INGENIERIA INDUSTRIAL

Estudio comparativo de algoritmos basados en cúmulo de partículas para resolver el problema de empaquetamiento en placas

David Álvarez-Martínez[1], Eliana M. Toro[2], Ramón A. Gallego[3§]

[1]*Facultade de Engenharia, Universidade Estadual Paulista Julio de Mesquita Filho, Ilha Solteira, Sao Paulo, Brasil, davidal@utp.edu.co*

[2]*Facultad de Ingeniería Industrial, Universidad Tecnológica de Pereira, Colombia, elianam@utp.edu.co*

[3§]*Programa Ingeniería Eléctrica, Universidad Tecnológica de Pereira, Colombia, ragr@utp.edu.co*

(Recibido: Octubre 22 de 2010 - Aceptado: Mayo 24 de 2011)

Resumen

En este artículo se presenta una metodología basada en técnicas metaheurísticas de optimización para la solución del problema de empaquetamiento óptimo bidimensional de piezas rectangulares en placas, considerando la posibilidad de rotar 90° las piezas y con restricciones de corte tipo guillotina. Este pertenece a la familia de problemas de corte y empaquetamiento considerados clásicos dentro de la investigación de operaciones, siendo de gran aplicación en la industria y caracterizados por su alta complejidad matemática y computacional. Para su solución se usa una codificación en árbol de cortes. En el análisis de resultados se realizó un estudio computacional con tres algoritmos basados en la técnica cúmulo de partículas. Con el fin de verificar la eficiencia de la metodología propuesta basándose en tiempos de cómputo y calidad de respuestas, se tomaron diferentes casos de prueba de la literatura especializada para realizar un benchmarking, que finalmente corroboró el desempeño de la metodología.

Palabras Claves: Empaquetamiento óptimo bidimensional guillotinado en placas, Cúmulo de partículas, Búsqueda en vecindario variable, Recocido simulado.

INDUSTRIAL ENGINEERING

A comparative study of algorithms based on particle swarm optimization for solving the 2D guillotine bin packing

Abstract

This article presents a methodology based on metaheuristic optimization techniques for the solution of the two-dimensional rectangular guillotineable bin packing problem, with and without 90 degrees items rotation. This one belongs to the family of cutting and packing problems, considered classic problems in the operations research, due to its big spectrum of application in the industry and its highly mathematical and computational complexity. An appropriate encoding called slicing tree was developed to work this problem. In the results analysis a computational study was presented using 3 algorithms based on the particle swarm technique. In order to check the efficiency of the presented methodology in terms of the computational time and the quality of the answers, different cases of study from the specialized literature were taken to make a benchmark, that finally corroborated the performance of the proposed methodology.

Keywords: Two-dimensional guillotineable bin packing problem, Particle swarm optimization, Variable neighborhood search, Simulated annealing.

1. Introducción

En el mundo de hoy, la globalización es una realidad que genera escenarios más complejos y competitivos para las empresas, donde para mantenerse se hace necesario pensar en términos de optimización en todos los frentes que componen a la organización, en especial en un panorama donde la escasez toma nuevas dimensiones.

Uno de los principales objetivos de las organizaciones es generar riqueza, los desperdicios representan dinero y tiempo perdido que se refleja en el aumento del precio del producto final y por consiguiente en la competitividad de la empresa, una mala decisión en la utilización de materiales conlleva aumentos significativos en el costo de los mismos, tal como se puede apreciar en la Figura 1.

El problema de empaquetamiento óptimo en placas (más conocido como problema de *bin packing*) consiste en minimizar el material desperdiciado. La solución de este es de gran interés en el sector industrial, comercial y académico. Enunciando algunos trabajos de esta temática con gran aplicación en la industria están relacionados con el problema de diseño de placas de metal Wy y Kim (2010), corte en la industria de la lona para la confección de carpas, toldos para jeep y otros, Farley (1990); corte en la industria del vidrio Dyson y Gregory (1976) y Madsen (1979); corte en la industria de ropas Farley (1988); la perdida residual en el corte de papel corrugado Haessler y Talbot (1983) ; cortes en la industria de la madera Morabito y Garcia (1997) y Venkateswarlu y Martyn (1992); corte en la industria del papel Westernlund *et al*. (1995).

En este documento se analiza el caso específico donde se debe atender completamente la demanda de piezas rectangulares a ser ubicadas en placas idénticas rectangulares y cuyo objetivo es minimizar el número de placas necesarias para realizar el embalaje. Teniendo en cuenta las siguientes consideraciones:

i) Todas las placas tienen las mismas dimensiones, un ancho *W* y un alto *H.*

ii) La orientación de las piezas demandadas, es decir, una pieza de alto *h* y ancho *w* es diferente de una pieza de ancho *w* y alto *h* (sin rotación). Si se considera que las dimensiones (*h*, *w*) y (*w, h*) representan las dimensiones de la misma pieza, se está abordando un problema con rotación.

iii) Los patrones de corte son del tipo guillotina. En estos cada corte produce dos sub-rectángulos y van de un extremo al otro del rectángulo original.

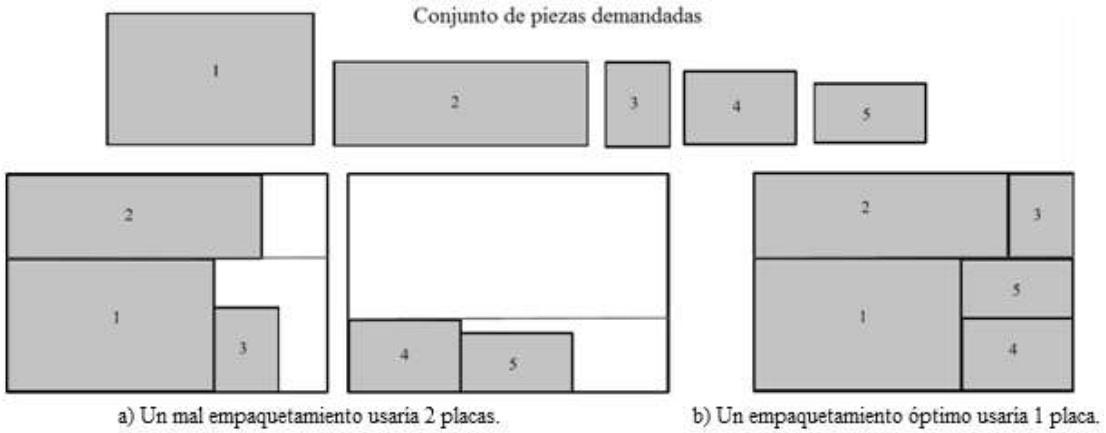

Figura 1. Empaquetamiento inadecuado y adecuado.

Los problemas de corte y empaquetamiento revisten una alta complejidad matemática, estos son considerados NP-Duros en un fuerte sentido. El problema de empaquetamiento óptimo de una dimensión fue probado NP-Duro por Garey y Johnson (1979) y Martello *et al*. (2003). El problema identificado en esta categoría no significa que no puede resolverse y se deben proponer algoritmos de solución que exploten de forma eficiente la estructura matemática del mismo para que se encuentren soluciones a la mayoría de las instancias del problema en tiempos de ejecución relativamente pequeños.

Este problema ha sido ampliamente tratado en diferentes campos de la optimización, como lo son la optimización exacta y la aproximada (heurísticas y metaheurísticas). Lodi *et al*. (1999) y (2002) realizan un resumen del estado del arte del problema de *bin packing*, describen los límites disponibles, algoritmos exactos y aproximados. De los métodos exactos con mejor desempeño está el propuesto por Amico et al. (2002), este determina un límite inferior al problema con rotación y lo resuelve con un algoritmo *branch and bound*. Puchinger y Raidl (2006) consideran un problema típico de manufactura de vidrio: *three-stage two-dimensional bin packing problem* donde el número de cortes guillotina no puede exceder 3 etapas. Estos diseñaron dos modelos lineales enteros de tamaño polinomial y un algoritmo *branch-and-price* basado en una formulación de cubrimiento para el problema bidimensional.

