ALGORITMO DE BÚSQUEDA TABÚ APLICADO A LA SOLUCIÓN DEL PROBLEMA DE CORTE BIDIMENSIONAL GUILLOTINADO

Tabu Search algorithm for solving the two dimensional guillotined problem

RESUMEN

El problema de corte de piezas guillotinadas bidimensional restringido es un problema clásico de optimización caracterizado por su alta complejidad computacional y sus aplicaciones prácticas en el área de la ingeniería.
En este documento se presenta la eficiencia de la técnica de Búsqueda Tabú, utilizando una propuesta de codificación basada en árboles binarios adaptada a la estructura del problema.
Con el fin de evaluar el desempeño del algoritmo utilizado, se presentan casos de prueba de la literatura especializada evaluando el porcentaje de uso del material disponible, este valor mide la calidad de la respuesta obtenida.

PALABRAS CLAVES: Búsqueda tabú, corte guillotina, empaquetamiento bidimensional, codificación de árbol binario.

ABSTRACT

The problem of restricted bidimensional guillotine cutting of pieces, is a classic optimization problem characterized by its high computational complexity, and its practical applications in the engineering area.
This document presents the efficiency of the Tabu Search technique using a codification approach based on binary trees adapted to the structure of the problem.
With the purpose of evaluating the performance of the used algorithm, test cases of specialized literature are presented to evaluate the percentage of use of the available material, this value measures the quality of the obtained answer.

KEYWORDS*: Tabú search, two dimensional packing, bidimensional packing, codification of binary tree.*

ELIANA M. TORO O
Ingeniera Industrial, M.Sc.
Profesor Asistente
Universidad Tecnológica de Pereira
elianam@utp.edu.co

AUGUSTO C. RUEDA M.
Ingeniero Electricista, M.Sc. (C)
Profesor Catedrático
Universidad Tecnológica de Pereira
aucer@ohm.utp.edu.co

MAURICIO GRANADA
Ingeniero Electricista, M.Sc.
Profesor
Facultad de Ingeniería Eléctrica
Universidad Tecnológica de Pereira
magra@utp.edu.co

1. INTRODUCCIÓN

El problema de corte de piezas se ha estudiado ampliamente en el campo de la Investigación de Operaciones. A partir de unos modelos básicos, existen gran cantidad de variantes, derivadas de la amplia gama de aplicaciones prácticas existentes y dependiendo de quién lo esté tratando.

Existen diferentes criterios para clasificar los problemas de corte y empaquetado, como son: el tamaño, la cantidad de las piezas, las dimensiones del problema, etc. Dyckhoff [1] desarrolló un esquema que generaliza la clasificación presentada por Hinxman [2] a finales de los años 80; el principal objetivo del artículo consiste en unificar las distintas notaciones existentes en la literatura relacionada con este tipo de problemas. En el documento, además, se pretende describir una clasificación de las diversas tipologías y clases de problemas relacionados.

Aunque este tipo de problemas se ha resuelto mediante técnicas heurísticas, el mayor número de procedimientos desarrollados para encontrar soluciones adecuadas a los diferentes problemas de corte y empaquetamiento se basan en metaheurísticas muy variadas.

El objetivo de este documento es presentar las soluciones obtenidas al aplicar el algoritmo de Búsqueda Tabú (BT) al problema de corte bidimensional tipo guillotina, usando codificación con árboles binarios en sistemas de prueba disponibles en la literatura especializada, donde el objetivo es obtener patrones de corte que generen el menor desperdicio posible de material.

Este trabajo se presenta de la siguiente forma: en la siguiente sección se explica el tipo de problema que se está resolviendo y su modelo matemático, en la sección 3 se muestra una heurística empleada como herramienta para obtener una solución de inicio, en la sección 4 se describe la codificación en árboles binarios, en la sección 5 se expone la adaptación al problema de la técnica BT, en la sección 6 se presentan las pruebas y resultados obtenidos comparándolos con casos de prueba de la literatura especializada, finalmente, se presenta una

Fecha de Recepción: 07 Septiembre de 2007
Fecha de Aceptación: 08 Noviembre de 2007

sección con conclusiones y recomendaciones para trabajos futuros.

