UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

FACULTAD DE CONTADURÍA, ADMINISTRACIÓN E INFORMÁTICA

Diseño e implementación de un algoritmo heurístico para el problema en el corte de vidrio

T E S I S

PARA OBTENER EL GRADO DE MAESTRA EN

OPTIMIZACIÓN Y CÓMPUTO APLICADO

PRESENTA:

L.I. Berenice Alonso Muñoz

DIRECTOR DE TESIS

Dr. Federico Alonso Pecina

CODIRECTOR DE TESIS

Dr. Martín Gerardo Martínez Rangel

Cuernavaca, Morelos Noviembre 2019

Resumen

En este trabajo de investigación se resuelve el problema en el corte de vidrio de Saint-Gobain Glass que publica la Sociedad Francesa de Investigación Operativa y Apoyo a la Decisión (ROADEF, por sus siglas en francés), el cual se desprende del conocido problema de optimización Corte y Empaquetado (cutting and packing, en inglés), este engloba a una gran familia de problemas con diferentes variantes que son de gran ayuda a la industria maderera, textil, metalúrgica, de vidrio, etc. La familia pertenece a los problemas NP-duros. Lo interesante de este problema en específico es que contiene varias restricciones que en la literatura se tratan en diferentes variantes, debido a esto la forma de resolverlo se vuelve más compleja.

Se han publicado trabajos donde los algoritmos glotones generan buenos resultados para problemas con un número menor de restricciones.

En esta tesis de maestría se proponen tres algoritmos glotones y la hibridación de ellos para resolver el problema y dar soluciones factibles, se experimentó con cincuenta instancias que proporciona ROADEF, la hibridación de los algoritmos glotones demuestra que son mejores al trabajar en conjunto que si se hace de forma individual.

Abstract

This master thesis solves the Cutting and Packing problem of Saint-Gobain Glass published by the French Society for Operational Research and Decision Support (ROADEF), which follows from the well-known problem of cutting and packing optimization, this encompasses a large family of problems with different variants that are of great help to the industry the wood, textile, metallurgical, glass, etc. This family problem belongs to the NP-hard problems. The interesting matter about this specific problem is that it contains several restrictions that in the literature are treated in different variants, because of this the way to solve it becomes more complex. Works have been published where greedy algorithms generate good results for problems with a smaller number of restrictions. In this master's thesis three greedy algorithms and hybridization of them are proposed to solve the problem and give feasible solutions, experimented with fifty instances provided by ROADEF, the hybridization of the greedy shows that they are better when working together than if individual way.

Agradecimientos

A Dios por permitirme adquirir más conocimientos a lo largo de este proyecto.

A CONACYT por brindarme el apoyo económico para desarrollar esta investigación.

A la secretaría de Investigación y Posgrado de la FCAeI, por darme la facilidad de trabajar en sus instalaciones durante el desarrollo del proyecto.

Muy en especial al Dr. Federico Alonso Pecina, director de esta tesis, por guiar y compartir sus conocimientos conmigo, por generar en mí el gusto por la optimización, por su apoyo para terminar este proyecto, pero sobre todo por creer en mí.

Al Dr. Martín Gerardo Martínez Rangel, codirector de esta tesis, por darme la oportunidad de entrar al bello mundo de la optimización y siempre brindarme palabras de aliento.

A los integrantes de mi comité tutoral y revisores de tesis: Dr. José Crispín Zavala Diaz, Dr. Martín Heriberto Cruz Rosales y Dr. José Alberto Hernández Aguilar. Por tomarse el tiempo para orientarme con sus consejos y comentarios para mejorar profesionalmente.

A mi madre hermosa Gloria Muñoz Rosales por confiar en mí y en las decisiones que tomo, brindándome su apoyo incondicional para culminar este proyecto de investigación.

"Dime y lo olvido, enséñame y lo recuerdo, involúcrame y lo aprendo". Benjamín Franklin.