Dentro de los algoritmos heurísticos con mejor desempeño están, *finite best fit strip* (FBS) y *finite first fit* (FFF) presentados por Berkey y Wang (1987) para el problema sin rotación. Lodi *et al*. (1999) adaptan los algoritmos FBS y FFF para el problema con rotación y presenta una aproximación *floor ceiling* (FC). Además, prueban que FC tiene un mejor desempeño que FBS y FFF.

Existe una gran cantidad de algoritmos aproximados que han venido siendo adaptados usando técnicas metaheurísticas. Lodi *et al*. (2002) introducen un algoritmo de búsqueda tabú que explora el espacio de solución independiente del problema haciendo uso de un tamaño y estructura de vecindad dinámica. La búsqueda considera conceptos de heurísticas constructivas. Lodi *et al*. (2002) adaptan una heurística constructiva para el problema con rotación introduciendo el concepto de pseudo-pieza y favorece la orientación vertical de la pieza cuando es posible.

Faroe *et al*. (2003) extiende la aproximación de Dowsland (1993) para el problema sin rotación, utilizando una búsqueda local guiada donde el vecindario es explorado a través de intercambios aleatorios de piezas embaladas de una placa a otra. La infactibilidad de las piezas sobrepuestas es penalizada en la función objetivo. La búsqueda finaliza cuando la incumbente logra un valor propuesto o un periodo de tiempo fijado es alcanzado.

Nuevos algoritmos aproximados han sido diseñados para resolver otras variantes del problema de *bin packing*. Por ejemplo, Binkley y Hagiwara (2007) describen para el problema no-guillotina la heurística de cuatro esquinas que es usada en conjunto con un algoritmo paralelo auto-adaptativo de recocido simulado y un algoritmo genético.

Las técnicas metaheurísticas han mostrado su gran potencial como herramientas de solución en variados campos de aplicación por su eficiencia en cuanto a tiempos de solución y calidad de las respuestas obtenidas. En este trabajo se están proponiendo 3 algoritmos basados en la técnica cúmulo de partículas para la solución del problema *bin packing* con y sin rotación, además de presentar un análisis y una comparación de las respuestas obtenidas respecto a las del estado del arte de la literatura especializada. La estructura de este artículo es la siguiente: descripción del problema, modelo matemático, metodología de solución, análisis de resultados y conclusiones.

2. Descripción del problema

El problema de *bin packing* presenta dos variantes: con y sin rotación de piezas. La primera consiste en cortar de un conjunto de rectángulos que se denominan placas (objetos) de alto H y ancho W, un conjunto de rectángulos de cardinalidad n que se denominan piezas (ítems), de alto h_i y ancho w_i

*(*donde $i = 1,..., n$*)*. Una pieza *(h, w)* es equivalente a una pieza *(w, h)*.

El objetivo dado por la ecuación (1), es minimizar Z_l, el número de placas necesarias para embalar la totalidad de las piezas demandadas, Z_l es una variable binaria que indica si la placa l fue o no usada.

$$\min \quad \sum_{l=1}^{M} Z_l \qquad (1)$$

Sujeto a:

- Las piezas empacadas no deben superar los límites de las placas.
- Las piezas no deben sobreponerse entre ellas mismas.
- Solo es permitido el uso de cortes tipo guillotina.

La segunda variante del problema (sin rotación) difiere en la definición anterior en la condición de orientación de las piezas, por lo tanto una pieza *(h, w)* no es equivalente a una pieza *(w, h)*.

2.1 Modelo Matemático

Ben et al. (2008) sugiere un modelo de programación lineal entera mixta, considerando todas las restricciones descritas en la definición del problema y utilizando un sistema de coordenadas para la ubicación de piezas.

Se asume que existen n piezas que pueden ser cortadas de un conjunto de placas de ancho W y altura H. Para un patrón de corte donde la esquina inferior izquierda de cada pieza k ($k = 1, 2, ..., n$) es ubicada en las coordenadas (α_k, β_k), se pueden obtener dos series $(x_1, x_2, ..., x_n)$ y $(y_1, y_2, ..., y_n)$ al realizar un ordenamiento decreciente de las series $(\alpha_1, \alpha_2, ..., \alpha_n)$ y $(\beta_1, \beta_2, ..., \beta_n)$ respectivamente, se obtiene $0 \le x_1 \le x_2 \le \ldots \le x_n \le W$ y $0 \le y_1 \le y_2 \le \ldots \le y_n \le H$.

Para obtener un modelo lineal entero, es utilizado el siguiente conjunto de variables binarias para representar la ubicación de las piezas en la placa:

$$Z_l = \begin{cases} 1 \text{ si la placa } l \text{ fue usada} \\ 0 \text{ de lo contrario} \end{cases}$$

$$z_{i,j,k,l} = \begin{cases} 1 \text{ si la pieza } k \text{ es empacada en la coordenada} \\ (i,j) \text{ de la placa } l \\ 0 \text{ de lo contrario} \end{cases}$$

Las siguientes variables de decisión intermedias también son necesarias para garantizar que no existan traslapes entre piezas:

$$u_{i,i',l} = \begin{cases} 1 \text{ si la pieza en } (i,j) \text{ no excede} \\ \text{(horizontalmente) } x_{i'} \text{ con } i' > i \text{ en la placa } l \\ 0 \text{ de lo contrario} \end{cases}$$

$$v_{i,j',l} = \begin{cases} 1 \text{ si la pieza en } (i,j) \text{ no excede} \\ \text{(verticalmente) } y_{j'} \text{ con } j' > j \text{ en la placa } l \\ 0 \text{ de lo contrario} \end{cases}$$

Los siguientes tres conjuntos de variables binarias son necesarios para garantizar las restricciones guillotina:

$$p_{i_1,i',j_1,j_2,l} = \begin{cases} 1, \text{si no hay pieza entre, } (i_1, j_1) \text{ y } (i'-1, j_2) \\ \text{que exceda } x_{i'} \text{(consecuentemente, un corte} \\ \text{vertical en } x_{i'} \text{ no cruza ninguna pieza} \\ \text{empacada entre, } (i_1, j_1) \text{ y } (i'-1, j_2)), \ i' > i_1 \\ \text{en la placa } l \\ \\ 0, \text{ de lo contrario} \end{cases}$$

$$q_{i_1,i_2,j_1,j'_2,l} = \begin{cases} 1, \text{si no hay pieza entre, } (i_1, j_1) \text{ y } (i_2, j'-1) \\ \text{que exceda } y_{j'} \text{(consecuentemente, un corte} \\ \text{horizontal en } y_{j'} \text{ no cruza ninguna pieza} \\ \text{empacada entre, } (i_1, j_1) \text{ y } (i_2, j'-1)), \ j' > j_1 \\ \text{en la placa } l \\ \\ 0, \text{ de lo contrario} \end{cases}$$

$$d_{i_1,i_2,j_1,j_2,l} = \begin{cases} 1, \text{ si existe mínimo una pieza empacada} \\ \text{entre } (i_1, j_1) \text{ y } (i_2, j_2-1) \text{ en la placa } l \\ \\ 0, \text{ de lo contrario} \end{cases}$$

La formulación completa del problema de empaquetamiento óptimo bidimensional en placas (sin rotación) es la siguiente:

Minimizar $\sum_{l=1}^{n} Z_l$, (2)

sujeto a $0 \le x_1 \le x_2 \le ... \le x_n$, (3)

$$0 \le y_1 \le y_2 \le ... \le y_n, \tag{4}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} z_{i,j,k,l} \le 1 \quad \forall k, \forall l \tag{5}$$

$$\sum_{i=1}^{n}\sum_{k=1}^{n} z_{i,j,k,l} \le 1 \quad \forall j, \forall l \tag{6}$$

$$\sum_{j=1}^{n}\sum_{k=1}^{n} z_{i,j,k,l} \le 1 \quad \forall i, \forall l \tag{7}$$

$$Z_l \ge z_{i,j,k,l} \quad \forall i, \forall j, \forall k, \forall l \tag{8}$$

$$x_{i'} - x_i - \sum_{k=1}^{n} w_k z_{i,j,k,l} \ge \left(u_{i,j,i',l} - 1\right)W \quad \forall i, \forall j, \forall l, \forall i' > i, \tag{9}$$

$$x_{i'} - x_i - \sum_{k=1}^{n} w_k z_{i,j,k,l} \ge u_{i,j,i',l} W \quad \forall i, \forall j, \forall l, \forall i' > i, \tag{10}$$

$$y_{j'} - y_j - \sum_{k=1}^{n} h_k z_{i,j,k,l} \ge \left(v_{i,j,i',l} - 1\right)H \quad \forall i, \forall j, \forall l, \forall j' > j, \tag{11}$$

$$y_{j'} - y_j - \sum_{k=1}^{n} h_k z_{i,j,k,l} \ge u_{i,j,j',l} H \quad \forall i, \forall j, \forall l, \forall j' > j, \tag{12}$$

$$\left(1 - d_{i_1,i_2,j_1,j_2,l}\right)n \ge \sum_{i=i_1}^{i_2}\sum_{j=j_1}^{j_2}\sum_{k=1}^{n} z_{i,j,k,l} - 1 \quad \forall i_1, \forall j_1, \forall l, \forall i_2 > i_1, \forall j_2 > j_1, \tag{13}$$

$$p_{i_1,i',j_1,j_2,l} \le \sum_{j=j_1}^{j_2}\sum_{k=1}^{n} z_{i',j,k,l} \quad \forall i_1, \forall j_1, \forall l, \forall j_2 > j_1, \forall i' > i_1, \tag{14}$$

$$p_{i_1,i',j_1,j_2,l} \le \sum_{i=i_1}^{i'-1}\sum_{j=j_1}^{j_2}\sum_{k=1}^{n} z_{i,j,k,l} \quad \forall i_1, \forall j_1, \forall l, \forall j_2 > j_1, \forall i' > i_1, \tag{15}$$

$$\left(i' - i_1\right)\left(j_2 - j_1 + 1\right)p_{i_1,i',j_1,j_2,l} \le \sum_{i=i_1}^{i'-1}\sum_{j=j_1}^{j_2} u_{i,j,i',l} \quad \forall i_1, \forall j_1, \forall l, \forall j_2 > j_1, \forall i' > i_1, \tag{16}$$

$$q_{i_1,i_2,j_1,j',l} \le \sum_{i=i_1}^{i_2}\sum_{k=1}^{n} z_{i,j',k,l} \quad \forall i_1, \forall j_1, \forall l, \forall i_2 > i_1, \forall j' > j_1, \tag{17}$$

$$q_{i_1,i_2,j_1,j',l} \le \sum_{i=i_1}^{i_2}\sum_{j=j_1}^{j'-1}\sum_{k=1}^{n} z_{i,j,k,l} \quad \forall i_1, \forall j_1, \forall l, \forall i_2 > i_1, \forall j' > j_1, \tag{18}$$

$$\left(j' - j_1\right)\left(i_2 - i_1 + 1\right)q_{i_1,i_2,j_1,j',l} \le \sum_{i=i_1}^{i_2}\sum_{j=j_1}^{j'-1} v_{i,j,i',l} \quad \forall i_1, \forall j_1, \forall l, \forall j_2 > i_1, \forall j' > j_1, \tag{19}$$

$$d_{i_1,i_2,j_1,j_2,l} + \sum_{i'=i_1+1}^{i_2} p_{i_1,i',j_1,j_2,l} + \sum_{j'=j_1+1}^{j_2} q_{i_1,i_2,j_1,j',l} \ge 1 \quad \forall i_1, \forall j_1, \forall l, \forall i_2 > i_1, \forall j_2 > j_1, \tag{20}$$

$$W \geq x_i + \sum_{j=1}^{n}\sum_{k=1}^{n} w_k z_{i,j,k,l} \quad \forall i, \tag{21}$$

$$H \geq y_j + \sum_{i=1}^{n}\sum_{k=1}^{n} h_k z_{i,j,k,l} \quad \forall j, \tag{22}$$

$$z_{i,j,k,l} \in \{0,1\} \quad \forall i, \forall j, \forall k, \forall l \tag{23}$$

$$u_{i,j,i',l} \in \{0,1\} \quad \forall i, \forall j, \forall l, \forall i' > i, \tag{24}$$

$$v_{i,j,j',l} \in \{0,1\} \quad \forall i, \forall j, \forall l, \forall j' > j, \tag{25}$$

$$d_{i_1,i_2,j_1,j_2,l}, p_{i_1,i_2,j_1,j_2,l}, q_{i_1,i_2,j_1,j_2,l} \in \{0,1\} \quad \forall i, \forall j, \forall l, \forall i_2 > i_1, \forall j_2 > j_1, \tag{26}$$

En la formulación anterior, las restricciones (5), (6) y (7) garantizan que cada posición horizontal o vertical sea ocupada por exactamente una pieza y cada pieza es empacada exactamente una vez. Con la restricción (8) se garantiza la generación de placas suficientes para embalar la totalidad de las piezas. Las restricciones (9)-(12) tienen como objetivo evitar traslapes entre piezas.

Las restricciones (13)-(20) determinan cortes tipo guillotina en cada área rectangular. Si las restricciones guillotina son satisfechas para cada área rectangular, entonces también se cumple para todo el patrón de corte.

Finalmente las restricciones (21) y (22) garantizan que ninguna pieza exceda horizontalmente el ancho *W* y que ninguna pieza exceda verticalmente la altura *H,* considerando que la función objetivo es minimizar el número de placas necesarias para embalar todas las piezas demandadas (ver ecuación (2)).

En este modelo existen muchas restricciones y variables binarias (cerca de $3n^4/4$ variables binarias y $2n^4$ restricciones). Sin embargo gran parte de las restricciones no son activadas en forma simultánea (por ejemplo, existe siempre una restricción redundante entre (9) y (10)).

El modelo presenta una alta complejidad matemática que en la práctica lo hace inexplotable a través del software y hardware actual disponible para la programación entera mixta (para más detalle de este modelo ver Ben *et al*. (2008)).

3. Metodología

3.1 Codificación

Wong *et al*. (1988) presentan un tipo de codificación de datos para el problema de diseño de planta a través de una estructura de árbol de cortes. Una de las grandes ventajas de la aplicación de esta representación en el problema de empaquetamiento, es la generación de patrones de corte tipo guillotina. Diferentes metodologías propuestas han corroborado la efectividad de la codificación en árbol de cortes, en especial la presentada por Kröger (1995), Cui (2007) y Toro *et al*. (2008). El éxito de las estructuras tipo árbol de cortes, es su capacidad de representación de datos jerárquicos y fácil adaptación de las técnicas numéricas de conglomerados (clusters).

Un árbol de cortes se define como: un árbol con raíz, donde cada nodo interno (padre) representa la posición y la forma como se realiza el corte sobre el material (horizontal o vertical), mientras los nodos hoja (nodos terminales) representan las dimensiones de los sub-espacios generados para cortar las piezas agrupadas.

La aplicación de técnicas de conglomerados permite agrupar las piezas a cortar en grupos de igual forma geométrica. En los problemas donde la rotación de piezas es permitida se generan $2n$ (donde n es el número de tipos de piezas disponibles) para su posterior agrupación.

Diferentes propuestas que usan codificación en árbol de cortes realizan el proceso de búsqueda

sobre todos los posibles arboles, lo que da lugar a un espacio de búsqueda de gran tamaño. Propuestas como Toro *et al*. (2008) delimitan y reducen el número de niveles y nodos por nivel del árbol de búsqueda obteniendo un espacio de búsqueda menor y con bajo riesgo de eliminar el árbol óptimo, en este el árbol de cortes se restringe al uso de árboles binarios completos con tres niveles.

La Figura 2 ilustra el árbol de cortes y la disposición de las piezas sobre los sub-espacios resultantes para el problema de empaquetamiento en placas. En cada sub-espacio generado se deben ubicar las piezas con igual dimensión, a fin de maximizar el área utilizada y conservar las restricciones tipo guillotina. En este árbol de cortes, los nodos que lo constituyen identifican la orientación (horizontal o vertical) y la distancia a la cual debe realizarse el corte sobre la placa.