2. PROBLEMA BIDIMENSIONAL DE CORTE TIPO GUILLOTINA

El problema de corte de piezas bidimensional restricto pertenece a la clase de problemas NP-completo, por lo tanto no existe ningún algoritmo polinomial para determinar la solución óptima del mismo.

Definiendo el problema formalmente se tiene *(W, L)* y (w_i, l_i) pares ordenados que denotan las dimensiones del tablero y de la pieza *i* a cortar respectivamente. Sea x_i la cantidad máxima demandada de cada una de las piezas y se denota con *n* el número total de piezas demandadas. La formulación del problema se describe con las siguientes ecuaciones [3]:

$$Min\ WL - \sum w_i l_i x_i \qquad (1)$$

$$s.a$$

$$1 \le i \le n \qquad (2)$$

$$x_i\ entero, \in i \qquad (3)$$

$$Cortes\ factibles \qquad (4)$$

La figura 1 muestra dos patrones de corte, uno factible (1a) y uno infactible (1b). Debido a que en este trabajo se estudia el corte tipo guillotina se considera como infactible cualquier corte de otro tipo.

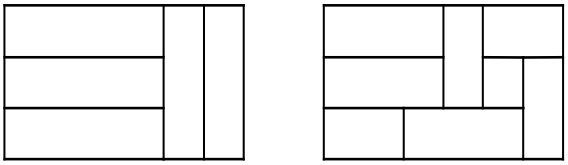

Figura 1. 1(a) Corte guillotina, 1(b) Corte no guillotina

3. TÉCNICA DE INICIALIZACIÓN

Para resolver los problemas de corte bidimensional se encuentran algunas técnicas heurísticas que proporcionan configuraciones iniciales tales como el *Bottom-Left* (BL), *Bottom Left Fill* (BLF), *Difference Process* (DP), las heurísticas de nivel (*First Fit Decrease Height* y *Next Fit Decrease Height*) y *hill-climbing,* entre otras [4].

La estrategia BL dicta que cada pieza a colocar se posiciona inicialmente en la esquina superior derecha de la estructura. A continuación, se desplaza hasta la posición más profunda posible. Una vez allí, la pieza es movida hacia la izquierda tanto como sea posible, repitiéndose esta rutina hasta que la pieza alcance una posición inamovible [4]. Este procedimiento debe realizarse pieza por pieza garantizando que la configuración que se obtenga conserve el corte tipo guillotina.

Debido a que con la implementación de estas técnicas se pueden obtener cortes factibles o infactibles, se hizo una adaptación del BL para encontrar patrones de tipo guillotina empleándola como un método constructivo inicializador de la técnica BT.

4. CODIFICACIÓN DE ÁRBOL

La creación de la estructura de árbol de cortes se realiza aplicando técnicas numéricas de conglomerados (clusters) y tiene por objeto ordenar elementos (piezas) en grupos. Cada nodo interno del árbol representa la forma en que se realiza el corte y los elementos que pertenecen a cada grupo [5]. En la figura 2 se muestra la representación del cluster 5 con las piezas 1 y 2.

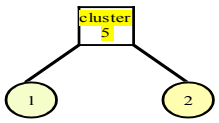

Figura 2. Representación de un cluster

En la figura 3 se observa la configuración con cuatro piezas. En el vector de configuración, la flecha vertical orientada hacia arriba indica la posición límite de ubicación de piezas; los valores en las siguientes posiciones representan los clusters formados. El tamaño del vector de configuración se define como:

Tamaño del vector = número piezas del árbol*2-2
Tamaño del vector = (4*2) -2=6

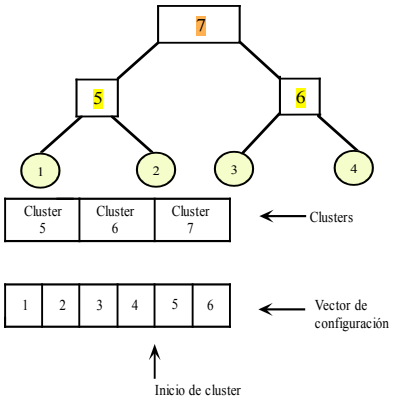

Figura 3. Árbol de 4 piezas y codificación

Obtenida la codificación de piezas y cluster se asocia a esta un vector (vector de cortes) que indica el tipo de corte de cada cluster. En la formulación de esta codificación de consideran dos tipos de corte: vertical, asociado al número cero en el vector de cortes, y horizontal, asociado al número uno en el mismo vector (figura 4).