Dedicatorias

A Dios por sostenerme siempre en los momentos difíciles de mi vida y acompañarme en esta aventura de dos años que me lleno de muchas satisfacciones personales y profesionales.

A mi familia fuente de inspiración para todo lo que hago.

De manera muy especial y con profunda admiración a mi amado hermano Milton José Alonso Muñoz, por enseñarme a tener valor y fuerza para enfrentar la vida, perseguir mis sueños, apasionarme por mi trabajo, por ser mi amigo y cómplice, por no dejar que me venza aunque las cosas parezcan difíciles, por ser mi ejemplo de perseverancia y tener la fortuna de verlo conmigo.

A mi madre M. Gloria Muñoz Rosales, que es mi raíz y mis alas, la persona que más admiro y amo, sin ella nada de esto sería posible.

A mi padre Vicente Alonso Martínez, que me enseño que no importan las adversidades, si luchas mucho lograrás todo lo que te propongas.

A mi asesor el Dr. Federico Alonso Pecina por la paciencia y comprensión que tuvo estos dos años.

A mis amigos, Diego Olmos Vázquez y Jorge Luis Aguilar Pita por su apoyo constante; a Rocio Diego Celis y Ana Karen Castañeda Escobar, que conocí en esta maestría, con las que pasé momentos increíbles profesionales y personales, a Verónica Oliver Aguilar por siempre preocuparse y cuidar de mí y de mi familia.

"Las mujeres que han cambiado el mundo no han necesitado mostrar otra cosa que su inteligencia". Rita Levi Montalcini.

Contenido

Índice de figuras

Índice de tablas

Glosario

Patrones	Un acomodo de las piezas y de los cortes de tal forma que respete las restricciones de precedencia y corte entre los artículos.
Figura	Es la representación que se realiza de un elemento geométrico (artículo) en un contenedor.
Palet o tarima	Es un armazón de madera, plástico u otro material empleado en el movimiento de carga, para facilitar el levantamiento y manejo con pequeñas grúas hidráulicas.
Stock	Conjunto de mercancías o productos que se tienen almacenados en espera de su venta o comercialización.
Eje x del contenedor	La línea en un contenedor que corre horizontalmente (izquierda-derecha).
Eje y del contenedor	La línea en un contenedor que corre verticalmente (abajo-arriba).
x_j	Vector de valores binarios para verificar si un patrón pertenece a la solución o no.
$\boldsymbol{O_I(it)}$	Función que obtiene el orden para los artículos cortados en un contenedor.
$\boldsymbol{O_P(p_i)}$	Función que obtiene el orden en los patrones seleccionados del posible conjunto de patrones P.
$\boldsymbol{\Delta}$	Conjunto de elementos bidimensionales que representan la pérdida.
$\boldsymbol{\delta}$	Subconjunto de elementos que representan la pérdida en una solución.

Árbol	Es un grafo en el que cualesquier dos vértices están conectados por exactamente un camino.
Bosque	Dícese del grafo que contiene dos o más árboles disjuntos.
$\boldsymbol{\beta^{it}}$	Variable que se le asigna el valor mínimo entre la longitud w y h de un artículo.
$\boldsymbol{\beta^{max}}$	Es el valor máximo entre un conjunto de variables β^{it}.
$\boldsymbol{\gamma^{it}}$	Variable que se le asigna el valor máximo entre la longitud w y h de un artículo.
$\boldsymbol{\gamma^{max}}$	Es el valor máximo entre un conjunto de variables γ^{it}.

Capítulo 1 Introducción

En este capítulo se explican los antecedentes de la investigación de operaciones, el planteamiento del problema, la hipótesis, los objetivos, la justificación, los alcances de esta investigación y la organización de la tesis.

