SOLUCIÓN DEL PROBLEMA DE EMPAQUETAMIENTO ÓPTIMO USANDO TÉCNICAS METAHEURÍSTICAS DE OPTIMIZACIÓN SIMULTÁNEAS A TRAVÉS DE PROCESAMIENTO PARALELO

LUIS MIGUEL ESCOBAR FALCÓN

TESIS PRESENTADA COMO REQUISITO PARA OPTAR AL TITULO DE MAGISTER EM INGENIERÍA ELÉCTRICA

DIRECTOR
M. Sc. ELIANA MIRLEDY TORO OCAMPO

MAESTRÍA EN INGENIERÍA ELÉCTRICA
FACULTAD DE INGENIERIAS ELÉCTRICA, ELECTRÓNICA FÍSICA Y CIENCIAS COMPUTACIONALES
UNIVERSIDAD TECNOLÓGICA DE PEREIRA
PEREIRA, NOVIEMBRE DE 2012

TABLA DE CONTENIDO

LISTA DE FIGURAS

1. INTRODUCCIÓN

1.1. Definición del problema

El problema de la mochila irrestricta bidimensional no guillotinada (U2DNGSKP) del inglés *unconstrained two-dimensional non-guillotine single knapsack problem*, es un problema de corte que se presenta cuando el material a ser utilizado es una pieza rectangular (hoja de material) donde se deben ubicar piezas rectangulares más pequeñas de las que se conoce el tamaño y un costo (bien sea su propia área o un valor establecido). El objetivo es maximizar el beneficio asociado a cada una de las piezas cortadas, sin sobreponer las piezas y sin sobrepasar los límites de la hoja de material.

Las características de este problema son las siguientes:

i) El costo asociado puede o no estar relacionado con el área de la pieza a ser ubicada; si el costo es igual al área de la pieza se está resolviendo el problema sin pesos (*unweighted version*) y si el costo es diferente del área del ítem el problema a resolver será el problema con pesos (*weighted version*).

ii) La orientación de las piezas a ser ubicada es fija, es decir, una pieza de alto h y ancho w es diferente de una pieza de longitud w y alto h (*without rotation*).

iii) Los patrones de corte son tipo no guillotina, es decir, los cortes para generar las piezas no necesariamente van de un extremo al otro del rectángulo original.

iv) No existe un límite máximo del número de piezas a cortar de cada tipo. (*unconstrained version*)

Formalmente el problema se define como: dado un rectángulo mayor de largo L y ancho W (placa o contenedor), y un conjunto de piezas rectangulares agrupadas en n tipos diferentes de longitud l_i, ancho w_i y valor v_i, $i = 1,\ldots,n$, el problema consiste en encontrar un patrón de corte (o empaquetamiento) que maximiza la suma de los valores de las piezas cortadas (empacadas). El patrón de corte es tratado como bidimensional dado que involucra dos dimensiones relevantes, las longitudes y los anchos de las piezas. Un patrón bidimensional factible para el problema es uno en el que las piezas ubicadas en la placa no se traslapen entre sí, tienen que estar completamente por dentro del contenedor y cada pieza debe tener una arista paralela a cualquiera de las aristas del contenedor (patrón ortogonal). En este estudio se asume que no existe un límite superior para el número de veces que cada tipo de pieza puede ser cortada (o empaquetada) de la placa, por lo tanto, el patrón bidimensional es llamado *irrestricto* (0 y $\left\lfloor \frac{L}{l_i} \right\rfloor \left\lfloor \frac{W}{w_i} \right\rfloor$ Resultan ser el mínimo y el máximo obvio de las piezas tipo i en el patrón); de otra manera sería llamada *restricta* o *doblemente-restricta.*
Sin perder generalidad, también se asume que los cortes son infinitamente delgados, la orientación de las piezas es fija (esto quiere decir que una pieza de tamaño (l_i, w_i) es

diferente de una pieza de tamaño (w_i, l_i,) si $l_i \neq w_i$) y además L, W, l_i, w_i son enteros positivos.
Por otra parte, se asume que el patrón de corte bidimensional irrestricto es *no-guillotina* porque no está limitado por los cortes tipo guillotina impuestos por algunas máquinas de corte (un corte ortogonal guillotina en un rectángulo es un corte realizado desde una de sus aristas hasta la arista opuesta, paralelo a la arista restante).

Se ha demostrado que este problema es NP-Duro debido a su alta complejidad matemática (Beasley, 2004). Se tratarán en este trabajo las versiones con pesos y sin pesos para el problema irrestricto. Este conjunto de problemas es aplicable en diversos sectores de la economía, entre los que destacan los sectores de industria, transporte y comercio. Así por ejemplo, su estudio es relevante por su aplicación en diferentes contextos industriales y en logística, como el corte de placas metálicas y de vidrio en tamaños requeridos, el corte de hojas de madera, corte de materiales textiles para hacer piezas con orientación (estampados), carga de diferentes ítems en pallets o la carga de diferentes pallets en camiones o contenedores, corte de cartones en cajas, ubicación de anuncios en páginas de periódicos y revistas, el posicionamiento de componentes en chips durante el diseño de circuitos integrados, entre otras.

1.2. Motivación

Los problemas de corte y empaquetamiento son de interés tanto industrial como académico, avanzar en la temática de corte y empaquetamiento óptimo bidimensional no guillotinado puede maximizar el uso de las materias primas, factor que incide directamente en el costo final del producto y son de aplicación directa en empresas de corte, transporte y almacenamiento de materiales, esto representa una estrategia de mejoramiento que redunda en el nivel de ganancias de una organización de esta naturaleza.

Este estudio esta basado en una revisión del estado del arte de los problemas clásicos de corte y empaquetamiento presenta una nueva metodología para la solución del problema de la mochila bidimensional irrestricta guillotinada, con base en esta pueden ser abordadas muchas otras variantes para este tipo de problemas.

La industria se ha concientizado de la importancia de la investigación y desarrollo en las áreas de investigación operativa e ingeniería de la producción, la carencia en el mercado de aplicativos informáticos que permitan al sector industrial disponer de herramientas que den solución eficiente a sus problemáticas de interés ha sido uno de los desafíos de este proyecto, en este orden de ideas se ha tomado como herramienta para implementar la solución del problema planteado las técnicas metaheurísticas de optimización ya que han mostrado un excelente desempeño en la solución de problemas complejos en diversas áreas de la ingeniería.

1.3. Objetivos

GENERAL

Desarrollar una metodología de solución para el problema de la mochila bidimensional irrestricta con patrones de corte tipo no guillotina, con y sin pesos asociados a las piezas, usando un algoritmo de optimización que combina las principales características de las técnicas metaheurísticas de optimización cúmulo de partículas, vecindario variable y algoritmos genéticos a través de técnicas de procesamiento paralelo.

ESPECÍFICOS

- Revisar el estado del arte de los modelos matemáticos que representan el problema de la mochila bidimensional irrestricta con patrones de corte tipo no guillotina, con y sin pesos asociados a las piezas.
- Revisar el estado del arte de las técnicas de solución utilizadas en la solución de este tipo de problemas.
- Proponer una codificación eficiente que permita la implementación de técnicas heurísticas y metaheurísticas de optimización, en la solución del problema de empaquetamiento.
- Desarrollar una nueva metodología para la solución del problema de la mochila bidimensional irrestricta con patrones de corte tipo no guillotina, con y sin pesos asociados a las piezas.
- Validar las metodologías propuestas con casos de prueba de la literatura especializada.

1.4. Alcance y contribuciones

Las principales contribuciones de este trabajo son las siguientes:

- Se desarrollará una metodología para la solución el problema de la mochila bidimensional irrestricta con patrones de corte tipo no guillotina, con y sin pesos asociados a las piezas.
- Se pretende que los resultados obtenidos por la metodología presentada en este trabajo alcancen las mejores respuestas conocidas hasta el momento, presentes en la literatura especializada.
- Todas las implementaciones realizadas en este trabajo se compararán contra técnicas representativas del estado del arte de optimización con técnicas exactas, heurísticas y metaheurísticas, con el fin de concluir si las metodologías desarrolladas en este trabajo presentan resultados de igual, mejor o peor calidad respecto a los conocidos.

- Se realizará la comparación de la metodología propuesta bajo una arquitectura secuencial contra la misma propuesta híbrida bajo una arquitectura de procesamiento paralelo para determinar las ventajas o desventajas que puede traer este recurso computacional aplicado en este trabajo.

1.5. Estructura general del documento

La organización de este documento es la siguiente:

En el capítulo 2 se plantea el problema de la mochila bidimensional irrestricta con patrones de corte tipo no guillotina, con y sin pesos asociados a las piezas.

En el capítulo 3 se presenta y se explica el modelo matemático del problema tratado en este trabajo, acogido por la comunidad académica.

En el capítulo 4 se realiza una descripción de las técnicas de optimización: búsqueda en vecindario variable, optimización con cúmulo de partículas y factor de turbulencia.

En el capítulo 5 se presenta la codificación del problema, la adaptación de las técnicas de optimización combinatoria para trabajar este caso particular de empaquetamiento y se explicitan las dos arquitecturas utilizadas para ejecutar las implementaciones resultantes (arquitectura secuencial y arquitectura paralela).

En el capítulo 6 se realiza un análisis formal de los resultados obtenidos por las diferentes metodologías presentadas.

En el capítulo 7 se presentan las conclusiones y recomendaciones para trabajos posteriores relacionados.

2. PLANTEAMIENTO DEL PROBLEMA DE LA MOCHILA BIDIMENSIONAL IRRESTRICTA DE PIEZAS RECTANGULARES

2.1. Generalidades de los problemas de la mochila

(Washer *et al*., 2007) caracterizan los problemas de la mochila como problemas de empaquetamiento con un surtido de piezas fuertemente heterogéneo que debe ser ubicado en un contenedor. La cantidad de piezas a empaquetar es ilimitada, por tanto para maximizar el beneficio asociado, se requiere seleccionar un subconjunto de las mismas.

Se estudia la generación de patrones bidimensionales tipo no guillotina, también referida por algunos autores como el problema de la mochila bidimensional o carga de contenedores para distribución bidimensional. Este problema es clasificado como 2/B/O de acuerdo a la tipología Dyckhoff para problemas de corte y empaquetamiento (Dyckhoff, 1990) y como el problema de empaquetamiento de un solo objeto bidimensional rectangular (*Single Large Object Packing Problem* - SLOPP) basado en la tipología (Wäescher *et al*., 2007), también llamada mochila 0-1 (Martello *et al*., 2000; Martello y Toth, 1990) la cual requiere ubicar un conjunto de piezas de áreas dadas y costos asociados dentro de una (*single*) mochila con un límite de volumen y donde el objetivo es maximizar el beneficio total de las piezas embaladas.

Por lo tanto, en este estudio se pretende resolver los problemas de la mochila bidimensional irrestricta, con y sin pesos asociados a las piezas, con orientación fija de las piezas y con patrones de corte tipo no guillotina de primer orden.

2.2. Aplicaciones del problema de la mochila en la industria

Este problema es aplicable en diversos sectores de la economía, entre los que destacan los sectores de industria, transporte y comercio. Así por ejemplo, sus aplicaciones se pueden observar en industrias de perfiles metálicos, corte de maderas, papel, plástico, piel, aluminio o vidrio en donde los componentes rectangulares tienen que ser cortados desde grandes hojas de material; en un contexto de depósitos o almacenes las mercancías deben ser ubicadas en estanterías; en periódicos, artículos y avisos tienen que ser organizados en páginas. La característica especial de estas aplicaciones, son las unidades de representación del material (rectángulos), y tienen funciones objetivo comunes que consisten en que se maximice el beneficio asociado de las piezas cortadas (con pesos) o que se minimice el área inutilizada (sin pesos). Diferente a otros contextos como las empresas papeleras o de industrias textiles donde la unidad de material es un rollo y la función objetivo es cortar todos los ítems usando la mínima dimensión del rollo (*two-dimensional strip packing problem*).

En la industria textil, el proceso de corte se realiza después del diseño de las prendas, el inventario de las telas necesarias y la descripción de los patrones que se desean obtener de cada una de ellas.

En el corte intervienen una o más personas especializadas que son quienes van distribuyendo y cortando los patrones sobre la tela. Para automatizar y optimizar este proceso, el sistema a desarrollar debería cumplir:

- El desperdicio de telas obtenida por el sistema debería ser igual o menor que la obtenida por un experto (humano especializado). El desperdicio producido por un profesional del corte es de aproximadamente el 30% de la tela original (Toro, 2007).
- El tiempo de ejecución para la obtención de un resultado admisible, no debe en ningún caso provocar una demora en el proceso global de producción, pues en tal caso, no proporcionaría ningún beneficio notable para la industria en cuestión.

A la hora de resolver el problema, se deben considerar y aplicar todas las restricciones referentes a la legalidad de los patrones. Por ejemplo, algunos patrones tienen orientación y ésta debe seguir la dirección del material. Además, para los problemas con pesos asociados a las piezas, donde los materiales cuya calidad no es la misma en toda su superficie, por ejemplo, con el cuero y la madera, se suele aprovechar las zonas de mejor calidad para fabricar las piezas de mayor valor.

2.3. Introducción Histórica de los problemas de corte y empaquetamiento

Los problemas de empaquetamiento y corte óptimo (*cutting/packing problems*) son considerados clásicos dentro de la investigación de operaciones. Este se comenzó a plantear y estudiar hace más de 6 décadas. Primero fue estudiado por (Kantorovich, 1939) y luego (Gilmore y Gomory, 1961) trabajaron problemas reales a través de la programación lineal. Gran cantidad de artículos con diferentes aproximaciones y estudios diversos acerca de las posibles variaciones del problema siguen apareciendo.

(Sweeney y Paternoster, 1992) realizaron una completísima recopilación de documentación entre 1940 y 1990 corte y empaquetamiento, (Wäescher *et al*., 2007) entre 1990 y 2005 y por último (Lodi *et al*., 2002) presentan los avances recientes en esta temática.

2.4. Patrones de corte bidimensional

Algunos procesos de corte industrial también limitan la manera de producir patrones de corte guillotina. En la primera etapa los cortes son realizados paralelos a uno de los lados de la placa, luego, en la siguiente etapa, ortogonal a los cortes previos y así sucesivamente. Si existe un límite impuesto al número de etapas, denotado como k, el patrón guillotina es llamado de *k-etapas*, de lo contrario se dice que es sin etapas (el patrón sin etapas es equivalente a definir k lo suficientemente grande).

(Morabito y Morales, 1998) clasifican de los patrones de empaquetamiento: patrones guillotina y no guillotina. Un patrón es de tipo guillotina si se puede obtener por sucesivos cortes de tipo guillotina (ver figura 2.1). Un patrón es no guillotina si es obtenido por sucesivos cortes de guillotina y no guillotina (ver figura 2.2).

Un corte es de tipo guillotina si cuando se aplica 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 del tipo no guillotina.

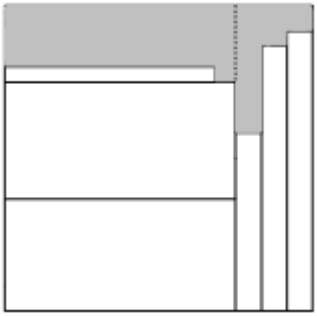

Figura 2.1. Corte Guillotina.

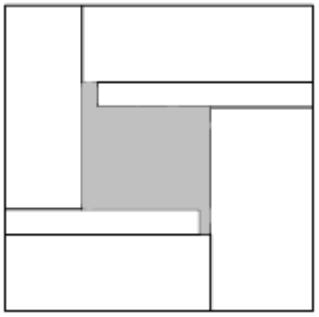

Figura 2.2. Corte no guillotina.

Dentro de los patrones de corte no guillotina podemos encontrar dos tipos. El primero, denominado *corte de primer orden*, consiste en un patrón de corte generado a partir del estampado mostrado en la figura 2.3 junto con cualquier corte guillotina en cualquiera de los subespacios generados. Igualmente un estampado de este tipo aplicado a cualquiera de los subespacios generados (aplicación recursiva del mismo) sigue siendo un corte de primer orden como se ilustra en las figuras 2.4 y 2.5.

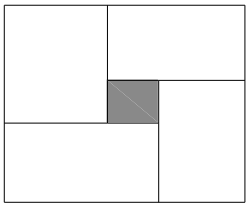

Figura 2.3. Estampado Corte No Guillotina (Primer Orden)

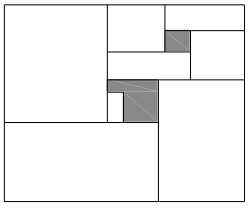

Figura 2.4. Ejemplo Corte Primer Orden Recursivo

Figura 2.5. Ejemplo corte primer orden recursivo con mayor anidación

El segundo tipo, denominado *corte de orden superior*, es un patrón de corte no guillotina que no puede ser alcanzado empleando el estampado de primer orden con cortes no guillotina como se ilustra en la figura 2.6.

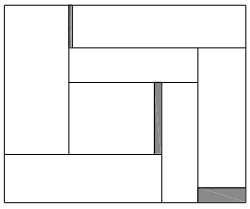

Figura 2.6. Patrón de Corte de Orden Superior

2.5. Extensión a polígonos

(Jakobs, 1996) propone para el problema de corte bidimensional de polígonos una reducción del problema a uno de empaquetamiento de rectángulos, al encontrar los rectángulos con la mínima área que contengan los polígonos, rotándolas alrededor del centro de gravedad (ver figura 2.7). Después de haber encajado cada polígono en rectángulos se resuelve el problema de empaquetamiento bidimensional rectangular.

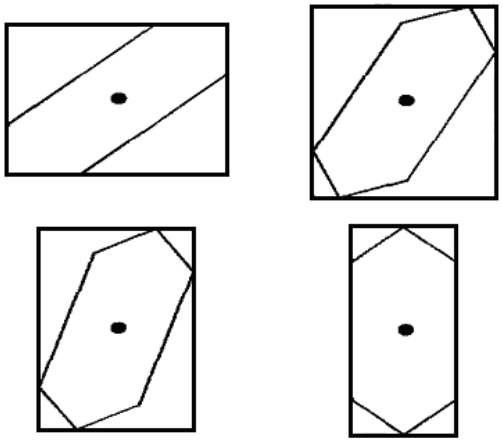

Figura 2.7. Empaquetamiento de un polígono en un rectángulo con mínima área.

(Jakobs, 1996) propone un algoritmo genético y lo compara respecto a la heurística constructiva esquina inferior izquierda (Chazelle, 1983, *bottom left*). Obteniendo excelentes resultados.

2.6. Aproximaciones al problema de la mochila bidimensional

Una gran cantidad de estudios en la literatura han trabajado problemas de corte y empaquetamiento bidimensionales con cortes tipo guillotina con etapas y sin etapas. En una proporción mucho menor se han realizado estudios de problemas de corte y empaquetamiento bidimensional no-guillotina (restrictos e irrestrictos), y sólo algunos de éstos han propuesto metodologías exactas para generar patrones tipo no-guillotina.

Para el problema de la mochila bidimensional con patrones tipo guillotina un resumen del estado del arte es el siguiente: (Gilmore y Gomory, 1965; 1966) proponen un algoritmo recursivo exacto sobre la base de la programación dinámica para resolver el problema. Este algoritmo es aplicable a las versiones con y sin pesos. (Herz, 1972) propone un método de búsqueda recursiva de árbol, su método es más eficaz que el algoritmo de Gilmore y Gomory para el problema sin pesos, pero no se aplica a los casos con pesos.

(Morabito *et al*., 1992) desarrollaron la heurística DH (de las siglas en ingles *Depth-first search y Hill-climbing strategies*) basado en un proceso de búsqueda primero en profundidad y estrategias de aceptación de empeoramientos. Estas dos estrategias son comúnmente usadas en métodos de inteligencia artificial (Gallego *et al*., 2007, Gallego *et al*., 2008). El algoritmo KD (de las siglas en ingles *Knapsack problem* usando *Dynamic programming*) fue presentado por (Fayard y Zissimopoulos, 1995) para resolver el problema basado en la solución de problemas de la mochila unidimensional, resultando eficiente para problemas de gran tamaño. Este algoritmo también usa una estructura de grafo, pero sólo considera el primer nivel del árbol, a diferencia del DH que consiste en desarrollar un árbol limitado por un parámetro de profundidad (Herz, 1972; Christofides y Whitlock, 1977; Morabito *et al*., 1992).

(Hifi, 2001) presenta un algoritmo híbrido que combina el algoritmo DH y KD, esto permite desarrollar un algoritmo general para el problema de la mochila bidimensional irrestricta guillotinada. Este prueba su algoritmo para problemas de gran escala, tanto en problemas con pesos como sin pesos, obteniendo excelente resultados.

(Hifi y Zissimopoulos, 1996) proponen un algoritmo recursivo exacto usando programación dinámica basada en eficientes límites inferiores y superiores para resolver el problema irrestricto. (G y Kang, 2002) proponen una mejora al algoritmo de (Hifi y Zissimopoulos, 1996) usando una cota superior más eficiente. Este es actualmente uno de los algoritmos exactos con mejor desempeño para resolver el problema irrestricto.

Existen dos técnicas generales utilizadas para resolver los problemas restrictos: *top-down* y *bottom-up*. (Christofides y Whitlock, 1977) propusieron originalmente el enfoque de *top-down*, el cual genera todos los posibles patrones de solución en forma recursiva, dividiendo cada placa en dos nuevas. La mayoría de los algoritmos exactos para el problema irrestricto

utilizan este enfoque. El enfoque de *bottom-up* genera todos los posibles patrones de corte mediante la combinación de dos patrones horizontalmente o verticalmente construidos a partir de otros dos patrones. El enfoque *bottom-up* requiere una gran cantidad de memoria, razón por la cual la implementación de un algoritmo de este tipo es poco atractivo. Sin embargo, (G *et al*., 2003) propusieron un algoritmo que parte de una solución inicial de buena calidad y utiliza el algoritmo constructivo *bottom-up* como estrategia para generar las ramas, disminuyendo el número de nodos a explorar.

Por otro lado, un resumen del estado del arte del problema de la mochila bidimensional con patrones de corte no guillotina es el siguiente: (Beasley, 1986) y (Hadjiconstantinou y Christofides, 1995) presentan una formulación de programación lineal 0-1 para el problema utilizando variables de decisión binarias para las posiciones donde serán cortadas las piezas de la placa; desarrollaron relajaciones Langrangeanas y las utilizaron como límites en los procedimientos de búsqueda en el árbol correspondiente (optimización de subgradiente fue utilizada para optimizar los límites).

