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

DAVID ÁLVAREZ MARTÍNEZ

PROPUESTA DE PROYECTO DE GRADO
PARA OPTAR AL TITULO DE MAGÍSTER EN INGENIERÍA ELÉCTRICA

DIRECTOR
PH.D. RAMÓN ALFONSO GALLEGO

MAESTRÍA EN INGENIERÍA ELÉCTRICA
FACULTAD DE INGENIERIAS: ELÉCTRICA, ELECTRÓNICA FÍSICA Y CIENCIAS COMPUTACIONALES
UNIVERSIDAD TECNOLÓGICA DE PEREIRA
PEREIRA, 26 DE ABRIL DE 2010

Resumen

Los problemas de empaquetamiento y corte óptimo son considerados clásicos dentro de la investigación de operaciones debido a su gran espectro de aplicación en la industria y su alta complejidad tanto matemática como computacional.

En este trabajo se presenta una discusión del estado del arte de los problemas de empaquetamiento óptimo bidimensional de piezas rectangulares en una sola placa, en rollos y placas infinitas, considerando la posibilidad de rotar 90° las piezas y con restricciones de corte tipo guillotina. Se describe el modelo matemático aplicado por los grupos de investigación en estos problemas. Se propone un tipo de codificación para ser aplicada en este problema y resolverla mediante técnicas aproximadas como lo son las heurísticas y metaheurísticas. Se implementa un conjunto de metodologías basadas en técnicas metaheurísticas bien conocidas como: recocido simulado, búsqueda en vecindario variable y optimización con cúmulo de partículas, las cuales van de la mano con heurísticas del problema, aplicando nuevos métodos de solución eficientes en cuanto a tiempo y calidad de las respuestas.
Para comprobar la eficiencia de las metodologías presentadas se tomaron casos de prueba de la literatura especializada, donde se analizan y comparan los métodos de solución presentados con los del estado del arte de los problemas, obteniéndose resultados de excelente calidad y nunca antes reportados en la literatura.

Abstract

The cutting and packing problems are considered classics problems in the operations research, due to its big spectrum of application in the industry and its highly mathematical and computational complexity for the academy.
In this thesis we show the state-of-the-art of two-dimensional cutting stock, strip and bin packing problems of rectangular items, with and without items rotations of 90° and with guillotine cuts. We describe for these problems the mathematical model recognized by the academic community. We develop an appropriate encoding of the problem to work on it by approximated techniques, such as heuristics and metaheuristics. We implement a set of methodologies based on metaheuristics techniques well known as: simulated annealing, variable neighborhood search and particles swarm optimization, these go along with heuristics of the problem, generating new and efficient methods of solution in relation to time and quality of the responses.
To check the efficiency of the presented methodologies, case of studies were taken from specialized literature, where it could be analyzed and compared the presented solution methods with the state-of-the-art of the problems, we obtained results of excellent quality and never reported in the literature.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1. Definición del problema

Dentro de la familia de los problemas de corte y empaquetamiento bidimensional está incluido el de empaquetamiento óptimo en placas (2D-BPP) del inglés *two-dimensional bin packing problem*, el de corte óptimo en una sola placa (2D-CSP) del inglés *two-dimensional cutting stock problem* y el de empaquetamiento óptimo en rollos (2D-SPP) del inglés *two-dimensional strip packing problem*. El primero consiste en un conjunto de rectángulos que deben ser ubicados en áreas rectangulares llamadas placas con anchos y largos definidos e idénticos, el objetivo es encontrar el número mínimo de placas necesarias de forma tal que la totalidad de las piezas sean ubicadas, sin sobreponer las piezas y sin sobrepasar los límites de las placas, el segundo consiste en un conjunto de rectángulos que deben ser cortados en un área rectangular llamada placa con un ancho y largo definido, el objetivo es encontrar la ubicación de las piezas en la placa para ser cortadas, minimizando el espacio desperdiciado de la placa, sin sobreponer las piezas y sin sobrepasar los límites de la placa. El tercer problema consiste en un conjunto de rectángulos que deben ser ubicados en un área rectangular llamada rollo con un ancho definido y un largo que se supone infinito, el objetivo es encontrar el largor mínimo del rollo de forma tal que la totalidad de las piezas sean ubicadas en el rollo, sin sobreponer las piezas y sin sobrepasar los límites del rollo.

Se ha demostrado que estos problemas son NP-Duros debido al tamaño de su espacio de solución [1]. Existen cuatro variantes de los mismos dependiendo de la posibilidad de rotar o no las piezas y del tipo de corte generado guillotina o no guillotina; esta problemática es de amplia aplicación en el sector textil, metalmecánico, papelero y en general en todos los procesos productivos donde la materia prima debe ser cortada o empacada en placas o rollos.

1.2. Motivación

Los problemas de corte y empaquetamiento son de interés tanto académico como industrial, avanzar en la temática de corte y empaquetamiento óptimo bidimensional guillotinado en empresas de corte, despacho y almacenamiento de materiales representa una estrategia de mejoramiento importante ya que estas pueden maximizar el uso de las materias primas, factor que incide directamente en el costo final del producto.

Este estudio esta basado en una revisión del estado del arte de los problemas clásicos de corte y empaquetamiento que puede ser un referente para la comunidad académica y presenta una nueva metodología para la solución de los problemas de *bin packing*, *cutting stock* y *strip* packing guillotinados, con base en esta pueden ser abordadas muchas otras variantes para este tipo de problemas.

En la última década 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 de los problemas planteados 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 para los problemas de corte y empaquetamiento óptimo bidimensional de piezas rectangulares, con cortes tipo guillotina, con y sin rotación de piezas, en placas y rollos infinitos usando técnicas metaheurísticas de optimización.

ESPECÍFICOS

- Revisar el estado del arte de los modelos matemáticos que representan el problema de empaquetamiento óptimo bidimensional tipo guillotina con y sin rotación de piezas, tanto en placas como en rollos infinitos.
- 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 de los problemas de empaquetamiento óptimo bidimensional de piezas rectangulares tanto en placas como en rollos infinitos.
- Validar las metodologías propuestas con casos de prueba de la literatura especializada.
- Implementar un aplicativo informático en versión Beta que entregue en tiempos computacionales razonables respuestas de buena calidad para los problemas propuestos en esta tesis.

1.4. Alcance y contribuciones

Las principales contribuciones de esta tesis son las siguientes:

- Se desarrollaron diferentes metodologías para la solución de los problemas de empaquetamiento óptimo bidimensional tipo guillotina con y sin rotación de piezas, tanto en placas como en rollos infinitos.
- Los resultados obtenidos por el conjunto de metodologías presentadas en esta tesis superaron a muchos de los que se encuentran propuestos en la literatura especializada.

- Todas las implementaciones realizadas en este trabajo se compararon contra técnicas representativas del estado del arte de optimización con técnicas exactas, heurísticas y metaheurísticas, concluyendo que las metodologías desarrolladas en esta tesis presentan resultados de igual o mejor calidad respecto a los conocidos.

- Se efectuó un análisis de varianza de las soluciones obtenidas con cada una de las técnicas propuestas a fin de conocer la sensibilidad de los algoritmos y la calidad de sus respuestas, permitiendo clasificar el conjunto de técnicas propuestas de acuerdo a su desempeño.

- Se formulo un software versión Beta que resuelve los problemas estudiados, a fin de que la comunidad académica y empresarial lo puedan aplicar en sus desarrollos.

1.5. Estructura general del documento

La organización de este documento es la siguiente:

En el capitulo 2 se explican los conceptos generales de los problemas de empaquetamiento óptimo bidimensional, así como su estado del arte.

En el capitulo 3 se presentan y se explican los modelos matemáticos de los problema de empaquetamiento óptimo bidimensional de piezas rectangulares acogidos por la comunidad académica.

En el capitulo 4 se realiza una descripción de las técnicas de optimización combinatoria que fueron aplicadas en la solución del modelo.

En el capitulo 5 se presenta una adaptación de las técnicas de optimización combinatoria a la solución del problema.

El capitulo 6 esta dedicado a un estudio y un análisis formal de los resultados obtenidos por las diferentes metodologías presentadas.

En el capitulo 7 se presentan las conclusiones y recomendaciones.

2. ANTECEDENTES DEL PROBLEMA

2.1. Introducción

Este capítulo está dedicado a proporcionar los conceptos básicos de los problemas de empaquetamiento óptimo a fin de establecer aspectos relevantes en la definición de los problemas de interés *bin packing, cutting stock y strip packing*.

Los problemas de corte y empaquetamiento fueron planteados y han sido estudiados ampliamente. En 1961 Gilmore y Gomory en [1] presentaron la primera aproximación para problemas reales, cabe recordar que Kantorovich en 1939 y Brooks en 1940 ya habían realizado algunos estudios de interés en el tema, pero tal vez el artículo de Gilmore y Gomory es una de las publicaciones con mayor relevancia dentro de esta área, a partir de éste cada año aparecen una gran cantidad de publicaciones con diferentes estudios con las posibles variaciones del problema, dado a el gran número de variantes del problema sea dificultado obtener información para un problema en particular.

Sweeney y Paternoster presentan en [2] una recopilación de más de 400 libros, artículos y demás documentación publicada entre 1940 y 1990, en los que se estudian los problemas de corte y empaquetamiento. La recopilación de la información es clasificada según la dimensión del problema, tipo del problema y tipo de aproximación a la solución empleada.

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 Johnson en [3] y S. Martello et al. en [4].

2.2. Generalidades de los problemas de corte y empaquetamiento

En [5], Dyckhoff propone una tipología para los problemas de corte y empaquetamiento basada en identificadores para las características básicas comunes de estos problemas. Para comprender su tipología Dyckhoff explica someramente en que consiste un problema de corte o empaquetamiento de la siguiente manera: existen dos elementos principales, un conjunto de ítems grandes y una lista de ítems pequeños, los ítems grandes también son llamados objetos (en este documento serán llamados placas). Siendo el objetivo ubicar los ítems en el conjunto de objetos. Los identificadores de las características básicas de los problemas de corte y empaquetamiento son enumerados a continuación:

a. Dimensionalidad (la característica más importante, define el número mínimo de dimensiones necesarias para describir la geometría de la disposición de los ítems en los objetos).

(1) Unidimensional

(2) Bidimensional
(3) Tridimensional
(N) Multidimensional con N>3

b. Clase de asignación (disponibilidad tanto de los ítems como de los objetos)
(B) Todos los objetos y una selección de ítems
(V) Una selección de objetos y todos los ítems

c. Surtido de los objetos (número de diferentes formas de los objetos)
(O) Un objeto
(I) Figuras idénticas
(D) Diferentes figuras

d. Surtido de los ítems (número de diferentes formas de los objetos)
(F) Pocos ítems (de diferentes figuras)
(M) Muchos ítems de muchas diferentes figuras
(R) Muchos ítems de relativamente pocas diferentes figuras (no-congruentes)
(C) Figuras congruentes

Además de estas características existen otras no tan generales que dependen particularmente del problema, a continuación se detalla una clasificación presentada por Lodi et al. en [6] (se modifican los identificadores presentados por Lodi et al. ya que algunos ya fueron usados en las anteriores características). Algunos autores [7, 8 y 9] diferencian dos tipos de problemas dependiendo del valor de las piezas: el problema sin pesos (*un-weighted*) donde el valor de cada ítem es el área de la misma y el problema con pesos (*weighted*) donde el valor de la pieza no tiene porque depender del área de la misma, por lo que pueden existir piezas pequeñas con valores superiores a otras más grandes.

e. Restricciones inherentes al patrón de corte (tipos de corte, separación entre los cortes)
(G) Exclusivamente cortes tipo guillotina
(U) No requiere cortes tipo no guillotina

f. Restricciones inherentes a la orientación de las piezas (posibilidad de que las piezas puedan rotar 90° o no)
(T) Las piezas pueden rotar 90°
(A) Las piezas no pueden rotar 90°

g. Valores de las piezas (beneficio que ofrece empacar una determinada pieza).
(W) Ítems con valores de beneficio diferente a su área (*weighted*)
(Z) Ítems con valores de beneficio igual a su área (*un-weighted*)

h. Demanda de las piezas
(E) Piezas con límite máximo de corte
(I) Piezas sin límite de corte (infinito)

i. Forma de las piezas

(L) Ítems con forma regular (rectángulos, círculos, cubos, esferas, cilindros, etc.)
(K) Ítems con forma irregular

La unión de estas dos clasificaciones genera una nomenclatura lo suficientemente abreviada y rica en información, para dejar explicito cuales son los problemas estudiados en esta tesis. Se define un problema por la nueve-tupla de identificadores de las características *(a, b, c, d, e, f, g, h, i)*, por ejemplo definir el problema de *cutting stock* a través de los identificadores de sus características bastaría con la nueve-tupla (2, V, I, *, U, A, Z, E, I) la cual describe el problema clásico de empaquetamiento bidimensional en objetos idénticos, teniendo en cuenta todos los posibles surtidos de ítems posibles, con restricciones tipo guillotina en los patrones de corte y donde no se permite rotar. La ausencia de un identificador de una característica representada por el símbolo (*) expresa que todas las opciones de esa característica son tomadas en cuenta. En este estudio todos los surtidos de piezas posibles son tomados en cuenta dado que las librerías de casos de prueba contienen los diferentes tipos de surtidos de piezas.

Una pregunta muy importante es si la forma de las figuras es regular o irregular, para este estudio solo se tiene en cuenta problemas donde la forma de las figuras es regular y en especial rectangular, ya que la mayor cantidad de las investigaciones en esta temática tratan los problemas de empaquetamiento de figuras rectangulares. Además de esto, existen diferentes objetivos en los problemas de empaquetamiento algunos de estos son con respecto al valor de las figuras, como la minimización de las perdidas de material, otros objetivos con relación a los precios de las figuras, como maximizar el costo del material embalado, mientras otros objetivos dependen del proceso de corte, por ejemplo en algunas aplicaciones el material tiene un costo muy inferior en comparación al costo del proceso que se realiza para el corte de este y se tiene como objetivo minimizar el número de cortes a realizar. En este trabajo se considera como único objetivo la minimización de pérdidas de material.

Una característica muy importante en las aplicaciones reales de los problemas de corte y empaquetamiento es la tecnología usada para realizar los cortes o embalaje del material. Esta puede reducir o aumentar la complejidad del problema, por ejemplo en el corte de goma crepé es usada una guillotina como tecnología de corte para generar los ítems demandados (ver figura 1), otro ejemplo conocido es el embalaje de cajas en un contenedor mediante el uso de paletas (ver figura 2). En ambos ejemplos aunque la dimensionalidad del problema es de grado 3, se trata como un problema de 2+1 dimensiones [1].

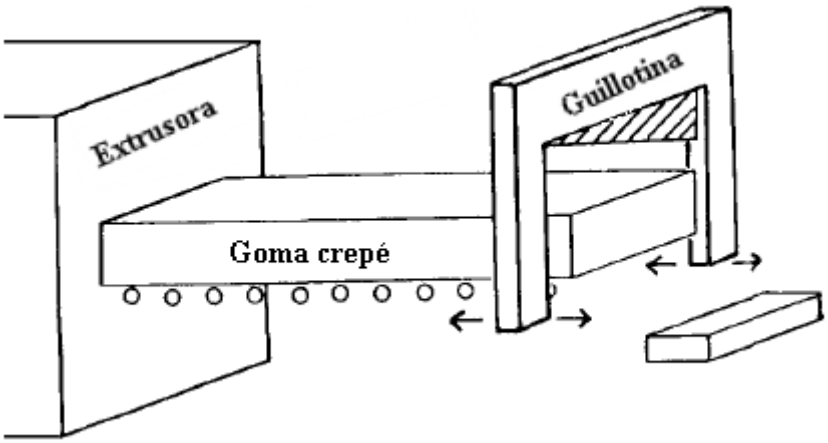

Figura 1. Uso de la guillotina para el corte de goma crepé.

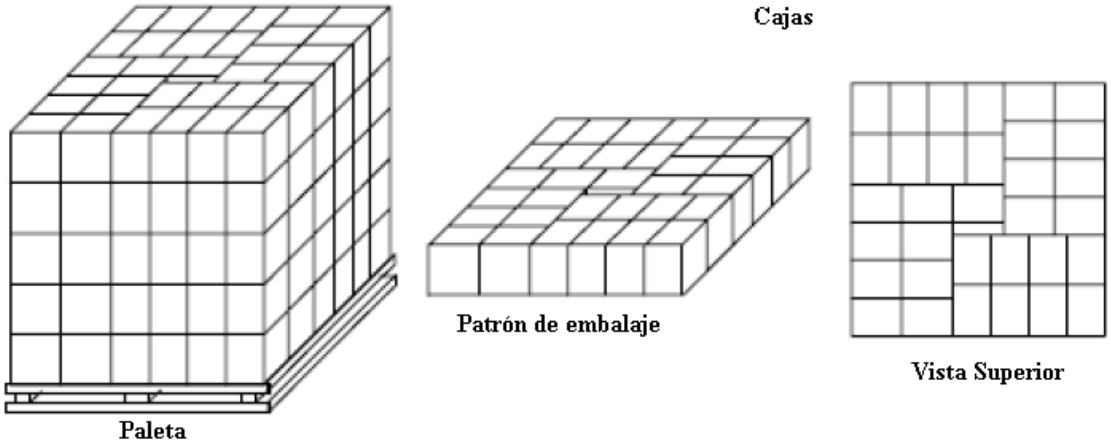

Figura 2. Uso de paletas para el embalaje de cajas.

A continuación se definen patrones de corte guillotina y no guillotina. Un corte tipo guillotina es aquel que al aplicarse sobre un rectángulo produce dos nuevos rectángulos, es decir, si el corte va de un extremo al otro del rectángulo original; en otro caso se denomina del tipo no guillotina. Un patrón es de tipo guillotina si se puede obtener por sucesivos cortes de tipo guillotina (ver figura 3). Un patrón es no guillotina si es obtenido por sucesivos cortes de guillotina y no guillotina (ver figura 4).

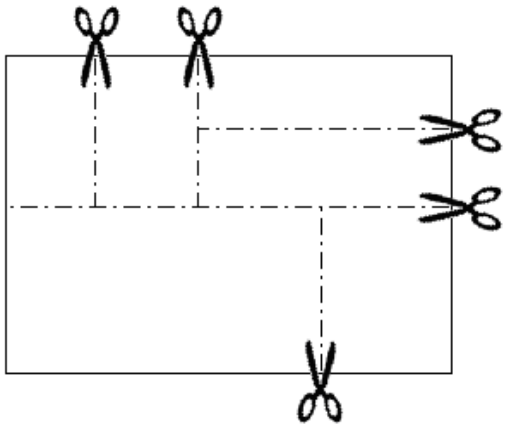

Figura 3. Patrón guillotina.

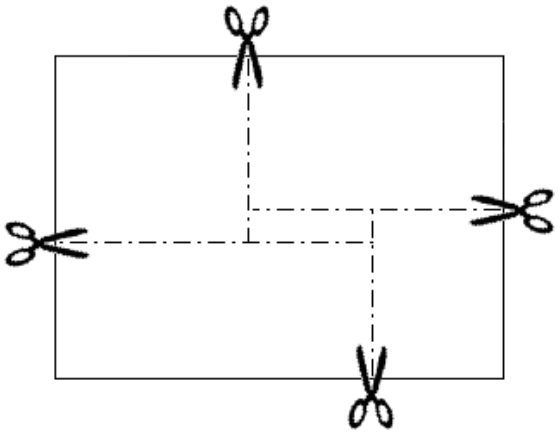

Figura 4. Patrón no guillotina.

2.3. Empaquetamiento óptimo bidimensional de piezas rectangulares en una sola placa (*cutting stock problem*)

Con las características expresadas anteriormente los identificadores que describen el problema de empaquetamiento óptimo bidimensional de piezas rectangulares en una sola placa trabajado en este estudio son *(2, V, O, *, G, *, Z, E, L)*. Consiste en el corte de una selección de ítems sobre un sola placa de dos dimensiones, se puede tener cualquier tipo de surtido de ítems, las piezas deben obtenerse como resultado de aplicar sucesivos cortes tipo guillotina y por último se consideran dos instancias del problema: las piezas pueden rotar 90° y las piezas tiene orientación (no pueden rotar).

La primera instancia del problema *(2, V, O, *, G, T, Z, E, L)* se define como: Cortar de un rectángulo que se denomina placa (objeto) de longitud *L* y anchor *W*, un conjunto de rectángulos de cardinalidad *n* que se denominan piezas (ítems), de longitud l_i y anchor w_i (donde $i = 1,..., n$). Una pieza *(l, w)* es equivalente a una pieza *(w, l)*.

Donde el objetivo es dado por la ecuación 1, maximizar el área del conjunto de piezas cortadas de la placa, z_i es una variable binaria que indica si la pieza *i* fue o no cortada.

Ecuación 1. $max \quad \sum_{i=1}^{n} l_i \cdot w_i \cdot z_i$

Sujeto a:

- Las piezas empacadas no deben superar los límites de la placa.
- Las piezas no deben sobreponerse entre ellas mismas.
- La generación de las piezas debe ser la consecución de realizar un corte de extremo a extremo sobre la placa de material o subdivisiones de esta.

La segunda instancia del problema *(2, V, O, -, G, A, Z, E, L)* solo difiere en la definición anterior por la condición de orientación de la piezas. Por lo tanto, una pieza *(l, w)* no es equivalente a una pieza *(w, l)*.

Los dos problemas definidos anteriormente han sido ampliamente trabajados a lo largo de diferentes campos de la optimización, como lo son la optimización exacta y la aproximada (heurísticas y metaheurísticas), a continuación se describe algunos de los trabajos más importantes en ambos campos.

Existen algunos métodos exactos propuestos para el problema: Beasley en [10], Tsai et al. en [11] y Scheithauer y Terno en [12] presentan formulaciones enteras. Hadjiconstantinou y Christofides en [13] también proponen una formulación mediante una relajación del problema y el uso de la programación dinámica. Arenales y Morabito en [14] presentan un algoritmo de búsqueda en árbol (*branch and bound)* basado en un tipo especial de grafo (*AND/OR graph*). Fekete y Schepers en [15] crean un procedimiento basado en la representación mediante grafos de la posición relativa de las piezas en un patrón posible. Finalmente, Caprara y Monaci en [16] presentan un algoritmo con algunas mejoras respecto a la propuesta de Fekete y Schepers.

Dentro de las heurísticas constructivas, el procedimiento más empleado para colocar las piezas es el algoritmo *bottom-left* (BL) presentado por Chazelle en [17] el cual alcanza patrones no guillotina. Otro método utilizado en la literatura para construir una solución a partir de un ordenamiento de las piezas, es el mecanismo *difference process* (DP) de Lai y Chan [18], al igual que el *bottom-left* este genera patrones no guillotina. Lodi et al. en [19] presentan una gran variedad de métodos constructivos (*Next-Fit Decreasing, First-Fit Decreasing, Best-Fit Decreasing, Hybrid First-Fit* entre otros) para los problemas de empaquetamiento bidimensional tanto en placas como en rollos, con patrones de corte tipo guillotina.

Beasley en [20] presenta un algoritmo metaheurístico para el problema de empaquetamiento óptimo bidimensional irrestricto guillotinado, métodos presentados para el caso restricto son: Lai y Chan en [18] presentan un procedimiento basado en recocido simulado, Leung en [21] utilizan un algoritmo evolutivo (algoritmo genético) y además presenta un recocido simulado, basado en la codificación de Lai y Chan [18], mostrando que existen patrones que no pueden alcanzarse por los procedimientos de BL y DP.

Beasley en [20] presenta un algoritmo genético para el caso general. El algoritmo está basado en una nueva formulación no lineal. La formulación también admite extensiones para el problema con más de una placa, el problema con algunas zonas de la placa que no se pueden utilizar y el problema donde las piezas se pueden rotar. Los resultados computacionales son los mejores hasta el momento, resuelve problemas con hasta 1000 tipos de piezas y recopila la mayoría de instancias de pruebas utilizadas para el problema. Además crea una nueva librería de problemas basados en los estudios de Fekete y Schepers.