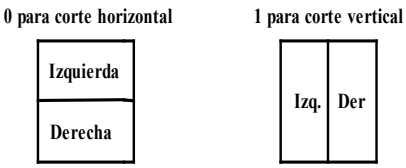

Figura 4. Tipos de corte

Si el tipo de corte de un cluster es horizontal, la pieza que está encima irá en la parte izquierda del par de posiciones del vector de codificación asociado a este cluster y la pieza que está debajo irá en la parte derecha del mismo. Si el corte es vertical, la ubicación real de cada pieza (lado izquierdo o lado derecho) indicará su ubicación en el par de posiciones del vector de codificación asociado al cluster que forman. En la figura 5 se muestran dos ejemplos de codificación de piezas según el tipo de corte.

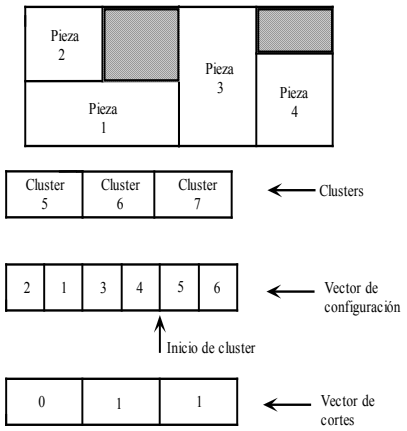

Figura 5. Clusters formados por corte vertical

Ejemplo 1. En este ejemplo se ilustra la codificación descrita.

En la tabla 1 se muestran las dimensiones del tablero y las piezas del ejemplo.

Tipo de pieza	Largo	Ancho	Número de piezas
Tablero	70	42	1
Pieza 1	22	18	2
Pieza 2	8	29	3
Pieza 3	19	19	2
Pieza 4	16	13	2
Pieza 5	4	16	1

Tabla 1. Datos del ejemplo 1

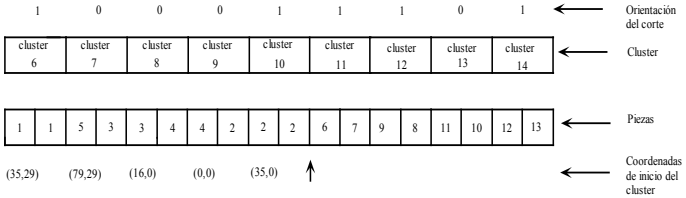

Figura 6. Codificación de una posible configuración

Con base en la figura 6 se puede observar que los cluster se clasifican por niveles a saber:

- Cluster básico: Es aquel que contiene dos piezas. Para el ejemplo, los nodos básicos son 6, 7, 8, 9 y 10.
- Cluster de segundo nivel: Está conformado por dos cluster básicos. Nodos 11 y 12.
- Cluster de tercer nivel: Nodo 13.
- Cluster raíz o nodo de inicio: Es el cluster que conforma todo el conjunto de piezas.

En la figura 7 se demarca el tablero con líneas rojas. Los clusters 8 y 10 son factibles, debido a que no superan el área del tablero, los demás clusters (9, 6 y 7) sobrepasan el área o están por fuera del mismo por lo que se consideran infactibles. Las áreas achuradas representan las áreas que aún no han sido utilizadas. Las coordenadas indican la ubicación de cada cluster.

