Solución al problema de empaquetamiento bidimensional usando un algoritmo híbrido constructivo de búsqueda en vecindad variable y recocido simulado

Two dimensional packing problem using a hybrid constructive algorithm of variable neighborhood search and simulated annealing

Eliana Mirledy Toro[1], Alejandro Garcés[1], Hugo Ruiz[2]*

[1]Facultad de Ingeniería Industrial, Universidad Tecnológica de Pereira, Apartado Aéreo 097, Risaralda, Colombia

[2]Programa de Ingeniería Eléctrica, Universidad Tecnológica de Pereira, Apartado Aéreo 097, Risaralda, Colombia.

(Recibido el 20 de febrero de 2008. Aceptado el 30 de junio de 2008)

Resumen

En este trabajo, se modela el empaquetamiento de rectángulos con base en divisiones del área disponible, se utiliza una codificación de árbol binario para disponer las piezas de forma que se garantice el corte de tipo guillotina y se propone un algoritmo dividido en tres etapas que trabajan con estrategias individuales inspiradas en algoritmos de vecindad variable, recocido simulado y técnicas constructivas para lograr la solución del problema. Se comparan las respuestas obtenidas con base en la función objetivo que cuantifica el área utilizada y el porcentaje de utilización del material disponible para cincuenta casos de prueba de la literatura especializada frente a sus respectivas respuestas reportadas obteniéndose excelentes resultados.

---------- *Palabras clave:* empaquetamiento bidimensional, vecindad variable.

Abstract

In this work, the packing of rectangles is modeled based on divisions of the available area, a binary tree codification is used to arrange the pieces so that the guillotines type cutting is guaranteed. A three stages algorithm with individual strategies inspired by algorithms of variable neighborhood

* Autor de correspondencia: Teléfono: + 57 + 6 + 313 73 00, fax + 57 + 6 + 321 32 06, correo electrónico: alejandrog@ohm.utp.edu.co (A. Garcés).

search simulated annealing and constructive techniques are used to obtain the solution of the problem. The results obtained are compared using the objective function and percentage used of available area with fifty test cases of the specialized literature and their respective well-known answer with excellent results.

---------- *Keywords:* Guillotine, two-dimensional cutting, variable neighborhood search, optimization.

Introducción

Los problemas de corte y empaquetamiento pertenecen a la categoría de problemas de optimización combinatoria denominados Np- completos debido a que el espacio de soluciones crece de forma exponencial de acuerdo al número de piezas a ser ubicadas. Si por ejemplo se tienen *n* piezas a ser ubicadas, entonces el espacio de soluciones estará dado por $2^n * n!$ [1]. Este tipo de problema es fácil de definir intuitivamente aunque no son fáciles de modelar formalmente y presentan grados de dificultad que hacen difícil el manejo computacional. En ellos se tiene un conjunto de piezas de diferentes tamaños y formas que deben ser localizadas sobre un tablero de material de mayor tamaño sin superponerse unas sobre otras. El objetivo de tal disposición es maximizar el área utilizada de forma que se generen la menor cantidad de área desperdiciada. Existen diferentes criterios para clasificar los problemas de empaquetado, como son el tamaño y la cantidad de las piezas, las dimensiones del problema, etc.

Un caso particular de esta familia de problemas, lo constituye el problema de corte de piezas rectangulares desde tableros también rectangulares con el fin de satisfacer una demanda predefinida y determinada con base en las solicitudes de los clientes. Además, todos los cortes deben ser de tipo guillotina, es decir, cortando el tablero o parte de él ortogonalmente de lado a lado se obtiene una pieza o un conjunto de piezas. Se exige además, que el número de veces que una determinada pieza sea cortada desde la placa no supere un valor preestablecido. Los problemas de empaquetado tienen una amplia aplicación en distintas facetas industriales: textil, cristal, piel, madera, entre otras. Sin embargo, diferentes inconvenientes particulares a cada problema determinan la disposición de las piezas sobre el patrón. Entre estos inconvenientes se pueden destacar las propiedades del material del patrón (falta de uniformidad en el color, calidad, textura), la tecnología de corte (que obliga a que exista una determinada distancia entre las piezas).

Para resolver este problema la mayor parte de procedimientos reportados en la literatura se basan en técnicas metaheurísticas. En 1977 Christofides y Withlock [2], propusieron un algoritmo exacto de búsqueda en árbol para resolverlo, utilizando para ello el algoritmo previamente propuesto en 1966 por Gilmore y Gomory [3] quienes resuelven un problema de características similares. Por otro lado en 1983 Wang [4], propone un algoritmo de desarrollo incremental del patrón solución. Tal algoritmo es posteriormente mejorado en los trabajos de Vasko en 1989 [5] y Oliveira y Ferreira en 1990[6]. En 1997, Lai y Chan [7] presentan un procedimiento basado en simulated annealing.