Para el problema de empaquetamiento óptimo bidimensional guillotinado en placas con y sin rotación de piezas, Yaodong Cui en [22] presenta un algoritmo que genera patrones de corte homogéneos en forma-T, mientras para este mismo problema sin rotación Toro et al. en [23] presentan un algoritmo híbrido (búsqueda en vecindario variable y recocido simulado).

2.4. Empaquetamiento óptimo bidimensional de piezas rectangulares en placas (*bin packing problem*)

Con las características expresadas anteriormente, los identificadores que describen el problema de empaquetamiento óptimo bidimensional de piezas rectangulares en placas

trabajado en este estudio son *(2, B, I, *, G, *, Z, E, L)*. Consiste en el corte de la totalidad de ítems demandados sobre placas idénticas de dos dimensiones, donde se tiene cualquier surtido de ítems, las piezas deben obtenerse como el resultado de aplicar sucesivos cortes tipo guillotina y por último se consideran dos instancias del problema: las piezas pueden rotar 90° y donde las piezas tiene orientación (no pueden rotar).

La primera instancia del problema *(2, B, I, *, G, T, Z, E, L)* se define como: Cortar de un conjunto de rectángulos que se denominan placas (objetos), un conjunto de rectángulos de cardinalidad *n* que se denominan piezas (ítems), de longitud l_i y anchor w_i (donde $i = 1,..., n$). Una pieza *(l, w)* es equivalente a una pieza *(w, l)*.

Donde el objetivo es dado por la ecuación 1, minimizar *b,* el número de placas necesarias para embalar la totalidad de las piezas demandadas, z_{ik} es una variable binaria que indica si la pieza *i* fue o no cortada en la placa *k*.

Ecuación 1. $\min \quad \sum_{k=1}^{b} L \cdot W - \sum_{i=1}^{n} l_i \cdot w_i \cdot z_{ik}, \forall k$

Sujeto a:

- Las piezas empacadas no deben superar los límites de la placa.
- Las piezas no deben sobreponerse entre ellas mismas.
- La generación de las piezas debe ser la consecución de realizar un corte de extremo a extremo sobre la placa de material o subdivisiones de esta.

La segunda instancia del problema *(2, B, I, -, G, A, Z, E, L)* solo difiere en la definición anterior por la condición de orientación de la piezas, por lo tanto una pieza *(l, w)* no es equivalente a una pieza *(w, l)*.

Los dos problemas definidos anteriormente han sido ampliamente trabajados a lo largo de diferentes campos de la optimización, como lo son la optimización exacta y la aproximada (heurísticas y metaheurísticas), a continuación se describe algunos de los trabajos más importantes en ambos campos.

Lodi et al. [6, 9] proveen un resumen de avances recientes en esta temática, describen los límites disponibles y algoritmos exactos y aproximados. Uno de los mejores métodos exactos es propuesto por Dell Amico et al. [91], este introduce un limite inferior del problema con rotación y lo resuelve con un algoritmo *branch and bound*.

La mayoría de los algoritmos aproximados [6, 55, 86, 89 y 92] usan un método de dos etapas [86]. La primera etapa empaca todas las piezas dentro de un rollo de longitud fija y ancho infinito y la segunda etapa subdivide el rollo en sub-rollos de longitud finita y ancho finito y empaca las piezas en placas.

Berkey y Wang [86] proponen los algoritmos *finite best fit strip* (FBS) y *finite first fit* (FFF) par el problema sin rotación. El algoritmo FBS empaca las piezas de acuerdo a la regla el mejor encaje en rollo de forma decreciente, es decir, donde la pieza es embalada en el rollo actual si la pieza encaja en este, de lo contrario es empacada en un nuevo rollo, empacando los sub-rollos resultantes dentro de las placas con ancho suficiente o en una nueva placa si el sub-rollo no encaja en cualquiera de las placas existentes. El algoritmo FFF, empaca la pieza en el rollo inferior o en un nuevo rollo de la primera

placa en la cual pueda acomodarse, de lo contrario, en la esquina inferior izquierda de una nueva placa.

Lodi et al. [89] adaptan los algoritmos FBS y FFF para el problema con rotación y presenta una aproximación *floor ceiling* (FC). Las piezas se ordenan de forma decreciente según la arista más corta, ubicando una pieza horizontalmente cuando un nuevo rollo o una nueva placa son creadas, favoreciendo la ubicación vertical en un rollo existente si el empaquetamiento es factible. Son modificadas las heurísticas: mayor esquina inferior izquierda, inferior derecha y la mayor esquina superior derecha del rollo para considerar las restricciones guillotina y resuelven la segunda etapa usando un algoritmo exacto para *unidimensional bin packing problem* (1BP). Lodi et al. prueban que FC tiene un mejor desempeño que FBS y FFF.

Algunos algoritmos aproximados son basados en la adaptación de técnicas metaheurísticas. Por ejemplo, Lodi et al. [6] introducen un algoritmo de búsqueda tabú que explora el espacio de solución independiente del problema de empaquetamiento haciendo uso de un tamaño y estructura de vecindad dinámica. La búsqueda recombina, a través de una heurística constructiva, un subconjunto de piezas embaladas dentro un subconjunto de placas y una pieza actualmente embalada en una placa objetivo vacía.

La heurística constructiva para el problema sin rotación es un algoritmo de dos fases, la fase de empaquetamiento en rollos requiere la solución de una serie de problemas de la mochila (*knapsack problem*) y la fase de empaquetamiento en placas requiere la solución del problema 1BP.

Lodi et al. [6] adaptan la heurística constructiva para el problema con rotación introduciendo el concepto de pseudo-pieza y favorece la orientación vertical de la pieza cuando es posible.

Faero et al. [88] extiende la aproximación de Dowsland's [45] para el problema sin rotación. Usando una búsqueda local guiada donde el vecindario es explorado a través de intercambios de piezas. Durante el proceso de optimización el algoritmo asigna aleatoriamente las piezas embaladas de una placa a otra. Cuando la nueva solución no es factible, una nueva función objetivo refleja el total de pares de piezas sobrepuestas y se minimiza la penalidad debido a los patrones de infactibilidad. La búsqueda es detenida cuando la incumbente alcanza un valor propuesto o un periodo de tiempo fijado es alcanzado.

Nuevos algoritmos aproximados han sido diseñados por resolver otras variantes de estos problemas. Por ejemplo, Binkley y Hagiwara [87] describen para el problema no-guillotina la heurística de cuatro esquinas que es usada en conjunto con una algoritmo paralelo auto-adaptativo de recocido simulado y un algoritmo genético paralelo auto-adaptativo. Puchinger y Raidl [90] consideran un problema típico de manufactura de vidrio: *three-stage two-dimensional bin packing problem* donde el número de cortes guillotina no puede exceder de 3. Estos diseñaron dos modelos lineales enteros de tamaño polinomial y un algoritmo *branch-and-price* basado en una formulación de cubrimiento para el problema bidimensional.

2.5. Empaquetamiento óptimo bidimensional de piezas rectangulares en rollos (*strip packing problem*)

Las características del problema de empaquetamiento óptimo bidimensional de piezas rectangulares en rollos (*strip packing problem*) también se puede describir a través de los identificadores enunciados anteriormente, la nueve-tupla correspondiente sería *(2, V, O, *, G, *, Z, E, L)*. Consiste en el corte de todos los de ítems demandados sobre un solo rollo de dos dimensiones, donde se tiene cualquier surtido de ítems, las piezas deben obtenerse como el resultado de aplicar sucesivos cortes tipo guillotina y por último al igual que en el problema de empaquetamiento en placas se consideran dos instancias del problema: donde las piezas pueden rotar 90° y donde las piezas tienen orientación (no pueden rotar).

La primera instancia del problema corresponde a la nueve-tupla *(2, V, O, *, G, T, Z, E, L)* y se define como: Cortar de un rectángulo de ancho W y longitud infinita que se denomina rollo, un conjunto de rectángulos de cardinalidad n, que se denominan piezas (ítems), de longitud l_i y anchor w_i (donde $i = 1,..., n$). Una pieza de largo l y ancho w es equivalente a una pieza de longitud w y ancho l. Donde el objetivo es dado por la ecuación 2, minimizar la longitud del rollo necesario para encontrar la ubicación de todas las piezas demandadas.

Ecuación 2. $min \quad L$

Sujeto a:

- Todas las piezas deben ser empacadas.
- Las piezas deben estar dentro de los límites del rollo.
- Las piezas no deben sobreponerse entre ellas mismas.
- La generación de las piezas debe ser la consecución de realizar un corte de extremo a extremo sobre el rollo de material o subdivisiones de este.

La segunda instancia del problema corresponde a la nueve-tupla *(2, V, O, *, G, A, Z, E, L)* y solo difiere en la definición anterior por la condición de orientación de la piezas, por lo tanto una pieza de longitud l y ancho w no es equivalente a una pieza de largo w y ancho l.

La mayoría de los trabajos que consideran el problema de empaquetamiento en rollos son algoritmos aproximados. Fernandez de la Vega y Zissimopoulos en [24], Lesh et al. en [25], Kenyon and Rémila en [26] y Zhang et al. en [27] presentan algoritmos aproximados para el problema de *strip packing*. Bortfeldt en [28] y Beltran et al. en [29] usan metaheurísticas. Hopper y Turton en [30] proveen una descripción general de los algoritmos metaheurísticos aplicados al problema de empaquetamiento óptimo bidimensional en rollos.

Existen pocos trabajos que utilizan algoritmos exactos para resolver el problema de empaquetamiento bidimensional en rollos. Hifi en [31] introduce el problema de corte y empaquetamiento en rollos con cortes tipo guillotina y propone dos algoritmos basados en *branch and bound*. Martello et al. en [32] proponen un nuevo limite inferior y usa un *branch and bound* para resolver el problema de empaquetamiento en rollos restricto y sin corte tipo guillotina.

Recientemente, tres trabajos sobre el problema de empaquetamiento en rollos con corte tipo guillotina han sido propuestos. Cui et al. en [33] proponen un *branch and bound* recursivo para obtener una solución aproximada. Un nuevo limite inferior y un *branch and bound* son propuestos por Bekrar et al. en [34]. Finalmente, Cintra et al. en [35] usan un método de generación de columnas y programación dinámica.

2.6. Resumen

El conjunto de problemas propuestos hacen parte de problemas clásicos y de gran interés dentro del área de la investigación operativa, dada su gran aplicabilidad en la industria como el corte de metal, madera, vidrio, plástico, papel, pieles, entre otras. La complejidad de estos problemas ha llevado que sean atacados desde diferentes áreas de la optimización como lo son la optimización exacta y la optimización combinatoria, una lista larga de trabajos y contribuciones sobre esta temática ha sido expresada anteriormente.

En este trabajo se pretende abordar el problema desde el campo de optimización combinatoria haciendo uso de una valiosa herramienta como lo son las técnicas metaheurísticas de optimización. Este estudio propone el uso de tres técnicas metaheurísticas: recocido simulado, búsqueda en vecindario variable y optimización con cúmulos de partículas, para generar cuatro diferentes métodos de solución a través de la creación de algoritmos híbridos de optimización, para dar solución a:

- Dos instancias del problema de empaquetamiento óptimo bidimensional en una sola placa con patrones de corte tipo guillotina: con rotación *(2, V, O, -, G, -, Z, E, L)* y sin rotación de piezas *(2, V, O, -, G, A, Z, E, L).*
- Dos instancias del problema de empaquetamiento óptimo bidimensional en placas con patrones de corte tipo guillotina: con rotación *(2, B, I, -, G, T, Z, E, L)* y sin rotación de piezas *(2, B, I, -, G, A, Z, E, L).*
- Dos instancias del problema de empaquetamiento óptimo bidimensional en rollos infinitos con cortes tipo guillotina: con rotación *(2, V, O, -, G, T, Z, E, L)* y sin rotación de piezas *y (2, V, O, -, G, A, Z, E, L).*

3. MODELOS MATEMÁTICOS DE LOS PROBLEMAS

3.1. Introducción

Debido al gran número de trabajos realizados sobre los problemas de corte y empaquetamiento se han generado distintos modelos matemáticos para representar un mismo problema, esto con el fin de encontrar mejores soluciones a través de un modelamiento eficiente. Este estudio dedica un capitulo a los modelos matemáticos que representan los problemas de corte y empaquetamiento óptimo bidimensional tanto en placas como en rollos, con patrones de corte tipo guillotina, con y sin rotación de piezas.

Gilmore y Gomory en [1] y [35] presentan el primer modelo matemático para el problema de corte y empaquetamiento unidimensional. Gilmore y Gomory en [36] y [37] estudian más cuidadosamente el problema de la mochila cuando este es aplicado para el corte de piezas en una y dos dimensiones.

Biro y Boros en [38] caracterizan los patrones no guillotina usando teoría de grafos. Beasley en [10] estudió el problema de empaquetamiento en una sola placa sin restricciones de corte tipo guillotina y donde el objetivo es maximizar el valor de los ítems embalados. Beasley 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 [13]. Scheithauer y Terno [12] proponen modelos para empaquetamiento de polígonos convexos y no convexos.

Fekete y Schepers en [39] usan la teoría de grafos para determinar un empaquetamiento factible sin sobreponer las piezas. Es importante denotar que ellos no consideraron las restricciones de corte tipo guillotina. Lodi et al. en [19] y [40] 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 en [20] presenta una nueva formulación de corte no guillotina. Una buena revisión de los modelos existentes esta disponible por Lodi et al. en [19].

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. en [41], este modelo puede ser visto como una extensión a la técnica de modelamiento propuesta por Onodera et al. en [42]. 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. [41] muestran sin embargo que el modelo propuesto es muy ineficiente en la solución de instancias prácticas de empaquetamiento. Además, no existe una manera fácil de adaptar las restricciones tipo guillotina. La misma técnica fue usada por Daniels et al. en [43] 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.

Como se muestra en la revisión bibliográfica anterior no existen modelos matemáticos que describan completamente los problemas de interés en este estudio, afortunadamente, Ben et al. en [44] presentan un modelo basado en la caracterización y modelamiento de los patrones guillotina que se ajusta perfectamente al problema de empaquetamiento óptimo en rollos propuesto en este trabajo, el cual es explicado a continuación.

3.2. Modelo matemático del problema de empaquetamiento óptimo bidimensional de piezas rectangulares en rollos

Se presenta un modelo de programación lineal entera mixta para el problema de empaquetamiento óptimo en rollos basado en la caracterización de los patrones guillotina y el uso de las coordenadas donde pueden ser ubicadas las piezas.

Se asume que n piezas tienen que ser embaladas en un rollo con ancho W y una altura infinita (lo anterior se obtiene cambiando el sentido del rollo, altura por longitud, el problema original continua intacto).

Para un patrón de corte donde la esquina inferior izquierda de cada pieza k (donde $k = 1, 2, \ldots, n$) es ubicada en las coordenadas (α_k, β_k), se tienen dos series ordenadas $(x_1, x_2, \ldots, x_n)$ y $(y_1, y_2, \ldots, y_n)$ por un ordenamiento decreciente de las series $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ y $(\beta_1, \beta_2, \ldots, \beta_n)$ respectivamente. Como consecuencia, se obtiene $0 \leq x_1 \leq x_2 \leq \ldots \leq x_n \leq W$ y $0 \leq y_1 \leq y_2 \leq \ldots \leq y_n$.

Para obtener un modelo lineal entero, se usaran las siguientes variables binarias para caracterizar la ubicación de las piezas en el rollo:

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

Las siguientes variables de decisión intermedias también son necesarias:

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

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

El siguiente conjunto de tres variables binarias son necesarias para garantizar las restricciones guillotina:

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

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

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

La formulación completa del problema de empaquetamiento óptimo guillotinado en rollos es la siguiente:

Minimizar H, *(1)*

sujeto a $0 \le x_1 \le x_2 \le \ldots \le x_n$, *(2)*

$0 \le y_1 \le y_2 \le \ldots \le y_n$, *(3)*

$$\sum_{i=1}^{n}\sum_{j=1}^{n} z_{i,j,k} = 1 \quad \forall k, \tag{4}$$

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

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

$$x_{i'} - x_i - \sum_{k=1}^{n} w_k z_{i,j,k} \ge \left(u_{i,j,i'} - 1\right) W \quad \forall i, \forall j, \forall i' > i, \tag{7}$$

$$x_{i'} - x_i - \sum_{k=1}^{n} w_k z_{i,j,k} \ge u_{i,j,i'} W \quad \forall i, \forall j, \forall i' > i, \tag{8}$$

$$y_{j'} - y_j - \sum_{k=1}^{n} h_k z_{i,j,k} \ge \left(v_{i,j,i'} - 1\right) \overline{H} \quad \forall i, \forall j, \forall j' > j, \tag{9}$$

$$y_{j'} - y_j - \sum_{k=1}^{n} h_k z_{i,j,k} \ge u_{i,j,j'} \overline{H} \quad \forall i, \forall j, \forall j' > j, \tag{10}$$

$$\left(1 - d_{i_1,i_2,j_1,j_2}\right) n \ge \sum_{i=i_1}^{i_2}\sum_{j=j_1}^{j_2}\sum_{k=1}^{n} z_{i,j,k} - 1 \quad \forall i_1, \forall j_1, \forall i_2 > i_1, \forall j_2 > j_1, \tag{11}$$

$$p_{i_1,i',j_1,j_2} \le \sum_{j=j_1}^{j_2}\sum_{k=1}^{n} z_{i',j,k} \quad \forall i_1, \forall j_1, \forall j_2 > j_1, \forall i' > i_1, \tag{12}$$

$$p_{i_1,i',j_1,j_2} \le \sum_{i=i_1}^{i'-1}\sum_{j=j_1}^{j_2}\sum_{k=1}^{n} z_{i,j,k} \quad \forall i_1, \forall j_1, \forall j_2 > j_1, \forall i' > i_1, \tag{13}$$

$$\left(i' - i_1\right)\left(j_2 - j_1 + 1\right) p_{i_1,i',j_1,j_2} \le \sum_{i=i_1}^{i'-1}\sum_{j=j_1}^{j_2} u_{i,j,i'} \quad \forall i_1, \forall j_1, \forall j_2 > j_1, \forall i' > i_1, \tag{14}$$

$$q_{i_1,i_2,j_1,j'} \le \sum_{i=i_1}^{i_2}\sum_{k=1}^{n} z_{i,j',k} \quad \forall i_1, \forall j_1, \forall i_2 > i_1, \forall j' > j_1, \tag{15}$$

$$q_{i_1,i_2,j_1,j'} \le \sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j'-1} \sum_{k=1}^{n} z_{i,j,k} \quad \forall i_1, \forall j_1, \forall i_2 > i_1, \forall j' > j_1, \tag{16}$$

$$(j'-j_1)(i_2-i_1+1)\, q_{i_1,i_2,j_1,j'} \le \sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j'-1} v_{i,j,i'} \quad \forall i_1, \forall j_1, \forall j_2 > i_1, \forall j' > j_1, \tag{17}$$

$$d_{i_1,i_2,j_1,j_2} + \sum_{i'=i_1+1}^{i_2} p_{i_1,i',j_1,j_2} + \sum_{j'=j_1+1}^{j_2} q_{i_1,i_2,j_1,j'} \ge 1 \quad \forall i_1, \forall j_1, \forall i_2 > i_1, \forall j_2 > j_1, \tag{18}$$

$$W \ge x_i + \sum_{j=1}^{n} \sum_{k=1}^{n} w_k z_{i,j,k} \quad \forall i, \tag{19}$$

$$H \ge y_j + \sum_{i=1}^{n} \sum_{k=1}^{n} h_k z_{i,j,k} \quad \forall j, \tag{20}$$

$$z_{i,j,k} \in \{0,1\} \quad \forall i, \forall j, \forall k, \tag{21}$$

$$u_{i,j,i'} \in \{0,1\} \quad \forall i, \forall j, \forall i' > i, \tag{22}$$

$$v_{i,j,j'} \in \{0,1\} \quad \forall i, \forall j, \forall j' > j, \tag{23}$$

$$d_{i_1,i_2,j_1,j_2}, p_{i_1,i_2,j_1,j_2}, q_{i_1,i_2,j_1,j_2} \in \{0,1\} \quad \forall i, \forall j, \forall i_2 > i_1, \forall j_2 > j_1, \tag{24}$$

donde

$\overline{H} = \sum_{k=1}^{n} h_k$, (límite superior del valor de la función objetivo)

En la formulación anterior, las restricciones (4), (5) y (6) aseguran que cada posición horizontal o vertical sea ocupada por exactamente una pieza y cada pieza es empacada exactamente una vez.

Las restricciones (7) y (8) garantizan el cumplimiento $u_{i,j,i'}$. Si $x_{i'} > x_i + \sum_{k=1}^{n} w_k z_{i,j,k}$ (es decir si la pieza empacada en la posición *(i, j)* no excede $x_{i'}$), entonces $u_{i,j,i'}$ debe ser igual a *1*. La restricción (8) requiere que $u_{i,j,i'} = 1$ considerando la restricción (7) redundante (ya satisfecha). Por otro lado, si $x_{i'} < x_i + \sum_{k=1}^{n} w_k z_{i,j,k}$ (es decir, si una pieza empacada en la posición *(i, j)* y además esta pieza excede $x_{i'}$), entonces la restricción (7) requiere que $u_{i,j,i'} = 0$ considerando la restricción (8) redundante. Note que $u_{i,j,i'}$ puede tomar el valor *0* o el valor *1,* si $x_{i'} - x_i = \sum_{k=1}^{n} w_k z_{i,j,k}$. En tal caso, en las restricciones (14) y (18) $u_{i,j,i'}$ sea hace igual a *1*.

Por simetría, las restricciones (9) y (10) aseguran la coherencia de la definición de $v_{i,j,j'}$.

La restricción (11) garantiza la coherencia de la definición de d_{i_1,i_2,j_1,j_2}. En efecto, si $\sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j_2} \sum_{i=1}^{n} z_{i,j,k} \ge 2$ (al menos dos piezas son empacadas entre (i_1, j_1) y (i_2, j_2)), la restricción (11) requiere que $d_{i_1,i_2,j_1,j_2} = 0$. De otra forma, d_{i_1,i_2,j_1,j_2} puede ser indiferentemente igual a 0 o 1, En la restricción (18) d_{i_1,i_2,j_1,j_2} sea hace igual a *1*, lo cual es consistente con la definición.

Las restricciones [(12)-(18)] son restricciones guillotina para cada área rectangular. Si las restricciones guillotina son satisfechas para cada área rectangular, entonces también se cumple para todo el patrón de corte.

Las restricciones [(12)-(14)] aseguran la coherencia de la definición de p_{i_1,i_2,j_1,j_2}. En particular, si $\sum_{j=j_1}^{j_2}\sum_{i=1}^{n} z_{i',j,k} = 0$; es decir, ninguna pieza es empacada entre *(i', j_1)* y *(i'-1, j_2)*, la restricción (12) requiere $p_{i_1,i',j_1,j_2} = 0$. Similarmente, si $\sum_{i=i_1}^{i'-1}\sum_{j=j_1}^{j_2}\sum_{i=1}^{n} z_{i,j,k} = 0$; es decir, ninguna pieza es empacada entre *(i_1, j_1)* e *(i'-1, j_2)*, la restricción (12) requiere $p_{i_1,i',j_1,j_2} = 0$. En cualquier caso, $x_{i'}$ no es el límite inferior de un intervalo disjunto con piezas empacadas entre *(i_1, j_1)* y *(i'-1, j_2)*. Además, si $p_{i_1,i',j_1,j_2} = 1$ en la restricción (14) entonces $u_{i,j,i'} = 1$ para todo *i* tal que $i_1 \le i \le i'$ y para todo *j* tal que $j_1 \le i \le j_2$. Esto significa que ninguna pieza empacada entre *(i_1, j_1)* e *(i'-1, j_2)* excedería $x_{i'}$. La fusión de intervalos corresponde a la proyección de estas piezas en el eje-*x,* formando al menos dos intervalos disjuntos. En algunos casos, p_{i_1,i',j_1,j_2} puede ser indiferentemente igual a *0* o *1*. En tales casos, la restricción (18) hace que p_{i_1,i',j_1,j_2} sea igual a *1*, lo cual es consistente con la definición.