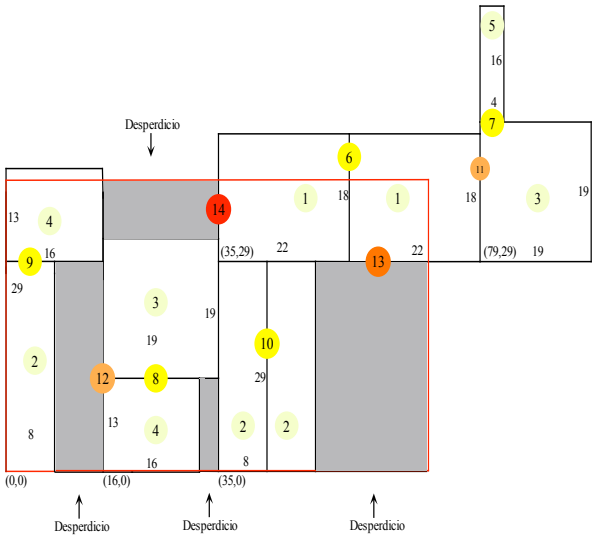

Figura 7. Gráfico de la configuración de la figura 7

Cálculo de la función objetivo

Para el cálculo de la función objetivo se debe conocer primero:

- Las dimensiones de cada cluster. Alto y ancho de un cluster. Para calcular las dimensiones de cada cluster se debe tener en cuenta su tipo de corte. Por ejemplo, si su corte es horizontal el ancho mayor de las dos piezas se escoge el como el ancho del cluster y la altura es la suma de las alturas de las dos piezas. En la tabla 2 se muestran los cálculos realizados para todos los clusters básicos del ejemplo 1.

Cluster	Tipo de corte	Tipos de piezas del	Largo	Alto

		cluster		
6	Vertical	1 y 1	$l_1=22,\ l_1=22$ $l_{c6}=l_1+l_2=44$	$a_1=a_2=18$ $a_{c6}=a_1=a_2$
7	Horizontal	5 y 3	$l_3=19,\ l_5=4$ $l_{c6}=19,\ l_3>l_5$	$a_3=19, a_5=16$ $a_{c7}=19+16=35$
8	Horizontal	3 y 4	$l_3=19,\ l_4=16$ $l_{c8}=19,\ l_3>l_4$	$a_3=19, a_4=13$ $a_{c8}=19+13=32$
9	Horizontal	4 y 2	$L_4=16,\ l_2=8$ $l_{c9}=16,\ l_4>l_2$	$a_4=13, a_2=29$ $a_{c9}=13+29=42$
10	Vertical	2 y 2	$L_2=8,\ l_2=8$ $l_{c10}=8+8=16$	$a_2=29, a_2=29$ $a_{c10}=a_2=a_2$

Tabla 2. Largos y anchos de los cluster

- Las coordenadas de cada cluster. Coordenada cartesiana de cada cluster considerando como origen el extremo inferior izquierdo del tablero. Para la determinación de las coordenadas de cada cluster se empieza el análisis de derecha a izquierda en el vector de codificación, observando el tipo de corte y dimensiones de cada cluster. El primer cluster es el cluster raíz, por tanto, la coordenada de éste siempre es la del origen. Así, por ejemplo, en la figura 9 el nodo de inicio o cluster raíz es el 14 que corresponde al primer valor (en el extremo derecho) del vector de codificación mostrado en la figura 6.

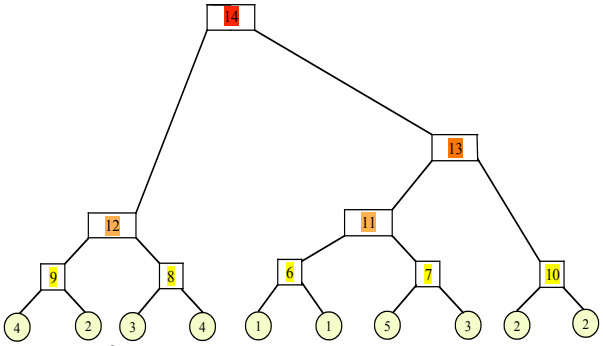

Figura 8. Árbol asociado al ejemplo 1

Este cálculo de la función objetivo se hace con base en el área no utilizada del tablero y las infactibilidades generadas en la ubicación de cada cluster (ecuación 5).

$$F.O. = \text{desperdicios} + \alpha(\text{infactibilidades}) \qquad (5)$$

El parámetro α corresponde al valor de penalización de las infactibilidades.