3.2 Cálculo de la función objetivo por placa

Obtenidos los sub-espacios se debe realizar la ubicación de las piezas. Este proceso se lleva a cabo usando el algoritmo constructivo mejor-ajuste *(best-fit)*, debido a que, conserva las restricciones tipo guillotina y embala la mayor cantidad de piezas por sub-espacio.

$$\max\left\{\min\left\{\left\lfloor\frac{W_j}{w_i}\right\rfloor\cdot\left\lfloor\frac{L_j}{l_i}\right\rfloor, Piezas\ disponibles\ tipo\ i\right\}\cdot w_i\cdot l_i\right\}$$
$$\forall i,\forall j; i=1,2,\cdots,n; j=1,2,\cdots,8 \qquad (27)$$

El algoritmo constructivo *best-fit* consiste en encontrar el conjunto de piezas idénticas que maximiza la utilización del área del sub-espacio *j*, sujeto a la restricción del número de piezas *i* disponibles. La ecuación (27) expresa formalmente el algoritmo *best-fit*. El cálculo de la función objetivo consiste en aplicar la ecuación (27) a cada sub-espacio generado por el árbol de cortes (donde W_j es el ancho y L_j es el largo del sub-espacio *j*, mientras w_i y l_i son el ancho y largo de las piezas demandadas).

La función objetivo general del problema es el número de placas necesarias hasta realizar el embalaje completo de las piezas demandadas. Por lo tanto, la idea es una metodología incremental y secuencial en la cual se van utilizando placas hasta suplir la demanda de piezas.

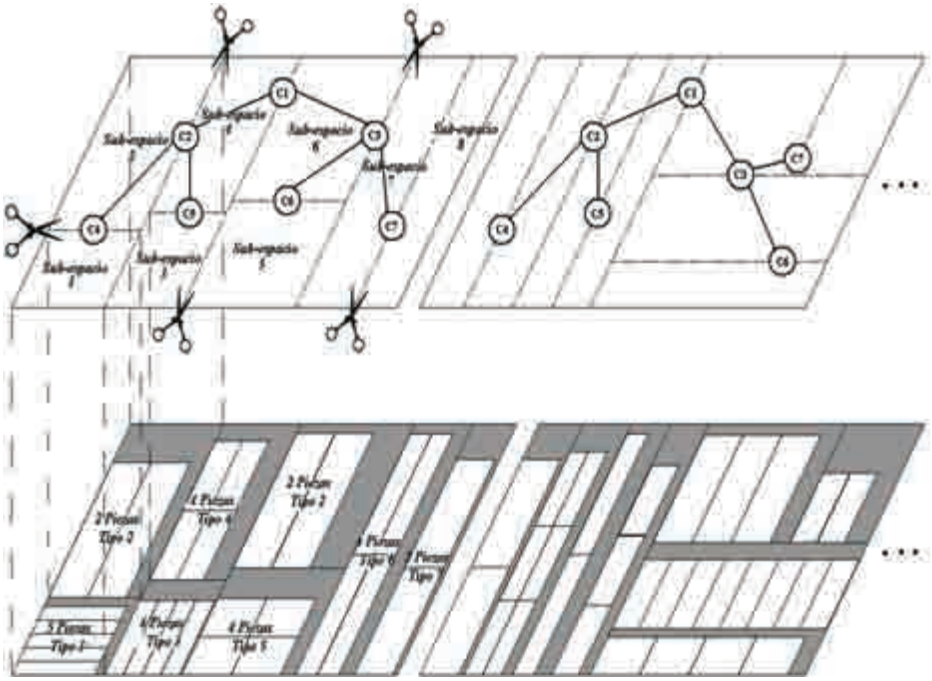

Figura 2. El árbol de cortes y disposición de las piezas.

3.3 Metodología de solución

La codificación propuesta en este estudio garantiza la factibilidad en cuanto a las restricciones de corte tipo guillotina. Este tipo de codificación y cálculo de la función objetivo permiten llevar las restricciones tecnológicas (traslapes, cortes guillotina y límites de la placa) del problema de forma implícita.

Como se mencionó el árbol de cortes se representa por dos árboles independientes entre sí, el primero representa la orientación de cortes y el segundo las distancias de los cortes (representados por las variables *O* y *D*). El esquema de optimización para una placa de material se ilustra en la Figura 3. El Algoritmo *I* realiza una búsqueda exhaustiva sobre el árbol *O,* el Algoritmo *II* recibe los árboles *O* y determina las distancia óptimas de estos y corresponde a la aplicación de las técnicas metaheurísticas.

Dado que el esquema de la Figura 3 representa el proceso de optimización realizado por placa, se deben resolver *l* problemas consecutivos, siendo *l* el número mínimo de placas necesarias para embalar la totalidad de las piezas. Para esto el esquema de optimización consiste en repetir los algoritmos *I* y *II* actualizando el inventario de piezas utilizadas en cada placa, por lo tanto la demanda de piezas va siendo satisfecha mientras evoluciona el proceso.

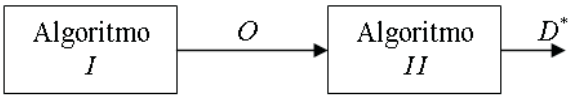

Figura 3. Esquema de optimización por placa.

Algoritmo I
Este algoritmo genera los posibles árboles de orientación de los cortes. Debido a que el árbol es restringido a sólo el uso de árboles binarios completos de 3 niveles, el resultado de este algoritmo siempre dará como resultado 128 diferentes árboles de orientación, cada uno de estos es usado como dato de entrada para el algoritmo *II.*

Algoritmo II
En este algoritmo se hace uso de las técnicas metaheurísticas de optimización, tal como se mencionó, se proponen tres diferentes algoritmos de optimización:

•Algoritmo Optimización Cúmulo de Partículas.

•Algoritmo Híbrido Optimización Cúmulo de Partículas y Búsqueda en Vecindario Variable.

•Algoritmo Híbrido Optimización Cúmulo de Partículas, Recocido Simulado y Algoritmo Genético.

3.4 Algoritmo Optimización con Cúmulo de Partículas (A_{PSO})

El algoritmo optimización con cúmulo de partículas (*Particle Swarm Optimization*, PSO) usa un simple mecanismo que imita el comportamiento de las bandadas de pájaros y los cardumes de peces en cuanto a la forma en que guían sus desplazamientos a fin de encontrar alimento y refugio, considerados como su objetivo primordial (Kennedy y Eberhart, 1995; Zhi-Hui *et al*. 2009).

Este algoritmo puede ser computacionalmente ineficiente al quedar atrapado fácilmente en óptimos locales cuando resuelve problemas como el de corte y empaquetamiento óptimo, cuyo espacio de solución es multimodal; esto ha hecho que el campo de aplicación de la metodología este un poco limitado. Sin embargo, acelerar la velocidad de convergencia y evitar los óptimos locales se han convertido en objetivos importantes en la investigación del PSO, (Liang *et al*. 2006).

Para mejorar la eficiencia y acelerar el proceso de búsqueda es fundamental determinar el estado de evolución y la selección de parámetros adecuados, con el fin de evitar la convergencia prematura, situación que genera óptimos locales de mala calidad. Algunas modificaciones se han planteado introduciendo operadores como la selección, cruzamiento, mutación y búsqueda local.

En PSO, las partículas representan soluciones potenciales, a cada partícula *i* se asocian dos vectores, velocidad $V_i = [v_i^1, v_i^2, ..., v_i^M]$ y posición $X_i = [x_i^1, x_i^2, ..., x_i^M]$ (siendo *M* el número de características que determinan el tamaño del

espacio de solución). La velocidad y la posición de cada partícula son inicializadas con vectores generados aleatoriamente en un rango (de 0 a 1) con una distribución uniforme. Durante el proceso de evolución, la velocidad y la posición de la partícula *i* se actualizan mediante las expresiones (28) y (29) respectivamente.

$$v_i^m = w \cdot v_{i-1}^m + c_1 \cdot rand_1^m \cdot \left(pbest_i^m - x_i^m\right) + c_2 \cdot rand_2^m \cdot \left(gbest_i^m - x_i^m\right) \quad (28)$$

$$x_i^m = x_{i-1}^m + v_i^m \quad (29)$$

En la población, cada partícula representa un árbol de alternativa (D_i), es decir, cada partícula está compuesta por un árbol binario completo de 7 nodos, donde cada nodo toma un valor de cero a uno, representando la distancia en proporción en la cual debe ser realizado el corte sobre la placa. El cálculo de la función objetivo para cada placa se realiza como fue definida anteriormente y como indica la metodología el algoritmo PSO es ejecutado para cada placa hasta que toda la demanda de piezas sea satisfecha. A diferencia de otros algoritmos de optimización, el PSO es del tipo elitista al conservar dentro de la población la configuración incumbente, la cual es actualizada en cada generación.