Por simetría, las restricciones [(15)-(17)] garantizan la coherencia de la definición de $q_{i_1,i_2,j_1,j'}$.

La restricción (18) requiere que al menos una de las siguientes condiciones se cumpla:

1. $d_{i_1,i_2,j_1,j_2} = 1$.
2. $p_{i_1,i',j_1,j_2} = 1$ para algún *i'* tal que $i_1 \le i' \le i_2$.
3. $q_{i_1,i_2,j_1,j'} = 1$ para algún *j'* tal que $j_1 \le j' \le j_2$.

Y finalmente las restricciones (19) y (20) garantizan que ninguna pieza exceda horizontalmente el ancho *W* y que ninguna pieza exceda verticalmente la altura *H,* considerando que la función objetivo es minimizar la altura total usada (ver ecuación (1)).

En este modelo, existen muchas restricciones y variables binarias (cerca de $3n^4/4$ variables binarias y cerca de $2n^4$ restricciones). Por otra parte, un grupo grande de restricciones presentan un comportamiento donde solo una restricción esta activa y las otras son redundantes (por ejemplo, existe siempre una restricción redundante entre (7) y (8)).

Por esta razón, la relajación PL del modelo obtiene un límite inferior de muy mala calidad. Todos estos factores hacen que el modelo sea inexplotable debido a su alta complejidad matemática en la práctica a través del software y hardware actual disponible para la programación entera mixta (para más detallada de este modelo ver [44]).

3.3. Modelo matemático del problema de empaquetamiento óptimo bidimensional de piezas rectangulares en una sola placa.

Es esta sección se plantea el problema de empaquetamiento óptimo en placas, en el cual solo se tiene una placa de material de ancho *W* y altura *H* (lo anterior se obtiene cambiando el sentido la placa, altura por longitud, el problema original continua intacto)

para empacar una selección de las piezas, donde el objetivo es maximizar el área usada de la placa.

El modelo presentado en la sección 3.2 será extendido para resolver el problema en una sola placa. Se modificará el valor del límite inferior $\overline{H}$ por *H*, para que este haga parte de la información del problema, además se modificará la función objetivo de tal manera que exprese la maximización del área usada (ver ecuación (25)). También deben ser modificadas las igualdades de las restricciones [(4)-(6)] para convertirlas en desigualdades, ya que no todo el conjunto de piezas esta restringido a ser empacado.

El modelo resultante se expresa a continuación, su definición y explicación es equivalente a la del modelo anterior.

$$\text{Maximizar} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} w_k h_k z_{i,j,k} \quad \forall k, \tag{25}$$

$$\text{sujeto a} \quad 0 \le x_1 \le x_2 \le \ldots \le x_n, \tag{26}$$

$$0 \le y_1 \le y_2 \le \ldots \le y_n, \tag{27}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} z_{i,j,k} \le 1 \quad \forall k, \tag{28}$$

$$\sum_{i=1}^{n}\sum_{k=1}^{n} z_{i,j,k} \le 1 \quad \forall j, \tag{29}$$

$$\sum_{j=1}^{n}\sum_{k=1}^{n} z_{i,j,k} \le 1 \quad \forall i, \tag{30}$$

$$x_{i'} - x_i - \sum_{k=1}^{n} w_k z_{i,j,k} \ge \left(u_{i,j,i'} - 1\right) W \quad \forall i, \forall j, \forall i' > i, \tag{31}$$

$$x_{i'} - x_i - \sum_{k=1}^{n} w_k z_{i,j,k} \ge u_{i,j,i'} W \quad \forall i, \forall j, \forall i' > i, \tag{32}$$

$$y_{j'} - y_j - \sum_{k=1}^{n} h_k z_{i,j,k} \ge \left(v_{i,j,i'} - 1\right) H \quad \forall i, \forall j, \forall j' > j, \tag{33}$$

$$y_{j'} - y_j - \sum_{k=1}^{n} h_k z_{i,j,k} \ge u_{i,j,j'} H \quad \forall i, \forall j, \forall j' > j, \tag{34}$$

$$\left(1 - d_{i_1,i_2,j_1,j_2}\right) n \ge \sum_{i=i_1}^{i_2}\sum_{j=j_1}^{j_2}\sum_{k=1}^{n} z_{i,j,k} - 1 \quad \forall i_1, \forall j_1, \forall i_2 > i_1, \forall j_2 > j_1, \tag{35}$$

$$p_{i_1,i',j_1,j_2} \le \sum_{j=j_1}^{j_2}\sum_{k=1}^{n} z_{i',j,k} \quad \forall i_1, \forall j_1, \forall j_2 > j_1, \forall i' > i_1, \tag{36}$$

$$p_{i_1,i',j_1,j_2} \le \sum_{i=i_1}^{i'-1}\sum_{j=j_1}^{j_2}\sum_{k=1}^{n} z_{i,j,k} \quad \forall i_1, \forall j_1, \forall j_2 > j_1, \forall i' > i_1, \tag{37}$$

$$\left(i' - i_1\right)\left(j_2 - j_1 + 1\right) p_{i_1,i',j_1,j_2} \le \sum_{i=i_1}^{i'-1}\sum_{j=j_1}^{j_2} u_{i,j,i'} \quad \forall i_1, \forall j_1, \forall j_2 > j_1, \forall i' > i_1, \tag{38}$$

$$q_{i_1,i_2,j_1,j'} \le \sum_{i=i_1}^{i_2} \sum_{k=1}^{n} z_{i,j',k} \quad \forall i_1, \forall j_1, \forall i_2 > i_1, \forall j' > j_1, \tag{39}$$

$$q_{i_1,i_2,j_1,j'} \le \sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j'-1} \sum_{k=1}^{n} z_{i,j,k} \quad \forall i_1, \forall j_1, \forall i_2 > i_1, \forall j' > j_1, \tag{40}$$

$$(j'-j_1)(i_2-i_1+1) q_{i_1,i_2,j_1,j'} \le \sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j'-1} v_{i,j,i'} \quad \forall i_1, \forall j_1, \forall j_2 > i_1, \forall j' > j_1, \tag{41}$$

$$d_{i_1,i_2,j_1,j_2} + \sum_{i'=i_1+1}^{i_2} p_{i_1,i',j_1,j_2} + \sum_{j'=j_1+1}^{j_2} q_{i_1,i_2,j_1,j'} \ge 1 \quad \forall i_1, \forall j_1, \forall i_2 > i_1, \forall j_2 > j_1, \tag{42}$$

$$W \ge x_i + \sum_{j=1}^{n} \sum_{k=1}^{n} w_k z_{i,j,k} \quad \forall i, \tag{43}$$

$$H \ge y_j + \sum_{i=1}^{n} \sum_{k=1}^{n} h_k z_{i,j,k} \quad \forall j, \tag{44}$$

$$z_{i,j,k} \in \{0,1\} \quad \forall i, \forall j, \forall k, \tag{45}$$

$$u_{i,j,i'} \in \{0,1\} \quad \forall i, \forall j, \forall i' > i, \tag{46}$$

$$v_{i,j,j'} \in \{0,1\} \quad \forall i, \forall j, \forall j' > j, \tag{47}$$

$$d_{i_1,i_2,j_1,j_2}, p_{i_1,i_2,j_1,j_2}, q_{i_1,i_2,j_1,j_2} \in \{0,1\} \quad \forall i, \forall j, \forall i_2 > i_1, \forall j_2 > j_1, \tag{48}$$

3.4. Modelo matemático del problema de empaquetamiento óptimo bidimensional de piezas rectangulares en placas.

Ben et al. en [44] presenta una extensión de su modelo para el problema de empaquetamiento óptimo guillotinado en placas sin rotación de piezas. Esta consiste en agregar un subíndice correspondiente a la placa y modificando la función objetivo de tal manera que esta exprese la minimización del número de placas necesarias para empacar la totalidad de las piezas. Es decir, la variable $z_{i,j,k,l}$ será igual a *1*, si la pieza *k* es embalada en la posición *(i, j)* de la placa *l*, de lo contrario, será igual a *0*.

Sea Z_l el indicador si la placa *l* fue usada, la función objetivo se convierte en Minimizar $\sum_{l=1}^{n} Z_l$, además de esto la siguiente restricción debe ser agregada $Z_l \ge z_{i,j,k,l} \quad \forall i, \forall j, \forall k, \forall l$.

3.5. Resumen

Los modelos presentados en este estudio a pesar de ser estudiados por más de 6 décadas no han sido bien definidos para todas las diferentes características que presentan, en especial los patrones de corte guillotina.

En este capitulo se presento un modelo propuesto recientemente por Ben et al. en [44], este describe perfectamente uno de los problemas presentados en este estudio, aunque

carece de la restricción de rotación de piezas. Algunas ideas son presentadas por Lodi et al. en [40] para incluir esta variante.

El problema de empaquetamiento óptimo en una sola placa se expresa con base en el modelo propuesto por Ben et al. en [44].

La conclusión de los estudios de Lodi et al. en [40] y Ben et al. en [44] es que actualmente no existe software y hardware para darle solución a instancias prácticas del problema mediante los modelos planteados. Por otro lado, en este estudio se plantea una metodología basada en novedosas técnicas metaheurísticas de optimización que de una solución eficiente.

4. DESCRIPCIÓN DE LAS TÉCNICAS METAHEURÍSTICAS DE OPTIMIZACIÓN: RECOCIDO SIMULADO, BÚSQUEDA EN VECINDARIO VARIABLE Y OPTIMIZACIÓN CON CÚMULO DE PARTÍCULAS

4.1. Introducción

Durante los últimos 60 años, se han desarrollado una gran cantidad de heurísticas [6, 17, 19, 55, 56, entre otras] para la solución del problema de empaquetamiento óptimo bidimensional tanto en placas como en rollos (Toro en [48] hace una recopilación detallada de estas). Las técnicas metaheurísticas han demostrado excelente desempeño en problemas de difícil solución, el problema de empaquetamiento esta incluido en esta categoría. Las heurísticas constructivas son una buena alternativa en el proceso de inicialización de un proceso metaheurístico o como factores de sensibilidad para guiar el proceso en cada paso [18, 21, 28, 29, 45, 46, 47, 49, entre otras]. Una de las primeras aproximaciones metaheurísticas al problema de *strip packing*, es el algoritmo de recocido simulado presentado por Dowsland en [45], el cual explora tanto un espacio solución factible como infactible en su proceso de búsqueda. Jackobs en [47] propone también una metaheurística para el problema de *strip packing*, mediante un algoritmo genético. Su aproximación hace uso de la heurística *bottom-left corner* (esquina inferior izquierda) para generar los patrones de corte del problema. Lai y Chan en [49] usan el algoritmo de recocido simulado para dirigir la heurística de *difference process* (proceso de diferencia).

La mayoría de los trabajos que consideran los problemas de *strip, cutting stock* y *bin packing* usan algoritmos aproximados. W. Fernandez de la Vega y V. Zissimopoulos en [24], N. Lesh et al. en [25], C. Kenyon y E. Remila en [26] y D. Zhang et al. en [27] presentan algoritmos de aproximación para el problema de empaquetamiento óptimo en rollos, mientras A. Gomes y J. F. Oliveira en [53], A. Bortfeldt en [28] y E. Toro en [48] usan metaheurísticas para el problema de empaquetamiento óptimo en placas, E. Hopper y B. Turton en [30] proveen una descripción general en cuanto técnicas metaheurísticas aplicadas al problema de *strip packing*. El problema de *bin packing* es presentado por Lodi et al. en [6].

Chung et al. en [55], Chazelle en [17], Lodi et al. en [6] proponen heurísticas para el problema de *bin packing*. Un estudio de comparación de heurísticas para resolver el problema de empaquetamiento óptimo bidimensional tipo guillotina en placas es presentado por Lodi et al. en [19] y en rollos es presentado por A. Bekrar y I. Kacem en [56].

Dentro del conjunto de técnicas metaheurísticas seleccionadas para la solución de los problemas de empaquetamiento propuestos en este estudio, están las técnicas de: recocido simulado, búsqueda en vecindario variable y optimización con cúmulo de partículas. A partir de ellas se desarrollaron 3 algoritmos mixtos y además se implementó el algoritmo de optimización con cúmulo de partículas propuesto por Lui et al. [57] para resolver el problema de *bin packing* en una versión multiobjetivo.

Dentro de las técnicas metaheurísticas propuestas en este proyecto, Parada et al. en [22] utilizan el algoritmo de recocido simulado para resolver *constrained guillotine cutting stock problem,* Chen et al. en [58] y L. Faina en [59] proponen el algoritmo de recocido simulado para trabajar los problemas de *one-dimensional y two-dimensional cutting stock*. Por otro lado, Toro et al. en [23], utilizan dos de las técnicas propuestas (recocido simulado y búsqueda en vecindario variable) para resolver el problema de *two-dimensional guillotine cutting stock*. Y el ya mencionado trabajo de Lui et al. en [57] presentan el algoritmo de optimización con cúmulo de partículas para resolver el problema de *bin packing* multiobjetivo.

A continuación se describen las técnicas metaheurísticas de optimización usadas en este estudio.

4.2. Recocido Simulado

El recocido simulado (RS) es una metaheurística de búsqueda local usada para direccionar problemas de optimización. La característica principal del recocido simulado es el proveer de medios para escapar de óptimos locales y lograr determinar soluciones de excelente calidad.

El recocido simulado (*simulated annealing*) es llamado así debido a su analogía con el proceso físico de recocido de sólidos, en el cual un sólido es llevado a temperaturas altas y luego a través de un programa de enfriamiento lento es llevado a una estructura cristalina perfecta (es decir, su estado mínimo de energía), y así se genera un cristal libre de defectos. Si el programa de enfriamiento es lo suficientemente lento, la estructura final será la de un sólido con integridad estructural superior. El algoritmo de recocido simulado establece la conexión entre este tipo comportamiento termodinámico y la búsqueda de un óptimo global en un proceso de optimización.

El RS inicia el proceso con un estado inicial, un cambio aleatorio es propuesto para este estado y un cambio de energía es generado, ΔE, es calculado. Si el nuevo estado tiene un nivel más bajo de energía que el estado anterior, $\Delta E \leq 0$, el nuevo estado es tomado para la siguiente iteración. Sin embargo, si el nuevo estado tiene un nivel de energía más alto que el anterior, este es aceptado con una probabilidad que depende de un parámetro temperatura, la cual es normalmente disminuida con cada iteración del algoritmo (ver Kirkpatrick en [60], Glover et al. en [51] y Gallego et al. en [52] para un descripción más detallada del recocido simulado).

Esta característica establece que el RS acepte con algún grado de probabilidad empeoramientos en la función objetivo y de esta forma escapar de óptimos locales lo que hace que el RS sea diferente a los algoritmos de búsqueda local. Como el parámetro temperatura decrece, los movimientos de empeoramiento ocurren con menor frecuencia. La distribución de soluciones asociadas con la cadena no homogénea de Markov que modela el comportamiento de convergencia del algoritmo es llevada a una forma en la cual toda la probabilidad se concentra en el conjunto de soluciones óptimas globales (siempre que el algoritmo sea convergente; de lo contrario el algoritmo convergerá a un óptimo local, el cual puede ser o no óptimo global).

El algoritmo RS es presentado en la tabla 1. El programa de enfriamiento en el paso 4.2 es crucial para el éxito del algoritmo, si se configura inteligentemente el enfriamiento, el algoritmo puede escapar de óptimos locales tempranamente y explorar profundamente otras regiones prometedoras.

RS	
1.	Escoja una solución inicial S
2.	Halle la temperatura inicial $T = T_0 > 0$
3.	Seleccione un programa de repetición M, que define el número de iteraciones en cada temperatura T
4.	Repita:
	4.1 Para $m = 0$ hasta M haga:
	a. Escoja una nueva solución S'
	b. Sea $\Delta E = E(S)' - E(S)$, donde $E(S)$ es la energía de la solución S
	c. Si $\Delta E \leq 0$, acepte la nueva solución, haga $S = S'$
	d. De lo contrario, Si $exp(-\Delta E/T) \geq rand(0,1)$, acepte la nueva solución
	e. De lo contrario rechace la solución
	4.2. Reduzca la temperatura de acuerdo al enfriamiento programado
5.	Hasta la condición de parada.

Tabla 1. Algoritmo recocido simulado

4.3. Búsqueda en Vecindario Variable

La búsqueda de vecindario variable (*Variable Neighbourhood Search, VNS*) es una metaheurística reciente 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 o pendientes de aparecer [61]. Se han realizado muchas extensiones, principalmente para permitir la solución de problemas de gran tamaño [64]. En la mayoría de ellas, se ha hecho un esfuerzo por mantener la simplicidad del esquema básico [62 y 63].

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.) para evitar el ciclado, que puede ocurrir si se usa cualquier regla determinística.

VNS Básico			
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.		

Tabla 2. Algoritmo búsqueda en vecindario variable

4.4. Optimización con Cúmulo de Partículas

El algoritmo optimización con cúmulo de partículas (*Particle Swarm Optimization*, PSO) usa un simple mecanismo que imita el comportamiento de las bandadas de pájaros y los cardumes de peces en cuanto a la forma en que guían sus desplazamientos a fin de encontrar alimento y refugio, considerados como su objetivo primordial [65].

PSO es fácil de implementar y ha sido ampliamente usado en variadas aplicaciones con resultados excelentes resolviendo problemas reales de optimización. Este algoritmo puede ser computacionalmente ineficiente por que puede quedar atrapado fácilmente en óptimos locales cuando resuelve problemas cuyo espacio de solución es multimodal; estas debilidades han hecho que el campo de aplicación de la metodología este un poco restringida, sin embargo acelerar la velocidad de convergencia y evitar los óptimos locales se han convertido en los dos más importantes objetivos en la investigación de PSO [66].
En este desarrollo, el control de parámetros del algoritmo y la combinación de los operadores de búsqueda se han convertido en auxiliares de dos de los tres enfoques más destacados y prometedores, el otro es la mejora de la estructura topológica [67].

Para mejorar la eficiencia y acelerar el proceso de búsqueda es fundamental determinar el estado de evolución y los mejores valores para los parámetros, con el fin de evitar posibles óptimos locales en el estado de convergencia. Algunas técnicas se han planteado introduciendo operadores como la selección [68], cruzamiento [69], mutación [70], búsqueda local [71], reinicio de algunos operadores [72] y [73], reinicialización de absolutamente todos los operadores dentro del proceso [74] y [75].

Estas operaciones híbridas normalmente se implementan en cada generación [70], [72] o en un intervalo prefijado [71] o son controladas por una función de adaptación definida para el caso específico.

En PSO, un conjunto de partículas representan soluciones potenciales, donde cada partícula *i* está asociada a dos vectores, el vector de velocidad $V_i = [v_i^1, v_i^2, ..., v_i^D]$ y a el vector posición $X_i = [x_i^1, x_i^2, ..., x_i^D]$ donde D es el número de características que determinan el tamaño del espacio de soluciones. La velocidad y la posición de cada partícula son inicializadas por vectores aleatorios con sus correspondientes rangos. Durante el proceso de evolución, la velocidad y la posición de la partícula *i* se actualizan mediante las expresiones (49) y (50) respectivamente.

$$v_i^d = wv_{i-1}^d + c_1 rand_1^d (pbest_i^d - x_i^d) + c_2 rand_2^d (gbest^d - x_i^d) \quad (49)$$

$$x_i^d = x_{i-1}^d + v_i^d \quad (50)$$

Donde w es el peso de inercia, c_1 y c_2 son los coeficientes de aceleración y $rand_1$ y $rand_2$ son números aleatorios uniformemente distribuidos y generados en el intervalo [0-1] para la *d-esima* dimensión.

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 actual del líder. En la tabla 3 se presenta los pasos básicos del PSO.

PSO	
1.	Generar una población inicial de partículas, con posiciones y velocidades en el espacio del problema *d* dimensional (generada de forma aleatoria).
2.	Repita:
	2.1. Evaluar la función objetivo a cada partícula
	2.2. Comparar el valor de la función de adaptación (*fitness*) de la partícula con el *pbest* de la partícula. Si su valor corriente es mejor que el *pbest* pasa a ser igual al valor de la *fitness* de la partícula y la localización de la *pbest* pasa a ser igual a la localización actual del espacio *d* dimensional
	2.3. Comparar el valor de la función *fitness* con el mejor valor de adaptación de la población. Si el valor actual es mejor que el g*best*, actualizar el valor del g*best*
	2.4. Modificar la velocidad de la posición de acuerdo a las ecuaciones (49) y (50) respectivamente.
3.	Hasta la condición de parada.

Tabla 3. Algoritmo optimización con cúmulo de partículas

4.5. Resumen

En este capítulo se realizo una revisión bibliográfica de las diferentes técnicas metaheurísticas de optimización propuestas para resolver los problemas de empaquetamiento óptimo bidimensional tanto en placas como en rollos. Al igual que son utilizadas diferentes técnicas se usan distintos enfoques que son propuestos para

resolver los problemas. Lo anterior conduce a enfoques diferentes aun usando una técnica ya utilizada de origen a metodologías interesantes que en la mayoría de los casos presentan excelentes resultados.

Las técnicas propuestas en este trabajo: recocido simulado, búsqueda en vecindario variable y optimización con cúmulo de partículas, son descritas someramente. El lector es invitado a las referencias bibliográficas presentadas donde se hace una descripción más detallada de las técnicas.

Las tres técnicas expresadas anteriormente conformaran las bases para 4 diferentes algoritmos presentados en este estudio, solo el algoritmo optimización con cúmulo de partículas será implementado como se describió en este capítulo, mientras los 3 algoritmos restantes son formulaciones híbridas entre estos. Los algoritmos propuestos en este estudio son:

- Algoritmo Optimización con Cúmulo de Partículas
- Algoritmo Híbrido Recocido Simulado y Búsqueda en Vecindario Variable
- Algoritmo Híbrido Optimización con Cúmulo de Partículas y Búsqueda en Vecindario Variable
- Algoritmo Híbrido Optimización con Cúmulo de Partículas con el operador mutación

En el capítulo siguiente se describe la adaptación de los algoritmos en la solución de problemas de empaquetamiento óptimo.

5. ADAPTACIÓN DE LAS TÉCNICAS DE OPTIMIZACIÓN COMBINATORIA A LA SOLUCIÓN DEL PROBLEMA

5.1. Introducción

Utilizar metodologías exactas en problemas de alta complejidad como los de corte y empaquetamiento presenta grandes dificultades respecto a la baja calidad de respuestas y altos tiempos de cómputo, a diferencia del uso de metodologías aproximados.

Por otra parte, para el desarrollo de una metodología aproximada es necesario de un buen entendimiento del problema. Este requiere describir la información, definir el flujo de datos a través de los algoritmos y la calibración de parámetros de los algoritmos (en especial si estos son metaheurísticas). Este esquema se puede resumir en: codificación, algoritmos y calibración de parámetros.

Distintas codificaciones son presentadas en la literatura especializada, dos de las más usadas son la representación mediante estructura de datos tipo vector y la tipo árbol de cortes (Toro en [48] realiza un descripción más detallada de las diferentes estructuras de datos). Por ejemplo, Jakobs en [47] hace uso de la estructura de datos tipo vector para codificar los patrones de empaquetamiento usando un algoritmo genético, por otro lado, Kröger en [46] utiliza la estructura de datos tipo árbol de cortes para codificar los patrones de empaquetamiento. En este estudio se ha optado por trabajar la estructura de datos tipo árbol de cortes.

En este capítulo se describen cuatro versiones de algoritmos que usan árbol de cortes para representar los patrones de empaquetamiento. Además, se detalla el proceso de calibración de los parámetros de cada algoritmo.

5.2. Codificación

En este estudio se seleccionó una representación presentada por Wong et al. en [76], esta estructura de datos se codifica a través de un árbol (llamado árbol de cortes) que determina 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 guillotina. Además, asegura que para todo el espacio de soluciones que conforman los diferentes patrones guillotina existe al menos un árbol de cortes que representa una de estas soluciones [76]. Diferentes metodologías propuestas han corroborado la efectividad de usar una codificación en árbol de cortes en especial la presentada por Cui en [22] y la de Toro et al en [23].