Según la ubicación de los clusters se pueden presentar los siguientes casos:

i) Que el cluster no genere infactibilidad. El ancho del cluster más la coordenada de inicio del mismo en el eje X es menor que el ancho del tablero. Además, la altura del cluster más la coordenada de inicio del mismo en el eje Y es menor que la altura del tablero.
ii) Que el cluster sea infactible por ancho. El ancho del cluster más su coordenada de inicio en el eje X es mayor que el ancho del tablero. Además, la altura del cluster más su coordenada de inicio en el eje Y es menor que la altura del tablero.
iii) Que el cluster sea infactible por altura. La altura del cluster más la coordenada de inicio del mismo en el eje Y es mayor que la altura del tablero. Además, el ancho del cluster más la coordenada de inicio del mismo en el eje X es menor que el ancho del tablero.
iv) Que el cluster sea infactible por ancho y por alto. La altura del cluster más la coordenada de inicio del mismo en el eje Y es mayor que la altura del tablero, pero la coordenada de inicio del cluster en el eje Y es menor que la altura del tablero. Además, el ancho del cluster más la coordenada de inicio de este en el eje X es menor que el ancho del tablero, pero la coordenada de inicio del cluster en el eje X es menor que el ancho del tablero.
v) Que el cluster generado quede por fuera del tablero. En este caso la coordenada de inicio del cluster en el eje X es mayor que el ancho del tablero y la coordenada de inicio del cluster en el eje Y es mayor que la altura del tablero.

Para calcular la función objetivo basta con calcular las infactibilidades porque a partir de estas, tal como se explica a continuación, se pueden determinar los desperdicios. De acuerdo a las posibles ubicaciones de los cluster, las infactibilidades individuales (infactibilidades de cada cluster) se calculan de la siguiente forma:

- Si se está en el caso i), donde el cluster es factible, la infactibilidad aportada por este es cero.
- Si el cluster es infactible y se está en los casos ii), iii) ó iv), la infactibilidad aportada por este cluster se encontrará calculando el área por fuera del tablero de la pieza que está traslapada.
- Si se está en el caso v), la infactibilidad aportada por este cluster es igual al área efectiva del mismo, es decir, la suma de las áreas de las piezas que lo forman.

Obtenida la infactibilidad total (*Infac*) mediante la suma de las infactibilidades individuales, se calcula el área efectiva del tablero (*Efec_tablero*) como se muestra a continuación:

$$Efec\_tablero = Efec - Infac \qquad (6)$$

donde *Efec* es la suma las áreas de todas las piezas que constituyen la configuración actual.

Con *Efec_tablero* se procede a calcular el valor de las áreas sin utilizar (*Desp*) a través de la ecuación 7.

$$Desp = Área\_tablero - Efec\_tablero \qquad (7)$$

Donde:

Área_tablero es igual al largo por el ancho del tablero.

Así, mediante la ecuación 8 se encuentra el valor de la función objetivo.

$$F.O = Desp + \alpha Infac \qquad (8)$$

El valor del parámetro α se determina experimentalmente.

5. BÚSQUEDA TABÚ

Búsqueda Tabú utiliza una estrategia de búsqueda local que le permite explorar el espacio de soluciones de manera eficiente a través de la estructura de vecindad. Gran parte de la potencialidad del método está basada en la manera como se define el conjunto de vecinos [6].

La idea básica es ir cambiando en forma sistemática la vecindad al momento de realizar la búsqueda mediante la imposición de reglas.

Criterios de vecindad

Para abordar los criterios de vecindad generados se deben definir dos conceptos: desperdicio general y desperdicio individual. El desperdicio individual es el presente en cada cluster calculado como el área del cluster (producto del alto por el ancho del cluster) menos la suma de las áreas de las piezas que lo forman. El desperdicio general es el área del tablero menos el área efectiva del mismo.