El algoritmo A_{PSO} propuesto se inicializa con una población generada de forma aleatoria (Paso 1), e identificada la mejor función objetivo de la población denominada líder (*gbest*) (Paso 2). Para el primer ciclo generacional calcular el *pbest (*mejor posición que ha visitado la partícula *i)* (Paso 3.1), de lo contrario se analiza si el *pbest* de cada partícula debe cambiar, es decir, si la nueva posición de la partícula es de mejor calidad que su anterior (Paso 3.2.1), además, se debe calcular si el líder de la población actual ha cambiado (actualizar el *gbest,* Paso 3.2.2). Por último se calcula la velocidad de cada partícula y se actualiza su posición (Paso 4). Estos pasos se repiten hasta un número máximo de ciclos generacionales (Paso 5). El diagrama del algoritmo A_{PSO} se presenta en la figura 4.

3.5 Algoritmo Híbrido Optimización con Cúmulo de Partículas y Búsqueda en Vecindario Variable ($A_{PSO+VNS}$)

El segundo algoritmo es una combinación entre el PSO y la búsqueda en vecindario variable. El algoritmo PSO presenta parámetros que deben ser calibrados, la inadecuada calibración de estos puede conducir el proceso a una convergencia prematura. Con el fin de dar solución a este problema, se propone una modificación al algoritmo básico PSO introduciendo en la metodología un procedimiento basado en la búsqueda en vecindario variable. En este se controla el número de nodos del árbol de distancias a los cuales se les actualiza la posición, lo cual hace que el algoritmo reduzca su velocidad de convergencia evitando quedar atrapado en óptimos locales, también permite visitar otras regiones del espacio de solución que sean interesantes.

El vecindario N_k (donde, $k \in \{1,2,..., k_{max}\}$; siendo, k_{max} el número de nodos padres del árbol de cortes) se define como *k* nodos padres aleatorios del árbol de cortes que deben mudar de valor, si:

k=1, entonces, N_1 = {nodo i_1 debe cambiar de valor; donde i_1 es un número aleatorio entre [$1 - k_{max}$]}

k=2, entonces, N_2 = {nodo i_1 y nodo i_2 deben modificar su valor; donde i_1 y i_2 son números aleatorios entre [$1-k_{max}$] e $i \neq i_2$}

k=l, entonces, N_l = {nodo i_k, $\forall$ *k* mudan de valor; siendo, *k = 1, 2,..., l;* donde i_k es un número aleatorio entre [$1 - k_{max}$] e $i_1 \neq i_2 \neq ... \neq i_l$}

Por lo tanto, el conjunto de vecindarios resultante es $N = \{N_1, N_2,..., N_l,..., N_{kmax}\}$.

La inclusión de este procedimiento en la metodología presenta como desventaja el incremento en el tiempo computacional y como ventaja la mejora en la diversificación de la población. Lo anterior se logra dado que en cada iteración solo se calculan las velocidades de los nodos que hacen parte del vecindario de la partícula. Al igual que el VNS Básico

Paso 1.	Inicializar Parámetros: Variable_Tamaño de la población, Variable_w (Inercia), Variable_C1 (Conocimiento Individua)l, Variable_C2 (Conocimiento Grupal), Variable_ Número de Ciclos
	Hacer Variable_t = 1
	Hacer Variable_Población = Función_Generar_Población_Inicial (Variable_Tamaño de la población)
	Ir al Paso 2
Paso 2.	Hacer Variable_Gbest = Función_Seleccionar_Lider (Variable_Población)
	Ir al Paso 3
Paso 3.	Si Variable_t = 1 ir al Paso 3.1. De lo contrario ir al Paso 3.2
Paso 3.1.	Hacer Variable_Pbest.Particula $_i$ = Variable_Población.Particula $_i$
	Ir al Paso 4
Paso 3.2.	Si Función_Calcular_Función_Objetivo(Variable_Población.Particula $_i$) es mayor que Función_Calcular_Función_Objetivo(Variable_Pbest.Particula $_i$) ir al Paso 3.2.1. De lo contrario ir al Paso 4
Paso 3.2.1.	Hacer Variable_Pbest.Particula$_i$ = Variable_Población.Particula$_i$
	Si Función_Calcular_Función_Objetivo (Variable_Población.Particula $_i$) es mayor que Función_Calcular_Función_Objetivo (Variable_Gbest) ir al Paso 3.2.2. De lo contrario ir al Paso 4
Paso 3.2.2.	Hacer Variable_Gbest = Variable_Población.Particula$_i$
	Ir al Paso 4
Paso 4.	Ecuación (28) - Hacer Variable_velocidad = Función_Calcular_Velocidad (Variable_w, Variable_Población, Variable_Pbest, Variable_Gbest, Variable_C $_1$, Variable_C $_2$)
	Ecuación (29) - Hacer Variable_Población = Función_Actualizar_Posición (Variable_ Población)
	Hacer Variable_t = Variable_t + 1
	Ir al Paso 5
Paso 5.	Si Variable_t = Variable_Número de Ciclos, terminar algoritmo. De lo contrario ir al Paso 2

Figura 4. Pasos del algoritmo A_{PSO}

(Mladenovic y Hansen (1997)) si la nueva partícula no presenta mejora se muda de vecindario a uno más grande, de lo contrario su vecindario cambia por la mínima vecindad definida (alterar solo un nodo). La combinación de estos algoritmos mejora significativamente la calidad de la solución.

El algoritmo híbrido PSO y VNS propuesto inicia definiendo un conjunto de vecinos para cada partícula, iniciando con un vecindario reducido. La población inicial se genera de forma aleatoria (Paso 1), el líder (*gbest*) de la población es identificado (Paso 2). Si es el primer ciclo generacional entonces debe calcularse el *pbest (*mejor posición que ha visitado la partícula *i)* (Paso 3.1). De lo contrario, se analiza si la nueva posición de la partícula es de mejor calidad que la anterior (Paso 3.2) y actualiza el *pbest,* el vecindario de la partícula se regresa al mínimo (Paso 3.2.1), además, se debe calcular si el líder de la población actual ha mudado (actualizar el *gbest,* Paso 3.2.3). De lo contrario, el vecindario de la partícula se promueve al siguiente (Paso 3.2.2). Por último, se calcula la velocidad de cada partícula solo para las características que hacen parte del vecindario y actualizada su posición (Paso 4). Estos pasos se repiten hasta un número

máximo de ciclos generacionales (Paso 5). La Figura 5 ilustra el diagrama del algoritmo $A_{PSO+VNS}$, en la Figura se expresa la función Calcular_Velocidad_Solo_Nodos_Vecindario, esta consiste en evaluar la ecuación (28) sólo en los nodos que pertenezcan al vecindario actual de cada partícula.

3.6 Algoritmo Híbrido Optimización con Cúmulo de Partículas con el operador mutación ($A_{PSO+Mutación}$)

El algoritmo propuesto se inspira en las características del cúmulo de partículas (PSO), recocido simulado (SA, Kirkpatrick, *et al.* 1983) y algoritmos genéticos (GA, Holland 1975).

El PSO opera como algoritmo maestro para guiar el proceso de búsqueda; el SA y GA se usan como estrategias para realizar la intensificación y diversificación en el proceso de exploración del espacio de soluciones, efectuando cambios en las posiciones de las partículas usando la filosofía de la mutación de los genéticos y la temperatura del recocido simulado.