En 1998 Parada [8] presenta la solución del problema mediante el simulated annealing usando una codificación de árbol binario. En 2001, Leung [9] utiliza un algoritmo evolutivo, en el que usan la representación de las soluciones de Lai y Chan . En 2003 Beasley [10] presenta un algoritmo genético para el caso general. El algoritmo está basado en una nueva formulación no lineal para el problema. La formulación también admite extensiones para el problema con más de un tablero, el problema con algunas zonas del tablero que no se pueden utilizar y el problema donde las piezas se pueden rotar.

En 2007 Yaodong Cui [11] presenta un algoritmo exacto que genera cortes homogéneos en dos segmentos de los que publica las características de los casos de prueba y las respuestas obtenidas.

En este trabajo se presenta una propuesta que emplea la codificación de árbol binario y un algoritmo dividido en tres etapas que trabajan con estrategias individuales inspiradas en algoritmos de vecindad variable [12], recocido simulado [13] y técnicas constructivas para lograr la solución del problema, para evaluar la eficiencia de la técnica de solución propuesta se utilizan los casos de prueba usados en la referencia [11].

Problema de Corte Bidimensional Tipo Guillotina

El problema de corte de piezas bidimensional restricto para cortes guillotinados consiste en la obtención de un patrón de cortes del tipo guillotina, para una lámina o tablero rectangular, desde donde se desea obtener un conjunto determinado de piezas rectangulares más pequeñas bajo una demanda establecida, con el propósito de minimizar la pérdida de material cortado o maximizar el área utilizada.

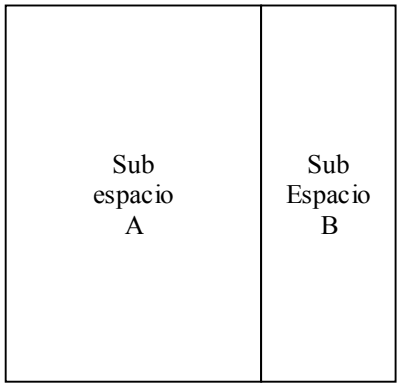
Definición del problema

Dado un número finito n de piezas rectangulares de área $A_i = x_i \text{A} y_i$ que deben ser ubicadas en un tablero igualmente rectangular de área $A_T = x_T \text{A} y_T$. El problema se define como restringido cuando se limita el número máximo de piezas (D) a ubicar de cada tipo. No es necesario que todas las piezas sean ubicadas en el tablero. La función objetivo consiste en maximizar el área efectiva utilizada en el tablero principal. Una variante al problema considera la rotación de piezas, aspecto que no será considerado en este trabajo.

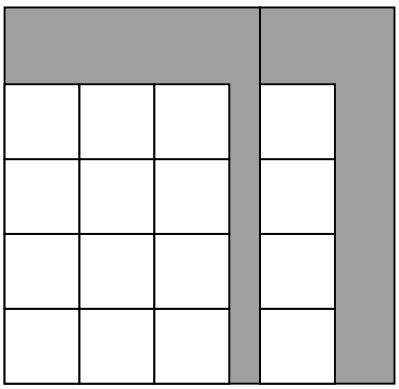
El modelo propuesto divide el tablero principal en sub-espacios (S) asegurando que en cada subespacio se ubiquen piezas de un mismo tipo formando una matriz rectangular de piezas como se muestra en la figura 1(a).

Adicional a esto, los sub-espacios deben ser seleccionados de tal forma que el corte sea tipo guillotina. Un corte es factible de tipo guillotina si cuando al ser aplicado sobre un rectángulo produce dos nuevos rectángulos, es decir, si el corte va de un extremo a otro del rectángulo original; en otro caso se denomina de tipo no guillotina [3].

En cada sub-espacio generado por los cortes tipo guillotina debe ser ubicada la mayor cantidad de piezas del mismo tipo de tal forma que el desperdicio total sea minimizado, un forma de lograrlo consiste en ubicar piezas del mismo tipo en un arreglo matricial como se muestra en la figura 1(b). Un arreglo de estas características puede ser generado por una secuencia de cortes guillotina aún en los casos en donde la demanda restrinja posiciones nulas (espacio sin utilizar).



(a)



(b)

Figura 1 Generación de subespacios

Cuando se resuelve un problema de tipo no guillotina el valor de la función objetivo considerada como el máximo de área utilizada será mayor que si se resuelve el problema donde se consideren los cortes de tipo guillotina, esta restricción hace más complejo el modelo matemático y la solución del mismo. El problema del corte de piezas ha sido ampliamente estudiado, debido a la variedad de aplicaciones prácticas en el ámbito de la Ingeniería y muchas de ellas requieren que los cortes sean de tipo guillotina.