Los criterios de vecindad que se definieron son los siguientes:

- Criterio 1 (C_1): Si la configuración es factible y hay desperdicio general pero ningún cluster presenta desperdicio individual, generar configuraciones vecinas insertando piezas en los clusters vacíos, es decir en los clusters que aún no han sido utilizados en la configuración actual.
- Criterio 2 (C_2): Si la configuración es factible y presenta clusters con desperdicios individuales, buscar e insertar piezas con áreas menores o iguales a los desperdicios de los clusters que los presenten.
- Criterio 3 (C_3): Si la configuración es factible y presenta clusters con desperdicios individuales, intercambiar piezas de la configuración actual con las aún disponibles, procurando disminuir el desperdicio de los clusters que los presenten.
- Criterio 4 (C_4): Si la configuración es infactible o si en los anteriores tres criterios no se generó el número requerido de vecinos, generar configuraciones vecinas retirando las piezas que generan infactibilidad.
- Criterio 5 (C_5): Si con los tres primeros criterios (configuración factible) o con el cuarto criterio (configuración infactible) no se generaron las configuraciones vecinas requeridas, crear las configuraciones restantes intercambiando piezas entre los clusters básicos vecinos no nulos completos (con por lo menos una pieza).
- Criterio 6 (C_6): Si con ninguno de los anteriores criterios se generó el número requerido de configuraciones vecinas, realizar cambios en los tipos de cortes de los clusters básicos dando prioridad a los que generen infactibilidad (caso infactible) y/o a los que presenten desperdicios (caso factible).

6. PRUEBAS Y RESULTADOS

Se hicieron pruebas con cuatro casos. Los Casos 1 y 2 aparecen en [7]. Los casos 3 y 4 corresponden a los casos 1 y 3 de la referencia [8].

Una parte importante de BT requiere la definición del número de iteraciones durante las cuales un atributo (en este caso una pieza) permanecerá tabú, es decir, sin poderse mover de su posición, ya sea dentro o fuera de la configuración actual o cualquier otra con relación inclusive con otros atributos, como por ejemplo, de intercambio. En el presente trabajo se definieron cuatro tipos de iteraciones tabú:

K_1 = Iteraciones tabú para piezas que entran a la configuración actual.
K_2 = Iteraciones tabú para que piezas salen de la configuración actual.
K_3 = Iteraciones tabú para piezas que se intercambian (dentro y fuera de la configuración actual).
K_4= Iteraciones tabú para piezas que intercambian (dentro de la configuración actual).

En la tabla 3 se muestran los valores usados como iteraciones tabú para los cuatro casos de prueba.

	K_1	K_2	K_3	K_4
Caso 1	7	7	7	7
Caso 2	7	7	7	7
Caso 3	5	5	5	5
Caso 4	7	7	7	7

Tabla 3. Iteraciones tabú

La cantidad de vecinos según cada criterio y el valor de α utilizados se especifican en la tabla 4.

	C_1	C_2	C_3	C_4	C_5	C_6	α
Caso 1	10	10	10	15	0	0	25
Caso 2	10	10	40	0	0	0	5
Caso 3	20	20	10	0	0	10	7
Caso 4	10	10	30	10	10	10	20

Tabla 4. Número de vecinos y penalización

En la figura 9 aparece la configuración inicial obtenida para el caso 3 con el constructivo BL modificado y en la

figura 10 se ilustra la configuración final obtenida para el mismo caso después de aplicar el algoritmo BT.

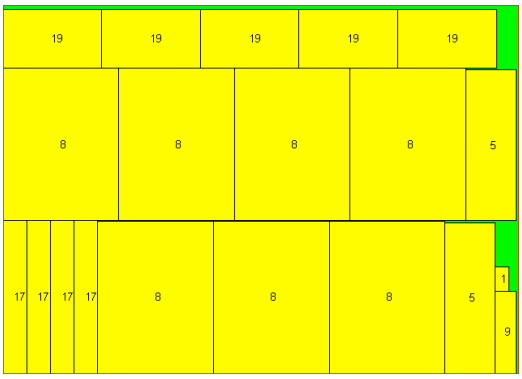

Figura 9. Configuración inicial del constructivo caso 3.

Para los casos que se corrieron se obtuvieron mejoras comparadas con las mejores soluciones reportadas en [8], como se observa en la tabla 5.