"El aprendizaje es experiencia, todo lo demás es información". Albert Einstein

1.1 Antecedentes

La investigación de operaciones tuvo sus inicios en 1759 cuando el economista Quesnay comenzó a utilizar modelos primitivos de programación matemática, pero sus actividades formales fueron a partir de la Segunda Guerra Mundial en la logística de estrategia para vencer al enemigo y en la distribución de los recursos materiales bélicos que representaban un costo alto económicamente, el doctor George Dantzig en 1947 inventó el método Simplex para ayudar en este y otro tipo de problemas, iniciando de esa manera con la programación lineal.

En la década de los cincuentas la investigación de operaciones se extendió dando partida a la optimización combinatoria que resuelve problemas complejos, donde el conjunto de posibles soluciones es discreto o puede reducirse a un conjunto discreto tratando de encontrar el mejor valor (Witenberg, 1999).

Por otra parte, la teoría de la complejidad clasifica a los problemas en tres clases identificadas como P, NP y NP-Completo, donde P es el conjunto de problemas que se puede resolver mediante un algoritmo determinístico en tiempos polinómicos (Bratley y Brassard, 1997), y NP es la clase de problemas que pueden ser resueltos por algoritmos no determinísticos acotados en tiempo polinomial. Por último "Los NP-Completos engloban a los problemas más difíciles de resolver y de acuerdo con sus características solo pueden ser abordados por algoritmos no determinísticos" (Martínez, 2015).

Actualmente los problemas de optimización se pueden encontrar en diferentes áreas: informática, gestión logística, telecomunicaciones, ingeniería, industria,

trasporte, procesos, etc. Dichos problemas se pueden discretizar y modelar para dar soluciones que ayudan a minimizar o maximizar un recurso.

La optimización combinatoria puede formar parte de un conjunto de estrategias para llevar al éxito a una industria, tratando de aprovechar sus recursos de tal manera que sus pérdidas sean mínimas.

1.2 Planteamiento del problema

Saint-Gobain Glass es uno de los líderes mundiales más importantes en la fabricación de vidrio, lo diseña, produce y distribuye.

“Se especializa en la fabricación de vidrio flotado y vidrio recubierto con magnetró, creando una variedad de tipos de vidrio con diferentes características: transparencia, seguridad, control solar, decoración, función de auto limpieza, aislamiento térmico y acústico, etc. Los productos están destinados para una amplia variedad de aplicaciones domésticas y comerciales, incluida el equipamiento de vivienda (ventanas, ventanas panorámicas, diseño de interiores), puertas, desarrollo urbano y la realización de proyectos importantes” (Tlilane y Viaud, 2018).

El vidrio plano se fabrica mediante un proceso de flotación que consiste en mezclar arena, aditivos, vidrio reciclado y carbonato de sodio, fundiéndolo en un horno y creando con ello vidrio líquido que pasa por un baño de estaño donde es moldeado para hacerlo plano e ir creando una cinta finita de vidrio, que corre sobre unos rodillos hasta llegar a un estado sólido para cortarse en hojas grandes llamadas “contenedores”, de un tamaño estándar de seis mil por tres mil doscientos diez milímetros, estos son almacenados para utilizarse en el futuro. En realidad, los contenedores no son perfectos, pueden tener defectos relacionados al proceso de flotación que son detectados y guardados en una base de datos para utilizarse cuando el contenedor se saque del almacén.

Por lo general los contenedores no se comercializan como tal, sobre ellos se crean patrones de corte para colocar pequeñas piezas de vidrio llamadas “artículos”. Todas las piezas para cortar vienen en un “lote” el cual está dividido en “pilas” que contienen en forma ordenada los artículos pedidos por cada cliente.

La Sociedad Francesa de Investigación Operativa y Apoyo a la Decisión(ROADEF) en colaboración con la Sociedad Europea de Investigación Operativa (EURO) y Sain Gobaint Glass, plantean el problema de Corte de Vidrio (ROADEF/EURO, por sus siglas en francés) el cual consiste, en generar un conjunto de patrones de corte bidimensionales que permitan cortar todos los elementos o artículos de un lote I, utilizando los contenedores disponibles de tal manera que se minimice la pérdida geométrica, respetando algunas restricciones, estas pueden ser físicas, relacionadas con el corte del vidrio u organizacional para satisfacer las órdenes de los clientes.

