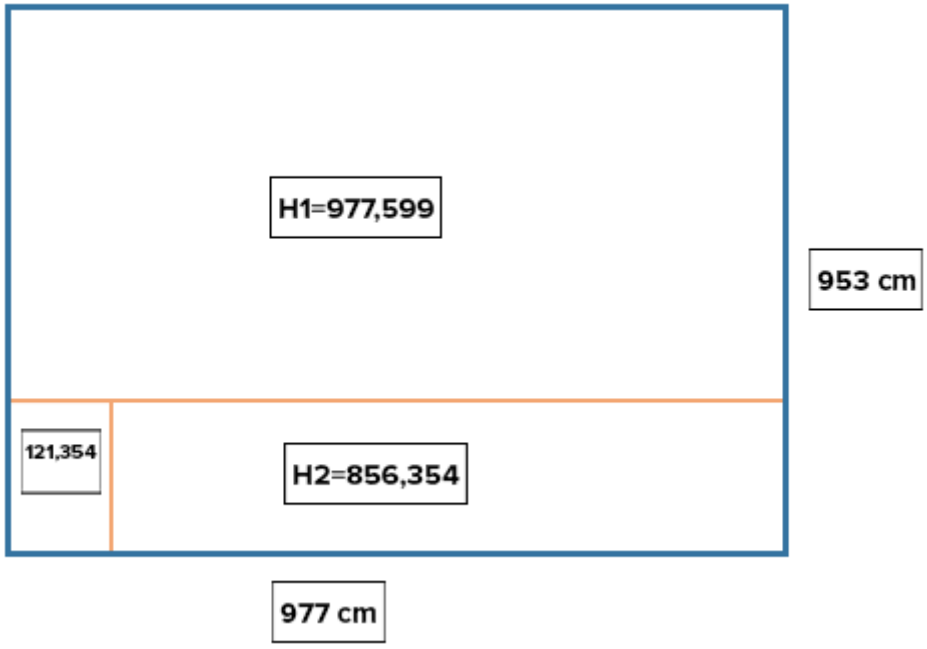

Figura 7. Lámina cortada en forma horizontal.

Tabla 4. Respuestas computacionales.

l_i, w_i	Cantidad que alcanzan en el área sobrante V_1
121,354	1
El resto	0
Función objetivo	**1**

l_i, w_i	Cantidad que alcanzan en el área sobrante V_2
121,354	3
254,321	2
261,308	4
451,187	2
124,237	1
El resto	0
Función objetivo	**12**

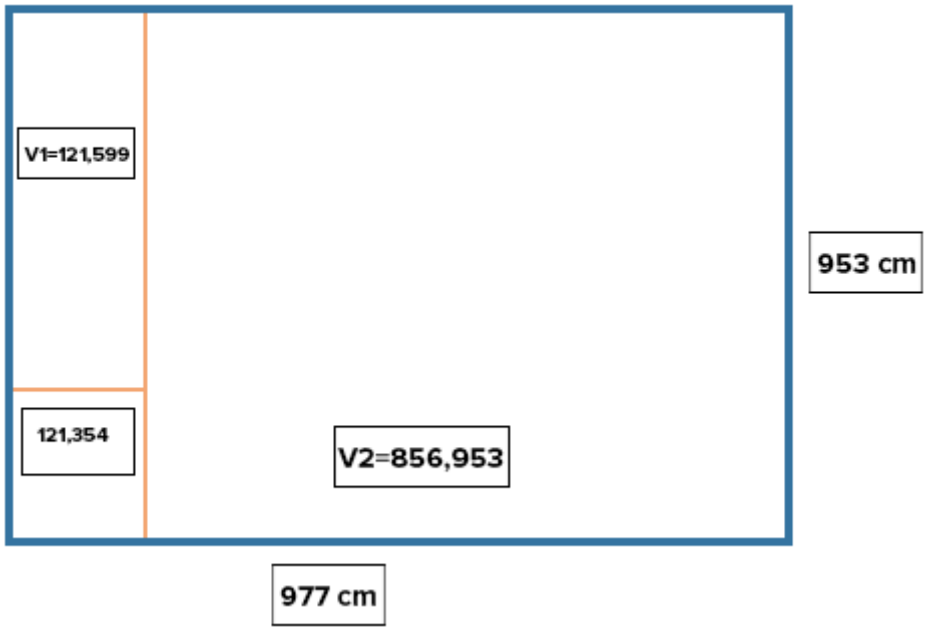

Figura 8. Lámina cortada en forma vertical.

Estas respuestas ofrecen una clara idea de los cortes ortogonales que se necesitan hacer en la lámina principal, para garantizar una mayor optimización del área de la misma. Por otro lado se obtienen las diferentes soluciones por cada lámina demandada en el problema.

CONCLUSIONES

El presente trabajo aporta una heurística capaz de solucionar problemas de corte bidimensional, generando respuestas que faciliten el acomodamiento de rectángulos propios de la demanda

requerida. Por otro lado, se aportan los códigos en C, para entusiastas que estudian problemas de optimización como el que se encuentra en este artículo.

El algoritmo permite seleccionar una dirección de corte (horizontal vs. vertical), para tratar de acomodar piezas adicionales, además, ofrece los costos generados por cada corte realizado. Es importante mencionar que los resultados se pueden mejorar empleando otros tipos de codificación, o mejor aún, haciendo que el programa ofrezca imágenes de los cortes por guillotina en cada lámina seleccionada, a como lo hacen software especializados como Smart2DCutting, CutLogin, CutMaster2D, CutOptimization pro y el Cutting3.

Se considera que se ha cumplido con el objetivo del trabajo, el cual era "Desarrollar un programa computacional que ofrezca una solución óptima al Cutting Stock Problem según la instancia ofrecida" y lo más importante, comprender la amplia gama de problemas que se pueden solucionar con la investigación de operaciones.

REFERENCIAS BIBLIOGRÁFICAS

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/lw4MOr

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.