La capacidad de las estructuras tipo árbol de cortes, es el poder que tiene esta como representación de datos jerárquicos y la fácil adaptación de técnicas numéricas de conglomerados (clusters) para agrupar elementos en grupos.

Un árbol de cortes se define como un árbol con raíz, donde cada nodo interno (padre) representa la posición y la forma como se realiza el corte sobre el material, mientras los

nodos hoja (nodos sin hijos) representan las dimensiones de los sub-espacios generados para cortar las piezas agrupadas.

La aplicación de técnicas de conglomerados se reduce a agrupar las piezas a cortar en grupos de piezas iguales, por lo que en los problemas donde no se permite la rotación de piezas este proceso es trivial, mientras que en los problemas donde la rotación de piezas de 90° es permitida se deben generar las diferentes *2n* piezas (donde *n* es el número de piezas disponibles) y luego realizar la agrupación.

Diferentes propuestas con árbol de cortes, proponen encontrar el árbol de cortes óptimo, mientras propuestas como la de Toro et al. en [23] delimitan y reducen el número de árboles diferentes en el proceso de optimización. Toro et al. en [23] sugieren como resultado de un estudio estadístico, delimitar el árbol de cortes al uso de solo árboles binarios completos con tres niveles. Para este estudio se fija esta restricción para los problemas de *strip packing, cutting stock y bin packing.*

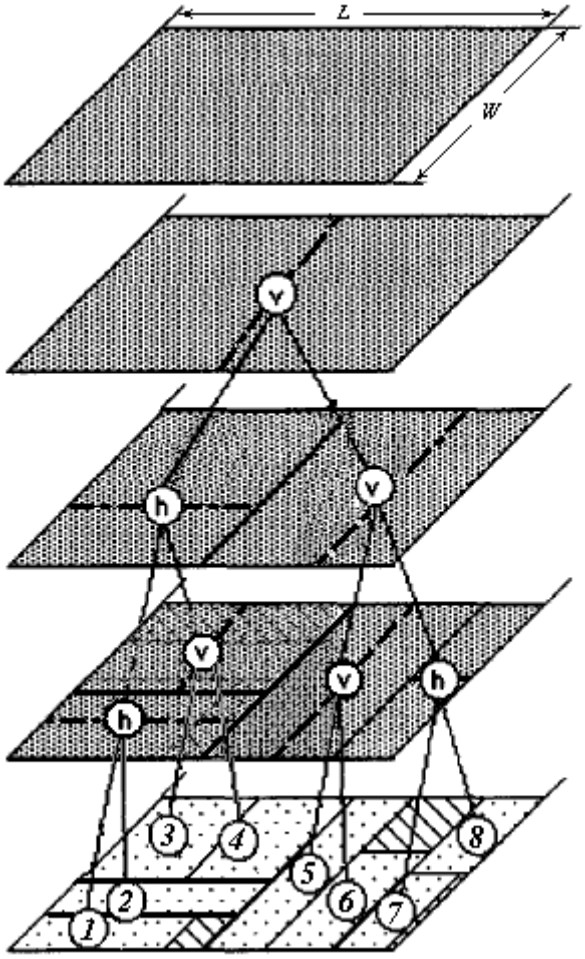

Figura 5. Representación mediante árbol de cortes.

La figura 5 ilustra un árbol para el problema de *cutting stock,* en esta el nodo raíz sugiere realizar un corte vertical sobre la placa de material. La jerarquía del árbol indica que el hijo izquierdo efectúa un corte horizontal sobre el rectángulo izquierdo, mientras el hijo derecho realiza un nuevo corte vertical sobre el rectángulo derecho, esto para el

primer nivel. En el segundo nivel se realiza un corte horizontal al rectángulo inferior izquierdo, obteniéndose en el tercer nivel dos sub-espacios 1 y 2, mientras que en el rectángulo superior izquierdo se efectúa un corte vertical obteniendo en el tercer nivel los sub-espacios 3 y 4. De la misma forma los sub-espacios 5, 6, 7 y 8 son el resultado de los cortes realizados en el segundo nivel. En cada sub-espacio generado se ubican las piezas con igual forma que maximice el área utilizada.

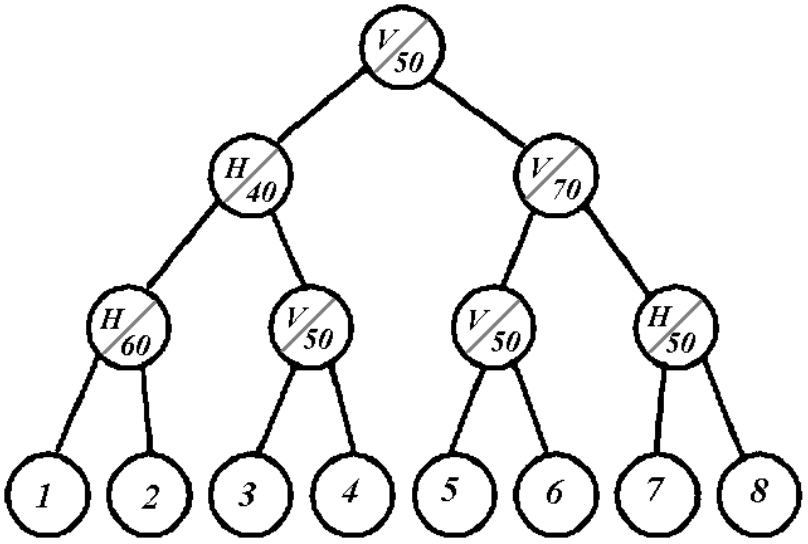

Figura 6. Ejemplo de árbol de cortes.

Por ejemplo, si se tiene una placa con longitud y ancho de 1800 y 1000 unidades respectivamente y en la figura 6 se presenta el árbol de cortes, las dimensiones de los sub-espacios resultantes serán las presentadas en la tabla 4 y la cual describe las dimensiones de las hojas del árbol denominados sub-espacios. Obsérvese en la figura 6 que cada nodo del árbol de cortes excepto las hojas contiene dos datos: orientación del corte (horizontal o vertical) y distancia del corte (porcentaje de cero a cien). Por lo tanto el nodo raíz indica que se debe realizar un corte vertical al 50% de la longitud de la placa de material, dividiendo la placa en dos nuevos rectángulos (fíjese que este corte es tipo guillotina) ambos con iguales dimensiones longitud 900 y ancho 1000 unidades. Así sucesivamente se recorre el árbol hasta sus hojas, donde son almacenadas las dimensiones de los sub-espacios.

Sub-espacio	Largo	Ancho
1	900	240
2	900	160
3	450	600
4	450	600
5	315	1000
6	315	1000
7	270	500
8	270	500

Tabla 4. Dimensiones de los sub-espacios

La metodología propuesta por Toro et al. en [23] delimita el problema de optimización a encontrar el árbol completo en tres niveles. Este árbol estará compuesto por 7 nodos,

donde cada nodo se conforma por una pareja de datos: orientación del corte y porcentaje de distancia. En este estudio se propone separar este árbol de cortes en dos, un árbol de orientación de los cortes y un árbol de distancias de cortes.

El tamaño del espacio de solución es igual al número de árboles de orientación de cortes (2^7 árboles diferentes), multiplicado por el número de árboles de distancia de cortes ($max(L, W)^7$ árboles diferentes). Dado que el tamaño del espacio de soluciones del árbol de orientación de los cortes es pequeño en comparación con el de distancias, se puede realizar una búsqueda exhaustiva sobre el primer árbol, mientras el segundo árbol se convertirá en el problema de optimización a resolver en este estudio.

Luego de obtener los sub-espacios se debe realizar la ubicación de las piezas en estos, este proceso se realiza mediante el algoritmo constructivo denominado mejor-ajuste *(best-fit)*, con este se logra que se conserven las restricciones tipo guillotina y se pretende embalar la mayor cantidad de piezas por sub-espacio.

El algoritmo constructivo *best-fit* consiste en encontrar el conjunto de piezas idénticas que maximiza el área del sub-espacio *j*, sujeto a la restricción del número de piezas *i* disponibles. La ecuación (51) expresa formalmente el algoritmo *best-fit*.

$$\max\left\{\min\left(\left\lfloor\frac{W_j}{w_i}\right\rfloor\cdot\left\lfloor\frac{L_j}{l_i}\right\rfloor, \textit{piezas disponibles tipo } i\right)\cdot w_i\cdot l_i; \forall i; i=1,2,...,n\right\} \qquad (51)$$

El cálculo de la función objetivo consiste en aplicar el algoritmo constructivo *best-fit* a cada sub-espacio, siendo el valor de la función objetivo la suma de las áreas de las piezas empacadas. El diagrama para el cálculo de la función objetivo es ilustrado en la figura 7.

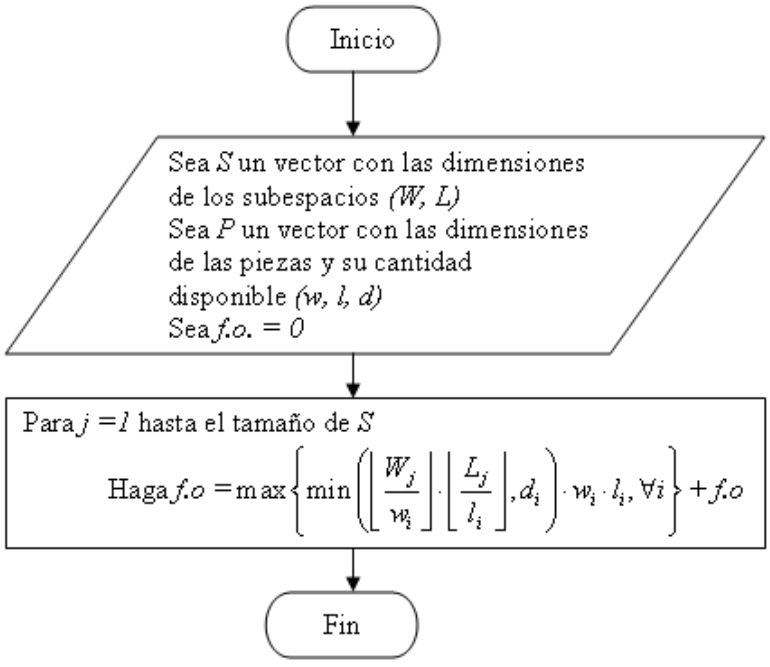

Figura 7. Algoritmo para el cálculo de la función objetivo.

A continuación en la tabla 5 se presenta un conjunto de piezas disponibles para ser embaladas en la placa del ejemplo anterior. Dado que en la tabla 4 se presentan las dimensiones de los sub-espacios generados por el árbol de cortes del ejemplo, ambas tablas contienen la información suficiente para calcular función objetivo.

Número de la Pieza	Longitud	Ancho	Disponibilidad
1	200	100	15
2	300	200	17
3	150	150	20
4	50	500	20
5	400	80	30

Tabla 5. Características de las piezas disponibles.

El valor de la función objetivo del ejemplo es igual a .1660.000 unidades cuadradas, las dimensiones de la placa son conocidas, por lo tanto, se puede calcular el porcentaje de uso de la placa 1.660.000/1.800.000 = 92,22%. La ubicación de las piezas son presentadas en la tabla 6 y la gráfica resultante de la colocación de estas es ilustrada en la figura 8. En la figura 8, los espacios en gris representan el área no utilizada de la placa.

Sub-espacio	Número de la Pieza	Cantidad de Piezas
1	2	3
2	3	6
3	4	9
4	3	12
5	2	5
6	2	5
7	4	5
8	4	5

Tabla 6. Ubicación de las piezas del ejemplo en formato tabla.

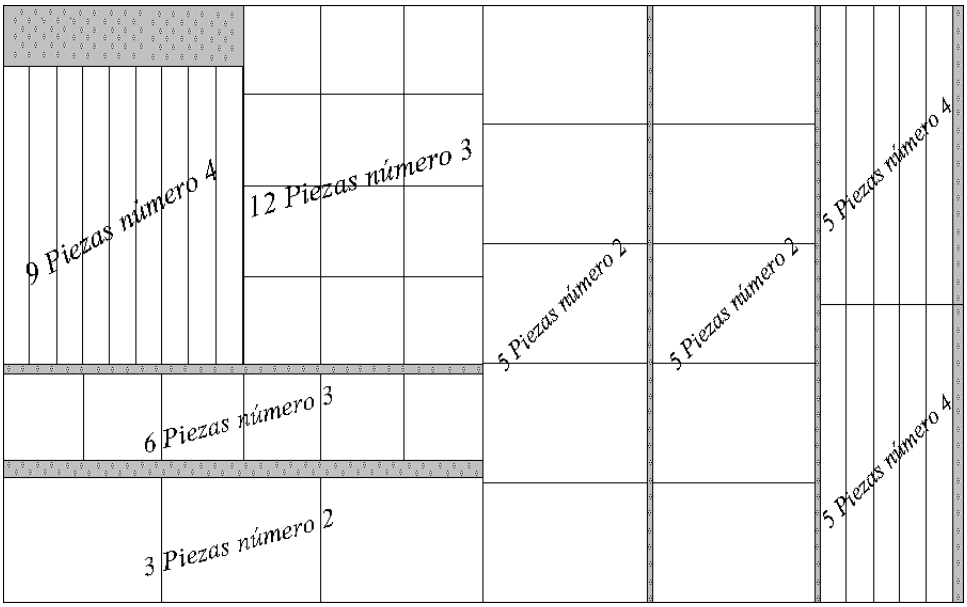

Figura 8. Ubicación de las piezas del ejemplo gráficamente.

Metodología

La codificación propuesta en este capítulo garantiza la factibilidad en cuanto a las restricciones de corte tipo guillotina. Este tipo de codificación permite transformar los problemas de *cutting stock* en problemas de minimización irrestrictos.

En estos el árbol de orientación de cortes y el árbol de distancias de los cortes son representados por las variables *O* y *D* respectivamente y los cuales son independientes entre sí, esto significa que para cada conjunto de valores de *O* existe una solución óptima D^*. El esquema de optimización para los problemas de *cutting stock* es ilustrado en la figura 9. En este el Algoritmo *I* realiza una búsqueda exhaustiva sobre el árbol *O,* mientras que el Algoritmo *II* recibe todos árboles *O* posibles, donde debe encontrar las distancias óptimas para cada uno de estos. El algoritmo II corresponde al escenario de las técnicas metaheurísticas.

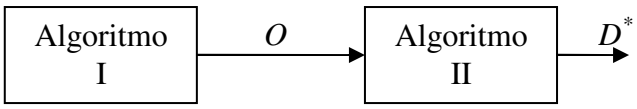

Figura 9. Esquema *Cutting Stock.*

Los problemas de *strip packing* aun siendo transformados en problemas de minimización irrestrictos, deben ser procesados para convertirse en *m* (donde, *m* es el número de placas de material necesarias para realizar la conversión de rollo a placa) diferentes problemas de *cutting stock*. Para esto el esquema de optimización debe ser modificado, ya que tendrá que ser introducido un nuevo algoritmo que transforme el problema de *strip* en *bins.* El esquema de optimización para los problemas de *strip packing* es ilustrado en la figura 10, en este el Algoritmo *T* se encarga de realizar la transformación de rollos a placas. Luego cada placa es procesada por los algoritmos I y II, los cuales equivalen a los algoritmos del esquema anterior.

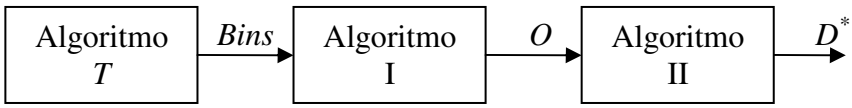

Figura 10. Esquema *Strip Packing.*

Los problemas de *bin packing* por otro lado se resuelven como solución de *l* problemas consecutivos de *cutting stock*, siendo *l* el número mínimo de placas necesarias para embalar la totalidad de las piezas. Para esto el esquema de optimización se resume en repetir los algoritmos I y II actualizando el inventario de piezas utilizadas en cada placa, por lo tanto la demanda de piezas ira disminuyendo mientras evoluciona el proceso.

Algoritmo *I*

Como fue enunciado el algoritmo I genera los posibles árboles de orientación de los cortes, Los problemas de empaquetamiento en placas se restringe a solo árboles completos de 3 niveles, el resultado de este algoritmo siempre dará como resultado 128 diferentes árboles de orientación, cada uno de estos es usado como dato de entrada para el algoritmo *II.*

Algoritmo *T (Transformación Strip a Bins)*

El algoritmo *T* consiste en determinar el número total de placas (*bins*) necesarias y las dimensiones de cada una, luego cada placa generada se resuelve como un problema de *cutting stock* a través de los algoritmos I y II. Las dimensiones de los *bins* están limitadas en todo momento a las restricciones tecnológicas. El procedimiento para calcular el número de *bins* es ilustrado en la ecuación (52) y se resume en los pasos de la tabla 7.

Número de *bins*	
1.	Calcular la sumatoria de las áreas de las piezas demandas y dividir esta por el ancho *W* del *strip,* este resultado es llamado L_{opt}.
2.	Definir l_{max} como la longitud de la pieza de mayor largor del conjunto de piezas demandadas.
3.	Definir w_{max} como el ancho de la pieza de mayor anchor del conjunto de piezas demandadas.
4.	Dividir el resultado del paso 1 entre l_{max} y dividir *W sobre* w_{max}.
5.	Seleccionar el máximo de los resultados del paso 4.

Tabla 7. Cálculo del número de *bins*

$$Número\ de\ Bins = \max\left(\left\lfloor \frac{\sum_{i=1}^{N} w_i \cdot l_i}{\frac{W}{l_{\max}}} \right\rfloor, \left\lfloor \frac{W}{w_{\max}} \right\rfloor\right) \qquad (52)$$

Teniendo el número y las dimensiones de los *bins* se resuelven como problemas de *cutting stock* secuenciales y dependientes, donde la demanda de piezas ira disminuyendo mientras evoluciona el proceso. Debido a que el largo del rollo L_{opt} es un valor ideal, durante el proceso se incrementará en caso de que no se logre satisfacer la demanda de piezas.

Algoritmo *II*

En el algoritmo *II* se hace uso de las técnicas metaheurísticas de optimización, en este estudio como ya fue mencionado se proponen cuatro diferentes algoritmos de optimización:

- Algoritmo Optimización Cúmulo de Partículas
- Algoritmo Híbrido Búsqueda en Vecindario Variable y Recocido Simulado
- Algoritmo Híbrido Optimización Cúmulo de Partículas y Búsqueda en Vecindario Variable
- Algoritmo Híbrido Optimización Cúmulo de Partículas y Recocido Simulado

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

El primer algoritmo propuesto para optimizar el árbol de distancias es el PSO, como fue presentado en el capítulo anterior es una herramienta poderosa en los problemas de optimización.

La figura 11 ilustra en un diagrama de flujo el algoritmo A_{PSO}. En la población, cada partícula representa un árbol de alternativa (D_i) en el proceso de optimización. El cálculo de la función objetivo se realiza como fue definida en la sección anterior. A diferencia de otros algoritmos de optimización, el PSO es del tipo elitista al conservar dentro de la población la configuración incumbente, la cual es actualizada en cada generación.

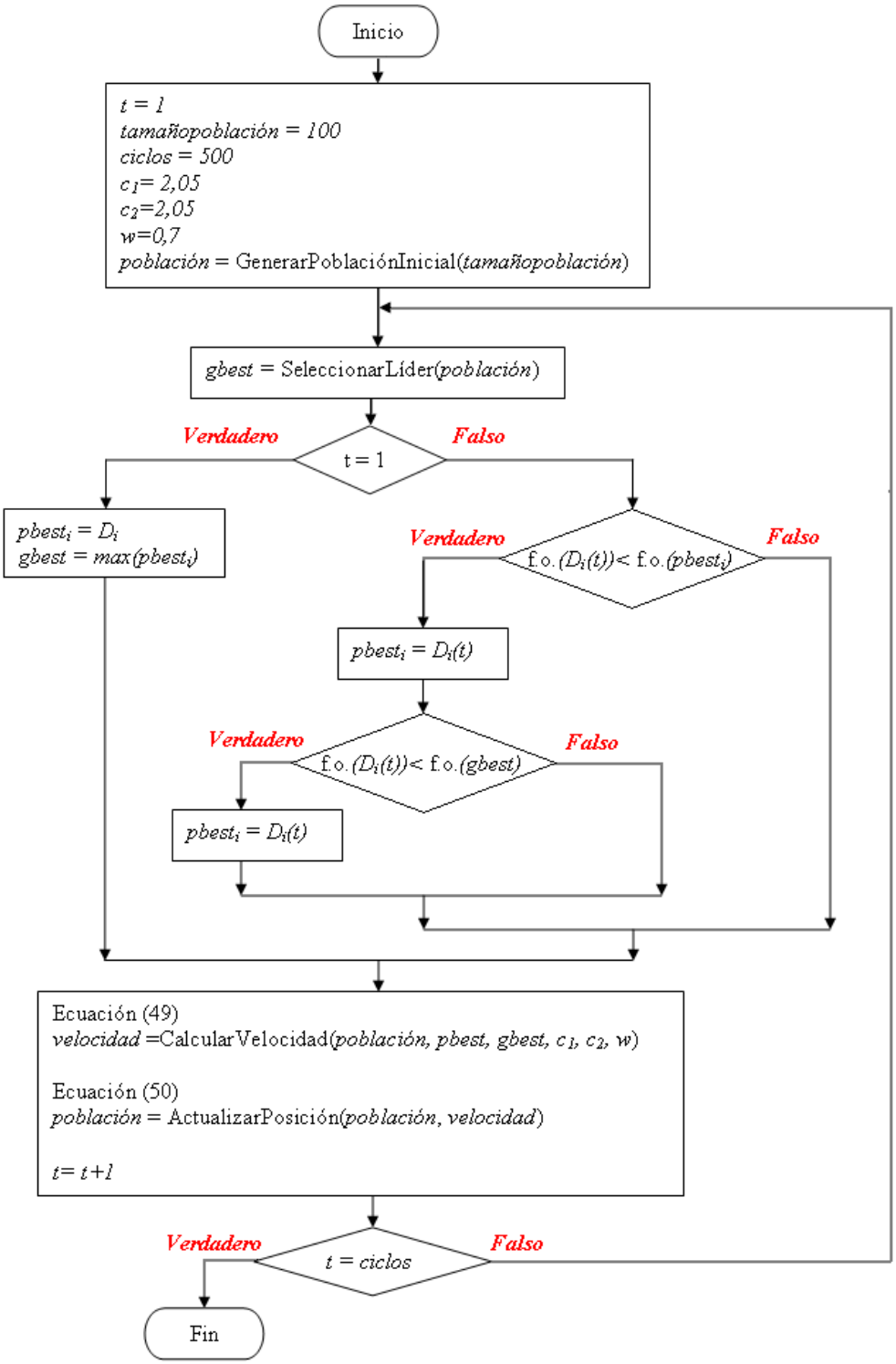

Figura 11. Algoritmo Optimización con Cúmulo de Partículas.

5.4. Algoritmo Híbrido Búsqueda en Vecindario Variable y Recocido Simulado (A_{VNS+RS})

El segundo algoritmo propuesto combina el algoritmo de búsqueda en vecindario variable y un mecanismo de transición inspirado en la técnica del recocido simulado.
El mecanismo de transición consiste en permitir grandes cambios en las distancias de los cortes durante las primeras iteraciones, al igual que el recocido simulado se permiten empeoramientos fuertes de la función objetivo al comienzo del proceso. En la medida en que avanza el proceso la probabilidad de aceptar empeoramientos es menor, como es el caso del recocido simulado.