En (Tsai *et al.*, 1993) y (Chen *et al.*, 1995), el problema es formulado como un modelo lineal binario utilizando las variables de decisión izquierda, derecha, arriba y abajo, relativas a la posición de cada par de piezas que se cortarán de la placa (con restricciones disyuntivas para las opciones múltiples); en éstos se sugiere resolver los modelos empleando algoritmos *branch-and-bound* explorando estructuras particulares de las mencionadas restricciones. Otra formulación lineal binaria aparece en (Boschetti *et al.*, 2002; Egeblad y Pisinger, 2009), y una formulación binaria no lineal fue presentada en (Beasley, 2004) (ver también (Boschetti *et al.*, 2002; Egeblad y Pisinger, 2009)). Otra manera exacta de abordar el problema utilizando *branch-and-bound* basado en el llamado enfoque de dos niveles (el primer nivel selecciona el conjunto de piezas a cortar sin tener en cuenta la capa, el segundo nivel revisa si existe una capa de corte factible para las piezas seleccionadas) (Baldacci y Boschetti, 2007; Caprara y Monaci, 2004; Fekete *et al.*, 2007).
El método presentado en (Fekete *et al.*, 2007) está basado en un árbol de búsqueda de dos niveles que combina el uso de una estructura de dato especial para caracterizar cortes factibles con límites superiores.

En (Birgin *et al*., 2012), se utiliza el enfoque de particionamiento recursivo presentado en (Birgin *et al.*, 2010) para la carga de manufacturas en *pallets* para los cortes bidimensionales ortogonales no guillotina (con pesos y sin pesos) sólo para generar patrones de primer orden en los contenedores. Este enfoque recursivo de particionamiento combina versiones refinadas de ambas heurísticas recursivas de cinco bloques presentadas en (Morabito y Morales, 1998; Morabito y Morales, 1999)y la estrategia en L para cortar rectángulos de rectángulos mayores y piezas en forma de L presentadas en (Lins *et al.*, 2003; Birgin *et al.*, 2005).

Diferentes heurísticas basadas en búsqueda local aleatoria, ubicación *bottom-left*, búsqueda en grafos, etc., y diferentes metaheurísticas basadas en algoritmos genéticos, búsqueda tabú, recocido simulado, GRASP, etc., para generar cortes bidimensionales no-guillotina para el problema restricto e irrestricto se encuentran en la literatura. Algunos ejemplos recientes son (Alvarez-Valdés, 2007; Gonçalves, 2007; Huang y Chen, 2007; Chen y Huang, 2007; Bortfeldt y Winter, 2009; Egeblad y Pisinger, 2009; Wei et al., 2009).

Los métodos basados en métodos no lineales para empaquetamiento de rectángulos dentro de regiones arbitrarias convexas, considerando diferentes tipos de restricciones de posicionamiento han sido presentadas en (Birgin *et al.*, 2006a; Birgin *et al.*, 2006b; Birgin y Lobato 2010; Mascarenhas y Birgin, 2010). La mayoría de estos estudios han sido desarrollados para el problema restricto, que puede ser más interesante para ciertas aplicaciones prácticas con relativa baja demanda de los ítems ordenados. Sin embargo, parte de esos métodos podrían no funcionar bien cuando se resuelve el problema irrestricto, especialmente aquellos cuyo comportamiento computacional es altamente dependiente en el total del número de ítems ordenados. Por otro lado, el problema irrestricto es particularmente interesante para aplicaciones del problema *cutting-stock* con producción de gran escala y con ítems levemente heterogéneos (es decir, relativamente pocos tipos de piezas pero muchas copias del mismo tipo), en el cual el problema utiliza el procedimiento de generación de columnas, como lo indican muchos autores desde el estudio pionero (Gilmore y Gomory, 1965).

Los problemas de corte y empaquetamiento revisten una alta complejidad matemática, estos son considerados NP-Duros en un fuerte sentido, luego de ser demostrados a través del caso especial donde todos los ítems tienen la misma longitud y diferente altura, el bien conocido problema de empaquetamiento unidimensional (*one-dimensional bin packing*). El problema de empaquetamiento óptimo de una dimensión fue probado NP-Duro por (Garey y Johnson, 1979) y (Martello *et al*., 2003). El que un problema este catalogado en esta categoría no significa que no puede resolverse, sino que 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.

En este estudio se propone el uso de tres metaheurísticas para el desarrollo de un algoritmo que reúna las características principales de las técnicas de cúmulo de partículas, vecindario variable y algoritmos genéticos (PSO, VNS y GA, de las siglas en ingles *particle swarm optimization, variable neighborhood search* y *genetic algorithms* respectivamente).

2.7. Resumen

Los problemas propuestos hacen parte de problemas clásicos y de gran interés dentro del área de la investigación operativa, dado que son aplicables en diversos sectores de la economía, entre los que se destacan los sectores de industria, transporte y comercio. Así por ejemplo, su aplicación se presenta en la industria textil, metalmecánica, papelera, vidriera, transporte y almacenaje de mercancías, entre otras. La alta complejidad matemática y computacional de estos problemas ha llevado que sean solucionados desde diferentes áreas de la optimización como lo son la optimización exacta y la optimización combinatoria, una lista de trabajos y contribuciones sobre esta temática ha sido expresada anteriormente.

Se pretende trabajar el problema desde la optimización combinatoria haciendo uso de herramientas como las técnicas metaheurísticas de optimización. Este estudio propone el uso de tres técnicas metaheurísticas: optimización con cúmulos de partículas, búsqueda en vecindario variable y algoritmos genéticos para desarrollar un método de solución a través de la creación de un algoritmo híbrido de optimización, para dar solución a:

- El problema de la mochila bidimensional irrestricta, con y sin pesos asociados a las piezas, con orientación fija de las piezas y con patrones de corte tipo no guillotina de primer orden.

3. MODELOS MATEMÁTICOS DE LOS PROBLEMAS

3.1. Introducción

Distintos modelos matemáticos para representar el mismo problema han sido propuestos, con el fin de encontrar mejoras en la solución a través de un modelamiento eficiente, es por esto que en este estudio se hace una revisión de los diferentes modelos y se expone solamente uno porque es el más cercano al problema tratado y presenta resultados satisfactorios para el problema de la mochila bidimensional irrestricta guillotinada con y sin rotación de piezas.

(Gilmore y Gomory, 1961; 1965) presentan el primer modelo matemático para el problema de corte y empaquetamiento unidimensional. (Gilmore y Gomory, 1965; 1966) estudian cuidadosamente el problema de la mochila cuando este es aplicado para el corte de piezas en una y dos dimensiones.

(Biro y Biros, 1985) caracterizan los patrones no guillotina usando teoría de grafos. (Beasley, 1986) estudió el problema de la mochila sin restricciones de corte tipo guillotina. (Beasley, 1986) propone un modelo de programación lineal entera para determinar las coordenadas discretas a las cuales los ítems pueden ser ubicados. Un modelo similar fue introducido por (Hadjiconstantinou y Christofides, 1995). (Scheithauer y Terno, 1993) proponen modelos para empaquetamiento de polígonos convexos y no convexos.

(Fekete y Schepers, 1997) usan la teoría de grafos para determinar un empaquetamiento factible sin sobreponer las piezas. (*Lodi et al*., 2002; 2004) presentan un modelo que maneja patrones formados por estantes (niveles), este modelo considera explícitamente restricciones de corte tipo guillotina (pero solo una parte de los patrones guillotina son considerados). Recientemente (Beasley, 2004) presenta una nueva formulación de corte no guillotina. Una buena revisión de los modelos existentes está disponible en (Lodi *et al*., 2004).

Un modelo matemático entero-mixto para el problema de empaquetamiento tridimensional en contenedores, con número polinomial de variables y restricciones, es presentado por (Chen *et al*., 1995), este modelo puede ser visto como una extensión a la técnica de modelamiento propuesta por (Onodera *et al*., 1991). Para un problema de ubicación de bloques bidimensionales, el modelo se basa en la enumeración de todas posibles ubicaciones relativas de cada par de piezas. Los experimentos computacionales presentados en (Chen *et al*., 1995) muestran sin embargo que el modelo propuesto es muy ineficiente en la solución de instancias prácticas de empaquetamiento. La misma técnica fue usada por (Daniels *et al*., 1994) para modelar el problema de empaquetamiento general (bidimensional) de polígonos, en este mismo caso, el uso directo del modelo prueba ser ineficiente en la práctica. (Ben *et al*., 2008) presentan un modelo basado en la caracterización y modelamiento de los patrones guillotina para el problema de

empaquetamiento en rollos infinitos (*strip packing problem*), el cual puede ser fácilmente adaptado para el problema de la mochila.

Como se muestra en la revisión bibliográfica anterior no existen modelos matemáticos que describan completamente los problemas de interés en este estudio. Pero algunos modelos propuestos en la literatura son fácilmente adaptables, por otro lado, se sabe que estos modelos no son satisfactorios en la resolución de casos prácticos del problema en la vida real (Beasley, 2004; Ben *et al*., 2008; Chen *et al.*, 1995). El enfoque de este trabajo es una aproximación dado que el objetivo es resolver aplicaciones reales. Por lo tanto el modelo matemático es sólo ilustrativo.

3.2. Modelo matemático de la mochila

A continuación se presenta el modelo matemático del problema de la mochila bidimensional no guillotina irrestricta presentado por (Beasley, 2004).

Se tiene un contenedor de dimensiones L_0 (largo del contenedor) y W_0 (ancho del contenedor) un número Q_i de copias idénticas de cada pieza i con un beneficio asociado v *(*área de la pieza cuando no se especifica*)* se define:

z_{ip} = 1, si la *p-ésima* copia (p = 1,…, Q_i) de la pieza i es cortada del contenedor (L_0, W_0),

z_{ip} = 0, de lo contrario,

donde la posición de cualquiera pieza cortada es tomada con referencia al centro de la pieza, es decir, sea

x_{ip} la coordenada x del centro del *p-ésima* pieza copia de la pieza i;

y_{ip} la coordenada y del centro del *p-ésima* pieza copia de la pieza i.

La figura 3.1 ilustra la ubicación. Las coordenadas centrales están limitadas por las ecuaciones (1) y (2):

$$L_i / 2 \le x_{ip} \le L_0 - L_i / 2, i = 1,\ldots,m; p = 1,\ldots,Q_i \qquad (1)$$

$$W_i / 2 \le y_{ip} \le W_0 - W_i / 2, i = 1,\ldots,m; p = 1,\ldots,Q_i \qquad (2)$$

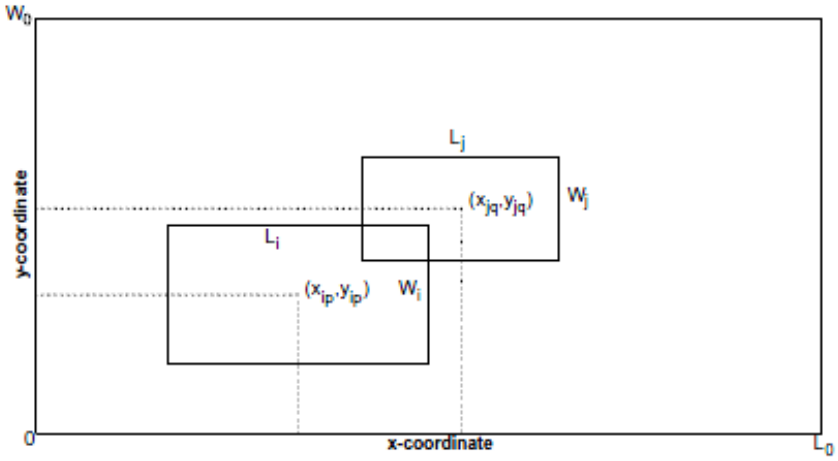

Figura 3.1. Representación de la codificación. Tomada de (Beasley, 2004)

Buscando asegurar que se corte un número mínimo apropiado de piezas del cada tipo que se tiene la ecuación (3):

$$z_{ip} = 1, i = 1, \ldots, m; P_i > 0; p = 1, \ldots, P_i \qquad (3)$$

Esta restricción simplemente expresa que arbitrariamente se eligieron cortar las primeras P_i copias de la pieza i. Naturalmente, las piezas que son cortadas del rectángulo contenedor (L_0, W_0) no se pueden traslapar y considerando la figura 3.1. donde se muestran dos piezas traslapándose (la *p-ésima* copia de i, donde p es la cantidad de copias que se utilizarán de la pieza i y la *q-ésima* copia de j, donde q es la cantidad de copias que se utilizarán de la pieza j) esto se puede expresar como la condición que se requiere

$$\left|x_{ip} - x_{jq}\right| \geq \left(L_i + L_j\right)/2 \quad \text{ó} \quad \left|y_{ip} - y_{jq}\right| \geq \left(W_i + W_j\right)/2 \qquad (4)$$

Esto quiere decir que la diferencia entre las coordenadas x centrales es suficiente $\left(\geq \left(L_i + L_j\right)/2\right)$ para asegurar que las piezas no se traslapan, o que la diferencia entre las coordenadas centrales y es suficiente $\left(\geq \left(W_i + W_j\right)/2\right)$ para asegurar que las piezas no se traslapan. Esta restricción de superposición fue planteada en primer lugar por (Christofides, 1974) en el contexto del problema del rodillo (strip packing). Para simplificar la notación se define $\alpha_{ij} = \left(L_i + L_j\right)/2$ y $\beta_{ij} = \left(W_i + W_j\right)/2$, luego la condición anterior (Ecuación 4) puede ser escrita de la siguiente manera:

$$\left|x_{ip} - x_{jq}\right| - \alpha_{ij} \geq 0 \quad \text{ó} \quad \left|y_{ip} - y_{jq}\right| - \beta_{ij} \geq 0 \qquad (5)$$

Por lo tanto, la restricción para prevenir traslape entre las piezas cortadas es:

$$\max\left[\left|x_{ip} - x_{jq}\right| - \alpha_{ij}, \left|y_{ip} - y_{jq}\right| - \beta_{ij}\right] z_{ip} z_{jq} \geq 0, \qquad (6)$$

Teniendo $i = 1,\ldots,m; j = 1,\ldots,m; p = 1,\ldots,Q_i; q = 1,\ldots,Q_j \left(i \neq j \text{ ó } p \neq q\right)$

Donde $z_{ip}\, z_{jq}$ inactiva la la restricción a menos que $z_{ip} = z_{jq} = 1$ (es decir, a menos que la *p-ésima* copia de la pieza *i* y la *q-ésima* copia de la pieza *j* sean cortadas).

La formulación completa del problema de corte y empaquetamiento bidimensional no guillotina irrestricto es entonces:

Maximizar $\sum_{i=1}^{m}\sum_{p=1}^{Q_i} v z_{ip}$ (7)

Sujeto a:

$$\max\left[\left|x_{ip} - x_{jq}\right| - \alpha_{ij}, \left|y_{ip} - y_{jq}\right| - \beta_{ij}\right] z_{ip} z_{jq} \geq 0, \qquad i = 1,\ldots,m; j = 1,\ldots,m; p = 1,\ldots,Q_i;$$
$$q = 1,\ldots,Q_j \left(i \neq j \text{ ó } p \neq q\right), \qquad (8)$$

$$z_{ip} = 1, \qquad i = 1,\ldots,m; P_i > 0; p = 1,\ldots,P_i, \qquad (9)$$

$$L_i / 2 \leq x_{ip} \leq L_0 - L_i / 2, \qquad i = 1,\ldots,m; p = 1,\ldots,Q_i, \qquad (10)$$

$$W_i / 2 \leq y_{ip} \leq W_0 - W_i / 2, \qquad i = 1,\ldots,m; p = 1,\ldots,Q_i, \qquad (11)$$

$$z_{ip} \in (0,1), \qquad i = 1,\ldots,m; p = 1,\ldots,Q_i, \qquad (12)$$

Acerca de la formulación es importante anotar:

(a) Si la *p-ésima* copia de la pieza *i* no es cortada ($z_{ip} = 0$), entonces los valores de x_{ip} e y_{ip} son irrelevantes como restricciones de traslape porque la Ecuación 8 es automáticamente satisfecha.
(b) La restricción de traslape, presentada previamente ecuación (8) contiene redundancia ya que si tuviéramos que escribir esta restricción de traslape completa para cualquier ejemplo en particular encontraríamos exactamente la misma ecuación matemática apareciendo dos veces (por ejemplo, para $i = 3$, $p = 1$ y $j = 7$, $q = 2$ se obtiene la misma ecuación que para $i = 7$, $p = 2$ y $j = 3$, $q = 1$).
(c) La formulación presentada anteriormente es válida sin importar si las dimensiones de las piezas son o no enteras.
(d) Las coordenadas centrales (x_{ip}, y_{ip}) pueden tomar valores fraccionarios, o si se requiere trabajar con coordenadas centrales sin una pérdida significativa de generalidad simplemente se asegura que todas las dimensiones (L_i, W_i), $i = 0,1,\ldots,m$, sean enteras y pares (se fija $L_i = 2L_i$ y $W_i = 2W_i$, $i = 0,1,\ldots,m$).

Debido a que la formulación presentada por (Beasley, 2004) es para el problema restricto, se debe adaptar limitando de manera implícita el número de copias posibles de cada pieza mediante la fórmula $Q_i = \left\lfloor \frac{L}{l_i} \right\rfloor \left\lfloor \frac{W}{w_i} \right\rfloor$.

La formulación del problema es no lineal, involucrando como sucede con la ecuación (8) la maximización de dos términos modulares y el producto de dos variables binarias. Utilizando $\sum_{i=1}^{m} Q_i$, luego la formulación tiene *3M* variables y restricciones O(M^2) (excluyendo los límites de las variables). La formulación presentada en (Beasley, 1985), el número de variables y restricciones estaba en función del tamaño del rectángulo del contenedor (L_0, W_0). La formulación presentada aquí, en términos del número de variables y restricciones no involucra el tamaño del rectángulo principal. En términos generales se puede decir que la formulación presentada es más compacta que la ofrecida en (Beasley, 1986), principalmente debido a la representación no lineal de la restricción de traslape en la ecuación (8).

3.3. Resumen

Los problemas presentados en este trabajo han sido estudiados por más de 6 décadas sin embargo, no presentan un modelo matemático bien definido para todas las diferentes características, en especial los patrones de corte no guillotina de primer orden.

En este capítulo se adaptó un modelo propuesto por (Beasley, 2004), este describe los patrones de corte tipo no guillotina.

(Lodi *et al*., 2004; Ben *et al*., 2008) concluyen que actualmente no existe software y hardware para darle solución a instancias prácticas del problema mediante los modelos planteados. Por otro lado, intentar abordar el problema a través de diferentes soluciones aproximadas, ofrece la posibilidad de construir una metodología novedosa que combine fortalezas de diferentes técnicas metaheurísticas.

4. TÉCNICAS DE OPTIMIZACIÓN: OPTIMIZACIÓN CON CÚMULO DE PARTÍCULAS, VECINDARIO VARIABLE Y ALGORITMO GENÉTICO

4.1. Introducción

En esta sección se presentan a grandes rasgos las características más importantes de las tres diferentes técnicas metaheurísticas de optimización. La filosofía, las codificaciones más comunes y el procedimiento general de cada una de las técnicas son presentados a continuación.

*4.2. Optimización con cúmulo de partículas (*Particle Swarm Optimización*, PSO)*

A principios del siglo XX se da inicio a estudios del comportamiento social de los individuos tratando de entender la influencia que presentaban sobre los demás integrantes de un grupo. Las investigaciones realizadas por (Reynolds, 1994) sobre las aves, simulando su comportamiento individual y colectivo dieron origen a la técnica combinatorial denominada PSO. Basados en estos conceptos (Kennedy y Eberhart, 1995) presentaron la formulación matemática del modelo PSO.

El método PSO es una metaheurística perteneciente al grupo de los algoritmos evolutivos y su estrategia se fundamenta en el comportamiento social de los animales tales como las bandadas de aves, bancos de peces o enjambres de abejas, los cuales actúan como si fueran un solo individuo. En estos grupos de animales se establecen relaciones entre los individuos, se crean jerarquías dependiendo de las características de los mismos, existiendo un líder grupal reconocido y seguido por los demás miembros del grupo. El papel de líder puede cambiar si existe otro individuo con mejores características que el líder existente. Cuando el grupo se organiza para realizar una tarea, el líder guía a los demás individuos hacia una región promisoria, sin embargo, los demás miembros del grupo durante su recorrido pueden inferir sobre una nueva dirección con el objetivo de tener un mejor éxito en la misión.

Esta técnica presenta cierta similitud con los algoritmos genéticos, ya que el proceso incluye una población, realizando una búsqueda óptima a través de actualizaciones de esta. A pesar de no poseer operadores de recombinación y mutación, cuenta con otros operadores que permiten trasladarse a través del espacio de solución y de esta manera detectar soluciones de muy alta calidad, con la posibilidad que una de ellas corresponda al óptimo global del problema.

En PSO la exploración del espacio de solución se realiza a través de una población de individuos conocidos como partículas, donde cada uno de ellas representa una posible solución del problema. La ubicación de cada partícula sobre la región de búsqueda está

determinada mediante su posición, la cual, representa el valor que toman las variables de decisión del problema.

Cada partícula cambia de posición de acuerdo a su velocidad teniendo en cuenta la mejor solución encontrada por ésta a lo largo del proceso (*pbest*) y a la información del líder del cúmulo (*gbest*). El operador *pbest* (*individual best*) compara la posición actual de una partícula con la mejor posición que ha presentado durante el proceso de búsqueda. Mientras que el operador *gbest* (*global best*) estudia el comportamiento del grupo, almacenando la posición del líder actual del cúmulo.

Población inicial

Las técnicas heurísticas dependiendo de la complejidad matemática del problema generan la población inicial de forma aleatoria o utilizando métodos constructivos basados en factores de sensibilidad cuya finalidad consiste en iniciar la búsqueda en una región atractiva con el fin de disminuir tiempo y esfuerzo computacional. El caso de estudio presentado en este trabajo inicia con un conjunto de partículas generadas de forma aleatoria alcanzando el óptimo del problema. Sin embargo, en la medida que se estudian sistemas grandes y de alta complejidad matemática cobra fuerza el uso de población inicial generada con base en métodos constructivos.