Modelo matemático

El problema propuesto puede ser modelado matemáticamente de la siguiente forma:

$$Max\ f = \sum_{k=}^{Ns} \sum_{i=1}^{n} A_i \cdot (U_{ki} \cdot V_{ki}) \quad (1)$$

Sujeto a:

$$U_{ki},\ V_{ki} \in N \quad (2)$$

$$\sum_{k=}^{Ns} U_{ki} \cdot V_{ki} \leq Di \quad (3)$$

$$x_i \cdot U_{ki} \leq X_k \quad (4)$$

$$y_i \cdot V_{ki} \leq Y_k \quad (5)$$

$$\sum_{i=1}^{n} U_{ki} = max\ \{U_{ki}\} \quad (6)$$

$$\sum_{i=1}^{n} V_{ki} = max\ \{V_{ki}\} \quad (7)$$

$$S = \{Subespacios\ de\ tipo\ guillotina\} \quad (8)$$

$$Ns = dim\ \{S\} \quad (9)$$

En donde

f: función objetivo : maximizar el área total utilizada.

A_i: área de cada una de las posibles piezas.

xi : ancho de la pieza i

yi : alto de la pieza i

U_{ki}: número de piezas del tipo i colocadas en el sub espacio k horizontalmente

V_{ki}: número de piezas del tipo i colocadas en el sub espacio k verticalmente

Ns : número de subespacios. (Ec. 9)

N : números naturales (enteros positivos).

n: número de tipos de piezas.

D_i: demanda por cada tipo de pieza.

X_k : ancho del subespacio.

Y_k: : altura del subespacio.

El producto $U_{ki} \cdot V_{ki}$ corresponde al número de piezas totales del tipo i localizadas en el sub-espacio k, mientras que el número de piezas ubicadas debe ser un número natural tal como se muestra en la Ec.(2). El total el número de elemento de cada tipo debe ser inferior a la demanda del mismo Ec.(3) y en cada sub-espacio los elementos ubicados en forma de arreglo matricial deben ser factibles lo cual significa que el número de elementos horizontales multiplicados por la longitud de cada elemento debe ser menor a la longitud del sub-espacio (X_k) como se muestra en la Ec.(4) y análogamente en la Ec.(5) para el caso vertical.

Para asegurar que las piezas ubicadas en cada subespacio sean del mismo tipo se plantean las Ecs. (6) y (7). Esa restricción además de permitir el desarrollo de un constructivo eficiente, garantiza que los cortes en los subespacios sigan siendo de tipo guillotina. Así mismo, la Ec.(8) indica que el corte debe ser tipo guillotina. La Ec.(8) es la restricción de mayor dificultad a ser modelada matemáticamente por tanto se propone un tipo de codificación que garantice su cumplimiento como se muestra a continuación.

Codificación

Para asegurar que los subespacios creados presenten cortes de tipo guillotina se define una codificación de árbol binario complementario el cual además permite un manejo eficiente de la información al utilizar un reducido número de variables para representar una solución.

El número de capas (c) es definido de forma aleatoria y determina el número de variables en la codificación así como el número de subespacios (S) creados. El número de cortes (p) está dado por la Ec. 10:

$$p = 2^c - 1 \quad (10)$$

El número de subespacios (N_S) está dado por la Ec. 11:

$$Ns = 2^c \quad (11)$$

Para representar la estructura que genera el subespacio son definidos dos vectores de tamaño

igual al número de cortes: el primero ($\vec{T}$) es un vector de tipo binario que define el tipo de corte (vertical u horizontal), el segundo ($\vec{H}$) es un vector real con valores entre 0 y 1 que determina la distancia porcentual a la cual se produce el corte con respecto al patrón superior.

Por ejemplo si se decide que el número de capas es 2 entonces se generarán tres cortes que a su vez darán lugar a 4 sub-espacios como se muestra en la figura 3. Un conjunto de posibles vectores generados aleatoriamente son mostrados en las Ec 12 y 13:

$$\vec{T} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \tag{12}$$

$$\delta = R \cdot \left(1 - \frac{k}{IP + 1}\right)^{1/2} \tag{13}$$

Una ventaja de este tipo de codificación es que cualquier conjunto $\{\vec{T}, \vec{H}\}$ es factible siempre y cuando $\vec{T}$ sea binario y $\vec{H}$ este en el intervalo $[0, 1]$. Por ejemplo, si las dimensiones del tablero base son de *70x42*, estas dimensiones representan el 100 % de la primera capa; el primer corte con parámetros $T_1 = 0$ y $H_1 = 0{,}65$ indica que se hace un corte horizontal al *65%* del tablero, esto genera dos espacios de dimensiones *70x14,7* y *70x27,3*; el segundo corte $T_2 = 1$ y $H_2 = 0{,}4$ indica un corte vertical sobre el espacio de *70x27,3*. El último corte es aplicado sobre el espacio de 70x14,7. Estos cortes sucesivos generan cuatro subespacios de dimensiones *S = {28x27,3 42x27,3 21x14,7 49x14,7}*. El proceso se representa mediante el árbol mostrado en la figura 2. Finalmente la figura 3 muestra la división del tablero base.