La ecuación (30) define el mecanismo de transición, en esta se involucran: el valor actual del nodo *i* (valor de la característica *i* de una partícula de la población) del árbol de distancias, el número de iteración actual, el número de iteraciones totales y un Épsilon (donde ε, es el mínimo porcentaje para generar un cambio en las distancias. La ecuación (30) se desarrolla emulando el comportamiento de la variable temperatura del algoritmo recocido simulado, con la cual al inicio del proceso se calculan valores que contienen una gran componente aleatoria y al final con una gran componente determinística.

$$nodo\ i \quad nodo\ i \quad rand \quad \frac{1}{2} \quad \sqrt{1 \quad \frac{k}{IteracionesTotales}} \qquad (30)$$

En este algoritmo se incluye el operador de mutación propio de los algoritmos genéticos, (Gallego *et al*. 2006) en el algoritmo PSO, donde la mutación se define como la modificación del valor de un nodo del árbol de distancias a través del mecanismo de transición de la ecuación (30). Esta característica evita la homogenización de la población, a la vez que utiliza un mecanismo especializado de transición entre soluciones que ha presentado excelentes resultados en trabajos como los de Álvarez *et al.* (2009) y (2010).

Dado que el algoritmo PSO presenta algunas similitudes con los algoritmos evolutivos, diferentes autores han propuesto la inclusión del operador mutación en el algoritmo PSO Zhi-Hui *et al*. (2009); Liang *et al*. (2006); Andrews (2006); Carlisle y Dozier (2000) entre otras. Como fue anteriormente mencionado estas operaciones híbridas normalmente se implementan en cada generación Liang *et al*. (2006), Andrews (2006) o en un intervalo prefijado, (Carlisle y Dozier 2000) o son controladas por una función de adaptación definida para el caso específico, en este estudio en cada generación tiene la probabilidad de utilizar el operador de mutación.

El algoritmo A_{PSO} con el operador mutación inicia con una población generada aleatoriamente (Paso 1), se calcula el líder (*gbest*) para esta población (Paso 2). Si es el primer ciclo generacional, calcular el *pbest (*mejor posición que ha visitado la partícula i*)* (Paso 3.1). De lo contrario se analiza si el *pbest* de cada partícula debe cambiar, es decir, si la nueva posición de la partícula es de mejor calidad que la anterior (Paso 3.2.1), además, se debe calcular si el líder de la población actual ha cambiado (actualizar el *gbest,* Paso 3.2.2). Por último, se genera un número aleatorio, si este es menor que la taza de mutación, entonces para una partícula aleatoria de la población es modificada su posición usando la ecuación (30) (Paso 4.1). De lo contrario, se calcula la velocidad de cada partícula y es actualizada su posición (Paso 4.2). Estos pasos se repiten hasta un número máximo de ciclos generacionales (Paso 5). La figura 6 ilustra el diagrama de flujo de datos del algoritmo $A_{PSO+Mutación}$.

3.7 Calibración de parámetros

El ajuste de parámetros es de gran relevancia en el proceso de implementación de una técnica metaheurística porque están directamente relacionados con la calidad de las respuestas encontradas en la solución de un problema específico. En general no existe un método exacto y eficiente para realizar la calibración de parámetros de las diferentes técnicas metaheurísticas, comúnmente estos algoritmos son parametrizados a través de la combinación de una búsqueda exhaustiva y un análisis estadístico de la calidad de los resultados.

Zhi-Hui *et al*. (2009) presentan un rango de valores reducido para los parámetros del algoritmo PSO, el cual se reduce considerablemente el espacio de búsqueda de los valores óptimos.

En este estudio se conserva la filosofía de los operadores de mutación de los algoritmos genéticos, donde la probabilidad de que ocurra una mutación en la población es muy pequeña. Para esto se realizó el mismo proceso de calibración para los rangos propuestos del parámetro de mutación en el trabajo de Gallego *et al*. (2006). Los valores resultantes de la calibración de parámetros se ilustran en la Tabla 1.

4. Análisis de resultados

Con el fin de validar la calidad de las respuestas obtenidas del problema solucionado se seleccionaron 145 casos de la literatura especializada así: 3 casos de prueba clásicos presentados por Christofides y Whitlock (1977), 12 casos de prueba propuestos por Beasley (1985) y 130 casos de prueba tomados de la gran librería

Tabla 1. Valores de parámetros de los algoritmos implementados

Parámetros	Algoritmo		
	A_{PSO}	$A_{PSO+VNS}$	$A_{PSO+Mutación}$
Tamaño de la P oblación	100	100	100
Número de Ciclos	100	100	100
c1 (Conocimiento Individual)	2.05	2.05	2.05
c2 (Conocimiento Grupal)	2.05	2.05	2.05
w (Inercia)	0.6	0.6	0.6
Número de Niveles	3	3	3
Tasa de Mutación			0.03

presentada por Lodi *et al*. (2002) y disponible en línea en Lodi *et al*. (1997). En resumen se usaron 70 casos de prueba que pertenecen a la categoría de problemas de empaquetamiento de gran escala, mientras los 75 restantes pertenecen a las categorías de pequeña y mediana escala (menos de 60 piezas). Diferentes estudios han utilizado estos casos de prueba para realizar sus análisis en una especie de benchmark de las metodologías propuestas, una descripción más detallada del desarrollo de los casos de prueba es presentada en Lodi *et al*. (1997).

Todos los algoritmos se desarrollaron en Delphi 7,0®, sobre un ordenador con unas especificaciones de un procesador Pentium 4 de 3.0 GHz y una memoria RAM de 1 GB. Cada algoritmo se ejecutò 20 veces para todos los 145 problemas, usando los parámetros encontrados con la recombinación realizada experimentalmente. Como la población inicial fue dada mediante una función aleatoria, todas las ejecuciones de cada algoritmo tienen una semilla diferente. La solución obtenida en cada ejecución, se midió con relación a la solución óptima y se compararon los resultados entre los 3 algoritmos.

4.1 Resultados computacionales

En las Tablas 2 y 3 se presentan para todos los casos de prueba la mejor solución alcanzada por los algoritmos propuestos y la mejor solución reportada en la literatura especializada con su respectivo autor. En estas tablas se utilizan diferentes símbolos para representar la calidad de las respuestas obtenidas, para esto se usa la siguiente codificación de los resultados:

El símbolo ^{a} representa una respuesta igual a la reportada en la literatura especializada.
El símbolo ^{b} representa una respuesta mejor que la reportada en la literatura.
El símbolo ^{c} representa una respuesta inferior a la reportada en la literatura.

Además de esto, para los primeros 15 casos de prueba los resultados están dados por el número de placas utilizadas, mientras en los 130 casos de la librería de Lodi *et al.* los resultados están dados como el promedio de cada grupo de 10 casos de la relación, número de placas de la solución sobre número de placas del problema relajado (más conocido como límite inferior).

Tabla 2. Mejor solución obtenida para los problemas de bin packing sin rotación.

Caso	Mejor Solución Obtenida	Mejor Solución Conocida	Autor	Clase	Casos Lodi *et al* Número de Piezas	Mejor Solución Obtenida Límite Inferior	Mejor Solución Conocida Límite Inferior	Autor
CGCUT1	2 [a]	2	Lodi	1	100	1.05 [c]	1.028	Vigo y Martello (1998)
CGCUT2	2 [a]	2	Lodi	3	20	1.18 [c]	1.120	Vigo y Martello (1998)
CGCUT3	23 [a]	23	Lodi	3	100	1.09 [c]	1.086	Vigo y Martello (1998)
NGCUT1	3 [a]	3	Vigo y Martello (1998)	5	20	1.13 [c]	1.040	Vigo y Martello (1998)
NGCUT2	4 [a]	4	Bekrar y Kacem (2008)	5	40	1.09 [c]	1.050	Vigo y Martello (1998)
NGCUT3	3 [a]	3	Bekrar y Kacem (2008)	5	100	1.09 [c]	1.086	Vigo y Martello (1998)
NGCUT4	2 [a]	2	Bekrar y Kacem (2008)	7	40	1.07 [c]	1.066	Vigo y Martello (1998)
NGCUT5	3 [a]	3	Bekrar y Kacem (2008)	7	100	1.04 [c]	1.036	Vigo y Martell (1998)
NGCUT6	3 [a]	3	Bekrar y Kacem (2008)	8	20	1.12 [c]	1.056	Vigo y Martello (1998)
NGCUT7	1 [a]	1	Bekrar y Kacem (2008)	8	100	1.04 [a]	1.040	Lodi
NGCUT8	2 [a]	2	Bekrar y Kacem (2008) Bekrar y Kacem (2008)	9	60	1.01 [c]	1.002	Vigo y Martello (1998)
NGCUT9	3 [a]	3	Bekrar y Kacem (2008)	9	100	1.01 [c]	1.000	Vigo y Martello (1998)
NGCUT10	3 [a]	3	Bekrar y Kacem (2008)	10	40	1.09 [c]	1.056	Vigo y Martello (1998)
NGCUT11	2 [a]	2	Bekrar y Kacem (2008)					
NGCUT12	4 [a]	4						

Tabla 3. Mejor solución obtenida para los problemas de bin packing con rotación.