Restricciones relacionadas con el corte de vidrio:

- La superposición entre elementos no está permitida.
- Los artículos se pueden rotar 90°, es decir pueden cortarse en forma horizontal o vertical.
- Todos los artículos de un lote I deben ser cortados, una solución dada debe contener todos los elementos de un lote.
- La sobre producción de artículos no está permitida, solo se deben cortar los artículos del lote a surtir, no es válida una solución con más elementos.
- Los artículos cortados deben estar libres de defectos.
- Los patrones de corte son bidimensionales y la única manera de obtenerlos es mediante cortes guillotina.
- El número máximo de cortes para obtener un artículo es de cuatro.

Restricciones relacionadas con la organización:

- Cada elemento pertenece a una pila y no está permitido modificar el orden en que se cortan los artículos. La precedencia es de suma importancia para poder cortar un artículo, sin embargo, dado que el orden de los elementos solo está dentro de las pilas, se pueden cortar artículos de diferentes pilas de manera desordenada.

- Los contenedores siempre son horizontales el ancho es mayor que el largo; la diferencia entre contenedores es el número de “defectos” que contiene cada uno, consecuencia del proceso de fabricación.

1.3 Motivación e hipótesis

Aunque en la literatura existe un amplio número de artículos científicos que hablan sobre el problema de Corte y Empaquetado o problemas muy semejantes, la mayoría de ellos carecen de restricciones tan específicas como las que encontramos en el problema de Corte de Vidrio ROADEF esto debido a que su enfoque es más general.

La problemática del Corte y Empaquetado no es propia de Saint-Gobain Glass, varias empresas del mismo ramo y otras industrias como metalúrgica, metalmecánica, textil, papelera, maderera, entre otras, se enfrentan a esta problemática con sus propias restricciones, es decir, generan variantes del problema Corte y Empaquetado como sucede con el problema de Corte de Vidrio ROADEF, donde todas las restricciones que contiene no se muestran en ninguna variante de la literatura porque es un problema con las necesidades específicamente de Saint-Gobain Glass, por lo tanto, es una oportunidad para trabajar en el diseño, implementación e investigación de técnicas algorítmicas que ayuden a generar resultados factibles en tiempos cortos.

Derivado de lo planteado en las anteriores secciones y en la presente, en esta tesis se trabajará con las siguientes hipótesis.

Hipótesis H_0: Es posible resolver el problema de corte de vidrio con una hibridación de algoritmos glotones, que genere soluciones factibles para minimizar la pérdida de vidrio.

Hipótesis H_1: No es posible resolver el problema de corte de vidrio con una hibridación de algoritmos glotones.

1.4 Objetivo general

Diseñar e implementar un algoritmo heurístico que genere un conjunto de patrones de corte para los contenedores donde se procure minimizar la perdida de vidrio.

1.5 Objetivos específicos

- Encontrar por lo menos una solución óptima para alguna instancia.
- Obtener soluciones factibles.

1.6 Justificación

El problema de corte y empaquetado es de gran importancia para empresas manufactureras, que involucran el proceso de cortar grandes rollos o tiras de algún material y pasarlo a piezas más pequeñas, “considerando el efecto que este tiene sobre los inventarios del producto en proceso y materias primas. Una apropiada gestión de este problema no solo genera beneficios económicos, sino que también genera impactos sostenibles, en la medida que busca un equilibrio entre el costo de inventario y los desperdicios del proceso” (Peña, Orejuela, González y Andrés, 2017).

La presente investigación se enfoca a resolver un problema de optimización donde el número de restricciones utilizadas es mayor a cualquier problema de Corte y Empaquetado trabajado en la literatura, la mayoría de los problemas que se muestran solo contienen algunas restricciones que maneja este problema.