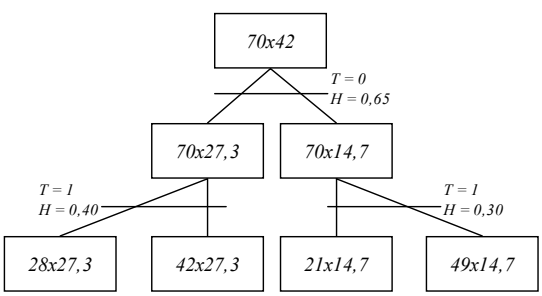


Figura 2 Árbol binario que representa el proceso

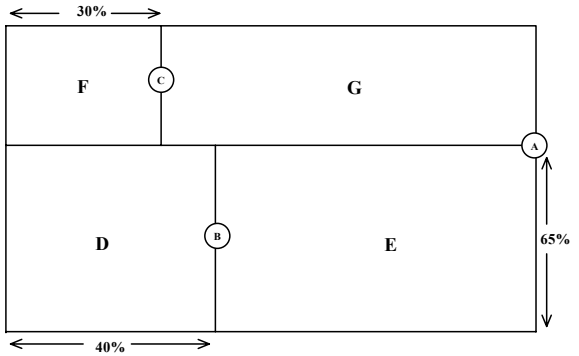


Figura 3 Cortes sobre el tablero según el árbol de corte propuesto en la figura 2

Metodología de solución

La codificación propuesta garantiza la factibilidad en cuanto al corte guillotina, esto permite que el problema de optimización con restricciones se transforme en un problema de minimización irrestricto entero mixto en donde las variables T y H son independientes entre sí, esto significa que para cada conjunto de valores de T existe una solución H la cual es óptima (Ec. 14):

$$f(T, H) = f(H(T)) \tag{14}$$

De esta forma se puede utilizar cualquier metaheurística para encontrar valores óptimos de T siempre y cuando se utilice un segundo algoritmo que permita encontrar los valores óptimos de H a partir de los valores de T. La ubicación final de las piezas se realiza mediante un tercer algoritmo, esta vez un constructivo que garantiza una solución única para cada conjunto de valores (T,H), este algoritmo es basado en arreglos del mismo tipo como se mostró en la figura 1(b). La figura 4 esquematiza el proceso.

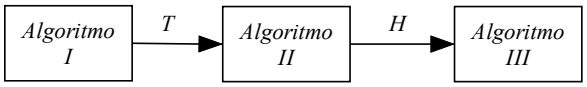


Figura 4 Secuencia de los algoritmos

Algoritmo de búsqueda aleatoria binaria

El algoritmo I determina el valor de la variable binaria T la cual representa el tipo de corte a utilizar y para resolver este algoritmo se utilizó un esquema de vecindario variable [12].

```
Algortimo I: Búsqueda aleatoria binaria
Datos de entrada ← IG
Tinc ←Aleatorio Binario
Incumbente ← 0
Vecindario ←1
For k = 1 To  IG
T ←Tinc
For i = 1 To Vecindario
    j ←Aleatorio Entero
    T[j] ←Not(T[j])
EndFor
    Vecindario ←Vecindario + 1
    [F, H] ←AlgoritmoII(T)
    If F > Incumbente
       Incumbente ← F
       Tinc ← T
       Hinc ←H
       Vecindario ← 1
    EndIf
EndFor
EndFor
Result ←Hinc, Tinc
```

Inicialmente el vecindario de búsqueda consiste en modificar de forma aleatoria una posición en el vector T, posteriormente, si esta modificación no genera una mejora en la función objetivo se procede a modificar dos posiciones y así sucesivamente, en el momento en que la incumbente sea actualizada se regresa nuevamente al primer vecindario. Esta metodología de búsqueda local permite diversificar cuando la función objetivo no es mejorada y hacer una búsqueda local más detallada en el momento de actualizarse la incumbente, de esta forma se logra un algoritmo computacionalmente eficiente.

Algoritmo de búsqueda aleatoria real

El segundo algoritmo determina la variable H a partir de un vector T, para ello utiliza un algoritmo de búsqueda aleatoria que modifica iterativamente una posición del vector H reduciendo paulatinamente el ancho de búsqueda, para ello se utiliza una función delta definida según la Ec. 15:

$$\delta = R \cdot \left(1 - \frac{k}{IP + 1}\right)^{1/2} \qquad (15)$$

En donde R corresponde a un número aleatorio en el intervalo [-1,1], mientras que k es la iteración actual e IP es el número total de iteraciones del algoritmo. En las primeras iteraciones, el factor que multiplica a R es cercano a 1, por tanto los movimientos son altamente aleatorios, a medida que el algoritmo evoluciona se hace más determinístico por lo cual este factor se acerca a cero. El valor de H_i estará dado por la Ec. 16:

$$H_i = H_i + \Delta \qquad (16)$$

Este algoritmo está inspirado en la filosofía del recocido simulado [13] en donde a medida que el proceso evoluciona entonces el grado de aleatoriedad disminuye.

```
Algoritmo II: Búsqueda aleatoria real
Datos de entrada ← IP, T
Hinc ←Aleatorio
Incumbente ← 0
For k = 1 To  IP
    H ←Hinc
    j ←Aleatorio Entero
    delta ← aleatorio(-1,1)*((1-k/(IP+1))^0,5)
    H[j] ←H[j] + delta
    If H[j] > 1 Or H[j] < 0
       H[j] ← Aleatorio
    EndIf
```