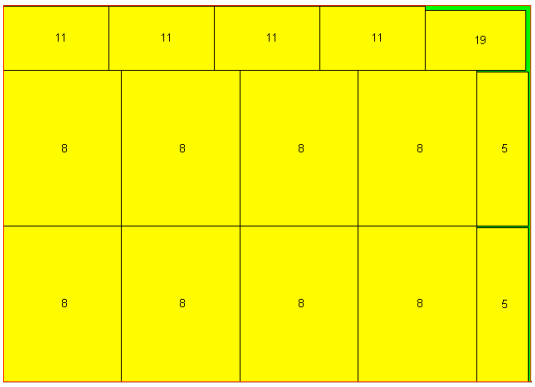

Figura 10. Configuración final con BT caso 3.

	Inicial		Final		Mejor Solución reportada	
	F. O.	% uso	F. O.	% uso	F. O.	% uso
Caso 1	869.95	95.39	0	100	2	99.78
Caso 2	419	36.71	0	100	5	99.12
Caso 3	62618	96.87	21354	98.93	39418	98.03
Caso 4	173958	93.57	44847	98.34	53071	98.04

Tabla 5. Comparación de resultados

7. CONCLUSIONES Y RECOMENDACIONES

El tipo de codificación para representar el problema bidimensional restricto basado en árboles binarios garantiza que los patrones que se obtengan sean de tipo guillotina; por la estructura de la misma, se facilita la generación de soluciones vecinas y el cálculo del valor de la función objetivo.

Se ha determinado el conjunto de parámetros que hacen que el método tenga un mejor desempeño computacional para cada uno de los casos de prueba calibrándolos por ensayo y error.

Se verificó la eficiencia de la técnica BT al superar las mejores respuestas reportadas para los casos 1, 2 y 3, siendo el más representativo el caso 3 que corresponde a un caso de la literatura especializada.

El tipo de codificación propuesta permite implementar otras técnicas metaheurísticas tales como los Algoritmos Genéticos y Colonia de Hormigas que requieren de una adaptación a los operadores específicos de cada método.

Dentro de los trabajos futuros podrían resolverse casos de prueba en donde las condiciones del problema permitan rotación de las piezas; además, se podría hacer la extensión para el problema de corte bidimensional cuando las figuras demandadas son polígonos, planteando que puede realizarse una reducción del problema a uno de empaquetamiento de rectángulos con algunas consideraciones.

8. BIBLIOGRAFÍA

[1] DYCKHOFF, H. Classification of real world trim loss problems. En:G.Fandel et al.(Eds), Essays on Production Theory and Planning, Springer-Verlag, Berlin, páginas 191-208.1990

[2] HINXMAN, A.I. A two dimensional trim-loss problem with sequencing constraints. Advanced Paper of IJCAI-77 MTI, páginas 859-854.1977

[3] PARADA V., SEPÚLVEDA M.,GÓMEZ A. Solution for the Constrained Guillotine Cutting Problem by Simulated Annealing. Journal on computers and operations research, 25 37-47. 1998.

[4]FERNÁNDEZ T, DUARTE A. Método Multi-arranque aplicado al problema del Strip Packing Problem bidimensional V Congreso de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados. Puerto de la Cruz (Tenerife), 14-16/02/2007

[5] DIEGO-MAS, J.A., "Optimización de la distribución de planta en instalaciones industriales mediante algoritmos genéticos. Aportación al control de la geometría de actividades". Tesis doctoral, Universidad Politécnica de Valencia, 2006.

[6] GALLEGO R. ESCOBAR A. ROMERO R. Técnicas de optimización combinatorial. Textos universitarios. Universidad Tecnológica de Pereira. Abril 2006.

[7] TORO, E. GRANADA M. Problema de empaquetamiento rectangular bidimensional tipo guillotina resuelto por algoritmos genéticos. Revista Scientia et Technica. Universidad Tecnológica de Pereira año XIII. número 35. Agosto 2007.

[8]CUI Y. An exact algorithm for generating homogenous T-shape cutting patterns. Computers & Operations Research. 2007, 34(4): 1107-1120. Disponible en Internet: www.gxnu.edu.cn/Personal/ydcui/English/index.htm