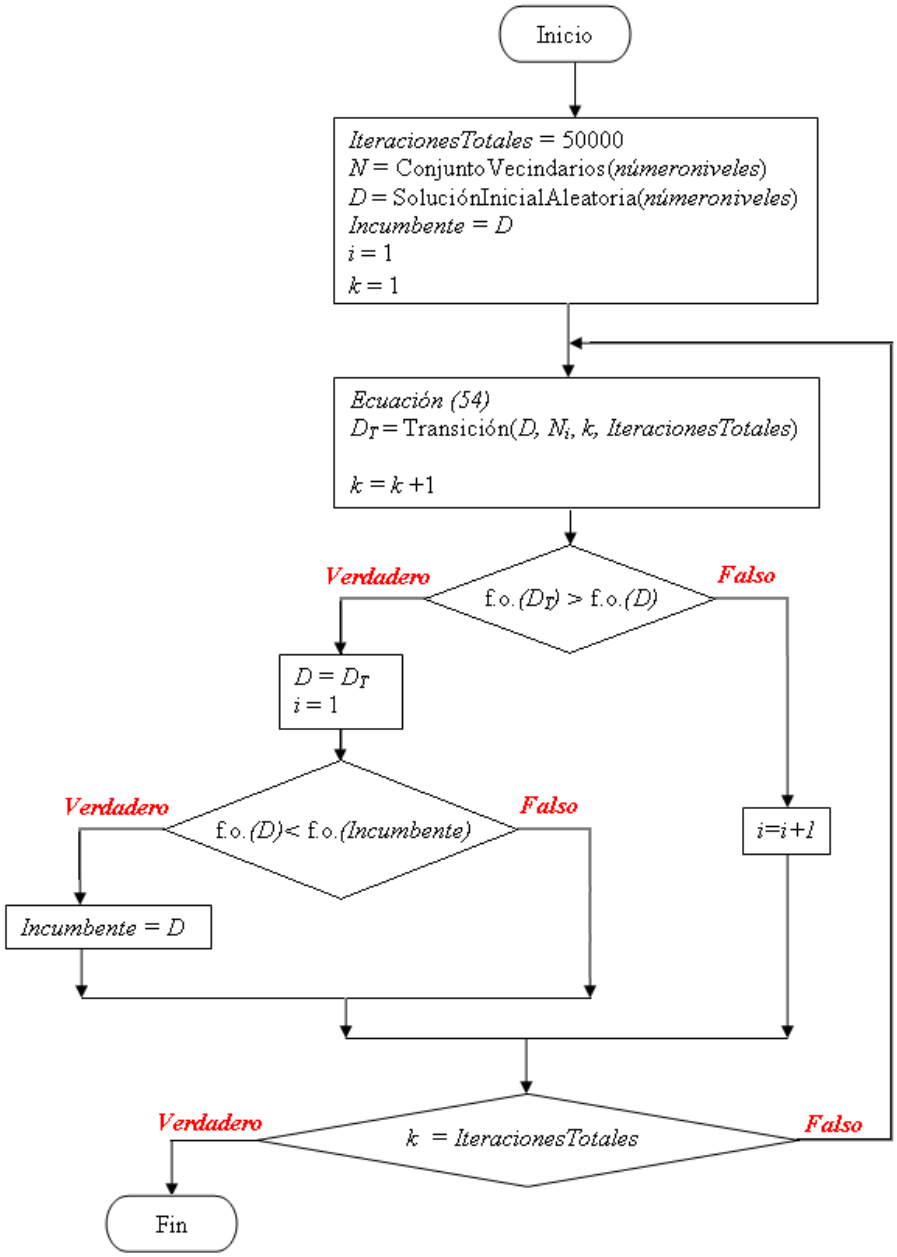

Figura 12. Algoritmo Híbrido Búsqueda en Vecindario Variable y Recocido Simulado.

La ecuación (54) define el mecanismo de transición, en esta se involucran: el valor actual del nodo *i* del árbol de distancias, el número de iteración actual, el número de

iteraciones totales y un Epsilon (donde ε, es el mínimo porcentaje para generar un cambio en la distancias) donde, $\varepsilon = 100/\max(L,W)$.

$$nodo\ i = nodo\ i + \left(rand - \frac{1}{2} \right) \cdot \left(\sqrt{1 - \frac{k}{IteracionesTotales}} + \varepsilon \right) \qquad (54)$$

El vecindario N_k (donde, $k \in \{1,2,..., k_{max}\}$; siendo, k_{max} el número de nodos padres del árbol de cortes) se define como k nodos padres aleatorios del árbol de cortes, es decir, si:
$k=1$, entonces, N_1 = {nodo i_1; donde i_1 es un número aleatorio entre [1 - k_{max}]}
$k=2$, entonces, N_2 = {nodo i_1, nodo i_2; donde i_1 y i_2 son números aleatorios entre [1-k_{max}] e $i_1 \neq i_2$}
$k=l$, entonces, N_l = {$nodo\ i_k, \forall k$; siendo, $k = 1, 2,..., l$; donde i_k es un número aleatorio entre [1 - k_{max}] e $i_1 \neq i_2 \neq ... \neq i_l$}
Por lo tanto, el conjunto de vecindarios $N = \{N_1, N_2,..., N_l,..., N_{kmax}\}$.

La figura 12 ilustra el diagrama de flujo del algoritmo A_{VNS+RS}. El proceso de optimización es dirigido a través de una búsqueda en diferentes vecindarios, organizados en orden ascendente por su tamaño. Al inicio el vecindario más pequeño consisten en realizar una transición en un nodo aleatorio, el siguiente es modificar dos nodos aleatorios, así sucesivamente como lo indica el algoritmo y esta definido el conjunto de vecindarios.

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

El tercer algoritmo es una combinación entre el algoritmo PSO y la búsqueda en vecindario variable. El algoritmo PSO presenta muchos para parámetros que deben ser calibrados, la inadecuada calibración de estos puede llevar el proceso a una convergencia prematura. Con el fin de dar solución a este problema, se propone una modificación al algoritmo básico PSO introduciendo en la metodología un procedimiento basado en el algoritmo de búsqueda en vecindario variable. En este se controla el número de nodos del árbol de distancias a los cuales se les actualiza la posición.

La inclusión de este procedimiento en la metodología presenta como desventaja la desmejora en el tiempo computacional y como ventaja una mejora en la diversificación de la población. Lo anterior se logra dado que en cada iteración solo se calculan las velocidades de los nodos que hacen parte del vecindario de la partícula. Al igual que el VNS Básico si la nueva partícula no presenta mejora se cambia el vecindario a uno más grande, de lo contrario su vecindario cambia por la mínima vecindad definida (cambiar solo un nodo). La combinación de estos algoritmos mejora significativamente la calidad de la solución.

La figura 13 ilustra en un diagrama de flujo el algoritmo $A_{PSO+VNS}$. Como es expresada en la figura la función CalcularVelocidadSoloVecindario solo evalúa la ecuación (49) en los nodos que pertenezcan al vecindario actual de cada partícula.

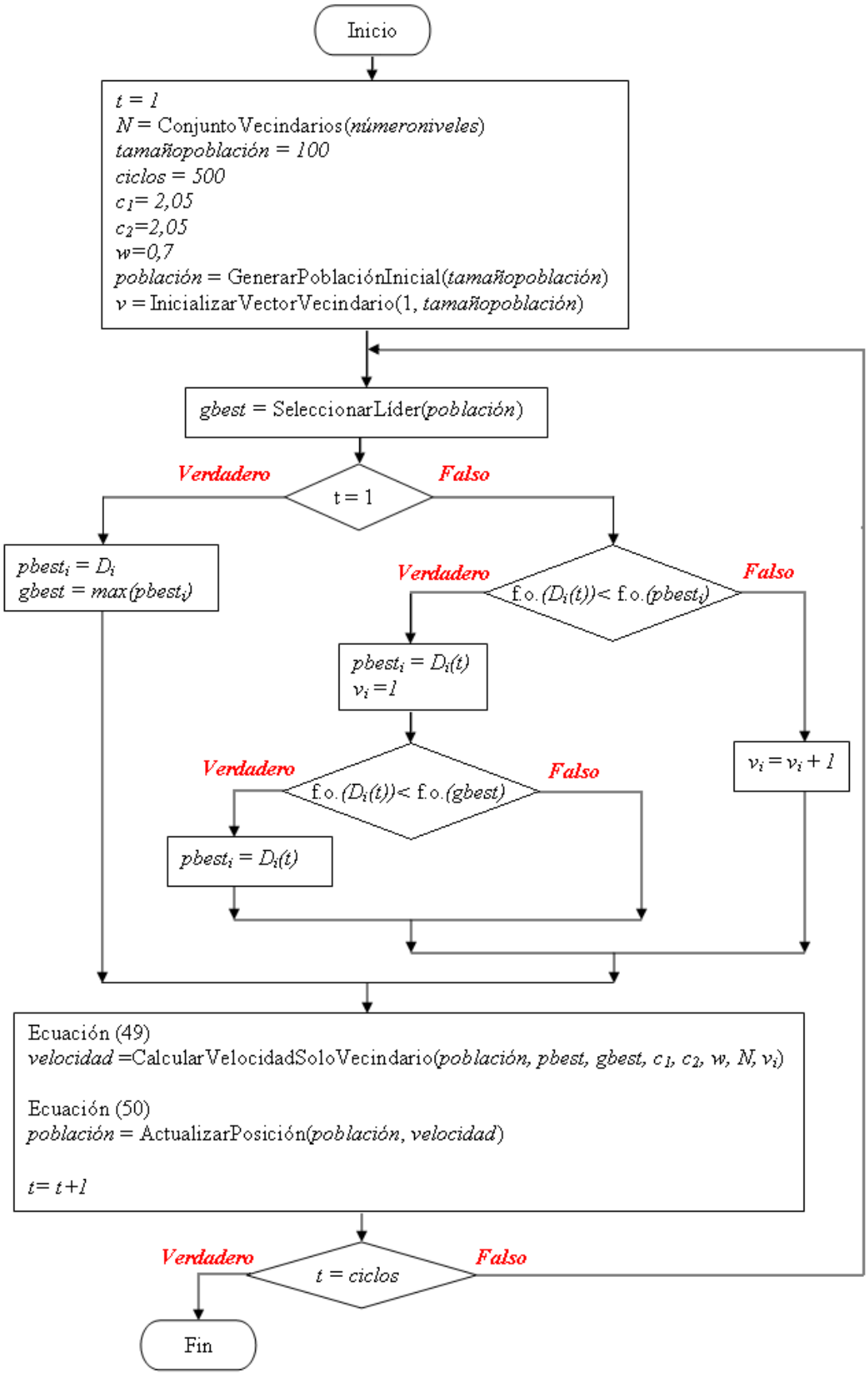

Figura 13. Algoritmo Híbrido PSO y Búsqueda en Vecindario Variable.

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

Este algoritmo de optimización combina las principales características de cúmulo de partículas, recocido simulado y algoritmos genéticos. El primero se considera como el algoritmo principal, el segundo y tercero se usan como mecanismo de perturbación especializado para realizar búsquedas locales, efectuando cambios en las posiciones de las partículas usando la filosofía de la mutación de los genéticos y la temperatura del recocido simulado.

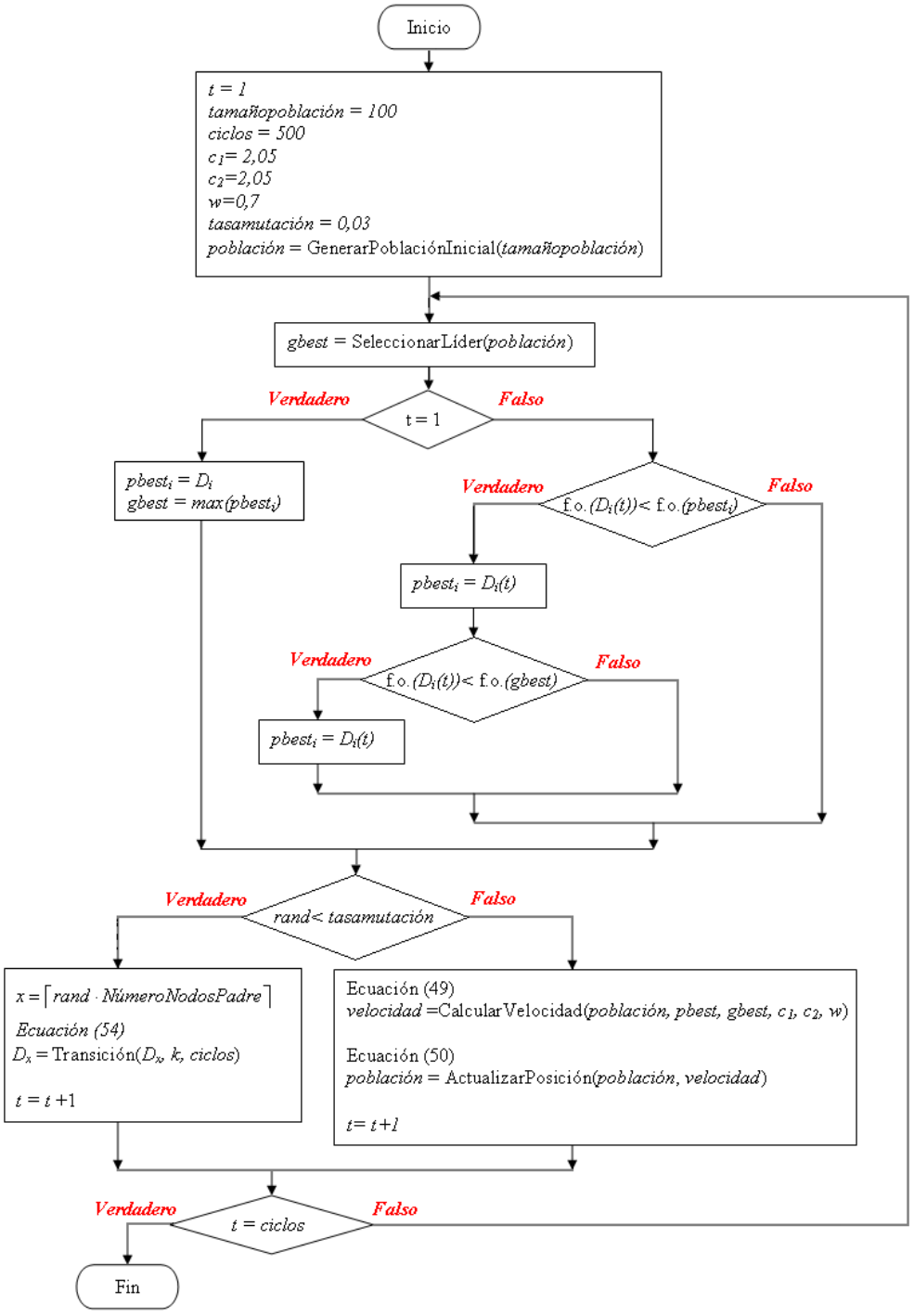

Figura 14. Algoritmo Híbrido PSO con el operador de mutación.

En este algoritmo se incluye el operador de mutación propio de los algoritmos genéticos [52] en el algoritmo PSO, donde la mutación se define como la modificación del valor de un nodo del árbol de distancias a través del mecanismo de transición de la ecuación (54).

Dado que el algoritmo PSO presenta algunas similitudes con los algoritmos evolutivos, diferentes autores han propuesto la inclusión del operador mutación en el algoritmo PSO [70, 71, 72, entre otras]. Como fue anteriormente mencionado estas operaciones híbridas normalmente se implementan en cada generación [70], [72] o en un intervalo prefijado [71] o son controladas por una función de adaptación definida para el caso específico, en este estudio en cada generación tiene la probabilidad de utilizar el operador de mutación.

La figura 14 ilustra el diagrama de flujo de datos del algoritmo $A_{PSO+Mutación}$.

5.7. Calibración de parámetros

Una etapa de gran importancia en el uso de técnicas metaheurísticas es el ajuste de parámetros, diferentes enfoques se presentan para abordarlo. Pepper et al. en [77] recaracterizan los parámetros del recocido simulado de forma tal que sin importar el tamaño del problema la selección de los parámetros es adecuada.

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

En este estudio son implementados 4 algoritmos de optimización, donde cada uno tiene sus propios parámetros. La tabla 8 resume los parámetros de cada algoritmo, obsérvese la gran cantidad de parámetros necesarios por cada algoritmo.

El algoritmo A_{VNS+RS} presenta pocos parámetros, en este la función de transición dada por la ecuación (54) puede ser modificada y puede ser vista también como parámetro.

El parámetro número de niveles para los problemas de *bin packing* y *cutting stock*, es fijado en *3*, distinto para los problemas de *strip packing* donde es fijado mediante la ecuaciones (52).

Para realizar la parametrización distintos estudios proponen: clasificar los problemas de prueba (si existen) por complejidad (matemática o computacional), escoger un problema representativo (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.

Algoritmo	Parámetros *Cutting Stock*	Parámetros *Strip Packing*	Parámetros *Bin Packing*
A_{PSO}	Tamaño de la Población Número de Ciclos c1 (Conocimiento Individual) c2(Conocimiento Grupal) w (Inercia) Número de Niveles	Tamaño de la Población Número de Ciclos c1 c2 w (Inercia) Número de *Bins* Número de Niveles	Tamaño de la Población Número de Ciclos c1 c2 w (Inercia) Número de Niveles
A_{VNS+RS}	Iteraciones Totales Número de Niveles	Iteraciones Totales Número de *Bins* Número de Niveles	Iteraciones Totales Número de Niveles
$A_{PSO+VNS}$	Tamaño de la Población Número de Ciclos c1 (Conocimiento Individual) c2(Conocimiento Grupal) w (Inercia) Número de Niveles	Tamaño de la Población Número de Ciclos c1 c2 w (Inercia) Número de *Bins* Número de Niveles	Tamaño de la Población Número de Ciclos c1 c2 w (Inercia) Número de Niveles
$A_{PSO+Mutación}$	Tamaño de la Población Número de Ciclos c1 (Conocimiento Individual) c2(Conocimiento Grupal) w (Inercia) Tasa de Mutación Número de Niveles	Tamaño de la Población Número de Ciclos c1 c2 w (Inercia) Tasa de Mutación Número de *Bins* Número de Niveles	Tamaño de la Población Número de Ciclos c1 c2 w (Inercia) Tasa de Mutación Número de Niveles

Tabla 8. Parámetros de los algoritmos implementados.

Este proceso requiere un gran esfuerzo computacional dado que el ajuste de parámetros a través de una búsqueda en malla representa otro proceso de optimización de complejidad similar al problema de este estudio, dado que cada parámetro pertenece a un gran rango de valores. Por otro lado, Zhi-Hui et al. en [65] 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.

Se ajustan primero los 5 parámetros del PSO ya que son comunes en 3 algoritmos para todos los problemas, obteniendo los valores de los parámetros tamaño de la población, número de ciclos, c_1 (conocimiento individual), c_2 (conocimiento grupal) y w (inercia).

Por último se calibran los parámetros número de iteraciones totales del algoritmo A_{VNS+RS} y tasa de mutación del algoritmo $A_{PSO+Mutación}$. El parámetro número de iteraciones totales se calibra con base en los valores de tamaño de la población y número de ciclos. Conservando 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, el parámetro tasa de mutación es calibrado en los rangos propuestos en [52]. Los valores resultantes de la calibración de parámetros son ilustrados en la tabla 9.

Algoritmo	Parámetros	Valor *Cutting Stock*	Valor *Strip Packing*	Valor *Bin Packing*
A_{PSO}	Tamaño de la Población	100	80	100
	Número de Ciclos	500	80	100
	c1 (Conocimiento Individual)	2,05	2,05	2,05
	c2(Conocimiento Grupal)	2,05	2,05	2,05
	w (Inercia)	0,7	0,7	0,6
	Número de Niveles	3	3	3
	Número de *Bins*		Ecuación (52)	
A_{VNS+RS}	Iteraciones Totales	50000	6000	10000
	Número de Niveles	3	3	3
	Número de *Bins*		Ecuación (52)	
$A_{PSO+VNS}$	Tamaño de la Población	100	80	100
	Número de Ciclos	500	80	100
	c1 (Conocimiento Individual)	2,05	2,05	2,05
	c2(Conocimiento Grupal)	2,05	2,05	2,05
	w (Inercia)	0,7	0,7	0,6
	Número de Niveles	3	3	3
	Número de *Bins*		Ecuación (52)	
$A_{PSO+Mutación}$	Tamaño de la Población	100	80	100
	Número de Ciclos	500	80	100
	c1 (Conocimiento Individual)	2,05	2,05	2,05
	c2(Conocimiento Grupal)	2,05	2,05	2,05
	w (Inercia)	0,7	0,7	0,6
	Tasa de Mutación	0,03	0,03	0,03
	Número de Niveles	3	3	3
	Número de *Bins*		Ecuación (52)	

Tabla 9. Valores de los parámetros de cada algoritmo.

5.7. Resumen

En este capítulo se presentó la codificación, algoritmos y la parametrización de los algoritmos que conforman la metodología de solución propuesta en este trabajo. El uso de una codificación en árbol de cortes, el cálculo de la función objetivo utilizando la heurística *best-fit*, la selección de tres técnicas de optimización para generar 4 algoritmos diferentes de optimización y un proceso cuidadoso de calibración de parámetros de los algoritmos representan la metodología de solución de los diferentes problemas de empaquetamiento propuestos en este estudio.

La estructura de datos tipo árbol de cortes garantiza las restricciones de corte tipo guillotina y presenta un escenario propicio para las técnicas metaheurísticas de optimización propuestas. Una rutina es la encargada de realizar la localización de las piezas sobre las hojas de material, además de medir la calidad de la solución propuesta calculando la función objetivo. El cálculo de la función objetivo utilizando la heurística *best-fit* permite agrupar las piezas mediante el criterio de igualdad de sus formas, lo que potencia el uso del árbol de cortes.

En este estudio se proponen cuatro algoritmos (A_{PSO}, A_{VNS+RS}, $A_{PSO+VNS}$, $A_{PSO+Mutación}$) para optimizar el árbol de distancias de los cortes, inspirados en tres técnicas metaheurísticas (VNS, RS y PSO). Con el fin de medir la eficiencia de la metodología propuesta, se realiza un estudio computacional usando bases de datos de prueba de la literatura especializada.

Calibrar los parámetros de las técnicas metaheurísticas es un paso esencial dado que 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 en rangos de valores sugeridos en la literatura especializada.

Un análisis estadístico permite concluir cuales son los algoritmos con mejor desempeño.

6. ANÁLISIS DE RESULTADOS

6.1. Introducción

Diferentes análisis se realizan a las metodologías de solución propuestas. Análisis de complejidad matemática son aplicados a metodologías. En este estudio se realiza un análisis estadístico entorno a la calidad de la solución y tiempos de respuesta de cada algoritmo propuesto.

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 descritas a continuación.

Fueron seleccionados 50 casos de prueba para el problema de empaquetamiento óptimo bidimensional guillotinado en una sola placa. Estos casos presentan 20 tipos de placas diferentes con distribuciones entre 88 y 139 piezas, la base de datos es presentada por el profesor Cui en [84] y disponible en línea en [78]. Los 50 casos pertenecen a la categoría de problemas de empaquetamiento de gran escala. Diferentes estudios han utilizado estos casos de prueba para realizar una especie de benchmark de las metodologías propuestas, una descripción más detallada del desarrollo de los casos de prueba es presentada en [78].

Para el problema de empaquetamiento óptimo bidimensional guillotinado en placas, fueron seleccionados en total 145 casos de prueba de tres diferentes librerías, 3 casos de prueba clásicos presentado por Christofides y Whitlock en [8], 12 casos de prueba propuestos por Beasley en [10] y 130 casos de prueba tomados de la gran librería presentada por Lodi et al en [6] y disponible en línea en [93], en esta se encuentra 500 casos divididos en 10 clases y agrupados por el número de piezas en 5 grupos (20, 40, 60, 80 y 100 piezas), con un tamaño de 10 casos de prueba por grupo. En este estudio fueron seleccionados 13 grupos aleatorios para un total de 130 casos de prueba. Obteniendo en total 20 tipos de placas diferentes con distribuciones entre 8 y 100 piezas. En resumen 70 casos de prueba pertenecen a la categoría de problemas de empaquetamiento de gran escala, mientras los 75 restantes pertenecen a las categorías de pequeña y mediana escala. Diferentes estudios han utilizado estos casos de prueba para realizar sus análisis en una especie de benchmark de las metodologías propuestas, una descripción más detallada del desarrollo de los casos de prueba es presentada en [93].