Una población está constituida por *k* partículas y cada una de estas en el estado *i* representa una alternativa de solución, la cual es interpretada en el problema a través de la posición de la partícula. Cada partícula se representa mediante un vector cuyas posiciones indican los valores de las variables del problema (ver figura 4.1) y simbolizan un punto dentro del espacio de solución. Inicialmente se generan *n* partículas con valores aleatorios dentro del rango de las variables y a través de la función objetivo se determina la calidad de la solución.

$$x_i = \boxed{x_{i1} \mid x_{i2} \mid \cdots \mid \ldots}$$

Figura 4.1. Posición de la i-ésima partícula

Selección del líder

En PSO durante cada iteración se debe seleccionar el líder del grupo comparando los valores de la función objetivo de cada partícula con la función objetivo del líder, siendo éste último aquella partícula que posee el mejor valor (incumbente global). Durante el proceso de optimización, cada vez que una partícula mejore el valor de su función objetivo, se debe actualizar el valor de su mejor ubicación local *pbest* y cada vez que exista un cambio de líder, se debe actualizar el valor de la mejor ubicación global *gbest*.

Función de velocidad

La velocidad permite actualizar la posición de cada partícula. El vector de velocidad representa el gradiente de cada individuo dentro del cúmulo, es decir, guía a las partículas durante el proceso de búsqueda. Al igual que la posición, la velocidad se representa a

través de un vector cuyas dimensiones deben ser iguales (ver figura 4.2). La velocidad contiene información de la experiencia local y colectiva del grupo. Para calcular la velocidad de cada partícula se usa la siguiente expresión:

$$v_i(t+1) = w \cdot v_{i-1}(t) + c_1 \cdot rand \cdot \left(pbest - x_i(t)\right) + c_2 \cdot Rand \cdot \left(gbest - x_i(t)\right) \tag{4.1}$$

$v_i =$ | v_{i1} | v_{i2} | $\cdots$ |

Figura 4.2. Velocidad de la i-ésima partícula

Donde:

Velocidad actual $\left(v_{i-1}\right)$: Dirección de vuelo que presenta una partícula. Al inicio del proceso las partículas parten del reposo.

Factor de inercia (w): Factor de ponderación que actúa sobre la velocidad actual de la partícula. Un valor elevado puede ocasionar una búsqueda muy exhaustiva mientras que un valor demasiado pequeño conlleva a que la exploración se realice de manera fugaz.

Constantes de aceleración c_1 y c_2: Valores que direccionan a las partículas hacia su mejor ubicación local o global respectivamente. La adecuada calibración de estos valores permite que la población no se homogenice durante el proceso.

Rand y *rand*: Valores aleatorios pertenecientes a una distribución de probabilidad uniforme.

Actualización de la posición

Para determinar la nueva posición de las partículas se aplica la ecuación 4.2:

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{4.2}$$

Como la posición actualizada de las partículas debe satisfacer las restricciones de las variables del problema, es necesario definir un rango de valores para las velocidades evitando de esta manera sobrepasar el límite impuesto. La figura 4.3 muestra un esquema de cómo es actualizada la posición de la partícula a través de los vectores de velocidad y posición.

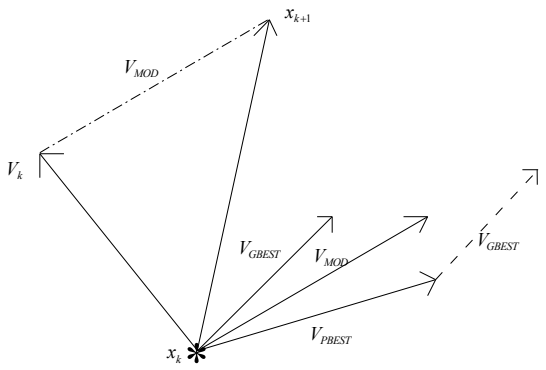

Figura 4.3. Esquema de la nueva posición de la partícula

Diferentes codificaciones

Las variables de un problema de optimización pueden ser representadas a través de valores enteros, reales o binarios. Si se usa una representación real no existe problema alguno con los procesos definidos anteriormente, pero si se desea una representación binaria diferentes problemas se encuentran. A continuación se ejemplifica el uso de una codificación real. Sea d_i las variables de decisión del problema ($i = 1, 2, ..., n$) y definida formalmente como un conjunto de variables binarias.

$$d_i \in \{0,1\}, \quad \forall i$$

Teniendo en cuenta lo anterior, una alternativa de solución dada por una partícula dentro del cúmulo se representa de la siguiente forma:

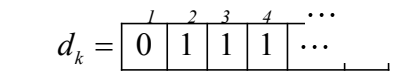

Figura 4.4. Codificación del problema

En la versión binaria, al igual que en la versión continua, la posición de una partícula es actualizada a través del operador de velocidad (ver ecuación (4.1)), el cual es representado con valores continuos y para este caso se hace necesario implementar una función de activación que permita transformar estos valores a números binarios. La función que se emplea para este propósito se conoce como función sigmoidal (habitualmente utilizada en redes neuronales) cuyo rango de valores se encuentra en el intervalo [0-1] (ver figura 4.5).

$$sig(v_i) = \frac{1}{1+\exp(-v_i)} \tag{4.3}$$

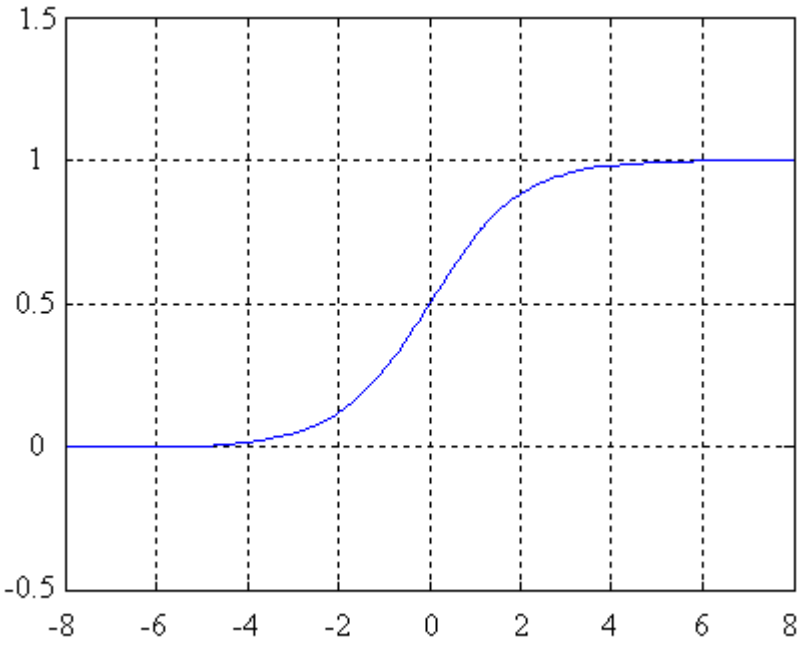

Figura 4.5. Gráfica de la función sigmoidal

La posición de la partícula asumirá un valor binario (0 ó 1). Dicha posición es actualizada teniendo en cuenta los valores de velocidad, los cuales son calculadas usando la ecuación (4.2). El valor obtenido es comparado con el umbral ρ, el cual puede tomar valores entre [0-1]. El rango típico sugerido es [0.5 - 0.7]. El procedimiento seguido para la actualización de la posición es el siguiente:

$$x_{i1} = \begin{cases} 1, & si\,\rho < sig\left(v_{i1}\right) \\ 0, & si\,\rho \geq sig\left(v_{i1}\right) \end{cases} \qquad (4.4)$$

El procedimiento anterior se constituye en una adecuación para el cálculo de la posición de las partículas para la versión binaria de PSO.

A continuación, en la figura 4.6 se muestra la estructura general del algoritmo PSO:

```
Inicio
        t = 1
        población = GeneraciónPoblaciónInicial(tamañopoblación)
        gbest = SeleccionarLíder(población)
        Si t = 1
                pbest_i = D_i
                gbest = max(pbest_i)
        De lo contrario
                Si f.o.(D_i(t)) < f.o.(pbest_i)
                        pbest_i = D_i
                        Si f.o.(D_i(t)) < f.o.(gbest)
                                pbest_i = D_i
                        Fin Si
                Fin Si
        Fin Si
        velocidad = CalcularVelocidad(población,pbest,gebest,c1,c2,w)
        población = ActualizarPosición(población,velocidad)
        t = t + 1
        Si el criterio de parada se satisface
                Se finaliza el algoritmo
        De lo contrario
                Volver a la etapa de Selección de Líder
        Fin Si

Fin
```

Figura 4.6. Algoritmo PSO.

Factor de Turbulencia de Vuelo

En PSO la velocidad de la partícula *i* está dada por $V_i = (v_{i1},\ldots,v_{iN})$. Comúnmente el algoritmo general para el ajuste de dicha velocidad es:

$$v_{i,j} := wv_{i,j} + c_1 r_1 (p_{i,j} - x_{i,j}) + c_2 r_2 (p_{g,j} - x_{i,j}) \qquad (4.5)$$

Durante el desarrollo inicial del PSO (Kennedy y Eberhart, 1995), una variable estocástica llamada "locura" se utilizó para alterar las partículas, de esta forma la ecuación (4.5) resultaría:

$$v_{i,j} := v_{i,j}^{\phi} + r_3 \qquad (4.6)$$

Donde $v_{i,j}^{\phi}$ es la velocidad del *j-ésimo* parámetro del vecino más cercano a X_i y r_3 es la variable de locura aleatoria. El parámetro de locura es utilizado para mantener la diversidad en la población En este trabajo se introduce esta variable dentro del PSO a través del

operador mutación propio de los algoritmos genéticos. Para mantener el diseño del PSO, este término es referido como una turbulencia (equivalente a una perturbación), porque refleja el cambio en el vuelo de las partículas cuando quedan fuera de control.

Criterio de parada

Los algoritmos de optimización precisan de un criterio que les permita decidir cuando finalizar la exploración del espacio de solución. Entre los criterios de parada más empleados en dichos algoritmos se encuentran:

- Si durante un número de iteraciones no se ha mejorado la incumbente, se escoge esta solución y se da por finalizado el proceso.
- Hasta cumplir un número máximo de iteraciones.

Siendo la última opción la considerada en este trabajo.

Descripción del algoritmo

El procedimiento empleado por el algoritmo PSO es presentado en la figura 4.6.

4.3. Búsqueda en Vecindario Variable

La búsqueda de vecindario variable (*Variable Neighbourhood Search, VNS*) es una metaheurística para resolver problemas de optimización cuya idea básica es el cambio sistemático de vecindario dentro de una búsqueda local.

La VNS está basada en tres hechos simples:

- Un mínimo local con una estructura de vecindad no lo es necesariamente con otra.
- Un mínimo global es mínimo local con todas las posibles estructuras de vecindarios.
- Para muchos problemas, los mínimos locales con la misma o distinta estructura de vecindad están relativamente cerca (este último es una aserción empírica).

El desarrollo de esta metaheurística ha sido rápido, con muchos artículos ya publicados (Hemmelmayr *et al.*, 2012; M'Hallah *et al.*, 2013). Se han realizado muchas extensiones, principalmente para permitir la solución de problemas de gran tamaño (Mladenović y Hansen, 2001b). En la mayoría de ellas, se ha hecho un esfuerzo por mantener la simplicidad del esquema básico (Mladenović, 1995; Mladenović y Hansen, 1997).

Las metaheurísticas de búsqueda local aplican una transformación o movimiento a la solución de búsqueda y por tanto utilizan, explícita o implícitamente, una estructura de vecindad. Sea N_k *(*donde, $k = 1, 2,..., k_{max}$*)* un conjunto finito de estructuras de vecindarios en el espacio X. Los vecindarios N_k pueden ser inducidos por una o más métricas introducidas en el espacio de soluciones X. La mayoría de las heurísticas de búsqueda local usan sólo una estructura de vecindad.

Una búsqueda local descendente cambia la solución actual por otra solución mejor de su vecindad, por tanto tienen el peligro de quedarse atrapada en un mínimo local. Las metaheurísticas basadas en procedimientos de búsqueda local aplican distintas formas de continuar la búsqueda después de encontrar el primer óptimo local.

La búsqueda en vecindario variable básica (*Basic Variable Neighborhood Search, BVNS*) combina cambios determinísticos y aleatorios de estructura de vecindad. La tabla 2 presenta los pasos de la VNS básica.

La condición de parada puede ser, por ejemplo, el tiempo máximo de procesamiento permitido, el número máximo de iteraciones, o el número máximo de iteraciones entre dos mejoras. Frecuentemente los vecindarios N_k sucesivos están anidados. Obsérvese que la solución *S'* se genera al azar en el paso (3.1.a.) de la figura 4.7 para evitar el ciclado, que puede ocurrir si se usa cualquier regla determinística.

Inicio

1. Seleccionar un conjunto de estructuras de entornos N_k;
   $k = 1,2,..., k_{max}$, que se usarán en la búsqueda
2. Encontrar una solución inicial *S*
3. Repita:
   - 3.1 Para $k = 1$ hasta k_{max} haga:
     - a. Generar al azar una solución *S'* del *k*-ésimo vecindario de *S* ($S' \in N_k(x)$)
     - b. Aplicar algún método de búsqueda local con *S'* como solución inicial; denótese con *S''* el mínimo local así obtenido
     - c. Si la solución obtenida *S''* es mejor que *S*, haga $S = S''$ y $k = 1$
     - d. De lo contrario, haga $k = k + 1$
4. Hasta la condición de parada.

Fin

Figura 4.7. Algoritmo búsqueda en vecindario variable (VNS Básico)

*4.4. Algoritmo genético (*Genetic Algorithm*, GA)*

Historia

El origen de los Algoritmos Genéticos se dio a finales de los años 50 con la realización de trabajos y publicaciones en el campo de los algoritmos evolutivos como alternativa de solución para problemas combinatorios. Esta técnica se define como una familia de modelos computacionales inspirados en los mecanismos de evolución natural perteneciente a la familia de los algoritmos evolutivos tomados de la teoría de (Darwin, 1859) sobre la evolución de las especies.

Filosofía

Los GA emplean un conjunto de soluciones en cada iteración del algoritmo en lugar de emplear una única solución como las metaheurísticas basadas en trayectorias. Mediante una imitación del mecanismo genético de los organismos biológicos, los algoritmos genéticos exploran el espacio de soluciones para un problema determinado. Una alternativa de solución se interpreta como el cromosoma del individuo compuesto de un cierto número de genes a los que les corresponden ciertos alelos. Los cromosomas son sometidos a un proceso de evolución que abarca la evaluación, selección, recombinación y mutación. Después de varios ciclos de evolución la población deberá contener individuos más aptos.

Un algoritmo genético elemental realiza la siguiente secuencia de operaciones:

1. Genera una población inicial después de escoger el tipo de codificación para representar cada configuración.
2. Calcula la función objetivo de cada configuración de la población y almacena la incumbente.
3. Realiza selección.
4. Realiza recombinación.
5. Realiza mutación y termina de generar la nueva población de la siguiente generación.
6. Si el criterio de parada (o criterios de parada) no se han cumplido el proceso regresa al paso 2.

Los pasos (2), (3), (4) y (5), en conjunto, son conocidos como ciclo generacional. También es necesario mencionar que existe una equivalencia entre los términos usados en genética y en un problema de optimización matemática.

Selección

Este operador genético permite seleccionar las configuraciones de la población actual que deben participar en la generación de las configuraciones de la nueva población (nueva generación). Por tanto la función del operador de selección termina después de decidir el número de descendientes que debe tener cada configuración de la población actual. En algunos casos las configuraciones puede generar varios descendientes mientras que en otras ninguno, desapareciendo la información de estas configuraciones que son consideradas de baja calidad.

Selección proporcional

La forma más simple de implementar la selección es usando el denominado esquema de selección proporcional. En esta estrategia cada configuración tiene derecho a generar un número de descendientes que es proporcional al valor de la función de adaptación. Así se tiene la siguiente relación.

$$Nd_i = \frac{z_i(x)}{z_m(x)} \tag{4.7}$$

Nd_i = Número de descendientes de la configuración i.
n = número de configuraciones de la población.
$z_i(x)$ = Función de adaptación.
$z_m(x)$ = Media de la función objetivo

$$z_m(x) = \frac{1}{n}\sum_{i=1}^{n} z_i(x) \tag{4.8}$$

$$Nd_i = n\frac{z_i}{\sum_{i=1}^{n} z_i(x)} \tag{4.9}$$

La propuesta inicial y aún usada para resolver el problema de número de descendientes no entero es el denominado esquema de selección de la ruleta. En el esquema de selección de la ruleta a cada configuración se le asigna una parte de la ruleta que es proporcional al número de descendientes, calculada con la siguiente relación:

$$2\pi\left(\frac{1}{n}\right)Nd_i \tag{4.10}$$

Selección por torneo

La selección por torneo, es bastante simple y consiste en escoger la mejor alternativa de las k alternativas seleccionadas aleatoriamente. El éxito de aplicar esta metodología radica en escoger adecuadamente un valor de k que se ajuste a cada problema en particular teniendo en cuenta tamaño y complejidad del problema, así como el tamaño de la población inicial. Un valor de k muy alto y la generación de una población inicial pequeña pueden, eventualmente, hacer elitista el proceso de selección y ocasionar convergencias locales. Un valor muy bajo de k puede ocasionar un mayor esfuerzo computacional (Gallego *et al.*, 2008).

Recombinación (*Crossover*)

Las configuraciones seleccionadas en el proceso de selección son sometidas a recombinación. Este operador consiste en intercambiar partes de dos vectores para formar dos nuevos vectores donde uno de los vectores nuevos tiene parte de los elementos de un vector y parte de los elementos de otro vector. Este operador se denomina también cruzamiento e intenta simular el fenómeno del *crossing over* en genética. Generalmente a

las configuraciones seleccionadas (originales) se les denomina configuraciones padres y a las nuevas configuraciones se les denomina configuraciones hijos.

Recombinación de Punto Simple

Es la forma más simple de recombinación y consiste en escoger un único punto para realizar la recombinación. Para explicar el concepto, se supone que una configuración tiene k elementos o celdas binarias; una vez elegidas las dos configuraciones para realizar recombinación, se genera un número aleatorio entre 1 y (k-1), y ese número indica el punto de recombinación. La parte que se encuentra hacia la derecha de ambas configuraciones son intercambiadas para formar las dos nuevas configuraciones. Las configuraciones que deben ser sometidas a recombinación son escogidas aleatoriamente entre las configuraciones seleccionadas que todavía tienen derecho a recombinación. Aquí están nuevamente presentes dos nuevas decisiones de carácter aleatorio: (1) Se escoge de forma aleatoria, las configuraciones que deben ser sometidas a recombinación y (2) Se escoge aleatoriamente el punto de recombinación. Aunque algunas decisiones determinísticas son usadas en algunos casos.

Algunos problemas que se presentan con este operador son:

El mismo par de configuraciones pueden ser escogidas varias veces, especialmente cuando existen configuraciones que tienen derecho a generar muchos descendientes.

Una misma configuración puede ser escogida para implementar recombinación con ella misma. Un mecanismo simple elimina este problema escogiendo otra configuración cada vez sucede este problema.

Para eliminar estos dos problemas se puede escoger una estrategia más determinista. Por ejemplo, se puede iniciar la recombinación escogiendo primero aquella configuración que tiene derecho a generar un mayor número de descendientes hasta agotar todas las recombinaciones a que esta configuración tiene derecho, después se escoge la segunda en prioridad y así sucesivamente.

Mutación

En la codificación binaria la mutación significa simplemente cambiar el valor de una variable de 0 para 1 ó viceversa. En los trabajos teóricos iniciales de algoritmos genéticos la mutación siempre se consideró un operador secundario, hoy en día se está reevaluando este concepto.

La tasa de mutación ρ_m indica la probabilidad de que una posición (celda binaria) puede tener su valor actual modificado. En el análisis teórico, y en las propuestas originales, se sugiere que la mutación debe ser intentada bit por bit (celda por celda) y así la decisión de mutación de una posición binaria es independiente de la mutación realizada de otras celdas binarias de una configuración.

Ciclo Generacional

Es el conjunto de procesos de selección, recombinación y mutación que permiten encontrar las configuraciones de la nueva generación (población) a partir de la población actual. El ciclo generacional es controlado por el programa de control del algoritmo genético.

```
Inicio
        población = GeneraciónPoblaciónInicial(tamañopoblación,gradodediversidad)
        [Padre1 Padre2] = Selección(población,TipodeSelección)
        Hijo = Recombinar(Padre1, Padre2, PuntosdeRecombinación)
        población = ActualizarPoblación(población, Hijo)
        población = Mutar(población, tasademutación)
        Incumbente = CalcularIncumbente(población)
        Si el criterio de parada se satisface
                Se finaliza el algoritmo
        De lo contrario
                Volver a la etapa de Selección
        Fin Si
Fin
```

Figura 4.8. Algoritmo GA básico.

Criterio de Parada

Existen varios criterios de parada.

- Se ha realizado un número específico de generaciones.
- La incumbente alcanza un valor de una calidad mínima especificada.
- La población es demasiada homogénea, es decir, las configuraciones son similares y no existe más evolución.

En implementaciones prácticas de problemas complejos se especifican criterios de parada más objetivos y generalmente se especifica más de un criterio de parada en un mismo algoritmo.

Descripción del algoritmo básico

El procedimiento empleado por el algoritmo GA básico es presentado en la figura 4.8.

Algoritmo genético de Chu-Beasley

El algoritmo genético de Chu-Beasley es una versión modificada del GA básico con ciertas características que lo convierten en un algoritmo más eficiente. La principal característica de este algoritmo consiste en mantener diversidad entre los cromosomas que conforman la población durante todo el proceso. En cada generación es reemplazado un solo cromosoma (alternativa) en la población, siempre y cuando, cumpla con las condiciones de optimalidad

y/o factibilidad establecidas. Dicho mecanismo busca que en cada ciclo generacional, la calidad de la solución sea mejorada por optimalidad y/o factibilidad. Durante el proceso en la población se reemplaza sistemáticamente un único descendiente. Esta estrategia tiene como ventaja encontrar soluciones de alta calidad y garantizar diversidad en la población a lo largo de las generaciones.

En el GA de Chu-Beasley el operador de selección escoge un número K de padres (generalmente se asume $K=2$). A estas configuraciones se les aplica el operador de recombinación dando como resultado K descendientes, de los cuales se escoge una sola configuración de manera aleatoria para continuar con el operador de mutación. Terminado el proceso generacional se tiene un único descendiente (configuración actual), el cual debe cumplir con las siguientes estrategias de optimalidad y/o factibilidad para formar parte de la población inicial:

- Si la alternativa actual es infactible y a su vez es menos infactible que la peor infactible de la población, entonces reemplazar la peor infactible por la alternativa actual.

- Si la configuración es factible y existe por lo menos una infactible en la población actual, entonces reemplazar la peor infactible por la alternativa actual.

- Si la configuración es factible y todas las alternativas de la población actual son factibles, entonces reemplazar la alternativa con peor función objetivo por la alternativa actual. Lo anterior se realiza sólo si la alternativa actual es de mejor calidad que la peor de la población.