F ← AlgoritmoIII(T,H)
If F > Incumbente
Incumbente ←F
Hinc ← H
EndIf
EndFor
Result ←Hinc

Algoritmo constructivo

El algoritmo constructivo ubica las piezas en los diferentes sub-espacios definidos por los vectores H y T. Como cada sub-espacio debe ubicar arreglos del mismo tipo solo vasta determinar cuál es el desperdicio para cada tipo de arreglo asegurando que se cumpla la restricción de demanda como se muestra a continuación.

Algoritmo III: Constructivo
Datos de entrada → T,H
For *k = 1* To *Ns*
(X_k, Y_k) ← Tamaño del sub-espacio k
Max ← 0
For *i* = 1 To *n*
a ← Entera(X_k / x_i)
b ← Entera (Y_k / y_i)
z ←min(a*b, D_i)
If z*A_i > Max
Max = z*A_i
Result(k) ←[z,i]
EndIf
EndFor
EndFor

La solución encontrada por este algoritmo es única ya que no incluye ninguna sentencia con aleatoriedad; por ejemplo para el tablero de 70x42 anteriormente descrito con los valores de T y H dados por las ecuaciones (12) y (13) y los parámetros mostrados en la tabla 1, el algoritmo constructivo obtiene el corte mostrado en la figura 5, en donde las áreas punteadas corresponden al material sin utilizar.

Tabla 1 Datos del problema

Tipo de pieza	*Largo*	*Ancho*	*Número de piezas*
Pieza 1	22	18	6
Pieza 2	8	29	5
Pieza 3	19	19	2
Pieza 4	16	13	2
Pieza 5	4	16	1

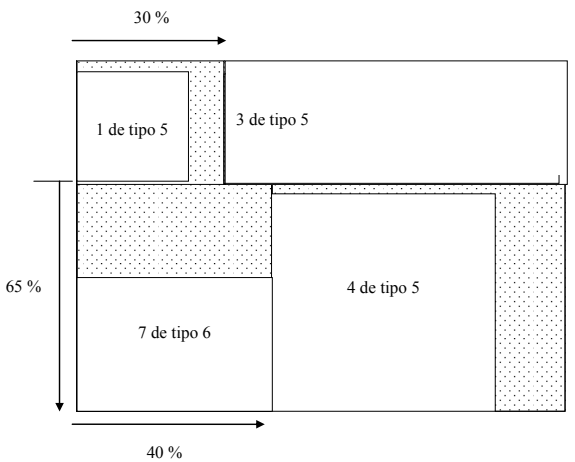


Figura 5 Solución obtenida

Resultados y discusión

Se tomaron 50 casos de prueba de la literatura especializada que pueden ser obtenidos de [11]. Cada caso presenta 20 tipos de piezas rectangulares a ser ubicadas sin permitir rotación.

Después de calibrar los parámetros a través de 1000 ejecuciones del algoritmo para distintos casos, los mejores resultados se obtuvieron con los valores que aparecen en la tabla 2.

Tabla 2 Valor de los parámetros

Número de capas	*3*
Iteraciones globales (IG)	5000
Iteraciones parciales (IT)	1000

Las iteraciones globales se refieren al número de iteraciones del algoritmo I mientras que las iteraciones parciales se refieren al número de iteraciones del algoritmo II.

En las tablas 3 y 4 se muestran los 50 casos donde se comparan las respuestas obtenidas en función del área utilizada y el porcentaje de utilización. Para 42 de ellos se obtuvieron mejores respuestas que las mejores conocidas. Cada caso se evaluó 10 veces con los parámetros de la tabla 2.