Además de los casos anteriores, también fueron seleccionados 34 casos de estudio del problema de empaquetamiento óptimo bidimensional guillotinado en rollos, los primeros 25 casos de estudio presentan distribuciones entre 7 y 22 piezas considerados de complejidad media y los 9 restantes presentan distribuciones entre 56 y 198 piezas considerados de alta complejidad matemática. Esta librería de casos de prueba es presentada por el profesor Hifi en [31] y [79] y disponible en línea en [80]. El primer conjunto de casos pertenece a la categoría de problemas de empaquetamiento de mediana escala, mientras el segundo conjunto pertenece a gran escala. Diferentes estudios han utilizado estos casos de prueba para realizar sus análisis en 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 Delphi 7,0 ®, sobre un ordenador con unas especificaciones de un procesador Pentium 4 de 3,0 GHz y una memoria RAM de 1 GB.

El software desarrollado visualiza los resultados de dos formas: en versión tabla (distancias exactas en la placa o el rollo de material donde se deben realizar los cortes para obtener las piezas), dado que esta metodología puede ser la rutina que presenta las distancias y orientaciones de los cortes al microcontrolador de la máquina corte (guillotina) y una versión gráfica, donde simplemente con la figura de la solución un experto puede realizar el corte.

Dada la metodología propuesta todos los algoritmos son altamente paralelizables lo que reduciría los tiempos de cómputo, sin aumentar los costos de implementación, ya que los algoritmos no requieren de grandes especificaciones de las máquinas y la conformación de un cluster de máquinas de bajo desempeño es un proyecto viable para la paralelización de los algoritmos.

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 a continuación para todos los casos de prueba de cada problema la mejor solución reportada en la literatura especializada con su respectivo autor: las tablas 10 y

11 presentan la mejor solución reportada para los problemas de *cutting stock* sin y con rotación respectivamente, por otro lado, las tablas 12 y 13 presentan la mejor solución reportada para los problemas de *strip packing* sin y con rotación y finalmente las tablas 14 y 15 presentan la mejor solución reportada para los problemas de *bin packing* sin y con rotación respectivamente. Es importante aclarar que si una respuesta es alcanzada entre diferentes autores, la primera respuesta publicada es la que se enumera en las tablas.

Caso	Solución	Autor	Caso	Solución	Autor
1	99,43	Álvarez et al.[81]	26	98,05	Álvarez et al. [81]
2	98,17	Toro et al. [23]	27	98,31	Álvarez et al. [81]
3	98,92	Álvarez et al.	28	98,51	Álvarez et al. [81]
4	98,53	Toro et al. [23]	29	98,39	Álvarez et al. [81]
5	98,36	Cui [84]	30	98,27	Cui [84]
6	98,49	Toro et al. [23]	31	97,78	Toro et al. [23]
7	98,91	Toro et al. [23]	32	97,57	Álvarez et al. [81]
8	98,9	Toro et al. [23]	33	98,31	Toro et al. [23]
9	98,53	Álvarez et al. [81]	34	98,09	Álvarez et al. [81]
10	98,64	Álvarez et al. [81]	35	97,31	Álvarez et al. [81]
11	97,98	Álvarez et al. [81]	36	97,52	Álvarez et al. [81]
12	97,79	Toro et al. [23]	37	97,36	Álvarez et al. [81]
13	98,91	Toro et al. [23]	38	98,08	Álvarez et al. [81]
14	96,98	Álvarez et al. [81]	39	98,98	Álvarez et al. [81]
15	98,71	Álvarez et al. [81]	40	98,21	Álvarez et al. [81]
16	98,27	Toro et al. [23]	41	98,84	Cui [84]
17	97,14	Álvarez et al. [81]	42	98,3	Toro et al. [23]
18	97,78	Álvarez et al. [81]	43	97,41	Toro et al. [23]
19	99,19	Álvarez et al. [81]	44	97,33	Toro et al. [23]
20	96,61	Toro et al. [23]	45	96,83	Cui [84]
21	98,2	Cui [84]	46	97,63	Toro et al. [23]
22	96,32	Toro et al.	47	97,29	Álvarez et al. [81]
23	97,73	Álvarez et al. [81]	48	98,76	Álvarez et al. [81]
24	95,98	Álvarez et al. [81]	49	98,26	Cui [84]
25	98,27	Toro et al. [23]	50	98,22	Toro et al. [23]

Tabla 10. Mejor solución reportada para *bin packing* 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 utilizo 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 (54) define formalmente el error porcentual. Los descriptores estadísticos de la calidad de la solución de cada algoritmo son la media del error (error medio) y la desviación estándar del error. El error medio se calcula como el error promedio de la muestra y la desviación estándar del error se define como la dispersión de los valores respecto al error medio.

$$error = \frac{MejorSoluciónReportada - SoluciónObtenida}{MejorSoluciónReportada} \qquad (54)$$

Caso	Solución	Autor	Caso	Solución	Autor
1	99,43	Álvarez et al.[83]	26	98,32	Álvarez et al.[82]
2	98,55	Álvarez et al.[82]	27	98,31	Álvarez et al.[83]
3	98,92	Álvarez et al.[83]	28	98,9	Cui [84]
4	98,53	Álvarez et al.[82]	29	99,45	Cui [84]
5	98,61	Álvarez et al.[82]	30	98,36	Álvarez et al.[82]
6	99,43	Álvarez et al.[82]	31	98,42	Álvarez et al.[82]
7	99,06	Álvarez et al.[82]	32	98,67	Álvarez et al.[83]
8	98,93	Álvarez et al.[82]	33	99,3	Cui [84]
9	98,54	Álvarez et al.[82]	34	99,2	Cui [84]
10	99,07	Cui [84]	35	98,39	Cui [84]
11	98,44	Cui [84]	36	98,37	Álvarez et al.[82]
12	98,54	Álvarez et al.[83]	37	98,26	Cui [84]
13	98,91	Álvarez et al.[82]	38	98,98	Álvarez et al.[83]
14	98,14	Cui [84]	39	98,98	Álvarez et al.[83]
15	99,15	Álvarez et al.[82]	40	98,28	Cui [84]
16	98,75	Cui [84]	41	99,4	Álvarez et al.[82]
17	98,33	Cui [84]	42	99,17	Cui [84]
18	98,44	Cui [84]	43	98,34	Cui [84]
19	99,19	Álvarez et al.[83]	44	98,2	Álvarez et al.[82]
20	97,17	Álvarez et al.[82]	45	98,73	Cui [84]
21	98,78	Cui [84]	46	98,13	Álvarez et al.[82]
22	97,4	Cui [84]	47	98,17	Álvarez et al.[83]
23	98,81	Álvarez et al.[82]	48	98,76	Álvarez et al.[83]
24	97,45	Cui [84]	49	98,85	Cui [84]
25	98,49	Álvarez et al.[83]	50	98,43	Álvarez et al.[82]

Tabla 11. Mejor solución reportada para *bin packing* con rotación.

Caso	Solución	Autor	Caso	Solución	Autor
SCP 1	13	Hifi [31]	SCP 18	101	Hifi [31]
SCP 2	40	Hifi [31]	SCP 19	26	Hifi [31]
SCP 3	14	Hifi [31]	SCP 20	21	Hifi [31]
SCP 4	20	Hifi [31]	SCP 21	145	Hifi [31]
SCP 5	20	Hifi [31]	SCP 22	34	Hifi [31]
SCP 6	36	Álvarez et al.[85]	SCP 23	34	Álvarez et al. [85]
SCP 7	14	Hifi [31]	SCP 24	114	Hifi [31]
SCP 8	17	Hifi [31]	SCP 25	36	Hifi [31]
SCP 9	68	Hifi [31]	SCP L1	54	Bekrar y Kacem [33]
SCP 10	80	Hifi [31]	SCP L2	154	Bekrar y Kacem [33]
SCP 11	48	Hifi [31]	SCP L3	178	Bekrar y Kacem [33]
SCP 12	34	Hifi [31]	SCP L4	66	Álvarez et al. [85]
SCP 13	49	Álvarez et al. [85]	SCP L5	140	Bekrar y Kacem [33]
SCP 14	68	Álvarez et al. [85]	SCP L6	22	Bekrar y Kacem [33]
SCP 15	34	Hifi [31]	SCP L7	125	Álvarez et al. [85]
SCP 16	33	Hifi [31]	SCP L8	71	Bekrar y Kacem [33]
SCP 17	38	Álvarez et al. [85]	SCP L9	99	Bekrar y Kacem [33]

Tabla 12. Mejor solución reportada para *strip packing* sin rotación.

Caso	Solución	Autor	Caso	Solución	Autor
SCP 1	13	Hifi [31]	SCP 18	101	Hifi [31]
SCP 2	40	Hifi [31]	SCP 19	26	Hifi [31]
SCP 3	14	Hifi [31]	SCP 20	21	Hifi [31]
SCP 4	20	Hifi [31]	SCP 21	145	Hifi [31]
SCP 5	11	Álvarez et al. [84]	SCP 22	34	Hifi [31]
SCP 6	35	Álvarez et al. [84]	SCP 23	32	Álvarez et al. [84]
SCP 7	12	Álvarez et al. [84]	SCP 24	114	Hifi [31]
SCP 8	17	Hifi [31]	SCP 25	36	Hifi [31]
SCP 9	68	Hifi [31]	SCP L1	54	Bekrar y Kacem [33]
SCP 10	80	Hifi [31]	SCP L2	154	Bekrar y Kacem [33]
SCP 11	48	Hifi [31]	SCP L3	178	Bekrar y Kacem [33]
SCP 12	34	Hifi [31]	SCP L4	66	Álvarez et al. [84]
SCP 13	48	Álvarez et al. [84]	SCP L5	140	Bekrar y Kacem [33]
SCP 14	67	Álvarez et al. [84]	SCP L6	22	Bekrar y Kacem [33]
SCP 15	34	Hifi [31]	SCP L7	125	Álvarez et al. [84]
SCP 16	33	Hifi [31]	SCP L8	71	Bekrar y Kacem [33]
SCP 17	39	Hifi [31]	SCP L9	99	Bekrar y Kacem [33]

Tabla 13. Mejor solución reportada para *strip packing* con rotación.

Caso	Solución	Autor	Casos Lodi et al.		Solución	Autor
			Clase	Número Piezas	*LowerBound*	
CGCUT1	2	Lodi et al. [6]	1	100	1,028	Bekrar y Kacem [33]
CGCUT2	2	Lodi et al. [6]	3	20	1,120	Bekrar y Kacem [33]
CGCUT3	23	Lodi et al. [6]	3	100	1,086	Bekrar y Kacem [33]
NGCUT1	3	Martello y Vigo [94]	5	20	1,040	Bekrar y Kacem [33]
NGCUT2	4	Martello y Vigo [94]	5	40	1,050	Bekrar y Kacem [33]
NGCUT3	3	Martello y Vigo [94]	5	100	1,086	Bekrar y Kacem [33]
NGCUT4	2	Martello y Vigo [94]	7	40	1,066	Bekrar y Kacem [33]
NGCUT5	3	Martello y Vigo [94]	7	100	1,036	Bekrar y Kacem [33]
NGCUT6	3	Martello y Vigo [94]	8	20	1,056	Bekrar y Kacem [33]
NGCUT7	1	Martello y Vigo [94]	8	100	1,040	Lodi et al. [6]
NGCUT8	2	Martello y Vigo [94]	9	60	1,002	Bekrar y Kacem [33]
NGCUT9	3	Martello y Vigo [94]	9	100	1,000	Bekrar y Kacem [33]
NGCUT10	3	Martello y Vigo [94]	10	40	1,056	Bekrar y Kacem [33]
NGCUT11	2	Martello y Vigo [94]				
NGCUT12	4	Martello y Vigo [94]				

Tabla 14. Mejor solución reportada para *bin packing* sin rotación.

Caso	Solución	Autor	Casos Lodi et al.		Solución	Autor
			Clase	Número Piezas	*LowerBound*	
CGCUT1	2	Lodi et al. [6]	3	60	1,07	Lodi et al. [6]
CGCUT2	2	Lodi et al. [6]	7	20	1,08	Lodi et al. [6]
CGCUT3	23	Lodi et al. [6]	7	60	1,05	Lodi et al. [6]
NGCUT1	3	Martello y Vigo [94]	7	80	1,05	Lodi et al. [6]
NGCUT2	4	Martello y Vigo [94]	8	40	1,04	Lodi et al. [6]
NGCUT3	3	Martello y Vigo [94]	8	60	1,03	Lodi et al. [6]
NGCUT4	2	Martello y Vigo [94]	8	80	1,03	Lodi et al. [6]
NGCUT5	3	Martello y Vigo [94]	8	100	1,04	Lodi et al. [6]
NGCUT6	3	Martello y Vigo [94]	9	20	1,00	Lodi et al. [6]
NGCUT7	1	Martello y Vigo [94]	9	40	1,01	Lodi et al. [6]
NGCUT8	2	Martello y Vigo [94]	9	60	1,01	Lodi et al. [6]
NGCUT9	3	Martello y Vigo [94]	9	80	1,01	Lodi et al. [6]
NGCUT10	3	Martello y Vigo [94]	9	100	1,01	Lodi et al. [6]
NGCUT11	2	Martello y Vigo [94]				
NGCUT12	4	Martello y Vigo [94]				

Tabla 15. Mejor solución reportada para *bin packing* con rotación.

Algoritmo	Error Medio	Desviación Estándar
A_{PSO}	0,0104	0,00510
A_{VNS+RS}	0,0105	0,00589
$A_{PSO+VNS}$	0,0131	0,00534
A_{PSO+RS}	0,0118	0,00500

Tabla 16. Error medio y desviación estándar para el *cutting stock* sin rotación

Algoritmo	Error Medio	Desviación Estándar
A_{PSO}	0,0098	0,00604
A_{VNS+RS}	0,0065	0,00377
$A_{PSO+VNS}$	0,0178	0,00370
A_{PSO+RS}	0,0132	0,00320

Tabla 17. Error medio y desviación estándar para el *cutting stock* rotación

Algoritmo	Error Medio	Desviación Estándar
A_{PSO}	0,1402	0,01572
A_{VNS+RS}	0,1373	0,00995
$A_{PSO+VNS}$	0,1442	0,02048
A_{PSO+RS}	0,1436	0,02028

Tabla 18. Error medio y desviación estándar para el *strip packing* sin rotación

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). Las tablas 16, 17, 18, 19, 20, y 21 resumen los errores medios y las desviaciones estándar para cada algoritmo en su respectivo problema. Mientras las tablas 22, 23, 24, 25, 26 y 27 presentan el mejor valor alcanzado en cada algoritmo.

Algoritmo	Error Medio	Desviación Estándar
A_{PSO}	0,1355	0,01921
A_{VNS+RS}	0,1377	0,02157
$A_{PSO+VNS}$	0,1388	0,02275
A_{PSO+RS}	0,1482	0,02410

Tabla 19. Error medio y desviación estándar para el *strip packing* con rotación

Algoritmo	Error Medio	Desviación Estándar
A_{PSO}	0,0701	0,02368
A_{VNS+RS}	0,0686	0,01592
$A_{PSO+VNS}$	0,0721	0,03082
A_{PSO+RS}	0,0718	0,03042

Tabla 20. Error medio y desviación estándar para el *bin packing* sin rotación

Algoritmo	Error Medio	Desviación Estándar
A_{PSO}	0,0162	0,00121
A_{VNS+RS}	0,0109	0,00751
$A_{PSO+VNS}$	0,0210	0,00740
A_{PSO+RS}	0,0230	0,00641

Tabla 21. Error medio y desviación estándar para el *bin packing* con rotación

Caso	Solución	Algoritmo	Caso	Solución	Algoritmo
1	99,43 [a]	PSO	26	98,05 [a]	PSO
2	98,46 [c]	PSO+VNS	27	98,31 [a]	PSO+RS
3	99,06 [c]	VNS+RS	28	98,74 [c]	PSO+VNS
4	98,53 [a]	PSO	29	98,39 [a]	PSO
5	98,36 [d]	PSO+VNS	30	98,27 [d]	PSO
6	98,49 [d]	VNS+RS	31	97,81 [c]	PSO+VNS
7	98,91 [d]	PSO+RS	32	98,23 [c]	VNS+RS
8	99,03 [c]	VNS+RS	33	99,46 [c]	VNS+RS
9	99,23 [c]	VNS+RS	34	98,36 [c]	VNS+RS
10	98,64 [a]	PSO	35	97,94 [c]	PSO+VNS
11	98,09 [c]	VNS+RS	36	97,57 [c]	VNS+RS
12	97,79 [d]	PSO	37	97,36 [a]	PSO
13	98,91 [d]	PSO	38	98,56 [c]	VNS+RS
14	96,98 [a]	VNS+RS	39	98,98 [a]	PSO
15	98,71 [a]	PSO	40	98,21 [a]	PSO
16	98,35 [c]	VNS+RS	41	98,92 [c]	VNS+RS
17	97,14 [a]	PSO	42	98,64 [c]	VNS+RS
18	97,78 [a]	PSO	43	97,41 [d]	PSO+RS
19	99,19 [a]	PSO	44	97,36 [c]	PSO+RS
20	96,61 [d]	VNS+RS	45	96,83 [d]	PSO+RS
21	98,25 [c]	VNS+RS	46	97,63 [c]	VNS+RS
22	97,40 [c]	VNS+RS	47	97,30 [c]	VNS+RS
23	98,19 [c]	PSO+VNS	48	98,76 [a]	PSO
24	95,98 [a]	PSO	49	98,26 [d]	VNS+RS
25	98,27 [d]	VNS+RS	50	98,22 [a]	PSO

Tabla 22. Mejor solución *cutting stock* sin rotación

Caso	Solución	Algoritmo	Caso	Solución	Algoritmo
1	99,43 [a]	PSO	26	98,32 [a]	VNS+RS
2	98,93 [c]	VNS+RS	27	98,53 [c]	VNS+RS
3	99,06 [c]	VNS+RS	28	98,80 [d]	VNS+RS
4	98,53 [a]	PSO	29	98,98 [d]	VNS+RS
5	99,19 [c]	VNS+RS	30	99,18 [c]	VNS+RS
6	99,43 [a]	PSO	31	98,42 [a]	PSO
7	99,06 [a]	PSO	32	98,67 [c]	PSO
8	99,08 [c]	VNS+RS	33	98,94 [d]	VNS+RS
9	99,05 [c]	PSO+RS	34	98,95 [d]	VNS+RS
10	99,44 [c]	VNS+RS	35	98,22 [d]	VNS+RS
11	98,86 [c]	VNS+RS	36	98,71 [c]	VNS+RS
12	98,85 [c]	VNS+RS	37	98,38 [c]	VNS+RS
13	99,29 [c]	VNS+RS	38	99,27 [c]	VNS+RS
14	97,14 [d]	VNS+RS	39	99,07 [c]	VNS+RS
15	99,15 [a]	PSO	40	98,64 [c]	VNS+RS
16	98,75 [c]	VNS+RS	41	99,40 [a]	PSO
17	98,43 [c]	VNS+RS	42	99,08 [d]	PSO
18	98,49 [c]	VNS+RS	43	98,08 [d]	VNS+RS
19	99,45 [c]	VNS+RS	44	98,50 [c]	VNS+RS
20	97,17 [a]	PSO	45	98,66 [d]	VNS+RS
21	98,94 [c]	VNS+RS	46	98,37 [c]	VNS+RS
22	98,33 [c]	VNS+RS	47	98,87 [c]	VNS+RS
23	99,05 [c]	VNS+RS	48	98,76 [a]	PSO
24	96,80 [d]	VNS+RS	49	98,99 [c]	VNS+RS
25	98,49 [c]	PSO	50	98,43 [a]	PSO+RS

Tabla 23. Mejor solución *cutting stock* con rotación.

Caso	Solución	Algoritmo	Caso	Solución	Algoritmo
SCP 1	13 [b]	Todos	SCP 18	104 [d]	Todos
SCP 2	40 [b]	Todos	SCP 19	26 [b]	PSO; VNS+RS
SCP 3	15 [d]	Todos	SCP 20	21 [b]	PSO; VNS+RS
SCP 4	20 [b]	PSO; VNS+RS	SCP 21	145 [b]	PSO; VNS+RS
SCP 5	20 [b]	PSO; VNS+RS	SCP 22	39 [d]	PSO; VNS+RS
SCP 6	36 [a]	PSO; VNS+RS	SCP 23	34 [a]	PSO; VNS+RS
SCP 7	14 [b]	PSO; VNS+RS	SCP 24	130 [d]	Todos
SCP 8	17 [b]	PSO; VNS+RS	SCP 25	37 [d]	PSO; VNS+RS
SCP 9	73 [d]	Todos	SCP L1	55 [d]	Todos
SCP 10	80 [b]	PSO; VNS+RS	SCP L2	180 [d]	PSO; VNS+RS
SCP 11	55 [d]	Todos	SCP L3	178 [b]	PSO; VNS+RS
SCP 12	37 [d]	PSO; VNS+RS	SCP L4	66 [a]	PSO; VNS+RS
SCP 13	49 [a]	PSO; VNS+RS	SCP L5	146 [d]	PSO; VNS+RS
SCP 14	68 [a]	PSO; VNS+RS	SCP L6	24 [d]	PSO; VNS+RS
SCP 15	38 [d]	Todos	SCP L7	125 [a]	PSO; VNS+RS
SCP 16	37 [d]	PSO; VNS+RS	SCP L8	76 [d]	Todos
SCP 17	39 [b]	PSO; VNS+RS	SCP L9	101 [d]	PSO; VNS+RS

Tabla 24. Mejor solución *strip packing* sin rotación.

Caso	Solución	Algoritmo	Caso	Solución	Algoritmo
SCP 1	13 [b]	Todos	SCP 18	101 [b]	Todos
SCP 2	40 [b]	Todos	SCP 19	26 [b]	PSO; VNS+RS
SCP 3	14 [b]	Todos	SCP 20	21 [b]	PSO; VNS+RS
SCP 4	20 [b]	Todos	SCP 21	139 [c]	PSO; VNS+RS
SCP 5	11 [a]	PSO; VNS+RS	SCP 22	38 [d]	PSO; VNS+RS
SCP 6	35 [a]	PSO; VNS+RS	SCP 23	32 [a]	PSO; VNS+RS
SCP 7	12 [a]	PSO; VNS+RS	SCP 24	120 [d]	Todos
SCP 8	17 [b]	Todos	SCP 25	37 [d]	PSO; VNS+RS
SCP 9	69 [d]	Todos	SCP L1	55 [d]	Todos
SCP 10	80 [b]	Todos	SCP L2	178 [d]	Todos
SCP 11	55 [d]	Todos	SCP L3	178 [b]	PSO; VNS+RS
SCP 12	37 [d]	Todos	SCP L4	66 [a]	PSO; VNS+RS
SCP 13	48 [a]	PSO; VNS+RS	SCP L5	140 [b]	PSO; VNS+RS
SCP 14	67 [a]	PSO; VNS+RS	SCP L6	23 [d]	PSO; VNS+RS
SCP 15	38 [d]	Todos	SCP L7	125 [a]	PSO; VNS+RS
SCP 16	37 [d]	Todos	SCP L8	71 [b]	VNS+RS
SCP 17	39 [b]	Todos	SCP L9	95 [c]	PSO; VNS+RS

Tabla 25. Mejor solución *strip packing* con rotación.

Caso	Solución	Algoritmo	Casos Lodi et al. [93]		Solución / *LowerBound*	Algoritmo
			Clase	Número Piezas		
CGCUT1	2 [b]	Todos	1	100	1,05 [d]	Todos
CGCUT2	2 [b]	Todos	3	20	1,18 [d]	Todos
CGCUT3	23 [b]	Todos	3	100	1,09 [d]	Todos
NGCUT1	3 [b]	Todos	5	20	1,13 [d]	Todos
NGCUT2	4 [b]	Todos	5	40	1,09 [d]	Todos
NGCUT3	3 [b]	Todos	5	100	1,09 [d]	Todos
NGCUT4	2 [b]	Todos	7	40	1,07 [d]	Todos
NGCUT5	3 [b]	Todos	7	100	1,04 [d]	Todos
NGCUT6	3 [b]	Todos	8	20	1,12 [d]	Todos
NGCUT7	1 [b]	Todos	8	100	1,04 [b]	Todos
NGCUT8	2 [b]	Todos	9	60	1,01 [d]	Todos
NGCUT9	3 [b]	Todos	9	100	1,01 [d]	Todos
NGCUT10	3 [b]	Todos	10	40	1,09 [d]	Todos
NGCUT11	2 [b]	Todos				
NGCUT12	4 [b]	Todos				

Tabla 26. Mejor solución *bin packing* sin rotación.