En caso contrario desechar la alternativa resultante e iniciar un nuevo ciclo generacional.

4.5. Resumen

Se presentó para las técnicas metaheurísticas: cúmulo de partículas, vecindario variable y algoritmo genético, las principales características, su filosofía, sus diferentes codificaciones y su procedimiento general.

En general las técnicas: algoritmo genético y vecindario variable son altamente reconocidas por sus excelentes resultados en diferentes problemas clásicos de optimización. Por otro lado, la técnica de cúmulo de partículas es relativamente nueva, pero estudios recientes han demostrado su gran desempeño y sus buenos resultados.

5. CODIFICACIÓN DEL PROBLEMA Y ADAPTACIÓN DE LAS TÉCNICAS DE OPTIMIZACIÓN COMBINATORIA

5.1. Introducción

La solución de problemas combinatoriales, normalmente está asociada a un proceso de búsqueda. Esta no puede realizarse en forma exhaustiva, por lo que los investigadores han desarrollado una gran cantidad de métodos alternativos de búsqueda que encuentran soluciones de "buena calidad" y en muchos casos inclusive óptimas.

La manera empírica de solucionar problemas combinatorios es a través del ensayo y el error. Enumerando todas las combinaciones posibles, de las cuales se seleccionan las soluciones que satisfagan las condiciones del problema planteado. En la mayoría de los casos, "generar y probar" no es aceptable en términos del espacio y tiempo computacional. Esto es obvio si el sistema de combinaciones es infinito. Pero aunque sea finito muchas veces es muy grande (el espacio de combinaciones crece de forma exponencial). En este caso, la generación de combinaciones genera una explosión combinatoria en el cuál sólo es posible realizar una búsqueda exhaustiva invirtiendo varios miles de millones de años.

5.2. Heurísticas en optimización combinatoria

El uso de métodos heurísticos se fundamenta en los problemas de optimización pertenecientes a la categoría denominada NP (no existe un algoritmo de solución de tiempo polinomial). Si se demuestra que un problema de optimización pertenece a esta categoría, es normal solucionarlo por medio de heurísticos de optimización. Problemas donde su espacio de búsqueda aumenta exponencialmente con el tamaño del problema, es lógica la utilización de algoritmos heurísticos, el problema de empaquetamiento presentado en este trabajo hace parte de esta categoría.

La gran ventaja del uso de heurísticas es la flexibilidad en el manejo del problema a diferencia de las rigurosas técnicas clásicas de la investigación operativa. La mayor desventaja de los métodos heurísticos es que se desconoce la calidad de la solución obtenida, por lo tanto, no se puede estimar la cercanía de dicha solución con respecto al óptimo global.

El uso de métodos heurísticos es apropiado bajo una o más de las siguientes condiciones del problema:

- No existe un método exacto de solución, o en el caso de que dicho método exacto exista, el mismo requiere de mucho gasto computacional o de memoria.
- No es necesario encontrar la solución óptima, en el sentido del óptimo global sino que es suficiente con obtener una solución suficientemente buena.

- Los datos son poco confiables y por tanto no tiene sentido el tratar de encontrar el óptimo global para dichos datos, ya que el mismo no puede ser más que una aproximación al óptimo global que corresponde a los datos correctos.
- Existen limitaciones de tiempo en proporcionar la respuesta o de memoria en computador que va a efectuar los cálculos
- Se va a utilizar el resultado proporcionado por el heurístico de optimización como solución inicial para un algoritmo exacto de tipo iterativo, el cual reduciría considerablemente el número de iteraciones si parte de una solución inicial suficientemente buena.

Se distinguen cuatro tipos de búsquedas heurísticas: constructivas, mejora de una solución, de descomposición y de reducción.

Las búsquedas heurísticas constructivas consisten en ir añadiendo componentes individuales a la solución inicial hasta que se obtiene una solución inicial factible.

Las búsquedas heurísticas basadas en la mejora de una solución, en cada paso se parte de una solución para buscar en su vecindad una solución mejor que la reemplace.

Las búsquedas heurísticas basadas en descomposición dividen el problema en subproblemas más manejables, de modo que al resolver dichos subproblemas se obtiene una solución al problema inicial por integración de las soluciones obtenidas en cada subproblema.

Finalmente, las búsquedas heurísticas basadas en la técnica de reducción. Se trata en este caso de identificar alguna característica que presumiblemente debe poseer la solución óptima, para de este modo simplificar el problema de búsqueda.

Las heurísticas pueden ser dependientes del problema o independientes del mismo. Las primeras, conocidas como heurísticas, son válidas únicamente para el problema particular para el que han sido diseñadas, mientras que las segundas, las llamadas metaheurísticas, pueden aplicarse a cualquier problema.

5.3. Heurísticas constructivas de la mochila bidimensional irrestricta.

Bottom Left (BL) y Bottom Left Fill (BLF)

El BL determina que cada pieza a colocar se posiciona inicialmente en la esquina superior derecha de la placa. A continuación, se desplaza hasta la posición más profunda posible. Luego, la pieza es movida hacia la izquierda tanto como sea posible, repitiéndose esta rutina hasta que la pieza alcance una posición inamovible.

El número máximo de patrones posibles de empaquetamiento usando BL son $2^n \cdot n!$ (donde n es el número de piezas). El valor anterior no considera la posibilidad de rotar las piezas, permitir esta variante aumenta considerablemente el número de patrones existentes. El

problema fundamental que presenta esta estrategia es que conduce a soluciones con gran número de desperdicios o residuos en la placa, además que no genera todos los patrones de corte existentes también genera patrones de corte no guillotina. La figura 5.1 ilustra el proceso de ubicación de las piezas mediante el BL.

El BLF consiste en encontrar un espacio para la pieza comenzando desde la esquina inferior izquierda de la placa, buscando nivel a nivel desde el fondo de la placa. En caso de no encontrar ubicación para la pieza dentro de un determinado nivel se pasa al nivel inmediatamente superior. Repitiendo este proceso para todas las piezas.

El BLF intenta minimizar los desperdicios de espacio del BL, diferenciándose del BL, en que antes de colocar la pieza se comprueba que ésta no cabe en ninguno de los espacios generados hasta el momento a un nivel inferior. La figura 5.1 ilustra el proceso de ubicación de las mediante el BLF.

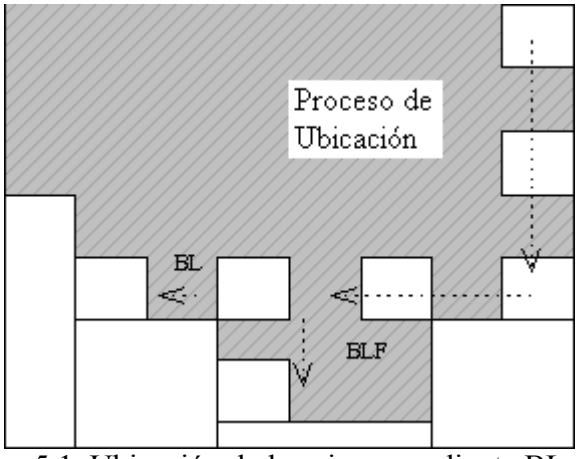

Figura 5.1. Ubicación de las piezas mediante BL y BF

Difference Process (DP)

El DP intenta colocar cada pieza en la esquina más cercana a la esquina inferior izquierda de la placa. Para medir cuál es la posición se emplea la distancia euclidiana. Al igual que con los métodos anteriores existen patrones que no puede alcanzarse con esta estrategia y también genera patrones no guillotina.

La figura 5.2 ilustra las diferencias entre los patrones BL y DP para la ubicación de 5 piezas.

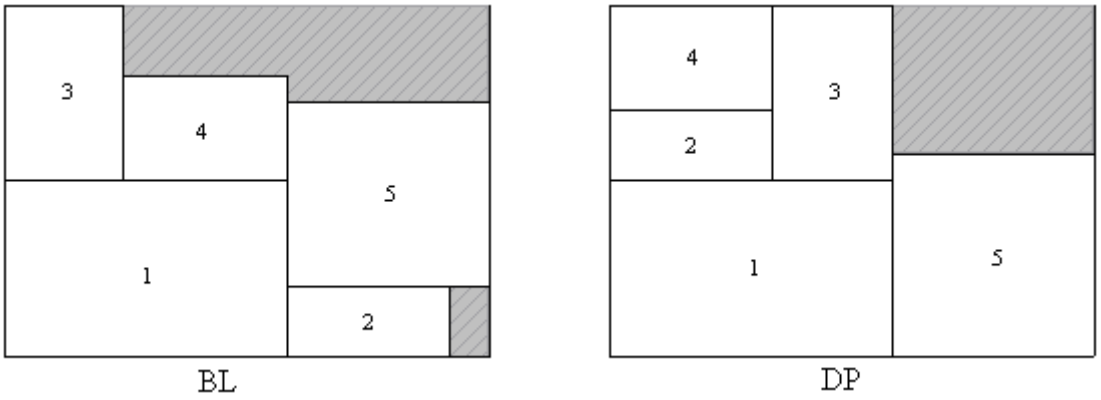

Figura 5.2. Diferencias entre los patrones BL y DP

Subespacios Maximales

Se trata de un constructivo (Parreño *et al*., 2008), cuya estrategia de empaquetamiento consiste en llenar primero las esquinas, luego los extremos (lados) del contenedor o plato principal y luego los espacios libres (subespacios maximales generados) que van quedando al interior del rectángulo mayor después de ingresar capas de piezas. En la figura 5.3 se muestra cómo se constituye una capa a partir de alguna de las piezas disponibles.

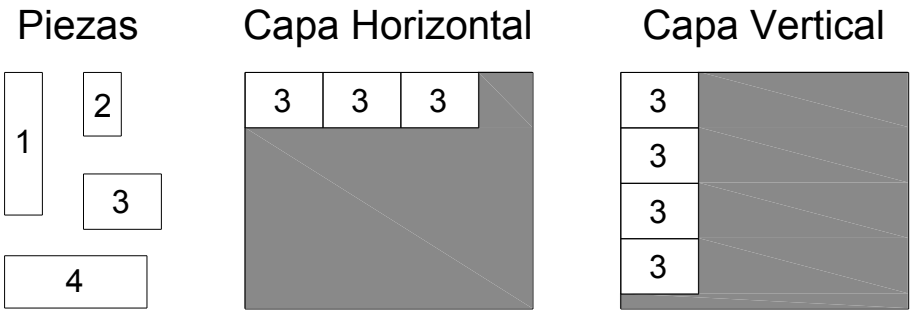

Figura 5.3. Capas vertical y horizontal generadas por alguna de las piezas que se va a cortar o empaquetar.

Un subespacio maximal, como su nombre lo indica, consiste en un subcontenedor máximo posible identificable en el espacio libre restante del contenedor después de ingresar una capa de piezas. En la figura 5.4 se muestran subespacios maximales en contenedores que están siendo llenados.

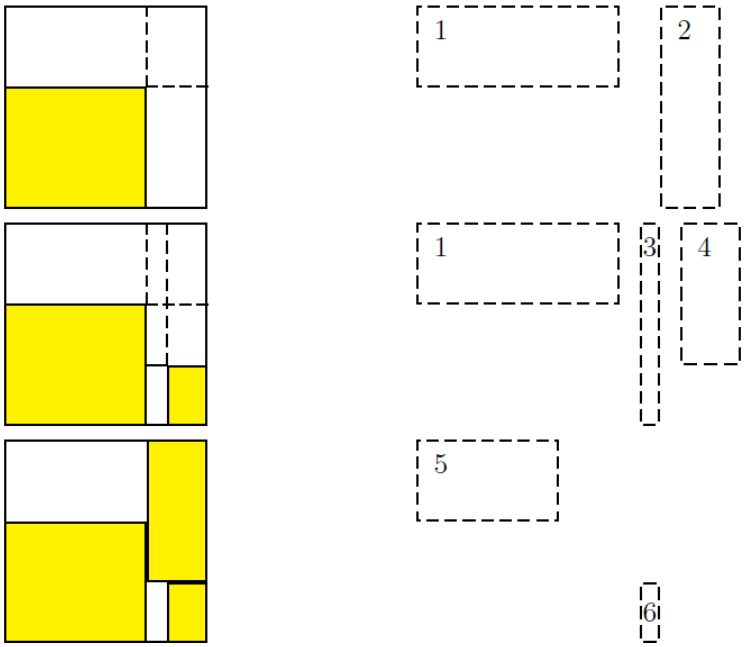

Figura 5.4. Subespacios maximales tomado de (Parreño *et al*., 2008).

Habiendo definido las capas y los subespacios maximales, la especificación del constructivo es la siguiente:

1) Se identifican los subespacios maximales disponibles en el contenedor y se elige el que se encuentre más cerca de cualquiera de las esquinas del mismo. En caso de empate, se selecciona el subespacio con mayor área.

2) Una vez elegido el subespacio se genera una capa vertical y una horizontal de cada una de las piezas que se van a cortar o empaquetar. Se elige la pieza que presente el mayor beneficio empleando cualquiera de los siguientes criterios:

   a) Beneficio máximo: Se elije la pieza que genere la capa (bien sea vertical u horizontal) con el mayor beneficio. La configuración ganadora, se adiciona en el subespacio maximal que se está llenando. En la versión sin pesos la configuración seleccionada se basa en el área que ocupa y en la versión con pesos sería el beneficio asociado a la capa.

   b) Mejor ajuste: Se elige la capa que queda más cerca de los límites del subespacio maximal que se está llenando. En la figura 5.5 se ilustra.

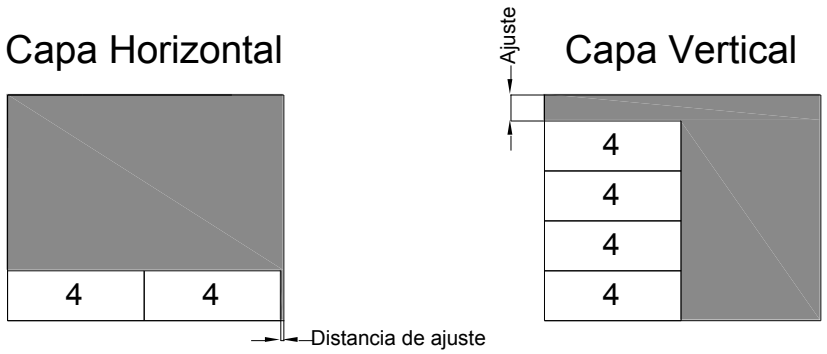

Figura 5.5. Si la distancia entre la capa y el límite de subespacio es menor, se dice que presenta un mejor ajuste (best-fit).

c) Índice área: En este trabajo se propone otro criterio para seleccionar las piezas que se adicionarán, denominado *índice área*, que consiste en relacionar el área ocupada por la capa que produce una pieza con el área de la capa ideal que sería equivalente a una capa con distancia de ajuste 0. En la figura 5.6 se ilustra la relación.

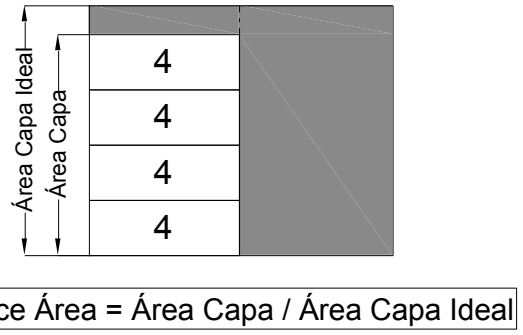

Figura 5.6. Cuando el índice es mayor, la capa generada presenta un mayor aprovechamiento del espacio delimitado por la línea discontinua denominado capa ideal.

d) Bloque trivial: Consiste en generar un patrón de corte o empaquetamiento a manera de arreglo (capa en sentido vertical y horizontal) utilizando un tipo de pieza. En la figura 5.7 se ilustra la metodología.

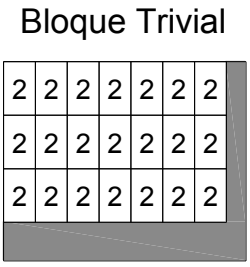

Figura 5.7. Criterio alternativo para empaquetar en el constructivo.

3) Una vez elegida la pieza y la orientación de la capa se actualiza el contenedor. Posteriormente se buscan los nuevos subespacios maximales.

4) Se evalúa si es posible ingresar alguna de las piezas disponibles, si no cabe ninguna se detiene el proceso.

5.4. Definición de la fase constructiva

Después de trabajar con las diferentes heurísticas presentadas, se hace necesario seleccionar aquellas que mejor se ajusten al problema abordado. Como se está trabajando la versión no guillotinada del problema de la mochila irrestricta, es preferible que el constructivo naturalmente (sin necesidad de recargar el procedimiento con restricciones y validaciones) genere patrones no guillotina y ese es el caso de la metodología de subespacios maximales.

La adaptación de subespacios maximales para este trabajo busca entonces obtener el mejor provecho de esta metodología que puede funcionar con alguno de los diferentes criterios a la hora de definir las capas de cada tipo de pieza que se va a empaquetar. Para escoger el mejor de ellos se realizaron pruebas de desempeño sobre la librería presentada en (Hifi, 2001) para las instancias UU1-UU11. En la tabla 5.1 se muestran los porcentajes de aprovechamiento y en la tabla 5.2 se muestran las áreas cubiertas por los patrones obtenidos con las diferentes variantes.

Instancias	Area Contenedor	Porcentajes Aprovechamiento		
		Area Máxima	Best-Fit	Capa Ideal
UU1	250000	86.9568	77.8544	77.8544
UU2	600000	97.6713	97.6713	97.6713
UU3	1100000	88.8567	86.8129	92.0624
UU4	1200000	96.6161	82.8045	82.8045
UU5	1885000	92.6545	99.1504	88.9789
UU6	2986850	93.2827	93.586	93.586
UU7	2965160	91.9909	98.6119	96.4424
UU8	4000000	95.562	90.805	90.805
UU9	6150000	99.1982	94.2281	94.2281
UU10	12075000	93.158	96.5314	96.5314
UU11	13177500	98.5182	99.1634	98.534

Tabla 5.1. Porcentajes de aprovechamiento obtenido por cada variante o criterio en la metodología de subespacios maximales.

Instancias	Area Contenedor	Área Cubierta		
		Area Máxima	Best-Fit	Capa Ideal
UU1	250000	217392	194636	194636
UU2	600000	586028	586028	586028
UU3	1100000	977424	954942	1012686
UU4	1200000	1159393	993654	993654
UU5	1885000	1746538	1868985	1677253
UU6	2986850	2786214	2795272	2795272
UU7	2965160	2727676	2924000	2859672
UU8	4000000	3822480	3632198	3632198
UU9	6150000	6100692	5795028	5795028
UU10	12075000	11248824	11656162	11656162
UU11	13177500	12982238	13067256	12984312

Tabla 5.2. Área cubierta (beneficio asociado sin pesos) por cada variante o criterio en la metodología de subespacios maximales.

En las tablas 5.1 y 5.2 se puede observar que el constructivo logra porcentajes de aprovechamiento altos empleando los diferentes criterios, pero ninguno se impone en todas las instancias. Para enriquecer el constructivo de este trabajo fueron utilizados todos de manera simultánea, teniendo en cuenta que esto genera una sobrecarga en la fase constructiva, pero más adelante se presentará la alternativa de procesamiento paralelo para aliviar este cuello de botella y construir una metodología enriquecida y más eficiente.

A continuación, entre las figuras 5.8a y 5.8g se presenta el constructivo de subespacios maximales paso a paso sobre la instancia UU11.

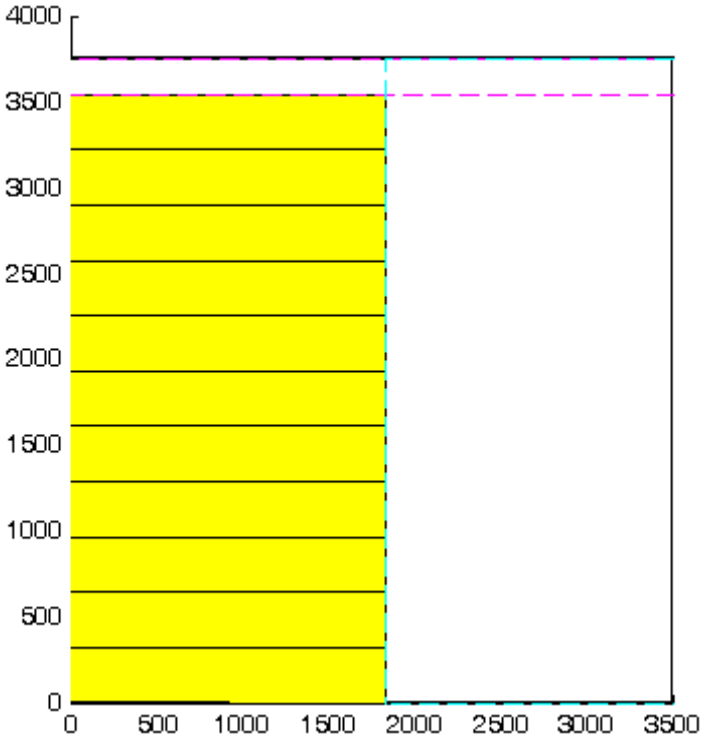

Figura 5.8a. Primer paso.

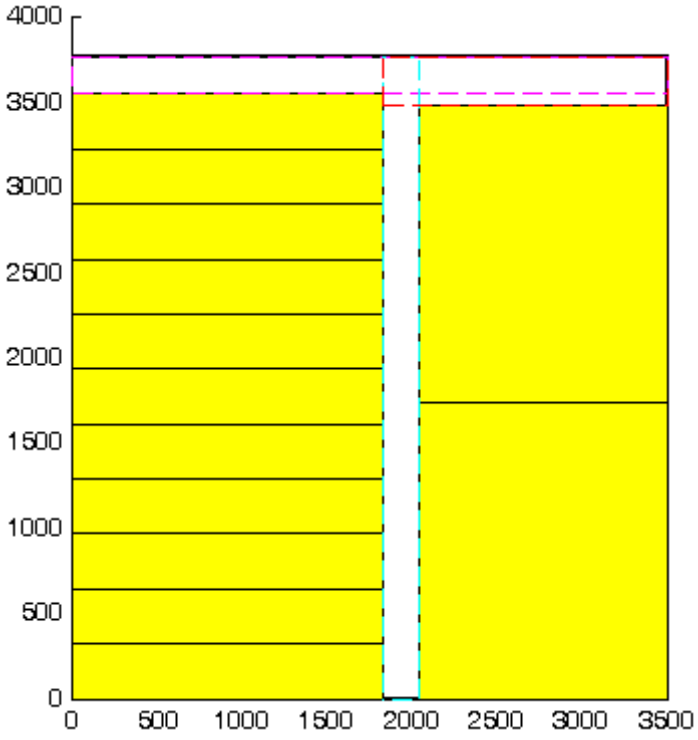

Figura 5.8b. Segundo paso.