Tabla 3 Resultados para los casos del 1 al 25

Caso de Prueba		*Respuesta de la Literatura*		*Respuesta del algoritmo propuesto*		*Diferencia*
Caso	*Área disponible*	*Área utilizada*	*Porcentaje de uso*	*Área utilizada*	*Porcentaje de uso*	
1	2004002	1964584	0,9803	1990174,386	0,9931	25590,3862
2	2941249	2867218	0,9748	2887424,143	0,9817	20206,1433
3	2709525	2656454	0,9804	2679178,320	0,9888	22724,3200
4	2479752	2370496	0,9559	2443299,646	0,9853	72803,6456
5	2319604	2281600	0,9836	2275995,445	0,9812	-5604,5552
6	2976190	2918503	0,9806	2931249,531	0,9849	12746,5310
7	2265650	2227422	0,9831	2240954,415	0,9891	13532,4150
8	2272792	2238658	0,9849	2247791,288	0,9890	9133,2880
9	2764384	2690038	0,9731	2717389,472	0,9830	27351,4720
10	2624232	2574750	0,9414	2579620,056	0,9830	4870,0560
11	3235950	3115873	0,9628	3162817,530	0,9774	46944,5300
12	2921333	2837743	0,9713	2856771,541	0,9779	19028,5407
13	2553588	2501400	0,9795	2525753,891	0,9891	24353,8908
14	2436480	2343169	0,9617	2350959,552	0,9649	7790,5520
15	3072540	3017184	0,9819	3031675,218	0,9867	14491,2180
16	2174340	2102756	0,9670	2136723,918	0,9827	33967,9180
17	2401595	2276323	0,9478	2317299,016	0,9649	40976,0155
18	2827455	2736919	0,9679	2751961,952	0,9733	15042,9515
19	2181600	2145051	0,9832	2149966,80	0,9855	4915,8000
20	3157786	2994717	0,9483	3050737,055	0,9661	56020,0546
21	2953340	2900376	0,9820	2892501,196	0,9794	-7874,8040
22	3204790	3066344	0,9568	3086853,728	0,9632	20509,7280
23	2763800	2676037	0,9682	2686966,36	0,9722	10929,3600
24	2408892	2287330	0,9495	2292060,738	0,9515	4730,7380
25	3163699	3037774	0,9601	3108967,007	0,9827	71193,0073

Tabla 4 Resultados para los casos del 26 al 50

Caso de Prueba		*Respuesta de la Literatura*		*Respuesta del algoritmo propuesto*		*Diferencia*
Caso	*Área disponible*	*Área utilizada*	*Porcentaje de uso*	*Área utilizada*	*Porcentaje de uso*	
26	2805712	2695345	0,9606	2748756,046	0,9797	53411,0464
27	2526230	2383242	0,9433	2479747,368	0,9816	96505,3680
28	2532060	2446799	0,9663	2490027,804	0,9834	43228,8040
29	2230605	2158548	0,9676	2186885,142	0,9804	28337,1420
30	2514609	2471150	0,9827	2462305,133	0,9792	-8844,8672
31	2727200	2583155	0,9471	2666656,160	0,9778	83501,1600
32	2699860	2618956	0,9700	2628853,682	0,9737	9897,6820
33	2262910	2216886	0,9796	2224666,821	0,9831	7780,8210
34	2330028	2279436	0,9782	2284359,451	0,9804	4923,4512
35	2882646	2787529	0,9670	2798761,001	0,9709	11232,0014
36	2658656	2508545	0,9435	2588999,213	0,9738	80454,2128
37	2817192	2667081	0,9467	2738592,343	0,9721	71511,3432
38	2618181	2566844	0,9803	2567650,107	0,9807	806,1067
39	2097396	2029177	0,9674	2068032,456	0,9860	38855,4560
40	2597700	2540141	0,9778	2531198,880	0,9744	-8942,1200
41	2718228	2686780	0,9884	2677182,757	0,9849	-9597,2428
42	2379541	2306316	0,9692	2339088,803	0,983	32772,8030
43	2519286	2419778	0,9605	2454036,493	0,9741	34258,4926
44	2575800	2493761	0,9681	2507026,140	0,9733	13265,1400
45	2497225	2418204	0,9683	2414067,408	0,9667	-4136,5925
46	2817360	2693446	0,9560	2750588,568	0,9763	57142,5680
47	2484896	2388374	0,9611	2414821,933	0,9718	26447,9328
48	3007466	2943027	0,9785	2933482,336	0,9754	-9544,6636
49	2868953	2819262	0,9826	2815016,684	0,9812	-4245,3164
50	2234752	2161976	0,9674	2194973,414	0,9822	32997,4144

En las gráficas 6 y 7, se comparan las mejores respuestas obtenidas para todos los casos con la mejor respuesta conocida en la literatura.

En la tabla 5 y la figura 8, se presenta un resumen de los resultados obtenidos con base en el porcentaje de mejora. Se observan respuestas de mejor calidad para la gran mayoría de los casos reportados en [11] y los no superados quedaron muy próximos a la mejor respuesta conocida.

Tabla 5 Resumen de respuestas

Resumen de Respuestas	*Número de casos*
Soluciones no superadas	8
Soluciones superadas entre 0 y 1 %	21
Soluciones superadas entre 1 y 2 %	14
Soluciones superadas entre 2 y 3 %	3
Soluciones superadas entre el 3 y 4 %	3
Soluciones superadas en mas de 4 %	1