Caso	Solución	Algoritmo	Casos Lodi et al. [93]		Solución / *LowerBound*	Algoritmo
			Clase	Número Piezas		
CGCUT1	2 [b]	Todos	3	60	1,07 [c]	Todos
CGCUT2	2 [b]	Todos	7	20	1,08 [c]	Todos
CGCUT3	23 [b]	Todos	7	60	1,05 [c]	Todos
NGCUT1	3 [b]	Todos	7	80	1,05 [c]	Todos
NGCUT2	4 [b]	Todos	8	40	1,04 [c]	Todos
NGCUT3	3 [b]	Todos	8	60	1,03 [c]	Todos
NGCUT4	2 [b]	Todos	8	80	1,03 [c]	Todos
NGCUT5	3 [b]	Todos	8	100	1,04 [c]	Todos
NGCUT6	3 [b]	Todos	9	20	1,00 [b]	Todos
NGCUT7	1 [b]	Todos	9	40	1,01 [b]	Todos
NGCUT8	2 [b]	Todos	9	60	1,01 [b]	Todos
NGCUT9	3 [b]	Todos	9	80	1,01 [b]	Todos
NGCUT10	3 [b]	Todos	9	100	1,01 [b]	Todos
NGCUT11	2 [b]	Todos				
NGCUT12	4 [b]	Todos				

Tabla 27. Mejor solución *bin packing* con rotación

En las tablas 22, 23, 24, 25, 26 y 27 se utilizan diferentes colores para representar la calidad de las respuestas, en este capítulo se utiliza la siguiente codificación para leer los resultados:

El símbolo [a] representa una respuesta reportada por el autor.
El símbolo [b] representa una respuesta igual a la reportada en la literatura especializada.
El símbolo [c] representa una respuesta mejor que la reportada en la literatura especializada.
El símbolo [d] representa una respuesta que no supera a las reportadas en la literatura especializada.

Problema	Soluciones Superadas	Soluciones Igualadas	Soluciones no Superadas
Cutting Stock sin rotación	39	0	11
Cutting Stock con rotación	40	0	10
Strip packing sin rotación	6	12	16
Strip packing con rotación	10	13	11
Bin packing sin rotación	0	16	12
Bin packing con rotación	8	20	0

Tabla 28. Comparación de resultados de los mejores resultados.

La tabla 28 presenta un resumen de las tablas 22, 23, 24, 25, 26 y 27. Esta tabla compara el mejor valor encontrado por los algoritmos para cada instancia del problema con la mejor solución reportada. De esta comparación se puede dar que la mejor respuesta supere la mejor respuesta reportada, que no la supere o que la iguale.

Las figuras 15, 16 y 17 representan gráficamente el desempeño de los algoritmos en los diferentes problemas. En estas gráficas se ilustran el error medio de cada algoritmo para cada problema.

La tabla 29 presenta los tiempos computacionales requeridos por los algoritmos para cada problema. En esta se ilustran dos tipos de tiempos: totales y promedios por conjunto de problemas.

El tiempo total se 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 de una categoría (pequeña, mediana o gran escala) sobre el número total de casos de prueba.

Para los problemas de *cutting stock* solo se reportan dos tiempos dado que todos los casos de prueba pertenecen a la clase de problemas de gran escala, mientras, para los problemas de *strip packing* y *bin packing* se reportan tres tiempos dado que el conjunto de casos contiene dos clases: problemas de mediana y gran escala.

Las figuras 18, 19 y 20 ilustran los tiempos totales utilizados por los algoritmos para resolver cada problema.

Problema	**Tiempos**	**Algoritmo**			
		PSO	VNS+RS	PSO+VNS	PSO+RS
Cutting Stock sin rotación	Tiempo total	86.750	79.756	83.772	82.383
	Tiempo promedio (gran escala)	86	79	83	82
Cutting Stock con rotación	Tiempo total	180.530	147.437	157.394	158.689
	Tiempo promedio (gran escala)	180	147	157	158
StripPacking sin rotación	Tiempo total	545.142	445.210	491.316	495.359
	Tiempo promedio (mediana escala)	144	118	130	131
	Tiempo promedio (gran escala)	1.380	1.127	1.243	1.254
StripPacking con rotación	Tiempo total	874.798	714437	788424	794.912
	Tiempo promedio (mediana escala)	242	197	218	220
	Tiempo promedio (gran escala)	2.185	1.784	1.969	1.986
Bin Packing sin rotación	Tiempo total	1.626.000	1.433.700	1.527.300	1.536.000
	Tiempo promedio (pequeña y mediana escala)	132	113	125	128
	Tiempo promedio (gran escala)	1.020	903	957	960
Bin Packing con rotación	Tiempo total	2.983.486	2.630.642	2.802.385	2.818.349
	Tiempo promedio (pequeña y mediana escala)	242	207	229	235
	Tiempo promedio (gran escala)	1.872	1.657	1.756	1.761

Tabla 29. Tiempos utilizados por cada algoritmo (segundos).

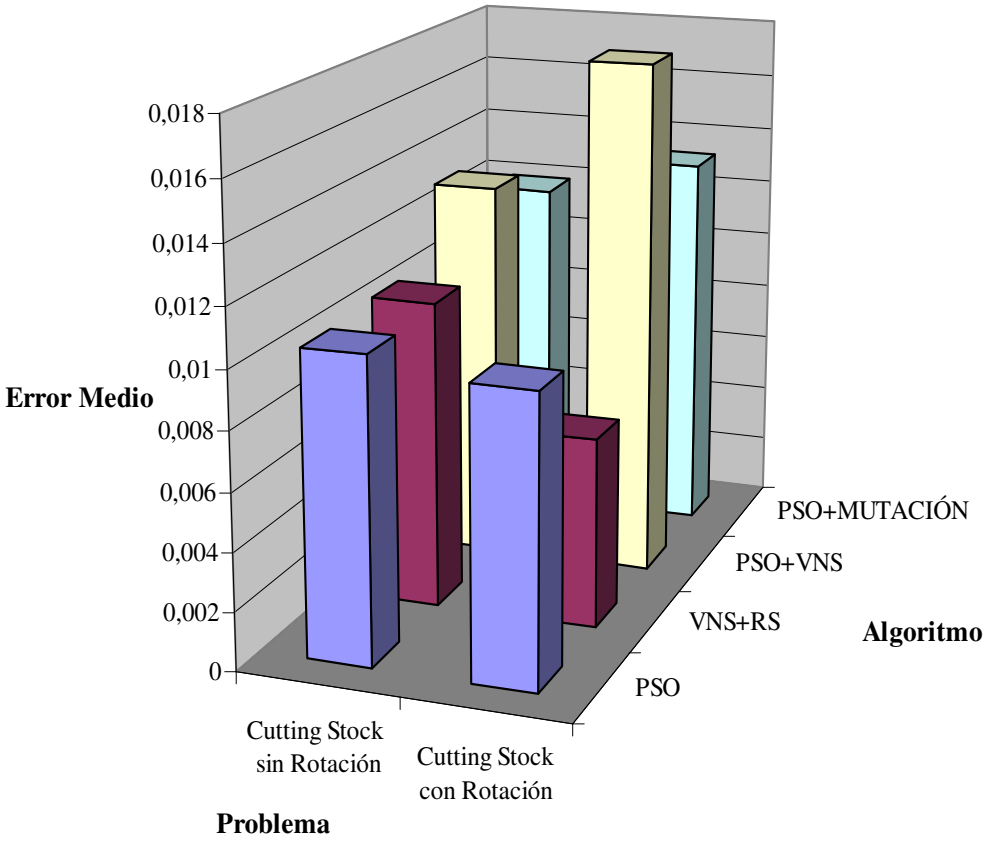

Figura 15. Gráfica error medio para los problemas de *cutting stock.*

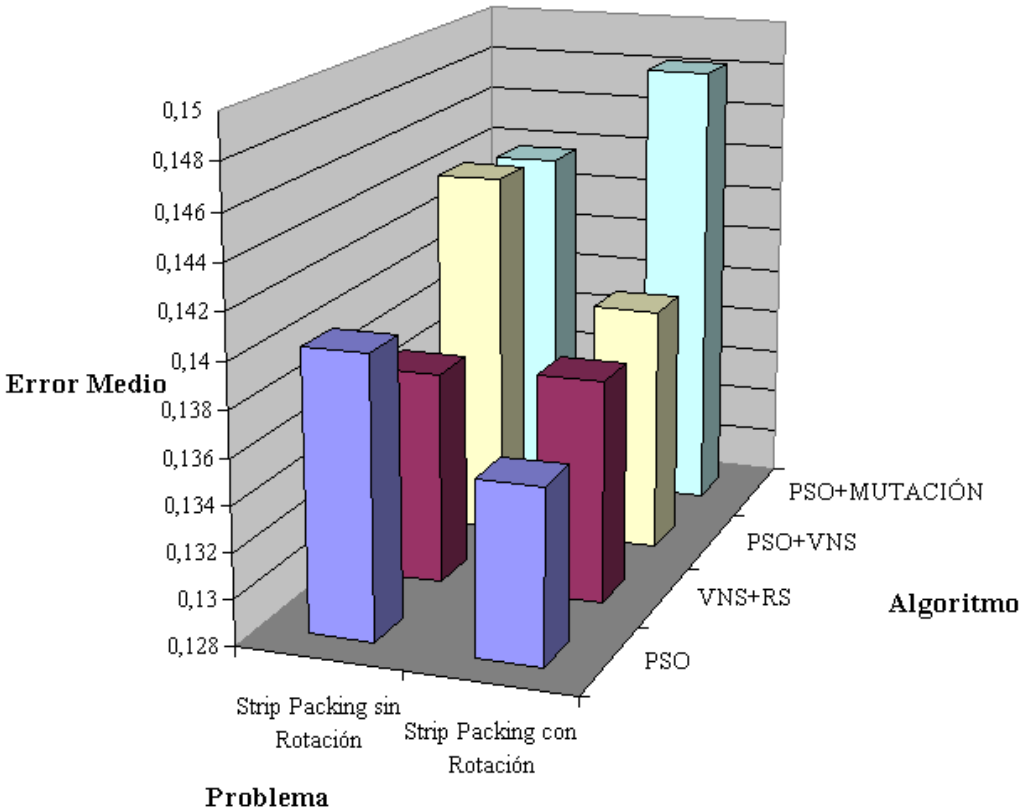

Figura 16. Gráfica error medio para los problemas de *strip packing.*

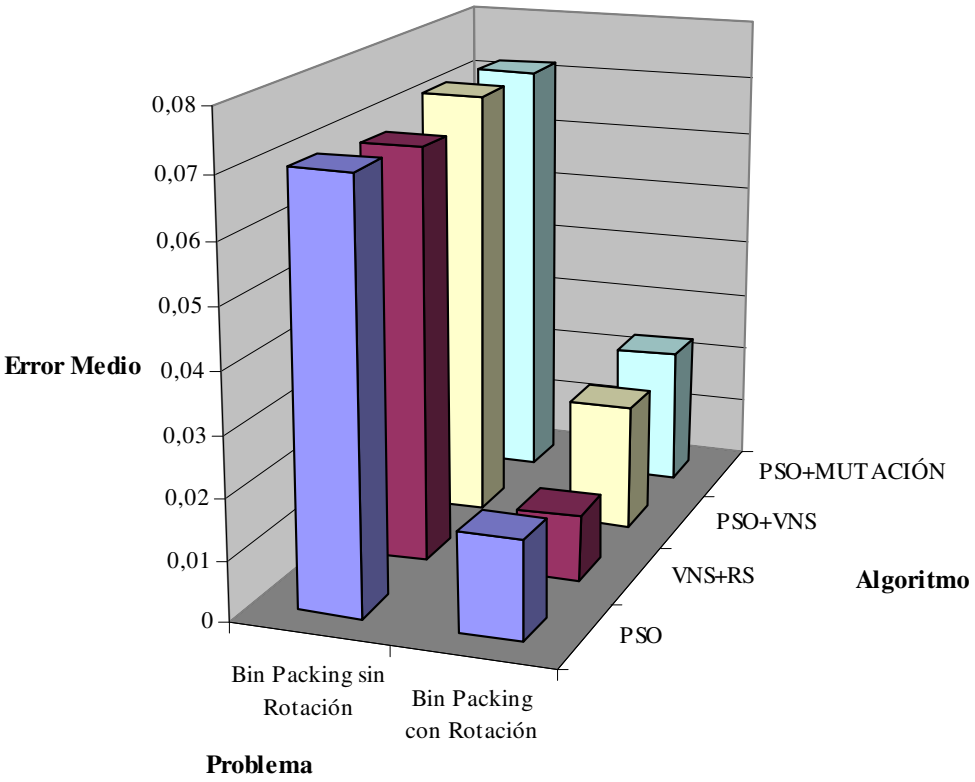

Figura 17. Gráfica error medio para los problemas de *bin packing*.

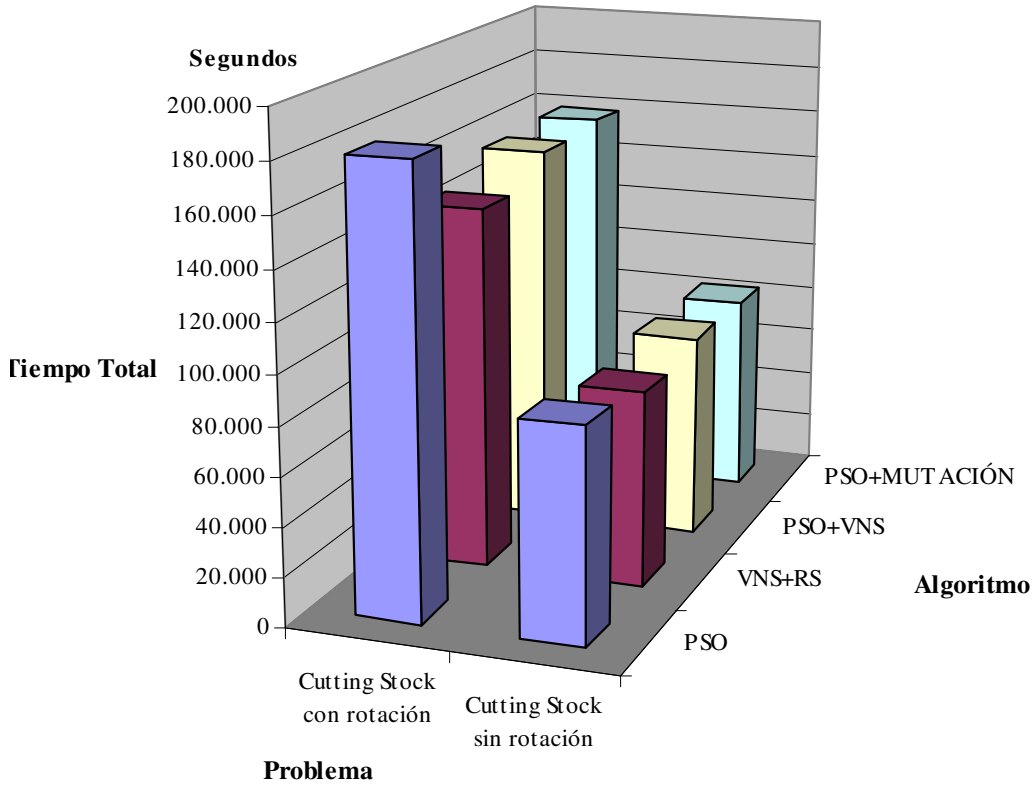

Figura 18. Gráfica tiempo total para los problemas de *cutting stock* (segundos).

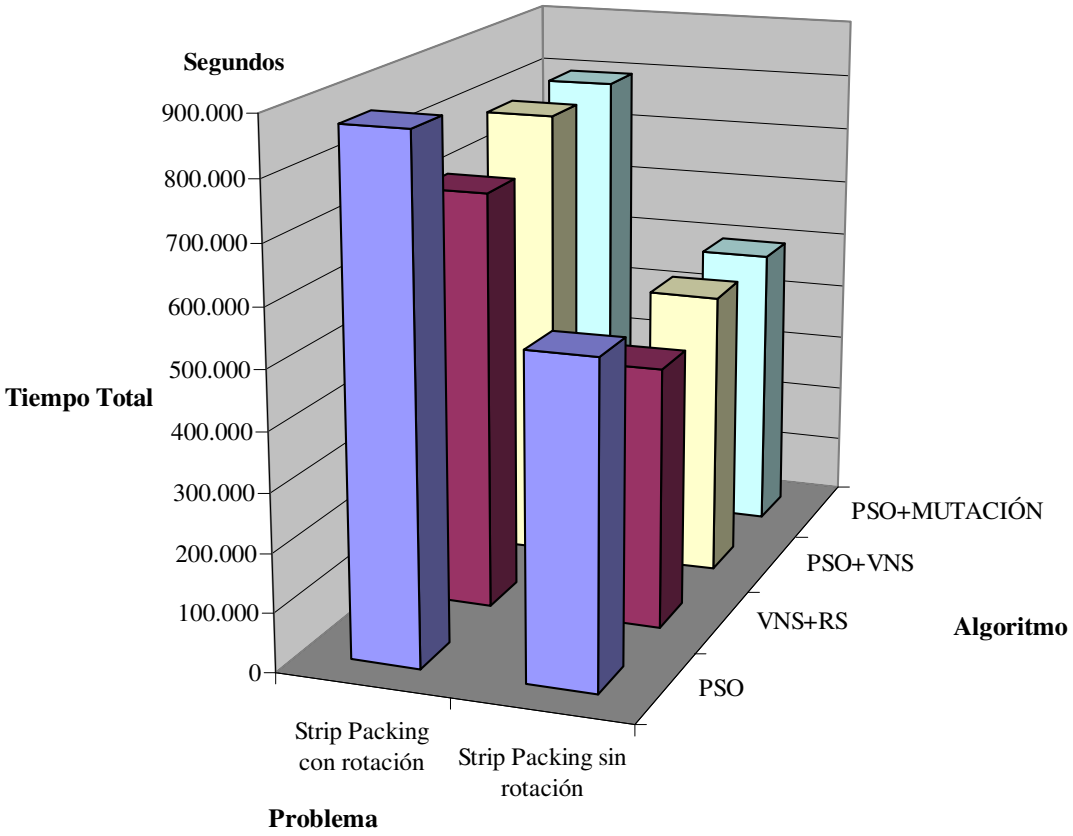

Figura 19. Gráfica tiempo total para los problemas de *strip packing* (segundos).

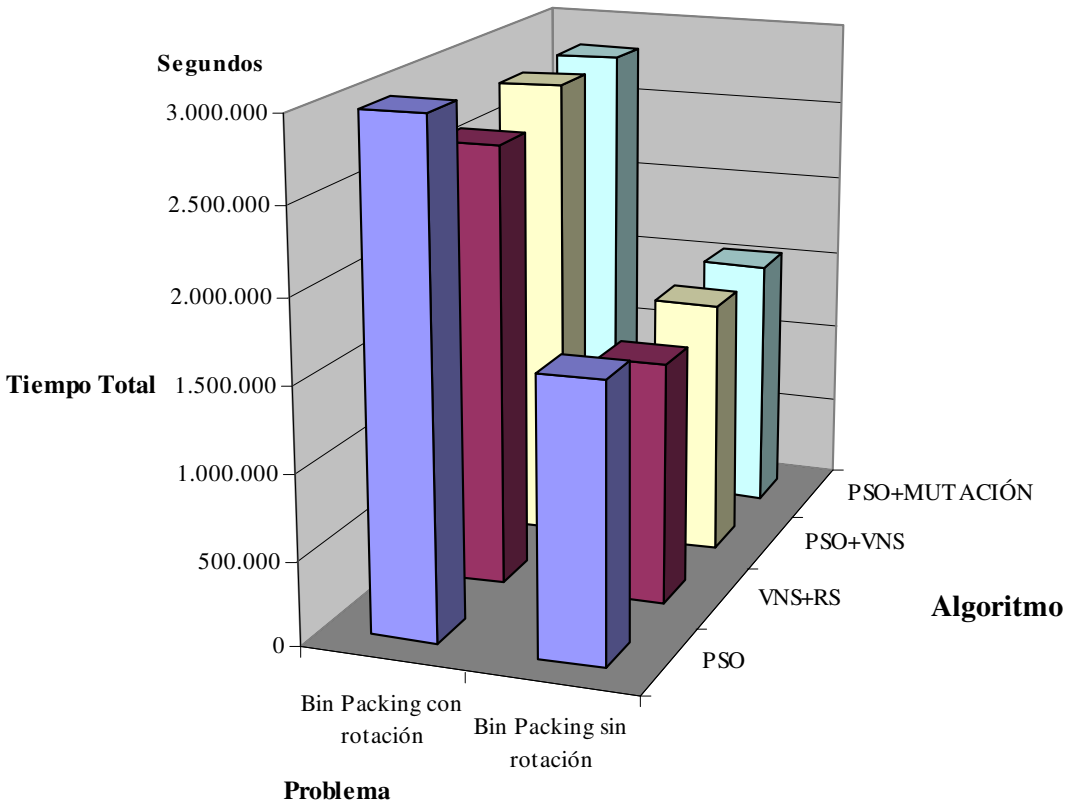

Figura 20. Gráfica tiempo total para los problemas de *bin packing* (segundos).

6.5. Análisis

En esta sección se realiza un análisis de los algoritmos y de la metodología desarrollada en cada problema. En este se utilizan los descriptores estadísticos (error medio y desviación estándar) de la calidad de las respuestas, el tiempo total utilizado por cada algoritmo y se emplean las mejores respuestas obtenidas para cada problema.

6.5.1. Análisis de los algoritmos

En las tablas 16, 17, 18, 19, 20, y 21, se identifica el algoritmo con mejor desempeño para todos los problemas propuestos en este trabajo. El algoritmo A_{VNS+RS} presenta mejor desempeño como puede observase en las tablas de resultados de los problemas de *cutting stock* con rotación, *strip packing* sin rotación y *bin packing* con y sin rotación, mientras el algoritmo A_{PSO} presenta el mejor desempeño en los problemas, *bin packing* sin rotación y *strip packing* con rotación. En estos dos últimos problemas el algoritmo A_{VNS+RS} esta muy cerca del algoritmo A_{PSO}. Por lo anterior, el A_{VNS+RS} es el algoritmo con más regularidad en todos los problemas, seguido del algoritmo A_{PSO}. Los algoritmos $A_{PSO+VNS}$ y $A_{PSO+Mutación}$ presentan un desempeño aceptable.

La tabla 29 presenta la diferencia entre el algoritmo A_{VNS+RS} y el resto de algoritmos. En los seis problemas el algoritmo A_{VNS+RS} presenta los menores tiempo de cómputo lo que lo hace muy eficiente. El algoritmo A_{PSO} requiere de los mayores tiempos de cómputo para alcanzar las respuestas. Los algoritmos $A_{PSO+VNS}$ y A_{PSO+RS} presentan desempeños similares, dados sus promedios de calidad y tiempos de respuestas.

6.5.2. Análisis de la metodología desarrollada en cada problema

En las tablas 22, 23, 24, 25, 26 y 27 se resumen los mejores resultados alcanzados por la metodología propuesta. En la tabla 28 se resumen los resultados reportados en la literatura especializada. De la tabla 28 se logra identificar que la metodología propuesta para resolver los problemas de *cutting stock* con o sin rotación supera en un 79% las mejores respuestas reportadas en la literatura especializada y sólo en un 21% las respuestas son de menor calidad.

La metodología propuesta aplicada a los problemas de *strip packing* presenta un comportamiento regular, debido a que en los problemas donde no se permite la rotación de piezas solo el 18 % de las respuestas reportadas son superadas, el 47% de las respuestas son igualadas y en el 35% restante no se logra alcanzar las reportadas. En los problemas donde se permite la rotación de piezas un 30% de las soluciones superan las reportadas, el 32% de las respuestas son igualadas y el 38% restante no logran alcanzar las reportadas.