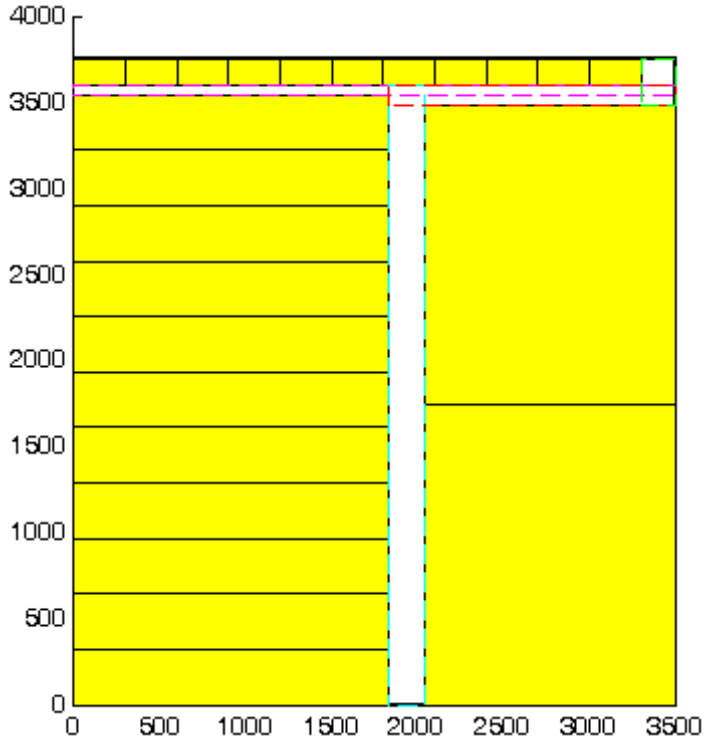

Figura 5.8c. Tercer paso.

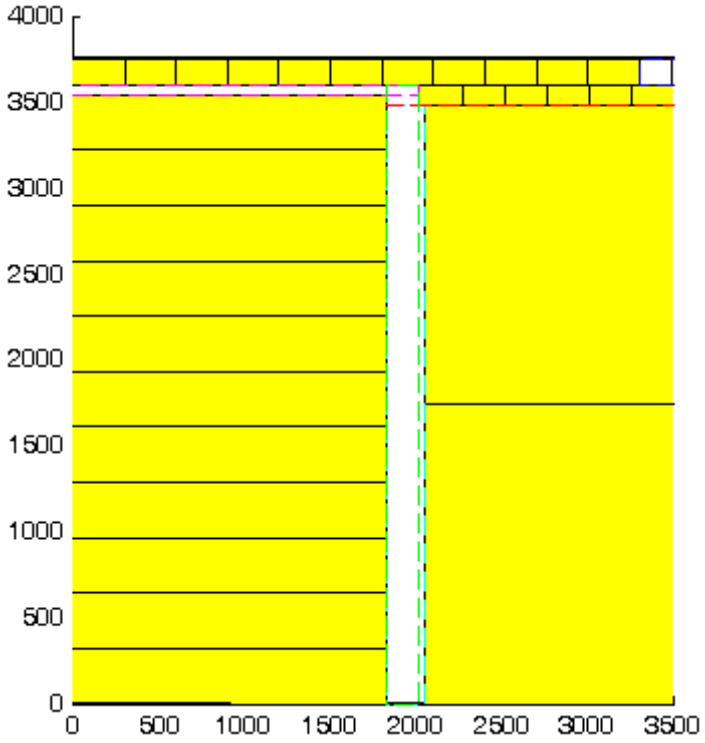

Figura 5.8d. Cuarto paso.

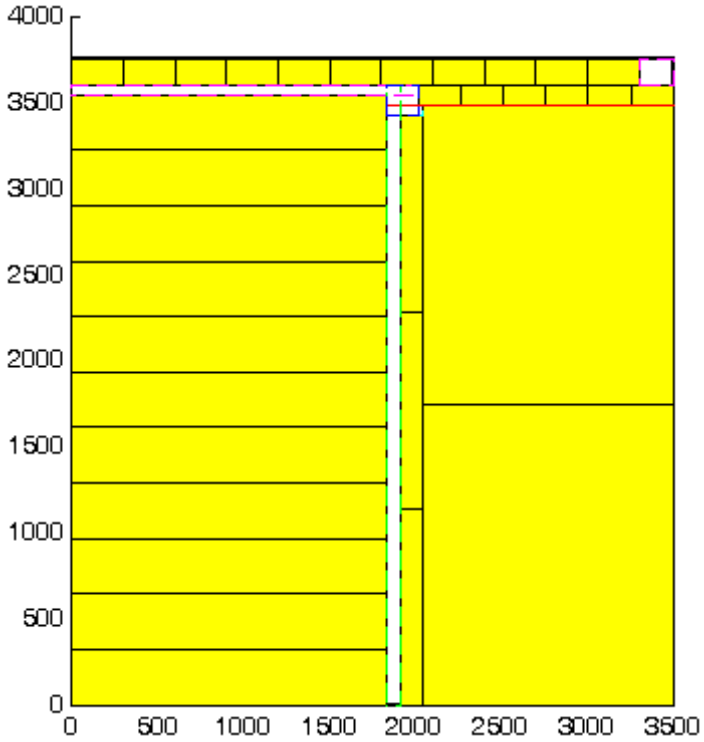

Figura 5.8e. Quinto paso.

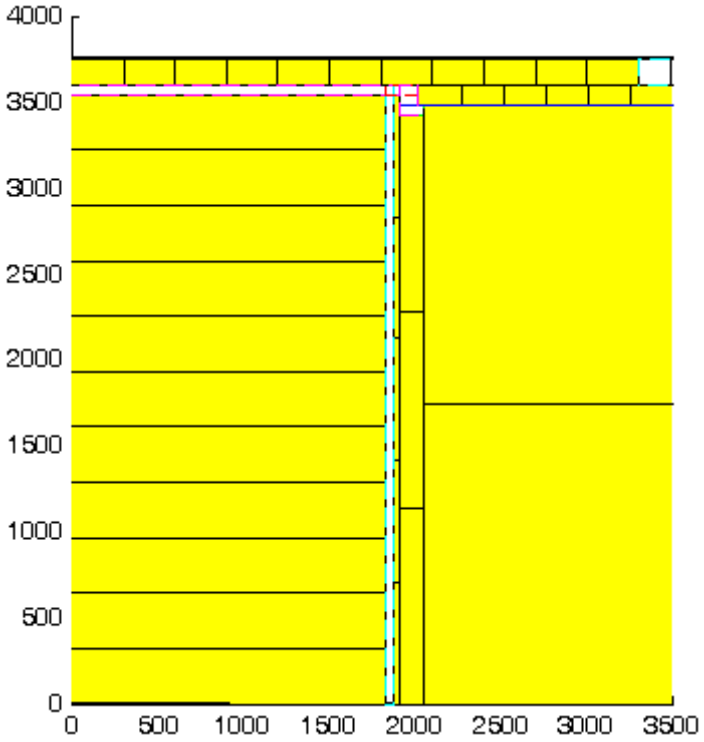

Figura 5.8f. Sexto paso.

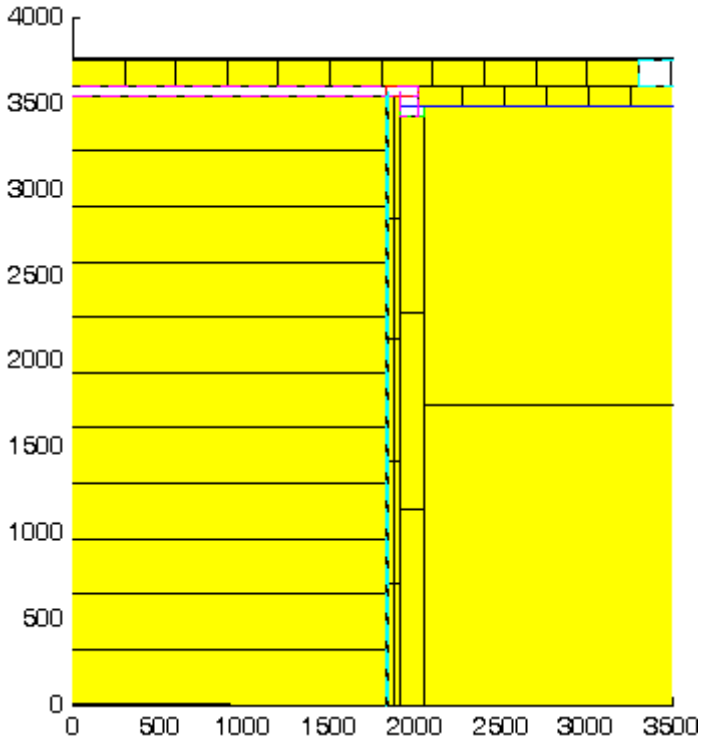

Figura 5.8g. Séptimo paso.

5.5. Codificaciones de la mochila bidimensional irrestricta

El uso de metaheurísticas requiere de una codificación apropiada del problema, existen distintas codificaciones propuestas en la literatura especializada, dos de las más usadas son la representación mediante estructura de datos tipo vector y la tipo árbol de cortes.

La representación natural de un patrón de empaquetamiento se basa en las coordenadas de ubicación de cada rectángulo en la placa. Si la esquina inferior izquierda y la esquina superior derecha de todos los rectángulos son conocidas, entonces el patrón de empaquetamiento puede ser reconstruido fácilmente. La ventaja de una representación natural está ligada con una fácil reconstrucción. Pero si hay cambios en las coordenadas es probable que el patrón de empaquetamiento se traslape.

Por ejemplo, (Jakobs, 1996) hace uso de la estructura de datos tipo vector para codificar los patrones de empaquetamiento de su algoritmo genético, por otro lado, (Kröger, 1995) utiliza la estructura de datos tipo árbol de cortes para codificar los patrones de empaquetamiento.

Codificación por permutaciones para corte no guillotina propuesta por Jakobs

Un patrón de empaquetamiento puede se representado por un permutación $\pi = (i_1,..., i_n)$, (donde, i_j indica que la pieza i ocupa la posición j del vector). La permutación representa la secuencia en la cual los rectángulos son empacados. La ventaja de este tipo de estructura es

la fácil creación de nuevas permutaciones cambiando la secuencia. Una consecuencia de esta estructura es que en cada permutación es asignado un único patrón de empaquetamiento. Para decodificar el genotipo es necesario un algoritmo eficiente que lo convierta en la representación natural.

Una dificultad encontrada es cuando dos permutaciones representan el mismo patrón de empaquetamiento (ver figura 5.9). Esta es difícil de subsanar.

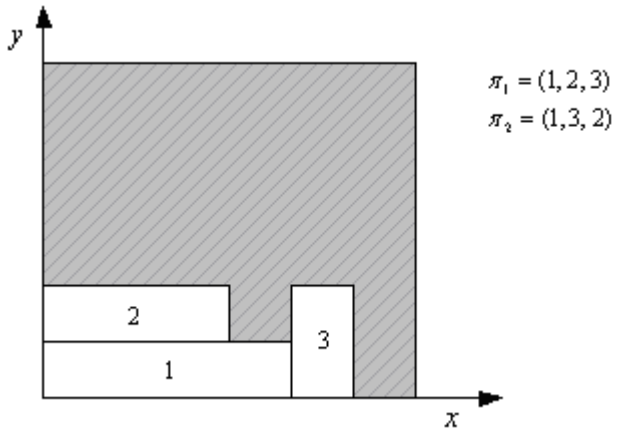

Figura 5.9. Dos permutaciones con el mismo patrón de empaquetamiento.

Codificación en diagrama de árbol para corte no guillotina de primer orden

La creación de la estructura de árbol de cortes se realiza aplicando técnicas numéricas de conglomerados (clusters) que tiene por objeto agrupar elementos. Cada nodo interno del árbol representa la forma en que se realiza el corte y los elementos que pertenecen a cada grupo (Wong *et al.*, 1988).

Se divide la placa original en subespacios. Para asegurar que los subespacios creados presenten cortes de tipo no guillotina se define una codificación de árbol 5-ario completo (Toro *et al.*, 2008).

El número de capas (c) es fijado 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 ecuación 5.1, mientras el número de subespacios (N_S) está dado por la ecuación 5.2.

$$p = 5^c - 1 \tag{5.1}$$

$$N_S = 5^c \tag{5.2}$$

Para representar la estructura que genera los subespacios se define un árbol 5-ario donde cada nodo padre (nodo interno) representa un estampado (un estampado se representa a

través de cuatro cortes) mientras los nodos hoja representan las dimensiones de los subespacios resultantes.

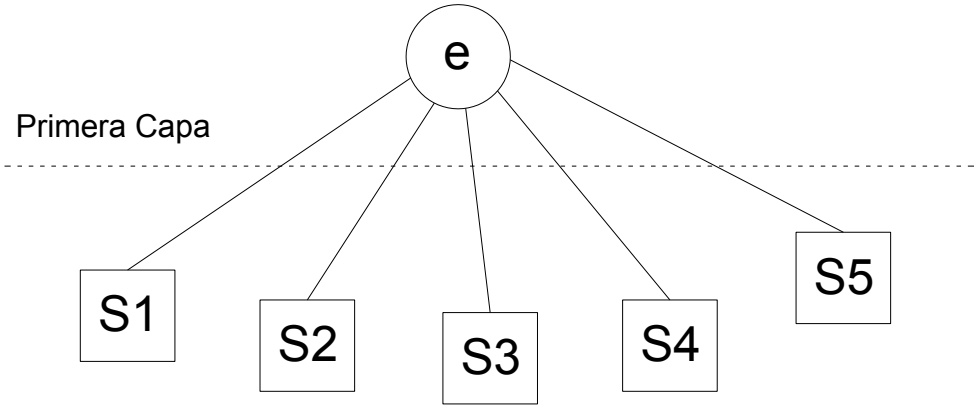

Figura 5.10. Primer nivel

Por ejemplo, la figura 5.10 (primer nivel) ilustra un árbol de cortes fijando el número de capas en 1. Una ventaja de este tipo de codificación es que cualquier conjunto de valores del árbol produce un patrón de corte factible.

La figura 5.11 ilustra un árbol de cortes de dos capas con 6 nodos estampados y 25 subespacios resultantes.

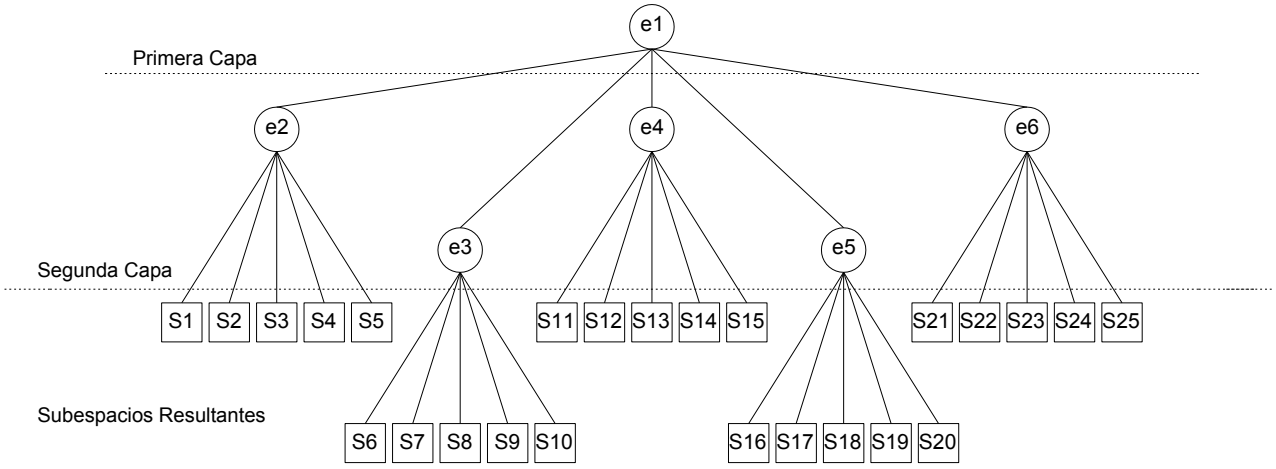

Figura 5.11. Árbol de cortes de dos capas (dos niveles)

La figura 5.12 muestra como se representa un estampado no guillotina gráficamente sobre la mochila.

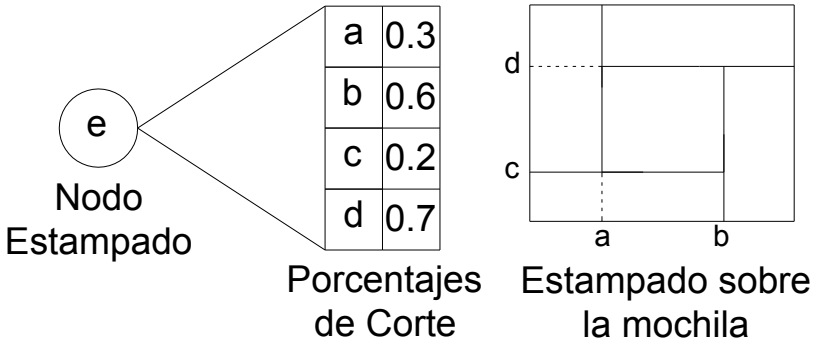

Figura 5.12. Estampado sobre la mochila

La figura 5.13 ilustra las dimensiones de los subespacios resultantes.

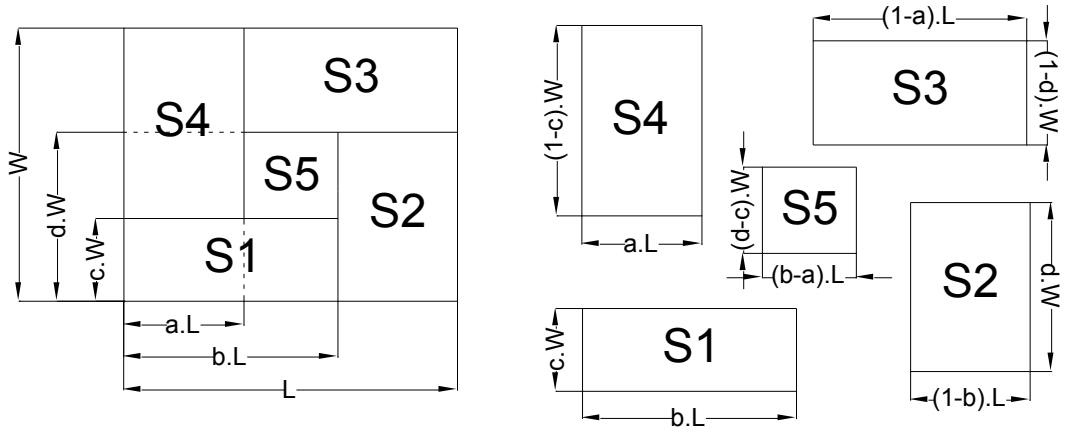

Figura 5.13. Árbol de cortes

Luego de obtener las dimensiones de los subespacios se debe dar paso a la ubicación de las piezas, para esto se utilizan dos heurísticas. La primera es la heurística del bloque trivial donde se acomoda la mayor cantidad del mismo tipo de pieza y la segunda heurística es la de máximos subespacios.

Debido a que en la heurística de los máximos subespacios son presentados 3 diferentes criterios de selección de piezas en total se obtienen cuatro diferentes heurísticas y de éstas se elige la que maximiza la función objetivo (área o beneficio).

En la figura 5.14 se presenta una configuración aplicando la codificación propuesta.

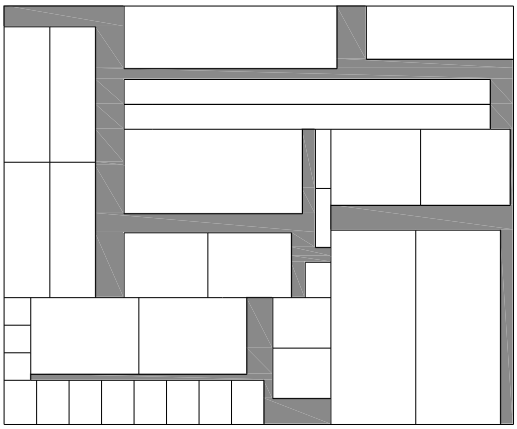

Figura 5.14. Configuración posible

Codificación Rectilinear propuesta por Chen, Fu, y Rodrigues

En (Chen *et al.*, 2002) se propone una matriz, que representa una aproximación discreta de las formas de las piezas a ser ubicadas. La codificación consiste en que se cuadricula todo la placa disponible para ubicar las piezas, cada cuadro tiene dimensiones de 1x1 e inicialmente la matriz que representa la placa esta llena de ceros lo que indica que no hay espacios ocupados, a medida que se van ubicando las piezas los valores de los cuadros que representan la pieza. La figura 5.15 ilustra la matriz y como se realiza el almacenamiento de piezas en esta.

0	1	1	1	1	0	0	0
0	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	2	0	0	3	3
0	0	0	2	2	0	0	0
0	0	0	2	2	0	0	0

Figura 5.15. Matriz de ubicación de las piezas.

En la figura 5.16 se muestra representación rectilínea de la matriz, que generan un patrón de empaquetamiento.

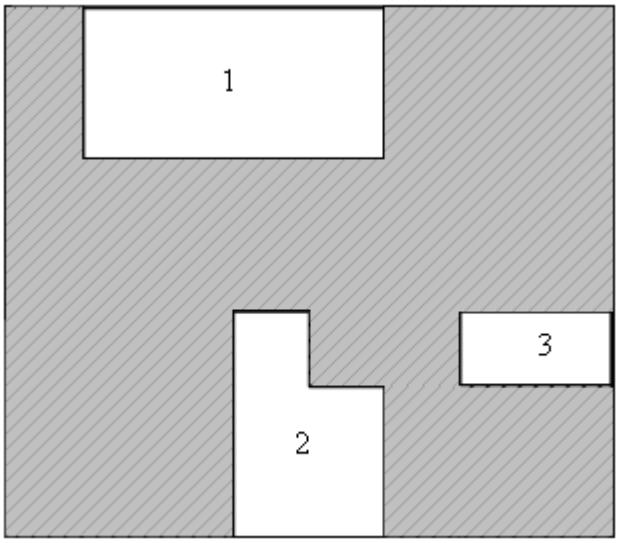

Figura 5.16. Representación rectilínea en la placa

Algunas ventajas de usar esta representación son:

- La representación de cualquier forma geométrica, incluyendo piezas cóncavas y convexas (ver figura 5.16).
- La fácil verificación de rotación y traslapes de piezas.
- Pueden ser aplicadas heurísticas y metaheurísticas.