Caso	Mejor Solución Obtenida	Mejor Solución Conocida	Autor	Casos Lodi *et al*. Número de Clase	Piezas	Mejor Solución Obtenida Límite Inferior	Mejor Solución Conocida Límite Inferior	Autor
CGCUT1	2 [a]	2	Lodi	3	60	1.07 [b]	1.12	Lodi
CGCUT2	2 [a]	2	Lodi	7	20	1.08 [b]	1.17	Lodi
CGCUT3	23 [a]	23	Lodi	7	60	1.06 [b]	1.16	Lodi
NGCUT1	3 [a]	3	Bekrar y Kacem (2008)	7	80	1.06 [b]	1.17	Lodi
NGCUT2	4 [a]	4	Bekrar y Kacem (2008)	8	40	1.07 [b]	1.19	Lodi
NGCUT3	3 [a]	3	Bekrar y Kacem (2008)	8	60	1.06 [b]	1.18	Lodi
NGCUT4	2 [a]	2	Bekrar y Kacem (2008)	8	80	1.05 [b]	1.15	Lodi
NGCUT5	3 [a]	3	Bekrar y Kacem (2008)	8	100	1.04 [b]	1.17	Lodi
NGCUT6	3 [a]	3	Bekrar y Kacem (2008)	9	20	1.00 [a]	1.00	Lodi
NGCUT7	1 [a]	1	Bekrar y Kacem (2008)	9	40	1.01 [a]	1.01	Lodi
NGCUT8	2 [a]	2	Bekrar y Kacem (2008)	9	60	1.01 [a]	1.01	Lodi
NGCUT9	3 [a]	3	Bekrar y Kacem (2008)	9	80	1.01 [a]	1.01	Lodi
NGCUT10	3 [a]	3	Bekrar y Kacem (2008)	9	100	1.01 [a]	1.01	Lodi
NGCUT11	2 [a]	2	Bekrar y Kacem (2008)					
NGCUT12	4 [a]	4	Bekrar y Kacem (2008)					

Como índice de calidad de las respuestas se utilizó el error porcentual (conocido en la literatura como *approximation ratio*), este se define como el porcentaje de desviación de la respuesta obtenida por el algoritmo con respecto a la mejor solución reportada en la literatura. La ecuación (31) define formalmente el error porcentual. Los descriptores estadísticos de la calidad de la solución de cada algoritmo son la media del error (error medio) y la desviación estándar del error. El error medio se calcula como el error promedio de la muestra y la desviación estándar del error se define como la dispersión de los valores respecto al error medio.

$$error \quad \frac{MejorSoluciónConocida \quad SoluciónObtenida}{MejorSoluciónConocida} \tag{31}$$

Los problemas propuestos tendrán un valor de error medio y desviación estándar, que representará la calidad de sus respuestas. Además de esto, interesa el mejor valor de cada muestra (mejor solución alcanzada por muestra). La Tabla 4 resume los errores medios y las desviaciones estándar para cada algoritmo en su respectivo problema.

Tabla 4. Error medio y desviación estándar para bin packing sin y con rotación (porcetanjes).

	Sin Rotación		Con Rotación	
Algoritmo	Error Medio	Desviación Estándar	Error Medio	Desviación Estándar
A_{PSO}	7.01	2.368	1.62	0.121
$A_{PSO+VNS}$	7.21	3.082	2.10	0.740
$A_{PSO+MUTACIÓN}$	7.18	3.042	2.30	0.641

La Tabla 5 presenta los tiempos computacionales requeridos por los algoritmos para cada problema. En esta se ilustran dos tipos de tiempos: totales y promedios por conjunto de problemas.

El tiempo total se define como el tiempo necesario para el algoritmo resolver todos los casos de prueba 20 veces, mientras el tiempo promedio se define como el tiempo necesario para resolver todos los casos de prueba de una categoría (pequeña, mediana o gran escala) sobre el número total de casos de prueba.

4.2 Análisis

En esta sección se realiza un análisis de los algoritmos y de la metodología desarrollada en cada problema. En este se utilizan los descriptores

Tabla 5. Tiempos utilizados por cada algoritmo (segundos).

Problema	Tiempos	Algoritmo PSO	PSO+VNS	PSO+Mutación
Bin packing sin rotación	Tiempo total	1,626,000	1,527,300	1,536,000
	Tiempo promedio (pequeña y mediana escala)	132	125	128
	Tiempo promedio (gran escala)	1,020	957	960
Bin packing con rotación	Tiempo total	2,983,486	2,802,385	2,818,349
	Tiempo promedio (pequeña y mediana escala)	242	229	235
	Tiempo promedio (gran escal a)	1,872	1,756	1,761

estadísticos (error medio y desviación estándar) de la calidad de las respuestas, el tiempo total utilizado por cada algoritmo y se emplean las mejores respuestas obtenidas para cada problema.

4.2.1 Análisis de los algoritmos

En la Tabla 4, se identifica el algoritmo con mejor desempeño para los problemas propuestos en este trabajo. El algoritmo A_{PSO} presenta mejor desempeño como puede observase en los descriptores estadísticos de los problemas *bin packing* con y sin rotación. Los algoritmos $A_{PSO+VNS}$ y $A_{PSO+Mutación}$ presentan un desempeño aceptable.

En la Tabla 5 se evidencia que el algoritmo A_{PSO} requiere de los mayores tiempos de cómputo para alcanzar las respuestas. Mientras los algoritmos $A_{PSO+VNS}$ y $A_{PSO+Mutación}$ presentan desempeños similares, dados sus promedios de calidad y tiempos de respuestas.

4.2.2 Análisis de la metodología desarrollada en cada problema

En la Tabla 6 se resumen y comparan los mejores resultados alcanzados por la metodología propuesta. En esta se identifica que la metodología propuesta para resolver los problemas de *bin packing* presenta un buen comportamiento, debido a que en los problemas donde no se permite la rotación de piezas el 57% de las respuestas reportadas son alcanzadas aunque en el 43% restante no se logran superar. Obteniéndose un comportamiento diferente cuando se permite la rotación de piezas, donde un 71 % de las respuestas son igualadas y el 29% restante de las soluciones superan las reportadas.

Tabla 6. Comparación de resultados de los mejores resultados

Problema	Soluciones Superadas	Soluciones Igualadas	Soluciones Inferiores
Bin packing sin rotación	0	16	12
Bin packing con rotación	8	20	0

Igual ocurre al analizar la tabla 5, los tiempos de respuestas de la metodología propuesta son excelentes para resolver los problemas de *bin packing* de pequeña y mediana escala (hasta 40 piezas) tiempos promedios de 128 y 235 segundos respectivamente. Sin embargo, en los problemas de *bin packing* de gran escala (hasta 100 piezas) con y sin rotación son requeridos unos tiempos promedios relativamente altos de 16 y 30 minutos respectivamente.

En la Tabla 4 se identifica que la metodología propuesta para problemas de *bin packing* con rotación de piezas se tiene error medio de 2% respecto a la mejor respuesta reportada, no muy lejos para los problemas con rotación de piezas con un error medio de 7% respecto a la mejor respuesta reportada.

Dado que en aplicaciones reales se manejan en el proceso de corte o empaquetamiento límites máximos de desperdicio del 30%. Se concluye que aplicando dichas metodologías se obtiene una reducción en las pérdidas del material de más del 50%. El inconveniente en la aplicación de estas metodologías es el alto tiempo de cómputo cuando se requiere resolver el problema en tiempo real.