Finalmente, la metodología propuesta aplicada a los problemas de *bin packing* presenta un comportamiento medio, debido a que en los problemas donde no se permite la rotación de piezas solo el 57% de las respuestas reportadas son alcanzadas y en el 43% restante no se logran superar, muy diferente cuando se permite la rotación de piezas, donde un 71 % de las respuestas son igualadas y el 29% restante de las soluciones superan las reportadas.

Igual ocurre al analizar la tabla 29, los tiempos de respuestas de la metodología propuesta son excelentes para resolver los problemas de *cutting stock* de gran escala (con distribuciones de hasta 139 piezas) permitiendo la rotación de piezas en un tiempo promedio de 160 segundos y sin rotación en un tiempo promedio de 80 segundos, problemas de *bin packing* de pequeña y mediana escala (hasta 40 piezas) tiempos promedios de 125 y 230 segundos respectivamente y problemas de *strip packing* de mediana escala (hasta 22 piezas) con y sin rotación en tiempos promedios de 130 y 220 segundos respectivamente. Sin embargo, en los problemas de *bin packing* de gran escala (hasta 100 piezas) con y sin rotación presentan unos tiempos promedios relativamente altos de 16 y 30 minutos respectivamente y en los problemas de *strip packing* de gran escala (hasta 198 piezas) con y sin rotación presenta unos tiempos promedio relativamente altos de 21 minutos y 33 minutos respectivamente.

En las tablas 16, 17, 18, 19, 20, y 21 se identifica que la metodología propuesta para problemas de *cutting stock* tiene un error medio de 1,1% respecto a la mejor respuesta reportada, lo que significa que para cualquier problema de *cutting stock* con o sin rotación la solución obtenida con la metodología propuesta esta cerca del óptimo global, igualmente, para problemas de *bin packing* con rotación de piezas con un error medio de 1,7% respecto a la mejor respuesta reportada, no muy lejos para los problemas de *bin packing* con rotación de piezas con un error medio de 7% respecto a la mejor respuesta reportada. Mientras, para los problemas de *strip packing* se encontró un error medio de 14% respecto a la mejor respuesta reportada y dado que en aplicaciones reales se manejan en el proceso de corte o empaquetamiento unos límites de desperdicio del 30%. Se concluye que aplicando dichas metodologías se obtiene una reducción en las pérdidas del material de más del 50%. El inconveniente en la aplicación de estas metodologías es el alto tiempo de cómputo.

6.6. Resumen

En este capítulo 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.

Fueron utilizados con el fin de verificar la eficiencia de las metodologías propuestas, 50 casos de prueba catalogados como instancias de gran escala para los problemas de *cutting stock*, 25 casos de estudio catalogados como instancias de mediana escala y 9 casos de estudio catalogados como instancias de gran escala para los problemas de *strip packing* y 70 casos de estudio catalogados como instancias de gran escala y 75 casos de estudio catalogados como instancias de pequeña y mediana escala para los problemas de *bin packing*.

Se efectuó un análisis estadístico con el fin de medir la calidad de los algoritmos 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 los diferentes algoritmos para cada muestra.

Finalmente un análisis comparativo es realizado tanto entre algoritmos como sobre la metodología propuesta. Para esto fueron usados descriptores estadísticos y tiempos de ejecución. Observándose que el algoritmo con mejor desempeño es el A_{VNS+RS} seguido del A_{PSO}. Los algoritmo $A_{PSO+VNS}$ y $A_{PSO+Mutación}$ presentaron comportamientos similares. Por otro lado, la metodología propuesta presenta un error de un 1,1% para los problemas de *cutting stock,* para los problemas de *bin packing* un error de un 4,4% y un error de un 14% para los problemas de *strip packing.*

7. CONCLUSIONES Y RECOMENDACIONES

En esta investigación se estudiaron cuatro algoritmos de optimización metaheurística para solucionar los problemas de: empaquetamiento óptimo bidimensional guillotinado en una sola placa (*two-dimensional cutting stock problem*), empaquetamiento óptimo bidimensional guillotinado en placas (*two-dimensional bin packing problem*) y empaquetamiento óptimo bidimensional guillotinado en rollos infinitos (*two-dimensional strip packing problem*), con y sin rotación de 90° de las piezas.

Se realizó la revisión bibliográfica de los modelos matemáticos de los problemas de empaquetamiento óptimo bidimensional guillotinado en una sola placa, en placas y en rollos con y sin rotación de 90° de las piezas. Los problemas estudiados en esta tesis han sido analizados 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 guillotina.

En este trabajo se presento un modelo propuesto recientemente por Ben et al. en [44], este describe perfectamente el problema de empaquetamiento óptimo bidimensional guillotinado en rollos infinitos y en placas presentado en este estudio, aunque carece de la restricción de rotación de las piezas 90°. Este es modificado para representar el problema de empaquetamiento óptimo bidimensional guillotinado en una sola placa.

Se utilizaron tres técnicas metaheurísticas (búsqueda en vecindario variable, recocido simulado y optimización con cúmulo de partículas), con base en estos se implementaron cuatro algoritmos para resolver los problemas de empaquetamiento óptimo bidimensional guillotinado en una sola placa, en placas y en rollos.

Se revisaron diferentes codificaciones propuestas en la literatura para resolver los problemas de empaquetamiento. En este trabajo fue seleccionada una codificación en árbol de cortes, que incluye un árbol de orientación de los cortes y un árbol de distancias de cortes. Los resultados obtenidos con esta codificación son de excelente calidad.

Los algoritmos propuestos presentan resultados de excelente calidad al resolver los casos de prueba de la literatura especializada. El algoritmo de mejor desempeño respecto a tiempos y calidad de respuestas es el A_{VNS+RS}.

En esta tesis se desarrollo un aplicativo informático en versión Beta para resolver los problemas propuestos, obteniéndose resultados de excelente calidad en tiempos de cómputo razonables.

Trabajos futuros propuestos en esta investigación.

Aplicar algoritmos paralelizables en la solución de los problemas estudiados, con el fin de mejorar tiempos y calidad de las respuestas. Esto sin aumentar los costos de

implementación, ya que los algoritmos no requieren de grandes especificaciones de las máquinas y la conformación de un cluster de máquinas de bajo desempeño es un proyecto viable para la paralelización de los algoritmos.

Aplicar los algoritmos propuestos en la solución de los problemas de empaquetamiento óptimo bidimensional guillotinado irrestricto (piezas infinitas) en una sola placa con y sin rotación de piezas, y a los problemas de empaquetamiento óptimo bidimensional guillotinado con pesos (piezas de valor diferente a su área).

Ampliar la codificación propuesta en el estudio problemas de empaquetamiento óptimo tridimensional guillotinado 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.

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

Aplicar esta metodología de solución en problemas de la vida real, lo cual se logra a través de información real.

8. BIBLIOGRAFÍA

[1] P. Gilmore y R. Gomory, "A linear programming approach to the cutting stock problem", Operations Research, Vol. 9, pp.849-859, 1961.

[2] P. Sweeney y R. Paternoster, "Cutting and Packing Problems: A categorized, application-orientated research bibliography". Journal of the Operational Research Society, Vol. 43, pp. 691-706, 1992.

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

[4] S. Martello, M. Monaci, y D. Vigo, "An exact approach to the strip-packing problem," INFORMS Journal on Computing, vol. 15, no. 3, pp. 310–319, 2003.

[5] H. Dyckhoff, "A typology of cutting and packing problems", European Journal of Operational Research, Vol. 44, pp. 145-160, 1990.

[6] A. Lodi, S. Martello, D. Vigo, "Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems", INFORMS J. Comput. Vol. 11, pp. 345–357, 1999.

[7] H. M. Salkin y C. A. Dekluyver, "The knapsack Problem: A Survey", Nav. Res. Log. Quart., Vol. 22, pp. 127-144, 1975.

[8] N. Christofides and C. Whitlock, "An algorithm for two-dimensional cutting problems", Operations Research, Vol. 25, pp. 30-44, 1977.

[9] B. MacLeod, R. Moll, M. Girkar y N. Hanifi, "An algorithm for the 2D guillotine cutting stock problem", European Journal of Operational Research, Vol. 68, pp. 400-412, 1993.

[10] J. E. Beasley, "An exact two-dimensional non-guillotine cutting tree search procedure", *Oper. Res.,* Vol. 33, pp. 49–64, 1985.

[11] R. D. Tsai, E. M. Malstrom y H.D. Meeks, "A two dimensional palletizing procedure for warehouse loading operations". *IEEE Transactions*, Vol. 20, pp.418-425, 1988.

[12] G. Scheithauer y J. Terno, "Modeling of packing problems", *Optimization*, Vol. 28, pp.63-84, 1993.

[13] E. Hadjiconsyantinou y N. Christofides, "An exact algorithm for general, orthogonal, two dimensional knapsack problems", *European Journal of Operacional Research*, Vol. 83, pp. 39-56, 1995.

[14] M. Arenales y R. Morabito, “An graph approach to the solution of two dimensional non-guillotine cutting problems”, *European Journal of Operational Research*, Vol. 84, pp.599-617, 1995.

[15] S. P. Fekete y J. Schepers, “On more-dimensional packing III: Exact algorithms”, *Technical Report ZPR97-290*, Mathematisches Institut, UniversitYat zu KYoln, 1997.

[16] A. Caprara y M. Monaci, “On the two dimensional knapsack problem”, *Operations Research Letters*, Vol. 32, pp. 5-14, 2004.

[17] B. Chazelle, “The bottom-left bin packing heuristic: An efficient implementation”, *IEEE Transactions on Computers*, Vol. 32, pp. 697–707, 1983.

[18] K. K. Lai y J. W. M. Chan, “A evolutionary algorithm for the rectangular cutting stock problem”, *International Journal of Industrial Engineering*, Vol. 4, pp. 130-139, 1997.

[19] A. Lodi, S. Martello y M. Monaci, “Two-dimensional packing problems: A Survey”, *European Journal of Operational Research*, Vol. 141, pp. 241–252, 2002.

[20] J. E. Beasley, “A population heuristic for constrained two-dimensional nonguillotine cutting”, *European Journal of Operational Research*, Vol. 156, pp. 601-627, 2004.

[21] T. W. Leung, C. H. Yung y M.D. Troutt, “Applications of genetic search and simulated annealing to the two-dimensional non-guillotine cutting stock problem”, *Computers and Industrial Engineering*, Vol. 40, pp. 201-214, 2001.

[22] Y. Cui, “An exact algorithm for generating homogenous T-shape cutting patterns”, *Computers & Operations Research*, Vol. 34, pp. 1107-1120, 2007.

[23] E. Toro, A. Garcés y H. Ruiz, “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.

[24] W. Fernandez de La Vega y V. Zissimopoulos, “An approximation scheme for strip packing of rectangles with bounded dimensions”, *Discrete Applied Mathematics*, Vol. 82, pp. 93–101, 1998.

[25] N. Lesh, J. Marks, A. McMahon y M. Mitzenmacher, “Exhaustive approaches to 2D rectangular perfect packings”, *Information Processing Letters*, Vol. 90, pp. 7–14, 2004.

[26] C. Kenyon and E. Rémila, “Approximate strip packing”, in Proceedings of the *37th Annual Symposium on Foundations of Computer Science* (FOCS ’96), pp. 31–36, Burlington, Vt, USA, Octubre 1996.

[27] D. Zhang, Y. Kang and A. Deng, “A new heuristic recursive algorithm for the strip rectangular packing problem”, *Computers & Operations Research*, Vol. 33, pp. 2209–2217, 2006.

[28] A. Bortfeldt, “A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces”, *European Journal of Operational Research*, Vol. 172, pp. 814–837, 2006.

[29] J. D. Beltran, J. E. Calderon, R. J. Cabrera, J. A. Moreno Perez y J. M. Moreno-Vega, “GRASP/VNS hybrid for the strip packing problem”, in Proceedings of the *1st International Workshop on Hybrid Metaheuristics* (HM ’04), pp. 79–90, Valencia, Spain, Agosto 2004.

[30] E. Hopper y B. C. H. Turton, “A review of the application of meta-heuristic algorithms to 2D strip packing problems”, *Artificial Intelligence Review*, Vol. 16, pp. 257–300, 2001.

[31] M. Hifi, “Exact algorithms for the guillotine strip cutting/packing problem”, *Computers & Operations Research*, Vol. 25, pp. 925–940, 1998.

[32] Y. Cui, Y. Yang, X. Cheng y P. Song, “A recursive branch-and-bound algorithm for the rectangular guillotine strip packing problem”, *Computers & Operations Research*, Vol. 35, pp. 1281–1291, 2008.

[33] A. Bekrar, I. Kacem y C. Chu, “A comparative study of exact algorithms for the two dimensional strip packing problem”, *Journal of Industrial and Systems Engineering*, Vol. 1, pp. 151–170, 2007.

[34] G. F. Cintra, F. K. Miyazawa, Y. Wakabayashi y E. C. Xavier, “Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation”, *European Journal of Operational Research*, Vol. 191, pp. 61–85, 2008.

[35] P. C. Gilmore y R. E. Gomory, “A linear programming approach to the cutting stock problem – part II”, *Operations Research*, Vol. 11, pp. 863–888, 1963.

[36] P. C. Gilmore y R. E. Gomory, “Multi-stage cutting stock problems of two and more dimensions”, *Operations Research,* Vol. 13, pp. 94–120, 1965.

[37] P. C. Gilmore y R.E. Gomory, “The theory and computation of knapsack functions”, *Operations Research*, Vol. 14, pp. 1045–1074, 1966.

[38] M. Biro y E. Boros, “Network flows and non-guillotine cutting patterns”, *European Journal of Operational Research,* Vol. 16, pp. 215–221, 1984.

[39] S. P. Fekete y J. Schepers, “A new exact algorithm for general orthogonal d-dimensional knapsack problems”, *Algorithms – ESA’97*, Springer Lecture Notes in Computer Science, Vol. 1284, pp. 144–156, 1997.

[40] A. Lodi, S. Martello y D. Vigo, "Models and bounds for the two-dimensional level packing problems", *Journal of Combinatorial Optimization*, Vol. 8, pp. 363–379, 2004.

[41] C. S. Chen, S. M. Lee y Q. S. Shen, "An analytical model for the container loading problem", *European Journal of Operational Research*, Vol. 80, pp. 68–76, 1995.

[42] H. Onodera, Y. Taniguchi y K. Tamaru, "Branch-and-bound placement for building block layout", in Proceedings of the *28th ACM/IEEE Design Automation Conference*, pp. 433–439, 1991.

[43] K. Daniels, V. J. Milenkovic y Z. Li, "Multiple containement methods", *Technical Report 12-94*, Center for Research in Computing Technology, Division of Applied Sciences, Harvard University, 1994.

[44] S. Ben, C. Chu y M. L. Espinouse, "Characterization and modelling of guillotine constraints", *European Journal of Operational Research*, Vol. 191, pp. 112–126, 2008.

[45] K. Dowsland, "Some experiments with simulated annealing techniques for packing problems", *European Journal of Operational Research,* Vol. 68, pp. 389–399, 1993.

[46] B. Kröger, "Guillotineable bin packing: A genetic approach", *European Journal of Operational Research*, Vol. 84, pp. 645-661, 1995.

[47] S. Jakobs, "On genetic algorithms for the packing of polygons", *European Journal of Operational Research*, Vol. 88, pp. 165-181, 1996.

[48] E. M. Toro, *Solución del problema de empaquetamiento óptimo bidimensional*, Tesis de Maestría, Universidad Tecnológica de Pereira, 2008.

[49] K. K. Lai y J. W. M. Chan, "A evolutionary algorithm for the rectangular cutting stock problem", *International Journal of Industrial Engineering*, Vol. 4, pp. 130-139, 1997.

[50] T. W. Leung, C. K. Chan y M. D. Troutt, "Application of a mixed simulated annealing-genetic algorithm heuristic for the two-dimensional orthogonal packing problem", *European Journal of Operational Research,* Vol. 145, pp. 530–542, 2003.

[51] F. Glover y G. Kochenberger, *Handbook of metaheuristics*, Kluwer Academic Publishers, Boston, 2003.

[52] R. Gallego, A. H. Escobar y E. M. Toro, *Técnicas metaheurísticas de optimización*, Textos universitarios, Universidad Tecnológica de Pereira, 2008.

[53] A. Gomes y J. F. Oliveira, "Solving Irregular Strip Packing problems by hybridizing simulated annealing and linear programming". *European Journal of Operational Research*, Vol. 171, pp. 811-829, 2006.

[54] E. Hopper y B. Turton, “An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem”, *European Journal of Operational Research*, Vol. 128, pp. 34–57, 2001.

[55] F. K. R. Chung, M. R. Garey, D. S. Johnson, “On packing two-dimensional bins”, *SIAM J. of Algebraic and Discrete Methods*, Vol. 3, pp. 66-76, 1982.

[56] A. Bekrar and I. Kacem, “A Comparison study of heuristics for solving the 2D Guillotine Strip and Bin Packing Problems”, *Service Systems and Service Management, 2008 International Conference on*, pp. 1–6, 2008.

[57] D. S. Liu, K. C. Tan, S. Y. Huang, C. K. Goh y W.K. Ho, “On solving multiobjective bin packing problems using evolutionary particle swarm optimization”, *European Journal of Operational Research*, Vol. 190, Issue 2, pp. 357-382, 2008.

[58] C. L. Chen, S. M. Hart y W. M. Tham, “A simulated annealing heuristic for the one-dimensional cutting stock problem”, *European Journal of Operational Research*, Vol. 93, pp. 33-43, 1996.

[59] L. Faina, “An application of simulated annealing to the cutting stock problem”, *European Journal of Operational Research*, Vol. 114, Issue 3, pp. 542-556, 1999.

[60] S. Kirkpatrick, C. Gelatt, y M. Vecchi, “Optimization by simulated annealing”, *Science*, Vol. 220, pp. 671-680, 1983.

[61] N. Mladenović y P. Hansen, “Industrial applications of the variable neighborhood search metaheuristics”, *Decisions and Control in Management Science*, pp. 261–274, 2001.

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

[63] N. Mladenović y P. Hansen, “Variable neighborhood search”, *Computers and Operations Research*, Vol. 24, pp. 1097–1100, 1997.

[64] N. Mladenović y P. Hansen, “Variable neighborhood search: Principles and applications”, *European Journal of Operational Research*, Vol. 130, pp. 449–467, 2001.

[65] Z. Zhi-Hui, Z. Jun, L. Yun, and S. Henry, “Adaptive Particle Swarm Optimization”, *IEEE Transactions On Systems, Man, And Cybernetics—Part B: Cybernetics*, This article has been accepted for inclusion in a future issue of this journal 2009.

[66] J. J. Liang, A. K. Qin, P. N. Suganthan y S. Baskar, “Comprehensive learning particle swarm optimizer for global optimization of multimodal functions”, *IEEE Trans. Evol. Comput.*, Vol. 10, pp. 281–295, 2006.

[67] R. C. Eberhart y Y. Shi, "Guest editorial", *IEEE Trans. Evol. Comput.—Special Issue Particle Swarm Optimization*, Vol. 8, pp. 201–203, 2004.

[68] P. J. Angeline, "Using selection to improve particle swarm optimization", *in Proc. IEEE Congr. Evol. Comput.*, Anchorage, AK, pp. 84–89 , 1998.

[69] Y. P. Chen, W. C. Peng y M. C. Jian, "Particle swarm optimization with recombination and dynamic linkage discovery", *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, Vol. 37, pp. 1460–1470, 2007.

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

[71] J. J. Liang y P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer with local search", *in Proc. IEEE Congr. Evol. Comput.*, pp. 522–528, 2005.

[72] A. Carlisle y Dozier G., "Adapting particle swarm optimization to dynamic environments", *in Proc. Int. Conf. Artif. Intell.*, Las Vegas, NV, pp. 429–434, 2000.

[73] X. Hu y R. C. Eberhart, "Adaptive particle swarm optimization: Detection and response to dynamic systems," *in Proc. IEEE Congr. Evol. Comput.*, Honolulu, HI, pp. 1666–1670, 2002.

[74] X. Xie, W. Zhang y Z. Yang, "Adaptive particle swarm optimization on individual level," *in Proc. Int. Conf. Signal Process.*, pp. 1215–1218, 2002.

[75] M. Clerc, "The swarm and the queen: Toward a deterministic and adaptive particle swarm optimization", *in Proc. IEEE Congr. Evol. Comput*., pp. 1951–1957, 1999.

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

[77] J. Pepper, B. Golden y E. Wasil, "Solving the Traveling Salesman Problem with Annealing-Based Heuristics: A Computational Study", *IEEE Trans. Syst., Man, Cybern. A*, Vol. 32, pp. 72-77, 2002.

[78] Y. Cui, *Problem instances for the Cutting Stock Problem,* [Online]. Available: http://www.gxnu.edu.cn/Personal/ydcui/English/Problems/RecPubC_G3.DTXT.

[79] M. Hifi, "The Strip Cutting/Packing Problem: Incremental Substrip Algorithms-Based Heuristics", *Pesquisa Operacional*, Special Issue on Cutting and Packing Problems, pp. 169-188, 1999.

[80] M. Hifi, *Problem instances for the Strip Cutting/Packing Problem*, [Online]. Available: ftp://cermsem.univ-paris1.fr/pub/CERMSEM/hifi/Strip-cutting/.

[81] D. Álvarez, E. Toro y R. Gallego, "Empaquetamiento óptimo bidimensional en rollos infinitos usando un algoritmo híbrido de búsqueda en vecindad variable y recocido simulado", *Scientia Et Technica*, Vol. 42, pp. 205-211, 2009.

[82] D. Álvarez, E. Toro y R. Gallego, "Empaquetamiento óptimo bidimensional con rotación de piezas usando un algoritmo híbrido de búsqueda en vecindad variable y recocido simulado", *Scientia Et Technica*, Vol. 41, pp. 279-284, 2009.

[83] D. Álvarez, E. Toro y R. Gallego, "Algoritmo de optimización cúmulo de partículas aplicado en la solución del problema de empaquetamiento óptimo bidimensional con y sin rotación", *Scientia Et Technica*, Vol. 43, pendiente de publicación, 2009.

[84] Y. Cui, "An exact algorithm for generating homogenous two-segment cutting patterns", *Engineering Optimization*, Vol. 39, pp. 365-380, 2007.

[85] D. Álvarez, E. Toro y R. Gallego, "Empaquetamiento óptimo en tiras con y sin rotación resuelto con cúmulo de partículas", *Scientia Et Technica*, Vol. 43, pendiente de publicación, 2009.

[86] J. O. Berkey y P. Y. Wang, "Two-dimensional finite bin packing algorithms", *Journal of the Operational Research Society*, Vol. 38, pp. 423–429, 1987.

[87] K. B. Binkley y M. Hagiwara, "Applying self-adaptive evolutionary algorithms to two-dimensional packing problems using a four corners' heurística", *European Journal of Operational Research,* Vol. 183, pp. 1230–1248, 2006.

[88] K. Dowsland, "Some experiments with simulated annealing techniques for packing problems", *European Journal of Operational Research,* Vol. 68, pp. 389–399, 1993.

[89] A. Lodi, S. Martello y D. Vigo, "Neighborhood search algorithm for the guillotine non-oriented two-dimensional bin packing problem", in: S. Voss, S. Martello, I.H. Osman, C. Roucairol (Eds.), *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer Academic Publishers, Boston, pp. 125–139, 1998.

[90] J. Puchinger y G.R. Raidl, "Models and algorithms for three-stage two-dimensional bin packing", *European Journal of Operational Research,* Vol. 183, pp. 1304–1327, 2006.

[91] M. Dell Amico, S. Martello y D. Vigo, "A lower bound for the nonoriented two-dimensional bin packing problem", *Discrete Applied Mathematics,* Vol. 118, pp. 13–24, 2002.

[92] A. Lodi, S. Martello y D. Vigo, "Approximation algorithms for the oriented two-dimensional bin packing problem", *European Journal of Operational Research*, Vol. 112, pp. 158–166, 1999.

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