Algunas desventajas de esta representación son:

- La complejidad de la representación y el crecimiento computacional cuadrático con el tamaño de cada objeto.
- Únicamente son posibles cuatro orientaciones (arriba, abajo, derecha, izquierda)
- Para evaluar la función objetivo se contabilizan el número de espacios sin ocupar para así calcular el porcentaje de utilización del tablero y el valor del área sin ocupar.

En este estudio se seleccionó una representación presentada por (*Wong et al.,* 1988), esta estructura de datos codifica a través de un árbol (llamado árbol de cortes) los patrones de empaquetamiento del problema. Una de las grandes ventajas de la representación en árbol de cortes es que este genera solo patrones de corte tipo no guillotina de primer orden. Además, asegura que para todo el espacio de soluciones que conforman los diferentes patrones no guillotina existe al menos un árbol de cortes que representa una de estas soluciones (Wong *et al.*, 1988). Diferentes metodologías propuestas han corroborado la efectividad de usar una codificación en árbol de cortes en especial la presentada por (Cui, 2007) y la de (Toro *et al.*, 2008).

5.6. Cálculo de la función objetivo

En los subespacios generados por el árbol de cortes se debe realizar la ubicación de las piezas, este proceso se realiza mediante las heurísticas de bloque trivial y subespacios maximales, esto hace que se conserven las restricciones de tipo no guillotina y sea embalada la mayor cantidad de piezas por subespacio (sin pesos) ó el conjunto de piezas que genere mejor beneficio (con pesos).

El algoritmo de *bloque trivial* consiste en encontrar el conjunto de piezas idénticas que maximiza el área (maximiza el beneficio para versión con pesos) del subespacio *j*. La ecuación (5.3) expresa formalmente el algoritmo *best-fit*.

$$\max\left\{\left\lfloor \frac{L_j}{l_i} \right\rfloor \left\lfloor \frac{W_j}{w_i} \right\rfloor\right\} \cdot c_i ; \forall i; i = 1,2,\ldots,n \tag{5.3}$$

El algoritmo de *subespacios maximales,* con los tres diferentes criterios mencionados en la sección 5.3, ubicará capas de piezas en los subespacios. La función objetivo será calculada sumando el área ocupada o el costo asociado a las mismas ubicadas en todos los subespacios en los que se ha dividido el plato original.

Las diferentes heurísticas empleadas para decidir la forma de llenar dichos subespacios en cada iteración, representan diferentes esfuerzos computacionales. Aplicarlos simultáneamente genera un cuello de botella notable especialmente cuando es incrementado el número de iteraciones en el contexto de la metaheurística. Por esta razón en este trabajo se acude al recurso computacional ofrecido por el procesamiento paralelo.

El procesamiento paralelo es una técnica que permite la realización de múltiples tareas de manera simultánea, bien sea aprovechando momentos de inactividad de un solo procesador para lograr dicha simultaneidad en las mismas o integrando varios procesadores para realizar diferentes tareas en el mismo instante de tiempo. Esta forma de ejecución se puede realizar en uno o muchos procesadores de una misma máquina (empleando OpenMP, librería de C++ para este propósito) o en uno o muchos procesadores distribuidos en varias máquinas interconectadas a través de una red (utilizando MPI igualmente para C++). Para el trabajo desarrollado se eligió la paralelización de tareas en una sola máquina, dado que si se cuenta con equipos con varios núcleos (como en este caso que se disponía de un procesador i7) se presenta un mejor rendimiento al evitar la latencia inherente a las redes de datos que comunican procesadores distribuidos en diferentes equipos (Escobar *et al.*, 2011).

En este trabajo, después de identificar el cuello de botella con las 4 variantes constructivas utilizadas, el proceso secuencial ilustrado en la figura 5.17 fue comparado con el proceso paralelizado como se muestra en la figura 5.18.

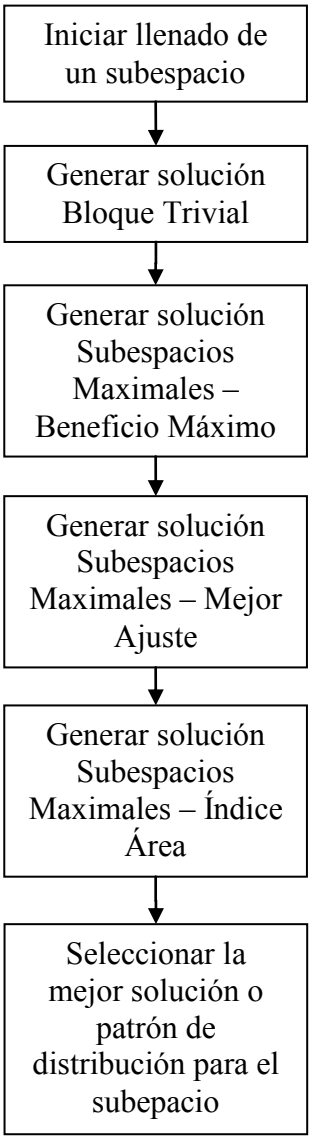

Figura 5.17. Fase constructiva en la versión secuencial del algoritmo

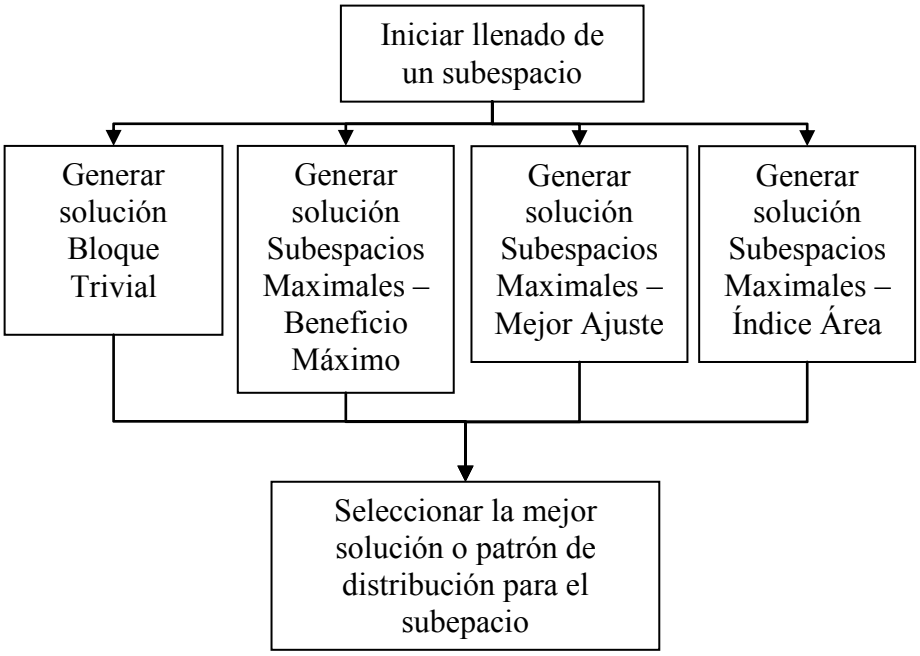

Figura 5.18. Fase constructiva en la versión paralela del algoritmo

Esta simultaneidad es posible gracias a que la librería OpenMP permite dividir un único curso de eventos en varios cursos simultáneos, denominados hilos de ejecución. Y estos hilos de ejecución se ejecutan de manera síncrona a nivel de tareas (todos inician y terminan al mismo tiempo) y asíncrona a nivel de bits (inician y terminan en momentos diferentes) de manera automática por la librearía OpenMP.

5.7. Metodología

La codificación propuesta en este capítulo garantiza la factibilidad en cuanto a las restricciones de corte tipo no guillotina. Por tanto, es necesario encontrar los valores óptimos de los cortes que dividen la mochila, para esto en este estudio se propone el siguiente algoritmo:

Algoritmo híbrido cúmulo de partículas, búsqueda de vecindario variable y algoritmos genéticos ($A_{PSO+VNS+TF}$**)**

El algoritmo propuesto en este trabajo utiliza diferentes técnicas de optimización, combinando características de cúmulo de partículas (PSO) con búsqueda de vecindario variable (VNS), y el operador de mutación de los algoritmos genéticos. El primero es considerado el algoritmo principal, el segundo fue utilizado como un limitador de las características de las partículas que deben ser actualizadas y el concepto de mutación es utilizado como el factor de turbulencia de vuelo de las partículas (Álvarez, 2010).

En este trabajo el VNS es incluído dentro del PSO con el propósito de definir cuáles características deben ser actualizadas por las ecuaciones (4.1) y (4.2). Por lo tanto, el conjunto vecindario N_i^l (donde, $i = 1,2,\ldots,D$) es definido como el número i de características que deben cambiar de valor en la partícula l.

$i=1$, entonces, N_1 = {una característica aleatoria debe cambiar su valor}

$i=2$, entonces, N_2 = {dos características aleatorias deben cambiar su valor}

$i=k$, entonces, N_k = {k características aleatorias deben cambiar su valor}

Por lo tanto, el conjunto resultante de vecindarios es $N = \{N_1, N_2, \ldots, N_D\}$

El operador de mutación comúnmente usado en los algoritmos genéticos es introducido en el algoritmo propuesto intentando emular el factor de turbulencia de vuelo, en este trabajo se presenta una adaptación de la ecuación para actualizar la variable umbral del algoritmo de optimización aceptando el umbral (Dueck y Scheuer, 1990).

El mecanismo de turbulencia consiste en permitir grandes cambios durante las primeras iteraciones, como el algoritmo aceptando el umbral permite la pérdida de calidad para la función objetivo al comienzo del proceso (debido a la relajación de los umbrales). Con el avance del proceso, la turbulencia se volverá más determinística. El factor de turbulencia es definido como la modificación del valor del nodo del árbol, a través del mecanismo mostrado en la ecuación (5.4).

$$node\ i = node\ i + \left(rand - \frac{1}{2} \right) \cdot \left(\sqrt{1 - \frac{k}{TotalIterations}} + \varepsilon \right) \qquad (5.4)$$

La ecuación (5.4) está compuesta por: el valor actual del nodo i del árbol, *rand* es un número aleatorio con distribución uniforme en el intervalo $[0,1]$, k es la iteración actual, *TotalIterations* es el número de cíclos y ε es el porcentaje mínimo para generar un cambio en el árbol, donde $\varepsilon = 100/\max(L,W)$. En este trabajo se introduce el operador de mutación presentado en los algoritmos genéticos, haciendo uso del parámetro llamado taza de mutación, permitiendo la mutación de partículas en cada iteración.

Dado que el algoritmo PSO presenta algunas similitudes con los algoritmos evolutivos como los algoritmos genéticos, diferentes autores han propuesto la inclusión del operador mutación en el algoritmo PSO.

Estas operaciones híbridas normalmente se implementan en cada generación (Andrews, 2006), (Carlisle y Dozier, 2000) o en un intervalo prefijado (Liang y Suganthan, 2005) o son controladas por una función de adaptación definida. En este estudio, en cada generación del algoritmo PSO, la población tiene la probabilidad de utilizar el operador de mutación.

La figura 5.19 muestra el pseudocódigo del algoritmo $A_{PSO+VNS+TF}$.

Inicio
Inicializar población y parámetros
Para *l = 1* hasta *NúmeroTotalCiclos*
Para *i = 1* hasta *TamañoPoblación*
Si $f(\overline{x_i}) < f(\overline{pbest_i})$, entonces
$\overline{pbest_i} = \overline{x_i}$
$N^i = N_1$
Si $f(\overline{x_i}) < f(\overline{gbest})$ entonces
$gbest = \overline{x_i}$
Fin Si
De lo contrario
$N^i = Siguiente\ \ N^i$
Fin Si
Para $k \in N^i$
$v_i^k = wv_{i-1}^k + c_1 rand_1^k \left(pbest_i^k - x_i^k\right) + c_2 rand_2^k \left(gbest_i^k - x_i^k\right)$
$x_i^k = x_{i-1}^k + v_i^k$
Siguiente *k*
Si *rand* < Taza de Mutación
$c\_rand = \lfloor rand \cdot D \rfloor$
$x_i^{c\_rand} = x_i^{c\_rand} + \left(rand - \frac{1}{2}\right) \cdot \left(\sqrt{1 - l / NúmeroTotalCiclos} + \varepsilon\right)$
Siguiente *i*
Siguiente *l*
Fin

Figura 5.19. Algoritmo híbrido de PSO, VNS y Factor de Turbulencia ($A_{PSO+VNS+TF}$)

En cada iteración del algoritmo híbrido $A_{PSO+VNS+TF}$ se genera un estampado no guillotina con un nivel de anidación (es decir, árbol de corte de dos niveles como se ha presentado) y en cada uno de los 25 subespacios generados se empaquetan las piezas con la metodología de bloque trivial o subespacios maximales según convenga más (mejor aprovechamiento del espacio o beneficio asociado a las piezas), de esa forma se articulan la metaheuristica presentada con las diferentes estrategias de construcción de soluciones ofreciendo flexibilidad al algoritmo para la prueba de vecindarios a través de una codificación eficiente.

El recurso del procesamiento paralelo en algoritmos de este tipo se debe realizar en los ciclos más internos del proceso para no generar sobrecarga y no entorpecer la eficiencia del mismo. En el diseño del trabajo presentado se concluyó que el mejor punto para aplicar este recurso era la etapa de llenado de subespacios, porque se aprovechaban las bondades de las

diferentes variantes (un barrido amplio de posibilidades) en la fase constructiva, disminuyendo el tiempo de espera antes de continuar con el PSO base involucrando las fortalezas de todos los criterios. Esta arquitectura permite entonces un mayor aprovechamiento del hardware como se muestra en las figuras 5.20 y 5.21 donde se contrasta el uso de los diferentes núcleos de procesamiento (procesadores) en la versión secuencial del $A_{PSO+VNS+TF}$ y la versión paralela del mismo.

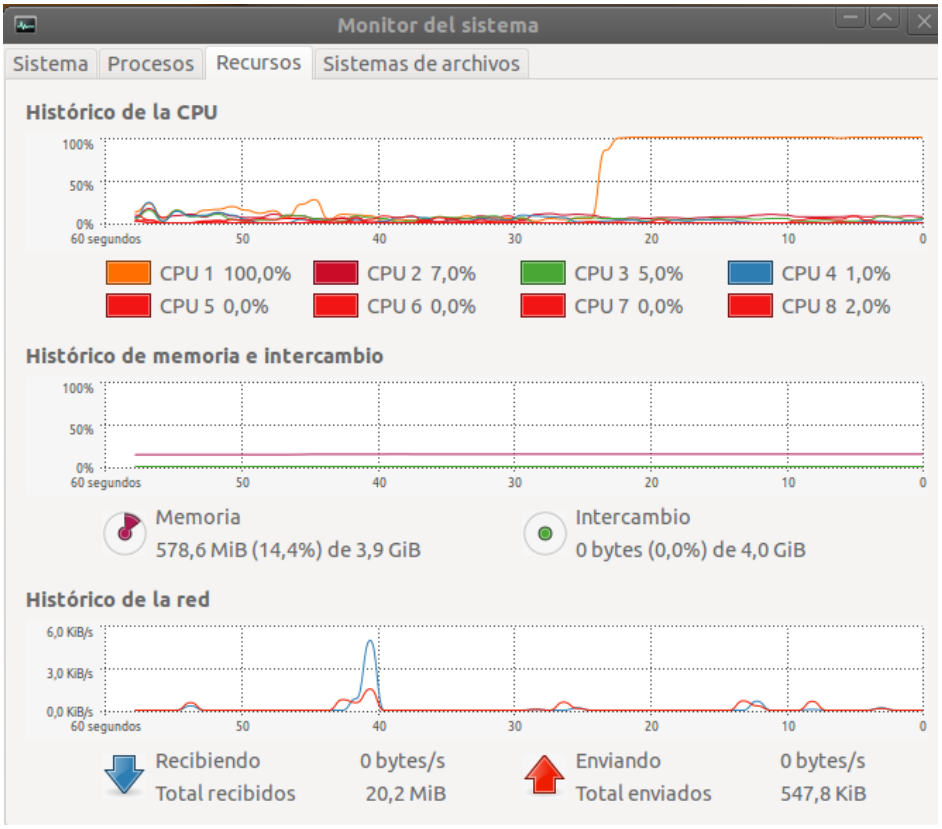

Figura 5.20. Uso de procesadores de la versión secuencial del $A_{PSO+VNS+TF}$

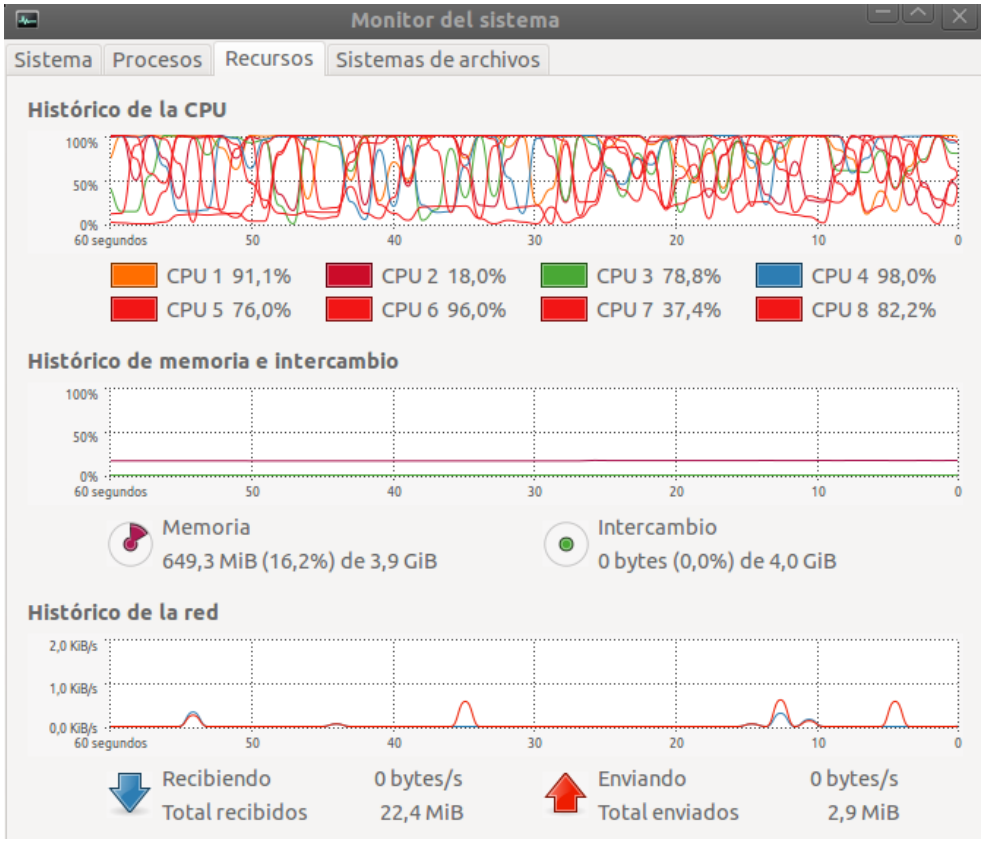

Figura 5.21. Uso de procesadores de la versión paralela del $A_{PSO+VNS+TF}$

Como se observa en las figuras 5.20 y 5.21, la versión secuencial utiliza de manera ineficiente (sólo utiliza la octava parte de la capacidad de procesamiento, es decir un solo núcleo mientras los otros están completamente inactivos) los procesadores para realizar el constructivo enriquecido, sin embargo, la versión paralela utiliza de manera intensiva la mayoría de los núcleos, dejando con poca carga pero utilizándolo igualmente en el proceso de optimización a uno de ellos para sostener los procesos del sistema operativo anfitrión, que para el trabajo presentado se trata de Ubuntu versión 12.04LTS (*Long Term Support*).

5.8. Calibración de parámetros

Una etapa de gran importancia en la aplicación de las técnicas metaheurísticas es el ajuste de parámetros. Diferentes enfoques se presentan para realizar la parametrización. (Pepper *et al*., 2002) recaracterizan los parámetros del recocido simulado de forma tal que sin importar el tamaño del problema la selección de parámetros permita encontrar soluciones de buena calidad. Los parámetros del algoritmo implementado son presentados en la tabla 5.3.

Algoritmo	Parámetros
$A_{PSO+VNS+TF}$	Tamaño de la Población Número de Ciclos c1 (Conocimiento Individual) c2(Conocimiento Grupal) w (Inercia) Tasa de Mutación

Tabla 5.3. Parámetros del algoritmo implementado.

El problema crítico es que se pueden encontrar unos parámetros que presentan respuestas de excelente calidad para un tamaño en especial del problema y que al usar estos mismos en problemas más pequeños o más grandes se ve reducida la calidad de las respuestas y a su vez la eficiencia del algoritmo.

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 una análisis estadístico de la calidad de los resultados. Para realizar la parametrización distintos estudios proponen: clasificar los problemas de prueba (si existen) por complejidad (matemática o computacional), escoger una problema representante (candidato) de cada clase, realizar un ajuste de los parámetros para cada candidato a través de una búsqueda exhaustiva en malla y por último recombinar los parámetros obtenidos para cada clase escogiendo la mejor combinación de estos.

(Zhi-Hui *et al*., 2009) presenta un rango de valores reducido para los parámetros del algoritmo PSO. Teniendo como base los rangos presentados por Zhi-Hui el tamaño de la malla se reduce considerablemente.

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 baja. Para esto se realizo el mismo proceso de calibración para los rangos propuestos del parámetro de mutación en (Gallego *et al*., 2008).

Los valores resultantes de la calibración de parámetros son ilustrados en la tabla 5.4.

Algoritmo	Parámetros	Valores
$A_{PSO+VNS+TF}$	Tamaño de la Población	100
	Número de Ciclos	100
	c1 (Conocimiento Individual)	2,05
	c2(Conocimiento Grupal)	2,05
	w (Inercia)	0,7
	Tasa de Mutación	0,03

Tabla 5.4. Valores de los parámetros de cada algoritmo.

5.9. Resumen

En este capítulo se presentaron las heurísticas, codificaciones, el algoritmo propuesto y la calibración de parámetros que conforman la metodología de solución presentada en este trabajo. El uso de una codificación en árbol de cortes, el cálculo de la función objetivo utilizando 4 heurísticas, la selección de tres técnicas de optimización para generar un algoritmo híbrido de optimización y un proceso cuidadoso de calibración de parámetros del algoritmo representan la metodología de solución de los dos problemas de empaquetamiento propuestos en este estudio.