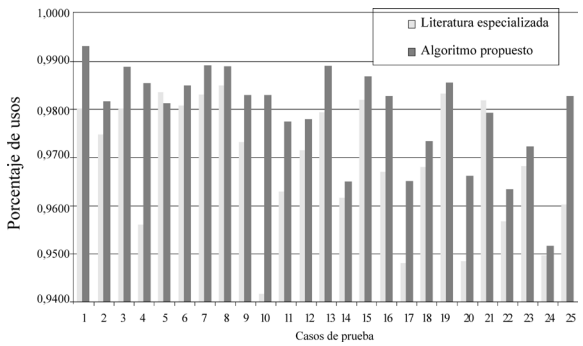


Figura 6 Comparación resultados del 1 al 25

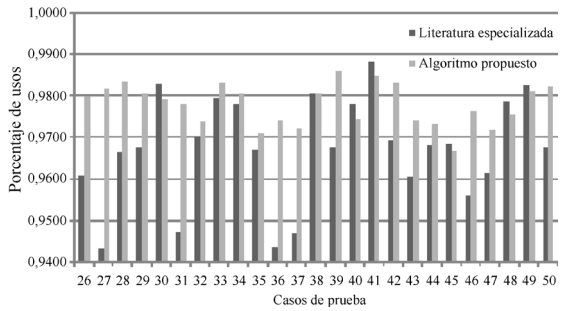


Figura 7 Comparación de resultados del 26 al 50

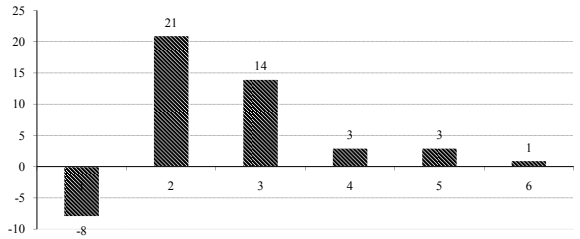


Figura 8 Resumen de respuestas

Con el fin de conocer la eficiencia del algoritmo propuesto se realizaron 90 corridas para el caso 25 cuyos resultados fueron agrupados según la siguiente distribución de frecuencias que se muestran en la tabla 6.

Para realizar el ajuste a una distribución de probabilidad fue usado el programa crystall ball 2000.2.2 obteniéndose el resultado mostrado en la figura 9.

Tabla 6 Distribución de frecuencias caso 25

$[l_{i-1}, l_i]$		x_i	n_i	*Probabilidad de ocurrencia*
[95,1409	95,9543]	95,5476	6	6,67 %
(95,9543	95,7678]	95,8611	4	4,44 %
(95,7678	96,0813]	95,9245	8	8,89 %
(96,0308	96,3949]	96,2381	5	5,56 %
(96,3949	97,7084]	96,5516	13	14,44 %
(97,7084	97,0219]	96,8652	5	5,56 %
(97,0219	97,3354]	97,1787	25	27,78 %
(97,3354	97,6489]	97,4922	7	7,78 %
(97,6489	97,9625]	97,8057	13	14,44 %
(97,9625	98,2760]	98,1192	4	4,44 %

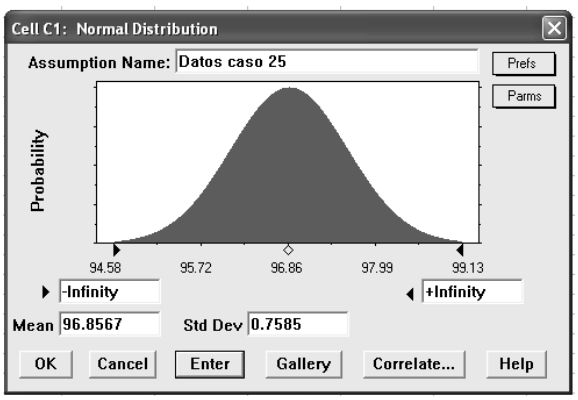


Figura 9 Distribución de probabilidad caso 25

Esta distribución pertenece a una normal con parámetros:

$$\mu = 96{,}8567 \quad \sigma = 0{,}7585$$

Con base en estos parámetros se calcula la probabilidad de encontrar un valor igual o mejor a la mejor solución reportada en la literatura que para este caso es de 96,019

$$P\left(X > 96,0197\right) = P\left(Z > \frac{96,0197 - 96,8567}{0,7585}\right) = P\left(Z > -1,10\right) = 1 - P\left(Z < -1,10\right)$$

$$P\left(Z > -1,10\right) = 0,8643$$

De acuerdo con lo anterior hay una probabilidad de 86,43% de obtener una respuesta mejor o igual a la reportada en la literatura.

Finalmente, se desea mostrar la mejor solución lograda por el algoritmo propuesto, se presenta la solución al caso 10 para el cual se obtuvo una solución 4,16 % mejor a lo reportada. En la figura 10 se presenta la configuración óptima en donde las áreas en negro representan el material sin utilizar.