5. Conclusiones

Se resolvió el problema de empaquetamiento óptimo bidimensional guillotinado en placas, con y sin rotación de las piezas mediante tres

algoritmos: algoritmo cúmulo de partículas, algoritmo híbrido de cúmulo de partículas y búsqueda en vecindario variable y algoritmo híbrido de cúmulo de partículas, recocido simulado y algoritmo genético, obteniendo resultados de buena calidad.

Se adaptó la codificación árbol de cortes presentada para el problema de diseño de planta, combinando un árbol de valores binarios, en el cual se orientan los cortes, con un árbol de valores reales y el cual determina las distancias de los cortes. Esta propuesta de codificación presenta un gran desempeño para este tipo de problemas debido a que logra reducir el espacio de búsqueda con bajo de riesgo de perder soluciones de buena calidad.

La metodología propuesta obtiene particulares resultados, en especial cuando se incluye la posibilidad de rotar las piezas, algunos de estos no están reportados en la literatura especializada. Los errores de desviación de la metodología propuesta respecto a las mejores soluciones reportadas en la literatura no son significativos y corroboran la eficiencia de esta.

La codificación propuesta es general y puede ser usada en metodologías que empleen otras técnicas de optimización tales como: *Branch and bound*, Búsqueda Tabú, GRASP, Colonia de Hormigas, Algoritmos Genéticos, etc.

La metodología propuesta podría adaptarse a problemas como: la mochila bidimensional, *strip packing*, entre otros, además la codificación usada en la solución de este problema podría extenderse a problemas de empaquetamiento en tres dimensiones.

6. Bibliografía

Álvarez D., Gallego R. G. y Toro E. M., (2009). Solución al problema de empaquetamiento óptimo bidimensional en rollos infinitos usando un algoritmo híbrido, *Revista Scientia Et Technica*, 42, 205-211.

Álvarez D., Gallego R. G. y Toro E. M., (2010). Problema de la mochila irrestricta bidimensional guillotinada, *Revista Ingenieria y Universidad*, 14, (2), 327-344.

Amico M. D., Martello S.y Vigo D., (2002). A lower bound for the non-oriented two-dimensional bin packing problem, *Discrete Applied Mathematics,* 118, 13-24.

Andrews P. S., (2006). *An investigation into mutation operators for particle swarm optimization*, IEEE Congr. Evol. Comput., Vancouver, 1044-1051.

Beasley J. E., (1985). An exact two-dimensional non-guillotine cutting tree search procedure, *Operations Research,* 33, 49-64.

Bekrar A. y Kacem I., (2008) *A Comparison study of heuristics for solving the 2D Guillotine Strip and Bin Packing Problems*, IEEE International Conference on Service Systems and Service Management, Melbourne, 1-16.

Ben S., Chu C. y Espinouse M. L., (2008). Characterization and modelling of guillotine constraints, *European Journal of Operational Research*, 191, 112-126.

Berkey J. O. y Wang P. Y., (1987). Two-dimensional finite bin packing algorithms, *Journal of the Operational Research Society*, 38, 423-429.

Binkley K. B. y Hagiwara M., (2007). *Applying self-adaptive evolutionary algorithms to two-dimensional packing problems using a four corners' heuristic*, European Journal of Operational Research, 183, 1230-1248.

Carlisle A. y Dozier G., (2000). Adapting particle swarm optimization to dynamic environments, *in Proc. Int. Conf. Artif. Intell.*, Las Vegas, 429-434.

Christofides N.y Whitlock C., (1977). An algorithm for two-dimensional cutting problems, *Operations Research*, 25, 30-44.

Cui Y., (2007). An exact algorithm for generating homogenous two-segment cutting patterns, *Engineering Optimization*, 39, 365-380.

Dowsland K., (1993). Some experiments with simulated annealing techniques for packing problems, *European Journal of Operational Research,* 68, 389-399.

Dyson R. G. y Gregory A. S., (1974). The cutting stock problem in the flat glass industry. *European Journal of Operation Research, 25*, 41-54.

Faroe O., Pisinger D. y Zachariasen M., (2003). Guided Local Search for the Three-Dimensional Bin-Packing Problem, *Informs Journal On Computing*, 15, 267-283.

Farley A., (1990). The Cutting Stock Problem in the Canvas Industry, *European Journal of Operation Research,* 44, 247-255.

Farley A., (1988). Mathematical Programming Model for Cutting Stock Problems in the Clothing Industry, *Journal of the Operation Research Society,* 39, 41-53.

Gallego R. A., Escobar A. H. y Romero R. A., (2006). *Técnicas de optimización combinatorial*, Textos universitarios, Universidad Tecnológica de Pereira.

Garey M. R. y Johnson D. S., (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, Calif., USA.

Haessler R. W. y Talbot F. B., (1983). A 0-1 Model for Solving the Corrugator Trim Problem, *Management Science,* 29, 200-209.

Holland J. H., (1975) *Adaptation in Natural and Artificial Systems,* University of Michigan Press, Ann Arbor, Michigan; re-issued by MIT Press (1992).

Kennedy J. y Eberhart R., (1995). *Particle Swarm Optimizations.* Proceedings of IEEE Intenational conference on Neural Networks, 1942 -1948.

Kirkpatrick S., Gelatt C., y Vecchi M., (1983). Optimization by simulated annealing, *Science*, 220, 671-680.

Kröger B., (1995). Guillotineable bin packing: A genetic approach, *European Journal of Operational Research*, 84, 645-661.

Liang J. J., Qin A. K., Suganthan P. N. y Baskar S., (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, *IEEE Trans. Evol. Comput.*, 10, 281295.

Lodi A., Martello S. y Monaci M., (2002). Two-dimensional packing problems: A Survey, *European Journal of Operational Research*, 141, 241-252.

Lodi A., Martello S. y Vigo D., (1999). Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems, *INFORMS J. Computing*, 11, 345-357.

Lodi A., Martello S. y Vigo D., (1997) *Problem instances for the two-dimensional Bin Packing Problem*, [Online]. Available: http://www.or.deis.unibo.it/research_pages/ORinstances/2BP.html.

Madsen O. G. B., (1979). Glass Cutting in Small Firm, *Mathematical Programming,* 17, 85-90.

Martello S., Monaci M., y Vigo D., (2003). An exact approach to the strip-packing problem, *INFORMS Journal On Computing*, 15, 310-319.

Morabito R. y Garcia V., (1997). The cutting stock problem in hardboard industry: a case study, *Computer Operation Research,* 25, 469-485.

Mladenovic N. y Hansen P., (1997). Variable neighborhood search, *Computers and Operations Research*, 24, 1097-1100.

Puchinger J. y Raidl G. R., (2006). Models and algorithms for three-stage two-dimensional bin packing, *European Journal of Operational Research,* 183, 1304-1327.

Toro E., Garcés A. y Ruiz H., (2008). Solución al problema de empaquetamiento bidimensional usando un algoritmo híbrido constructivo de búsqueda en vecindad variable y recocido simulado, *Revista Facultad de Ingeniería Universidad de Antioquia*, 46, 119-131.

Venkateswarlu P. y Martyn C. W., (1992). *The Trim loss problem in a wooden drum industry*, Proceeding of the Convention of Operation Research Society of India.

Vigo D. y Martello S., (1998). Exact solution of the two-dimensional finite bin packing problem, *Management Science*, 44, 388-399.

Westernlund T., Isaksoon J. y Harjunkoski I., (1995). *Solving a production optimization problem in the paper industry*, Report 95-146A, Processs Desing Laboratory, Abo Akademi University.

Wy J. y Kim, B. I., (2010). Two-staged guillotine cut, two-dimensional bin packing optimisation with flexible bin size for steel mother plate design, *International Journal of Production Research*, 48, 6799-6820.

Wong D. F., Leong H. W. y Liu C. L., (1988). *Simulated Annealing for VLSI Design*, Kluwer Academic Publishers, 1988.

Zhi-Hui Z., Jun Z., Yun L. y Henry S., (2009). *Adaptive Particle Swarm Optimization*, IEEE Transactions On Systems, Man, And CyberneticsPart B: Cybernetics, 39, 1362-1381.