Utilizar estructuras de datos tipo árbol de cortes garantiza las restricciones de corte tipo no guillotina y presenta un escenario propicio para las técnicas metaheurísticas de optimización propuestas. Dado que los árboles de corte no representan la ubicación de las piezas sobre el material, una rutina debe realizar este proceso además de medir la calidad de la ubicación propuesta. En este estudio esta rutina es llamada cálculo de la función objetivo. Realizar el cálculo de la función objetivo utilizando 4 heurísticas, potencia el uso del árbol de cortes.

En este estudio se propone un algoritmo híbrido $A_{PSO+VNS+TF}$ para optimizar el árbol de cortes, inspirado en tres técnicas metaheurísticas (PSO, GA y VNS).

Calibrar los parámetros de las técnicas metaheurísticas es un proceso importante dado que este afecta directamente la calidad y el tiempo de respuesta de los algoritmos. En este estudio se realiza un ajuste detallado y cuidadoso de todos los parámetros, mediante una búsqueda exhaustiva en malla sobre los rangos de valores sugeridos en la literatura especializada.

Un análisis estadístico permitirá verificar el desempeño en la solución de los diferentes problemas propuestos en este estudio. En el siguiente capítulo se describe el estudio computacional realizado del algoritmo de solución sobre los problemas propuestos.

6. ANÁLISIS DE RESULTADOS

6.1. Introducción

En este estudio se realiza un análisis estadístico entorno a la calidad de la solución y tiempos de respuesta del algoritmo propuesto, utilizando como descriptores estadísticos la media y la desviación estándar para representar el comportamiento de la calidad de respuesta del algoritmo implementado para cada problema de este estudio.

Se describen los casos de prueba utilizados, los detalles de la implementación de los algoritmos, los resultados computacionales y por último se realiza un análisis comparativo de los resultados obtenidos.

6.2. Casos de prueba

Dado que en este estudio se tiene en cuenta todos los posibles surtidos de piezas, las librerías de casos de prueba seleccionadas deben cumplir con todas las diferentes características necesarias, las librerías utilizadas para los diferentes problemas son variados en cuanto a la complejidad matemática y son diseñados especialmente para cada tipo de problema.

Fueron seleccionados 35 casos de prueba para el problema de la mochila bidimensional irresticta no guillotina, 24 casos para la versión sin pesos ([UU1-UU11] y [GCUT1-GCUT13]) y 11 casos para la versión con pesos ([UW1-UW11]). Estos casos presentan diferentes tipos de mochilas con distribuciones entre 10 y 60 piezas, la base de datos es presentada por (Hifi, 2001) y disponible en línea en (Hifi, 1997); (Beasley, 2002). Diferentes estudios han utilizado estos casos de prueba para realizar una especie de *benchmark* de las metodologías propuestas.

Existe gran cantidad de librerías para los problemas presentados en este estudio, pero luego de ser analizadas muchas caen en la categoría de problemas de empaquetamiento perfecto, es decir, los casos prueba son desarrollados a partir de la solución óptima del problema, lo cual hace que metodologías exactas parezcan un buen método de solución.

Las bases de datos utilizadas en este estudio no necesariamente presentan un óptimo global en su límite inferior, que hacen que las metodologías exactas renuncien en su proceso de optimización debido al costo computacional.

6.3. Detalles de la implementación

Todos los algoritmos fueron desarrollados en C++ ®, sobre un ordenador con unas especificaciones de un procesador Pentium i7 de 2,99 GHz por núcleo de procesamiento y una memoria RAM de 4 GB.

6.4. Resultados computacionales

Los sistemas de prueba usados en este estudio fueron tomados de la literatura especializada después de realizar una revisión bibliográfica. Las metodologías usadas en la solución de dichos problemas son resueltos con métodos aproximados o exactos. Los problemas seleccionados son variados en cuanto a la complejidad matemática y tipo de problema a solucionar.

Se presentan para todos los casos de prueba de cada tipo de problema la mejor solución reportada en la literatura especializada (*Best Known Solution*). Los resultados de (Birgin et al., 2010) presentan las mejores soluciones aunque trabajos como (Parreño *et al*., 2008) alcanzan buenos resultados.

Las tablas 6.1 y 6.2 presentan los mejores resultados reportados en la literatura especializada y su respectivo autor.

Caso	Mejor Solución Conocida	Tiempo Utilizado (Segundos)	Caso	Mejor Solución Conocida	Tiempo Utilizado (Segundos)	Autor
GCUT1	58480	0	UU1	245205	0.02	(Birgin, 2010)
GCUT2	61146	0	UU2	595288	0.08	(Birgin, 2010)
GCUT3	61275	0.02	UU3	1088154	0.04	(Birgin, 2010)
GCUT4	61918	0.06	UU4	1191071	0.82	(Birgin, 2010)
GCUT5	246000	0	UU5	1870038	4.52	(Birgin, 2010)
GCUT6	243598	0	UU6	2950760	0.16	(Birgin, 2010)
GCUT7	244306	0	UU7	2943852	10.76	(Birgin, 2010)
GCUT8	247815	0.1	UU8	3969784	3.3	(Birgin, 2010)
GCUT9	971100	0	UU9	6100692	0.03	(Birgin, 2010)
GCUT10	982025	0	UU10	11955852	0.04	(Birgin, 2010)
GCUT11	980096	0	UU11	13157811	988.84	(Birgin, 2010)
GCUT12	979986	0.06				
GCUT13	8997780	10.4				

Tabla 6.1. Mejor solución reportada con y sin pesos, sin rotación.

Para realizar un estudio estadístico de la calidad de las respuestas obtenidas, los casos de prueba fueron ejecutados 20 veces. Como índice de calidad de las respuestas se utilizó el error porcentual, 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

(6.1) 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.

Caso	Mejor Solución Conocida	Tiempo Utilizado (Segundos)	Autor
UW1	6036	0	(Birgin, 2010)
UW2	8720	0	(Birgin, 2010)
UW3	6652	0	(Birgin, 2010)
UW4	8326	0.01	(Birgin, 2010)
UW5	7780	0	(Birgin, 2010)
UW6	6803	0.03	(Birgin, 2010)
UW7	10464	0.01	(Birgin, 2010)
UW8	7692	0.05	(Birgin, 2010)
UW9	7128	0.02	(Birgin, 2010)
UW10	7507	0.12	(Birgin, 2010)
UW11	16400	1.56	(Birgin, 2010)

Tabla 6.2. Mejor solución reportada con y sin pesos, con rotación.

$$error = \frac{BestKnownSolution - SoluciónObtenida}{BestKnownSolution} \tag{6.1}$$

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 6.3 resume los errores medios y las desviaciones estándar para cada problema. Mientras que entre las tablas 6.4, y 6.12 se presentan los mejores valores obtenidos en cada muestra comparados con los obtenidos en la literatura especializada, además de la comparación de tiempos entre las versiones secuencial y paralela de la metaheurística presentada.

	Algoritmo Secuencial		Algoritmo Paralelo	
Problema	**Error Medio**	**Desviación Estándar**	**Error Medio**	**Desviación Estándar**
Con pesos	0.040402	0.0124	0.035902	0.0223
Sin pesos	0.011148	0.00732	0.00423	0.00621

Tabla 6.3. Error medio y desviación estándar para cada problema.

	Algoritmo Secuencial	**Mejor Solución Conocida**	**Diferencia Algoritmo Secuencial y Solución Conocida**
GCUT1	56460	58480	-2020
GCUT2	61146	61146	**0**
GCUT3	61036	61275	-239
GCUT4	61698	61918	-220
GCUT5	246000	246000	**0**
GCUT6	238998	243598	-4600
GCUT7	242567	244306	-1739
GCUT8	247815	247815	**0**
GCUT9	971100	971100	**0**
GCUT10	982025	982025	**0**
GCUT11	980096	980096	**0**
GCUT12	979986	979986	**0**
GCUT13	8997780	8997780	**0**

Tabla 6.4. Comparación de los mejores resultados de la versión secuencial del $A_{PSO+VNS+TF}$ para el problema de la mochila bidimensional no guillotinada frente a los reportados en (Birgin, 2010) para las instancias GCUT (sin pesos).

	Algoritmo Paralelo	**Mejor Solución Conocida**	**Diferencia Algoritmo Paralelo y Solución Conocida**
GCUT1	56460	58480	-2020
GCUT2	61146	61146	**0**
GCUT3	61036	61275	-239
GCUT4	61698	61918	-220
GCUT5	246000	246000	**0**
GCUT6	238998	243598	-4600
GCUT7	242567	244306	-1739
GCUT8	247815	247815	**0**
GCUT9	971100	971100	**0**
GCUT10	982025	982025	**0**
GCUT11	980096	980096	**0**
GCUT12	979986	979986	**0**
GCUT13	8997780	8997780	**0**

Tabla 6.5. Comparación de los mejores resultados de la versión paralela del $A_{PSO+VNS+TF}$ para el problema de la mochila bidimensional no guillotinada frente a los reportados en (Birgin, 2010) para las instancias GCUT (sin pesos).

	Tiempo en segundos algoritmo secuencial	Tiempo en segundos algoritmo paralelo	Diferencia Tiempo Algoritmo Secuencial y Algoritmo Paralelo
GCUT1	106.35	60.88	**45.47**
GCUT2	127.19	73.55	**53.64**
GCUT3	146.76	87.97	**58.79**
GCUT4	190.1	111.21	**78.89**
GCUT5	98.21	71.58	**26.63**
GCUT6	119.84	59.04	**60.8**
GCUT7	123.21	67.63	**55.58**
GCUT8	201.74	88.67	**113.07**
GCUT9	107.94	70.05	**37.89**
GCUT10	116.01	54.76	**61.25**
GCUT11	159.21	51.26	**107.95**
GCUT12	185.45	104.82	**80.63**
GCUT13	127.24	79.85	**47.39**

Tabla 6.6. Comparación del tiempo utilizado por las versiones secuencial y paralela del $A_{PSO+VNS+TF}$ para el problema de la mochila bidimensional para las instancias GCUT (sin pesos).

	Algoritmo Secuencial	Mejor Solución Conocida	Diferencia Algoritmo Secuencial y Solución Conocida
UU1	245205	245205	**0**
UU2	595288	595288	**0**
UU3	1081136	1088154	-7018
UU4	1178295	1191071	-12776
UU5	1870038	1870038	**0**
UU6	2950760	2950760	**0**
UU7	2932092	2943852	-11760
UU8	3969784	3969784	**0**
UU9	6100692	6100692	**0**
UU10	11955852	11955852	**0**
UU11	13141175	13157811	-16636

Tabla 6.7. Comparación de los mejores resultados de la versión secuencial del $A_{PSO+VNS+TF}$ para el problema de la mochila bidimensional no guillotinada frente a los reportados en (Birgin, 2010) para las instancias UU (sin pesos).

	Algoritmo Paralelo	Mejor Solución Conocida	Diferencia Algoritmo Paralelo y Solución Conocida
UU1	245205	245205	**0**
UU2	595288	595288	**0**
UU3	1081136	1088154	-7018
UU4	1178295	1191071	-12776
UU5	1870038	1870038	**0**
UU6	2950760	2950760	**0**
UU7	2932092	2943852	-11760
UU8	3969784	3969784	**0**
UU9	6100692	6100692	**0**
UU10	11955852	11955852	**0**
UU11	13141175	13157811	-16636

Tabla 6.8. Comparación de los mejores resultados de la versión paralela del $A_{PSO+VNS+TF}$ para el problema de la mochila bidimensional no guillotinada frente a los reportados en (Birgin, 2010) para las instancias UU (sin pesos).

	Tiempo en segundos algoritmo secuencial	Tiempo en segundos algoritmo paralelo	Diferencia Tiempo Algoritmo Secuencial y Algoritmo Paralelo
UU1	150.62	103.725116	**46.894884**
UU2	175.63	136.125594	**39.504406**
UU3	192.59	85.913802	**106.676198**
UU4	233.36	104.345034	**129.014966**
UU5	235.36	140.419368	**94.940632**
UU6	188.39	128.744809	**59.645191**
UU7	249.74	164.001438	**85.738562**
UU8	263.24	153.533357	**109.706643**
UU9	243.88	144.431402	**99.448598**
UU10	268.82	183.194972	**85.625028**
UU11	227.49	137.974819	**89.515181**

Tabla 6.9. Comparación del tiempo utilizado por las versiones secuencial y paralela del $A_{PSO+VNS+TF}$ para el problema de la mochila bidimensional para las instancias UU (sin pesos).

	Algoritmo Secuencial	Mejor Solución Conocida	Diferencia Algoritmo Secuencial y Solución Conocida
UW1	6036	6036	**0**
UW2	8720	8720	**0**
UW3	6652	6652	**0**
UW4	8326	8326	**0**
UW5	7780	7780	**0**
UW6	6615	6803	-188
UW7	10464	10464	**0**
UW8	7692	7692	**0**
UW9	7038	7128	-90
UW10	7507	7507	**0**
UW11	15920	16400	-480

Tabla 6.10. Comparación de los mejores resultados de la versión secuencial del $A_{PSO+VNS+TF}$ para el problema de la mochila bidimensional no guillotinada frente a los reportados en (Birgin, 2010) para las instancias UW (con pesos).

	Algoritmo Paralelo	Mejor Solución Conocida	Diferencia Algoritmo Paralelo y Solución Conocida
UW1	6036	6036	**0**
UW2	8720	8720	**0**
UW3	6652	6652	**0**
UW4	8326	8326	**0**
UW5	7780	7780	**0**
UW6	6615	6803	-188
UW7	10464	10464	**0**
UW8	7692	7692	**0**
UW9	7038	7128	-90
UW10	7507	7507	**0**
UW11	15920	16400	-480

Tabla 6.11. Comparación de los mejores resultados de la versión paralela del $A_{PSO+VNS+TF}$ para el problema de la mochila bidimensional no guillotinada frente a los reportados en (Birgin, 2010) para las instancias UW (con pesos).

	Tiempo en segundos algoritmo secuencial	Tiempo en segundos algoritmo paralelo	Diferencia Tiempo Algoritmo Secuencial y Algoritmo Paralelo
UW1	155.116128	92.161773	**62.954355**
UW2	212.382875	135.485146	**76.897729**
UW3	188.865993	106.526342	**82.339651**
UW4	244.01729	132.461649	**111.555641**
UW5	188.921776	109.015751	**79.906025**
UW6	245.638779	141.823401	**103.815378**
UW7	247.849019	131.640604	**116.208415**
UW8	280.420116	155.20341	**125.216706**
UW9	225.353528	131.927302	**93.426226**
UW10	276.807234	169.749345	**107.057889**
UW11	317.207316	174.579342	**142.627974**

Tabla 6.12. Comparación del tiempo utilizado por las versiones secuencial y paralela del $A_{PSO+VNS+TF}$ para el problema de la mochila bidimensional para las instancias UW (con pesos).

La tabla 6.13 presenta un resumen de las tablas 6.4 a la 6.12. Esta tabla compara el mejor valor encontrado por los algoritmos para cada instancia del problema con la mejor solución reportada. En esta comparación puede ocurrir que la mejor respuesta supere la mejor respuesta reportada, que no la supere o que la iguale.

	Con Pesos		Sin Pesos	
	Calidad	**Tiempo**	**Calidad**	**Tiempo**
Igualados (Empate con la literatura especializada)	8	-	15	1
Inferiores (Por debajo de la literatura especializada)	3	11	9	23

Tabla 6.13. Comparación global de resultados obtenidos con los presentes en (Birgin, 2010).

La figura 6.1 representa gráficamente los valores de los descriptores estadísticos (media y desviación estándar) de la calidad de solución en los diferentes problemas.

La tabla 6.14 presenta los tiempos computacionales requeridos por el algoritmo para cada problema. En esta se ilustran dos tipos de tiempos: totales y promedios por conjunto de problemas.

El tiempo total se definen 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 con igual complejidad (mediana o gran escala) sobre el número total de casos de prueba.

La figura 6.2 ilustra los tiempos totales utilizados por el algoritmo para resolver cada problema.

Problema	Tiempos	Algoritmo Secuencial	Algoritmo Paralelo
Con Pesos	Tiempo total	51651.60108	29611.4813
	Tiempo promedio	234.7800049	134.597642
Sin Pesos	Tiempo total	84767.39	49273.3929
	Tiempo promedio	176.60	102.652902

Tabla 6.14. Tiempos utilizados por cada algoritmo (segundos).

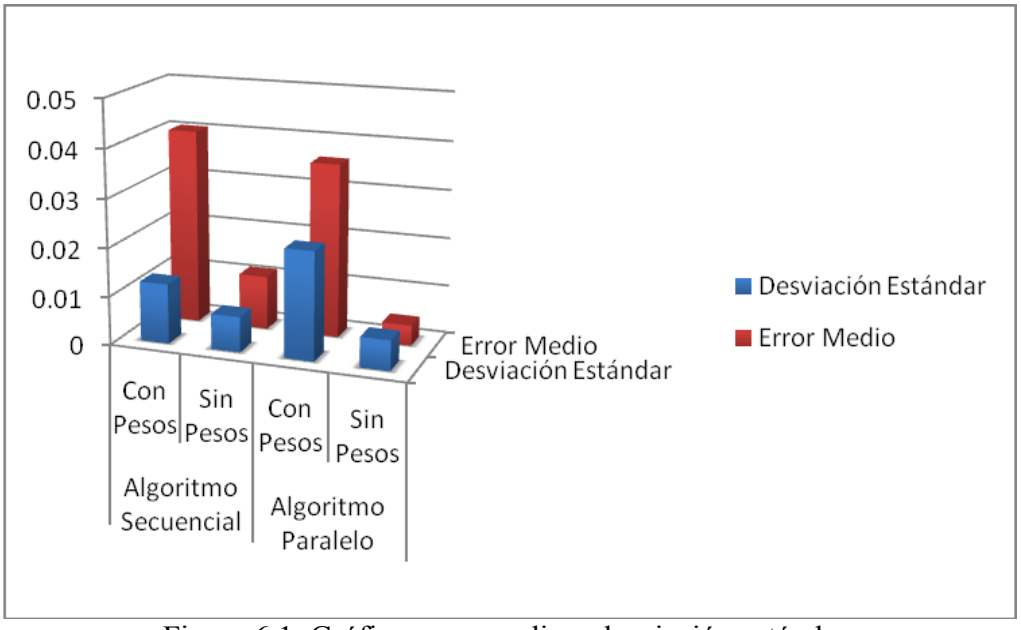

Figura 6.1. Gráfica error medio y desviación estándar.

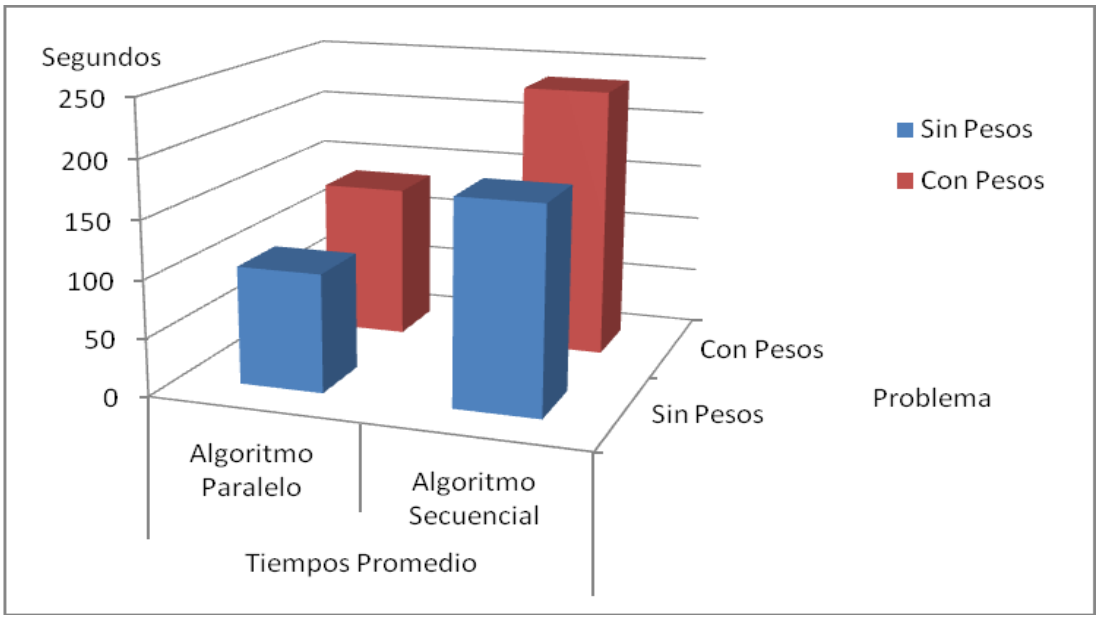

Figura 6.2. Gráfica de tiempos promedio (segundos).

6.5. Análisis

Entre las tablas 6.4, y 6.12 se resumen los mejores resultados alcanzados por la metodología propuesta. En la tabla 6.13 se resumen los resultados comparados con los reportados en la literatura especializada. De la tabla 6.13 se logra identificar que la metodología propuesta para resolver los problemas de la mochila sin pesos asociados, alcanza en un 63% las mejores respuestas reportadas en la literatura especializada y para el problema con pesos se alcanzan el 73% de los óptimos reportados.

La metodología propuesta aplicada a los problemas de la mochila en general presenta un comportamiento regular, alcanzando un 66% de las respuestas reportadas en la literatura.

Al analizar la tabla 6.14, los tiempos de respuesta de la metodología propuesta son razonables para resolver los problemas de la mochila. Obteniendo un tiempo promedio para el algoritmo secuencial de 235 segundos para el problema con pesos y 177 segundos para el problema sin pesos (con distribuciones de hasta 60 tipos de piezas). Estos tiempos son mejorados al incluir la arquitectura de procesamiento paralelo en la metodología, paralelizando las diferentes heurísticas aplicadas a los subespacios generados en los estampados. Alcanzando respuestas con la misma calidad para ambas versiones del problema (con y sin pesos) en unos tiempos promedio de 135 segundos y 107 segundos respectivamente. Esta incursión en paralelo reduce los tiempos de cómputo en un 42%.

La metodología propuesta utilizando el algoritmo secuencial para los problemas de la mochila tiene un error medio del 2,6% respecto a la mejor respuesta reportada, mientras que empleando el algoritmo paralelo se obtiene un error medio de 1,8%, lo que significa que para el problema de mochila con o sin pesos asociados, la solución obtenida con la metodología propuesta está muy cerca del óptimo global, dado que en aplicaciones reales se

manejan en el proceso de corte o empaquetamiento unos límites de desperdicio del 30%. En este estudio 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.