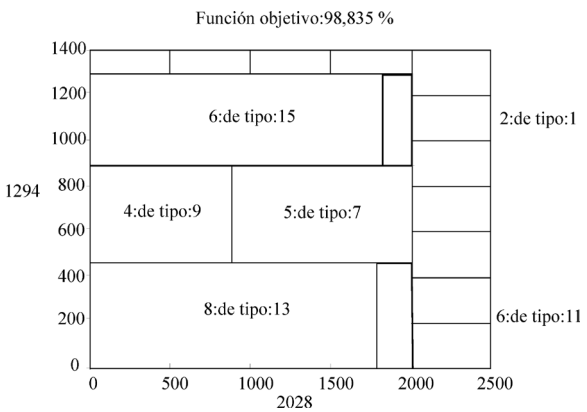


Figura 10 Solución obtenida para el caso 10

Conclusiones y recomendaciones

Se implementó un algoritmo para la solución del problema de corte bidimensional tipo guillotina y se solucionó usando un algoritmo híbrido que combina técnicas inspiradas en búsqueda en vecindario variable, recocido simulado y técnicas constructivas.

Los patrones de corte obtenidos son de fácil implementación práctica, especialmente cuando no se cuenta con máquinas de corte de control numérico ya que se entregan disposiciones homogéneas para cortar las piezas de acuerdo a la demanda.

La escala de los problemas de prueba es comparable con muchos problemas prácticos de corte. Los resultados computacionales fueron satisfactorios comparados con las mejores respuestas conocidas de la literatura especializada.

Se ha realizado un análisis estadístico para demostrar la eficiencia de la metodología propuesta y tener conocimiento sobre el porcentaje de mejora al momento de realizar un ensayo en particular. Para ello se tomaron 90 muestras de un mismo caso, cuyos parámetros se ajustan a una distribución normal. Los resultados obtenidos muestran que el algoritmo propuesto presenta una alta eficiencia en la gran mayoría de los casos estudiados.

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 destaca la importancia de aplicar conjuntamente una codificación eficiente con técnicas metaheurísticas que se adapten adecuadamente a la naturaleza del problema en estudio.

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, formulando el problema de forma que pueda realizarse una reducción del problema a uno de empaquetamiento de rectángulos con algunas consideraciones. Adicionalmente, se desarrolla un método de solución eficiente para resolver este tipo de problemas.

Agradecimientos

Los autores desean expresar su agradecimiento a la Maestría en Investigación de Operaciones y Estadística de la Facultad de Ingeniería Industrial y al grupo de Planeamiento en sistemas eléctricos del programa de Ingeniería Eléctrica de la Universidad Tecnológica de Pereira por su apoyo en la realización de esta investigación.

Referencias

1. S. Jakobs. "Theory and methodology on genetic algorithms for the packing of polygons". *European Journal of Operational Research*. Vol. 88. 1996. pp. 87-100.
2. N. Christofides, A. Whitlock. "An algorithm for twodimensional cutting problems". *Operational Research.* Vol. 25. 1977. pp. 30-44.
3. P. C. Gilmore, R.E. Gomory. "The theory and computation of knapsack functions". *Operations Research*. Vol 15. 1967. pp. 1045-1074.
4. P. Wang. "Two algorithms for constrained twodimensional cutting stock problems". *Operations Research*. Vol. 31. 1983. pp. 573-586.
5. F. A. Vasko. "Computational improvement to Wang's two-dimensional cutting stock algorithm". *Computers and Industrial Engineering*. Vol. 16. 1989. pp. 109-115.
6. J. F. Oliveira, J. S. Ferreira. "An improved version of Wang's algorithm for two - dimensional Cutting Problems". *EJOR 44*. 1990. pp. 256-266.
7. K. Lai, J. Chan. "A evolutionary algorithm for the rectangular cutting stock problem". *International Journal of Industrial Engineering*. Vol 4. 1997. pp.130-139.
8. V. Parada, M. Sepúlveda, A. Gómez. "Solution for the Constrained Guillotine Cutting Problem by Simulated Annealing". *Journal on computers and operations research.* Vol 25. 1998. pp. 37-47.
9. T. W. Leung, C. H. Yung, M. D. Troutt. "Applications of genetic search and simulated annealing to the two-dimensional non-guillotine cutting stock problem". *Computers and Industrial Engineering*. Vol. 40. 2001. pp. 201-214.
10. J. E. Beasley. "A population heuristic for constrained two-dimensional non guillotine cutting". *European Journal of Operational Research.* Vol. 156. 2004. pp. 601-627.
11. C. Yaodong. "An exact algorithm for generating homogenous T-shape cutting patterns". *Computers & Operations Research*. Vol 34. 2007. pp. 1107-1120. Disponible en Internet en: http://www.gxnu.edu.cn/Personal/ydcui/English/Paper.htm. Consultada el 10 de abril de 2007.
12. P. Hansen, M. Nenad, J. Moreno. "Búsqueda de entorno variable. Inteligencia Artificial". *Revista Iberoamericana de Inteligencia Artificial*. Vol. 19. 2003. pp. 77-92.
13. R. Gallego, A. Escobar, R. Romero. *Técnicas de optimización combinatorial*. Textos universitarios. Universidad Tecnológica de Pereira. Pereira. 2006. pp. 27-47.