Los tiempos de cómputo utilizados por la metodología son mayores a los reportados en la literatura especializada pero debido a las diferencias entre las arquitecturas de cómputo y lenguajes de programación no se puede concluir entre metodologías. Por otro lado, los tiempos de cómputo de la metodología propuesta presentan un comportamiento regular y no dependen fuertemente de los parámetros del problema (número de tipos de piezas, tamaño de las piezas y tamaño de la mochila). En particular los tiempos de cómputo utilizados por la metodología propuesta son razonables.

Además, con la metodología desarrollada se alcanzan resultados de mejor calidad que los reportados a través de métodos aproximados (Egeblad y Pisinger, 2009) en la literatura especializada.

6.6. Resumen

Se presentaron los casos de prueba utilizados en este estudio, las especificaciones del hardware y software sobre los cuales se ejecutaron los algoritmos propuestos, los resultados obtenidos para realizar el análisis estadístico, tiempos de respuesta de la metodología y análisis de resultados.

Se efectuó un análisis estadístico con el fin de medir la calidad del algoritmo implementado sobre cada problema, utilizando la media y la desviación estándar del error (diferencia entre la solución alcanzada y la mejor solución reportada en la literatura) como descriptores estadísticos y definiendo un tamaño de muestra de 20 ejecuciones por instancia de prueba. Además fueron medidos los tiempos de ejecución utilizados por el algoritmo.

La metodología propuesta presenta un error de 1,8% para el algoritmo en paralelo y 2,6% para el algoritmo secuencial para ambos problemas de la mochila y logra alcanzar gran parte de las soluciones óptimas reportadas en la literatura.

7. CONCLUSIONES Y RECOMENDACIONES

En esta investigación se estudió un algoritmo híbrido de optimización metaheurística para solucionar el problema de la mochila bidimensional irrestricta con patrones de corte tipo no guillotina con y sin pesos asociados a las piezas (*unconstrained (un)weighted two-dimensional non-guillotine knapsack problem*).

Se realizó la revisión bibliográfica de los modelos matemáticos de los problemas de la mochila bidimensional irrestricta con patrones de corte tipo no guillotina, con y sin pesos asociados a las piezas. Los problemas propuestos en este trabajo han sido estudiados por más de 6 décadas, sin embargo no se ha llegado a un consenso general que determine un modelo matemático definido y que incluyan las diferentes características que den solución al problema de la vida real, en especial los patrones de corte no guillotina. En este trabajo se adaptó un modelo propuesto por (Beasley, 2004), para describir los problemas presentados en este estudio.

Se realizó una revisión bibliográfica de las diferentes técnicas metaheurísticas de optimización propuestas para resolver los problemas de de la mochila bidimensional irrestricta. Con base en las diferentes técnicas de optimización estudiadas en este trabajo se propuso un algoritmo híbrido, resultante de la combinación entre estas. En este estudio se utilizaron tres técnicas metaheurísticas (cúmulo de partículas, búsqueda en vecindario variable y algoritmos genéticos), con base en estos se generó un algoritmo híbrido de optimización que reúne características de cada uno.

Se realizó una revisión de las diferentes codificaciones propuestas para los problemas de de la mochila bidimensional irrestricta. En este estudio se optó por una codificación en árbol de cortes. Además de esto también se revisaron las heurísticas propuestas en la literatura.

Se desarrolló una metodología de solución que consiste en utilizar un árbol de cortes para dividir la mochila en un conjunto de subespacios. La división óptima de la mochila es encontrada a través de un algoritmo híbrido que combina tres técnicas metaheurísticas de optimización, mientras que la disposición de piezas sobre el conjunto de subespacios resultantes se realiza a través de cuatro diferentes heurísticas de empaquetamiento.

El algoritmo propuesto presenta resultados de buena calidad al resolver casos de prueba presentados en la literatura especializada. Aunque este incluye una gran cantidad de parámetros que deben ser ajustados y emplea un alto tiempo de cómputo.

El algoritmo propuesto explota las principales propiedades de cada técnica metaheurística de acuerdo a la codificación de árbol de cortes para mejorar los procesos de exploración y explotación que son la base de los procesos búsqueda de las técnicas combinatorias de optimización.

Se desarrolló una versión paralela de la metodología que disminuyó los tiempos de respuesta junto con un mejoramiento de la calidad de los resultados.

Trabajos futuros propuestos en esta investigación:

Implementar el algoritmo propuesto en la solución de los problemas de empaquetamiento óptimo bidimensional no guillotina en una sola placa, en placas y en rollos infinitos con y sin rotación de piezas.

Adaptar la codificación propuesta en el estudio problemas de empaquetamiento óptimo tridimensional no guillotina y extruir la codificación para el problema de empaquetamiento óptimo de cajas en contenedores usando paletas.

Usar otras técnicas metaheurísticas de optimización utilizando la codificación propuesta con el fin de mejorar la calidad de las respuestas.

Implementar una metodología exacta para realizar el ajuste óptimo de parámetros de las técnicas metaheurísticas de optimización propuestas en este trabajo.

Utilizar esta metodología de solución en problemas de la vida real, realizando las adaptaciones necesarias y haciendo uso de información real.

REFERENCIAS BIBLIOGRÁFICAS

ALVAREZ, D. "Solución del problema de empaquetamiento óptimo bidimensional en una sola placa, en placas y rollos infinitos con y sin rotación de piezas usando técnicas metaheurísticas de optimización". *Tesis de Maestría Universidad Tecnológica de Pereira*, 2010.

ALVAREZ-VALDÉS, R.; PARREÑO F. y TAMARIT J. M. "A tabu search algorithm for a twodimensional non-guillotine cutting problem". *European Journal of Operational Research*, 183:1167–1182, 2007.

ANDREWS, P. S. An investigation into mutation operators for particle swarm optimization. *in Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, pp. 1044–1051, 2006.

BALDACCI, R. y BOSCHETTI M. A. "A cutting-plane approach for the two-dimensional orthogonal non-guillotine cutting problem". *European Journal of Operational Research*, 183:1136–1149, 2007.

BEASLEY J. E. "Algorithms for unconstrained two-dimensional guillotine cutting". *Journal of the Operational Research Society*, vol. 36, pp. 297-306, 1985.

BEASLEY, J. E. "An exact two-dimensional non-guillotine cutting tree-search procedure". *Operations Research*, 33:49–64, 1986.

BEASLEY, J. E. "A population heuristic for constrained two-dimensional nonguillotine cutting". *European Journal of Operational Research*, vol. 156, pp. 601- 627, 2004.

BEN, S. CHU, C. y ESPINOUSE, M. L. "Characterization and modelling of guillotine constraints". *European Journal of Operational Research*, vol. 191, pp. 112–126, 2008.

BIRO, M. y BOROS, E. "Network flows and non-guillotine cutting patterns". *European Journal of Operational Research*, vol. 16, pp. 297–306, 1985.

BIRGIN, E. G.; MARTÍNEZ, J. M.; MASCARENHAS, W. F. y RONCONI, D. P. "Method of sentinels for packing items within arbitrary convex regions". *Journal of the Operational Research Society*, 57:735–746, 2006.

BIRGIN, E. G.; MARTÍNEZ, J. M.; NISHIHARA, F. H. y RONCONI, D. P. "Orthogonal packing of rectangular items within arbitrary convex regions by nonlinear optimization". *Computers and Operations Research*, 33:3535–3548, 2006.

BIRGIN, E. G.; LOBATO, R. D. y MORABITO, R. "An effective recursive partitioning approach for the packing of identical rectangles in a rectangle". *Journal of the Operational Research Society*, 61:306–320, 2010.

BIRGIN, E. G. y LOBATO, R. D. "Orthogonal packing of identical rectangles within isotropic convex regions". *Computers & Industrial Engineering*. DOI: 10.1016/j.cie.2010.07.004.

BIRGIN, E. G.; LOBATO, R. D. y MORABITO, R.. "Generating unconstrained two-dimensional non-guillotine cutting patterns by a recursive partitioning algorithm". *Journal of the Operational Research Society 63*, pp. 183-200, 2012.

BISCHOFF, E. E. y MARRIOTT, M.D., "A comparative evaluation of heuristics for container loading". *European Journal of Operational Research*, vol. 44, pp. 267–276, 1990.

BORTFELDT, A. y GEHRING, H. "A hybrid genetic algorithm for the container loading problem". *European Journal of Operational Research*, vol. 131, pp. 143–161, 2001.

BORTFELDT, A. y WINTER, T. "A genetic algorithm for the two-dimensional knapsack problem with rectangular pieces". *International Transactions in Operational Research*, 16:685–713, 2009.

BOSCHETTI M. A.; MINGOZZI A. y HADJICONSTANTINOU, E. "New upper bounds for the twodimensional orthogonal non-guillotine cutting stock problem". *IMA Journal of Management Mathematics*, 13:95–119, 2002.

CAPRARA, A. y MONACI, M. "On the two-dimensional knapsack problem". *Operations Research Letters*, vol. 32, pp. 5–14, 2004.

CARLISLE, A. y DOZIER, G. Adapting particle swarm optimization to dynamic environments. *in Proc. Int. Conf. Artif. Intell*., Las Vegas, NV, pp. 429–434, 2000.

CHEN, C. S.; LEE, S. M., y SHEN Q. S. "An analytical model for the container loading problem". *European Journal of Operational Research*, 80:68–76, 1995.

CHEN, P.; FU, Z.; LIM, A. y RODRIGUES B. "Two-Dimesional Packing for irregular shaped objects". *36th Hawaii International Conference on System Sciencies*. Computer Society. IEEE. 2002.

CHEN, D. y HUANG, W. "A new heuristic algorithm for constrained rectangle-packing problem". *Asia-Pacific Journal of Operational Research*, 24:463–478, 2007.

CHAZELLE, B. "The bottom-left bin packing heuristic: An efficient implementation". *IEEE Transactions on Computers*, vol. 32, pp. 697–707, 1983.

CHRISTOFIDES, N. y WHITLOCK, C. “An algorithm for two-dimensional cutting problems”. *Operations Research*, vol. 25, pp. 31-44, 1977.

CHRISTOFIDES, N. “Optimal cutting of two-dimensional rectangular plates”, *CAD74 Proceedings*, pp. 1–10, 1974.

CUI, Y. An exact algorithm for generating homogenous T-shape cutting patterns. *Computers & Operations Research*, vol. 34, pp. 1107-1120, 2007.

DANIELS, K. MILENKOVIC, V. J. y LI, Z. “Multiple containement methods”. Technical Report 12-94, *Center for Research in Computing Technology*, Division of Applied Sciences, Harvard University, 1994.

DARWIN, C. “The Origin of Species”, *John Murray*, 1859.

DUECK, G. y SCHEUER, T. “Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing”, *J. Computat. Phys.*, vol. 90, pp. 161–175, 1990.

DYCKHOFF, H. “A typology of cutting and packing problems”. *European Journal of Operational Research*, 44:145–159, 1990.

EGEBLAD, J. y PISINGER, D. “Heuristic approaches for the two- and three-dimensional knapsack packing problem”. *Computers and Operations Research*, 36:1026–1049, 2009.

ESCOBAR, L. M.; ÁLVAREZ, D. y ROMERO, R. A. “Equipo asíncrono de agentes basados en recocido simulado aplicado al problema del agente viajero simétrico”. *Scientia et Technica* Año XVI, No 49, pp. 122-127, Diciembre de 2011.

FAYARD, D. y ZISSIMOPOULOS, V. “An approximation algorithm for solving unconstrained two-dimensional knapsack problems”. *European Journal of Operational Research*, vol. 84, pp. 618-632, 1995.

FEKETE, S. P. y SCHEPERS, J. “A new exact algorithm for general orthogonal d-dimensional knapsack problems”. In: Burkard, R., Woeginger, G.J. (Eds.), Algorithms-Proceedings of the 5th Annual European Symposium, Graz, Austria, Sept., 1997, *Lecture Notes in Computer Science*, vol. 1284. Springer, Berlin, pp. 144–156, 1997.

FEKETE, S. P. y SCHEPERS, J. “On more-dimensional packing III: Exact algorithms”. Technical Report ZPR97-290*, Mathematisches Institut*, UniversitYat zu KYoln, 1998.

FEKETE, S. P.; SCHEPERS, J. y VAN DER VEEN, J. C. “An exact algorithm for higher-dimensional orthogonal packing”. *Operations Research*, 55:569–587, 2007.

G, Y-G. y KANG, M-K. “A new upper bound for unconstrained two-dimensional cutting and packing”. *J. Oper. Res. Soc*., vol. 53, pp. 587–591, 2002.

G, Y-G. KANG, M-K. y SEONG, J. “A best-first branch and bound algorithm for unconstrained two-dimensional cutting problems”. *Operations Research Letters*, vol. 31, pp. 301–307, 2003.

GALLEGO, R.; ESCOBAR, A. H. y ROMERO, R. A. *Programación lineal entera*, Textos universitarios, Universidad Tecnológica de Pereira, 2007.

GALLEGO, R.; ESCOBAR, A. H. y TORO, E. M. *Técnicas metaheurísticas de optimización*, Textos universitarios, Universidad Tecnológica de Pereira, 2008.

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

GILMORE, P. C. y GOMORY, R. E. “A linear programming approach to the cutting-stock problem”. *Operations Research*, vol. 9, pp. 849–859, 1961.

GILMORE, P. C. y GOMORY, R. E. “Multistage cutting stock problems of two and more dimensions”. *Operations Research*, vol. 13, pp. 94–120, 1965.

GILMORE, P. C. y GOMORY, R. E., “The theory and computation of knapsack functions”. *Operations Research*, vol. 14, pp. 1045-1074, 1966.

GONÇALVES, J. F. “A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problem”. *European Journal of Operational Research*, 183:1212–1229, 2007.

HADJICONSTANTINOU, E. y CHRISTOFIDES, N. “An exact algorithm for general, orthogonal, 2-dimensional knapsack-problems”. *European Journal of Operational Research*, 83:39–56, 1995.

HEALY, P. y MOLL, R. “A local optimization-based solution to the rectangle layout problem”. *Journal of the Operational Research Society*, vol. 47, pp. 523–537, 1996.

HERZ, J. C. “A recursive computing procedure for two-dimensional stock cutting”. *IBM Journal of Research and Development*, vol. 16, pp. 462-469, 1972.

HIFI, M. “Exact algorithms for large-scale unconstrained two and three staged cutting problems”. *Computational Optimization and Applications*, vol. 18, pp. 63–88, 2001.

HIFI, M. *Problem instances for the 2D Cutting/Packing Problems*, [web en línea], 1997. <ftp://cermsem.univ-paris1.fr/pub/CERMSEM/hifi/2Dcutting/> [Consulta: 01-02-2010].

HIFI, M. y ZISSIMOPOULOS, V. “A recursive exact algorithm for weighted two-dimensional cutting”. *European J. Oper. Res.*, vol. 91, pp. 553–564, 1996.

HUANG, W. y CHEN, D. “An efficient heuristic algorithm for rectangle-packing problem”. *Simulation Modelling Practice and Theory*, 15:1356–1365, 2007.

JAKOBS, S. “Theory and methodology on genetic algorithms for the packing of polygons”. *European Journal of Operational Research*, vol. 88, pp. 87-100, 1996.

KANTOROVICH, L. V. “Mathematical methods of organizing and planning production”. *Management Science*, vol. 6, pp. 363-422, 1960.

KENNEDY, J. y EBERHART, R. C. “Particle Swarm Optimization”. *In Proceedings of the 1995 IEEE International Conference on Neural Networks*, pp. 1942–1948, Piscataway, New Jersey, IEEE Service Center, 1995.

KIRKPATRICK, S. GELLATT, C. y VECCHI, M. “Optimization by Simulated Annealing”. *Science*, vol. 220, pp. 671–680, 1983.

KRÖGER B., “Guillotineable bin packing: A genetic approach”, *European Journal of Operational Research*, Vol. 84, pp. 645-661, 1995.

LIANG, J. J. y SUGANTHAN, P. N. Dynamic multi-swarm particle swarm optimizer with local search. *in Proc. IEEE Congr. Evol. Comput*., pp. 522–528, 2005.

LINS, L.; LINS, S. y MORABITO, R. “An L-approach for packing (l,w)-rectangles into rectangular and L-shaped pieces”. *Journal of the Operational Research Society*, 54:777–789, 2003.

LODI, A. MARTELLO, S. y MONACI, M. “Two-dimensional packing problems: A Survey”. *European Journal of Operational Research*, vol. 141, pp. 241–252, 2002.

LODI, A. MARTELLO, S. y VIGO, D. “Models and bounds for the two-dimensional level packing problems”. *Journal of Combinatorial Optimization*, vol. 8, pp. 363–379, 2004.

MARTELLO, S. MONACI, M. y VIGO, D. “An exact approach to the strip-packing problem,” *INFORMS Journal on Computing*, vol. 15, pp. 310–319, 2003.

MARTELLO, S. PISINGER, D. y TOTH, P. “New trends in exact algorithms for the 0-1 knapsack problem”, *European Journal of Operational Research*, vol. 123, pp. 325–332, 2000.

MARTELLO, S. y TOTH, P. *Knapsack Problems - Algorithms and Computer Implementations*. John Wiley & Sons, Chichester, 1990.

MASCARENHAS, W. F. y BIRGIN, E. G. “Using sentinels to detect intersections of convex and nonconvex polygons”. *Computational & Applied Mathematics*, 29:247–267, 2010.

METROPOLIS, N., ROSENBLUTH, A., ROSENBLUTH, M., TELLER, A. y TELLER, E. “Equation of state calculations by fast computing machines”. *Journal of Chemical Physics*, vol. 21, pp. 1087–1092, 1953.

M'HALLAH, R.; ALKANDARI, A y MLADENOVIĆ, N. “Packing unit spheres into the smallest sphere using VNS and NLP”. *Computers & Operations Research*, Volume 40, Issue 2, Pages 603-615, February 2013.

MLADENOVIĆ, N. “A variable neighborhood algorithm - a new metaheuristic for combinatorial optimization”. *In Abstracts of Papers Presented at Optimization Days*, page 112, 1995.

MLADENOVIĆ, N. y HANSEN, P. “Variable neighborhood search”, *Computers and Operations Research*, Vol. 24, pp. 1097–1100, 1997.

MLADENOVIĆ, N. y HANSEN, P. “Industrial applications of the variable neighborhood search metaheuristics”, *Decisions and Control in Management Science*, pp. 261–274, 2001.

MLADENOVIĆ, N. y HANSEN, P. “Variable neighborhood search: Principles and applications”, *European Journal of Operational Research*, Vol. 130, pp. 449–467, 2001.

MORABITO, R. y ARENALES, M. “An graph approach to the solution of two dimensional non-guillotine cutting problems”. *European Journal of Operational Research*, vol. 84, pp. 599-617, 1995.

MORABITO, R. ARENALES, M. y ARCARO, V. “An and-or-graph approach for two-dimensional cutting problems”. *European Journal of Operational Research*, vol. 58, pp. 263-271, 1992.

MORABITO, R. y MORALES, A. “A simple and effective recursive procedure for manufacturer’s ballet loading problem”. *Journal of the Operational Research Society*, vol. 49, pp. 819-828, 1998.

MORABITO, R. y MORALES, S. Errata “A simple and effective recursive procedure for the manufacturer’s pallet loading problem”. *Journal of the Operational Research Society*, 50:876–876, 1999.

ONODERA, H. TANIGUCHI, Y. y TAMARU, K. “Branch-and-bound placement for building block layout”. *in Proceedings of the 28th ACM/IEEE Design Automation Conference*, pp. 433–439, 1991.

PARREÑO, F.; ALVAREZ-VALDES, R.; TAMARIT, J. M. y OLIVEIRA, J. F. "A maximal-space algorithm for the container loading problem". *INFORMS Journal on Computing archive*, Volume 20 Issue 3, Páginas 412-422, 2008.

PEPPER, J.; GOLDEN, B. y WASIL, E. Solving the Traveling Salesman Problem with Annealing-Based Heuristics: A Computational Study. *IEEE Trans. Syst., Man, Cybern. A*, vol. 32, pp. 72-77, 2002.

PISINGER, D. "Heuristics for the container loading problem". *European Journal of Operational Research*, vol. 141, pp. 382–392, 2002.

REYNOLDS, R. G. "An Introduction to Cultural Algorithms". In A. V. Sebald and L. J. Fogel, editors, *Proceedings of the Third Annual Conference on Evolutionary Programming*, pp. 131–139. World Scientific, River Edge, New Jersey, USA, 1994.

SCHEITHAUER, G. "LP-based bounds for the container and multi-container loading problem". *International Transactions in Operational Research*, vol. 6, pp. 199–213, 1999.

SCHEITHAUER, G. y TERNO, J. "Modeling of packing problems". *Optimization*, vol. 28, pp.63-84, 1993.

SWEENEY, P. y PATERNOSTER, R. "Cutting and Packing Problems: A categorized, application-orientated research bibliography". *Journal of the Operational Research Society*, vol. 43, pp. 691-706, 1992.

TORO, E.; GARCÉS, A. y RUIZ, H. "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*, vol. 46, pp. 119-131, 2008.

TORO, E. "Solución del problema de empaquetamiento óptimo bidimensional". *Tesis de Maestría Universidad Tecnológica de Pereira*, 2007.

TSAI, R. D.; MALSTROM, E.E. y KUO, W. "3-dimensional palletization of mixed box sizes". *IIE Transactions*, 25:64–75, 1993.

HEMMELMAYR, V.; SCHMID, V. y BLUM, C. "Variable neighbourhood search for the variable sized bin packing problem Original Research Article". *Computers & Operations Research*, Volume 39, Issue 5, Pages 1097-1108, May 2012.

WÄSCHER, G. HAUßNER H. y SCHUMANN H. "An improved typology of cutting and packing problems". *European Journal of Operational Research*, vol. 183, pp. 1109–1130, 2007.

WEI, L.; ZHANG, D. y CHEN, Q. “A least wasted first heuristic algorithm for the rectangular packing problem”. *Computers and Operations Research*, 36:1608–1614, 2009.

WONG, D. F.; LEONG, H. W. y LIU, C. L. *Simulated Annealing for VLSI Design*. Kluwer Academic Publishers, 1988.

ZHI-HUI, Z.; JUN, Z.; YUN, L. y Henry, S. Adaptive Particle Swarm Optimization. *IEEE Transactions On Systems, Man, And Cybernetics—Part B: Cybernetics*, vol. 39, Issue 6, pp. 1362-1381, 2009.