Particularmente el corte de vidrio representa un problema de gran relevancia para Saint-Gobain Glass, debido a las pérdidas económicas que representan el mal corte del producto, esto sucede por no contar con un buen mecanismo que permita generar un conjunto de patrones de cortes, que ayuden a minimizar el desperdicio de material y por consecuencia las pérdidas económicas. La idea de esta tesis es ayudar a generar patrones de corte que de soluciones factibles en tiempos razonables, reduciendo las pérdidas de vidrio mediante el diseño e implementación de un algoritmo heurístico, en esta clase de problemas NP-duros.

1.7 Alcances y limitaciones de la investigación

1.7.1 Alcances

- Se trabajará con las 50 instancias existentes en el estado del arte, publicadas en la página oficial del Challenge ROADEF/EURO 2018 (ROADEF, 2018).
- Se validará la factibilidad de los resultados (que cumplan las restricciones del problema mencionadas en la sección 1.2).

1.7.2 Limitaciones

- Se limitará a trabajar con algunos algoritmos glotones.
- Solo se utiliza C++ como lenguaje de programación.
- No se trabajará con los defectos porque se pretende abordarlos en trabajos futuros.

1.8 Organización de la tesis

La presente tesis está organizada de la siguiente manera: En el capítulo 1 se habla de cómo surgió la investigación de operaciones y la manera en la que ayuda al ámbito industrial. Se explica el planteamiento del problema del grupo Saint-Gobain, la hipótesis y motivación, los objetivos, la justificación, los alcances y limitaciones de esta investigación. En el capítulo 2 se desarrolla el marco teórico de este problema, de donde proviene, el primer modelo matemático, las técnicas algorítmicas con las que se han resuelto algunas variantes, la explicación y modelación matemática del corte de vidrio ROADEF. En el capítulo 3, se explican los algoritmos implementados para resolver el problema, los criterios de selección y la forma en que fueron hibridados. En el capítulo 4, se muestran las tablas de resultados de todos los algoritmos glotones implementados, al final se toma el algoritmo que dio mejores resultados y se muestra el promedio de la calidad de la solución, su desviación estándar y los tiempos de ejecución. En el capítulo 5 se describen las conclusiones y se habla de trabajos futuros para mejorar las soluciones que se tienen hasta el momento con la hibridación de los glotones.

Capítulo 2 Problemas de Corte y Empaquetado

En este capítulo se desarrolla el marco teórico de los problemas de corte y empaquetado, su definición y clasificación de acuerdo con las características que contienen, el primer modelo creado para este tipo de problemas, los métodos de solución que se han utilizado a través de los años en la literatura, la explicación detallada del problema de corte de vidrio ROADEF y su modelación matemática.

"Lo que conocemos es una gota de agua, lo que ignoramos es el océano". Isaac Newton

2.1 Definición de los problemas de Corte y Empaquetado

Los problemas de corte y empaquetado (C&P, por sus siglas en inglés) surgen de la estrecha relación de los problemas de empaquetado que consisten en llenar contenedores con pequeños elementos, y los problemas de corte que se caracterizan por cortar grandes objetos en pequeños elementos, donde ambos tienen como objetivo principal minimizar el material utilizado, estos problemas al entrelazarse crean el problema C&P que consiste en "la combinación geométrica de artículos pequeños a patrones de empaque que pueden asignarse a contenedores del inventario"(Dyckhoff, 1990).

De estos problemas C&P se desprenden muchas variantes procedentes de la extensa gama de aplicaciones industriales como la madera, aluminio, cuero, vidrio, el paginado de periódicos, etc.

Este tipo de problemas en general son difíciles de solucionar en la práctica, se clasifican en problemas NP-duros, donde actualmente no se sabe de algoritmos exactos con complejidad polinómica que permitan resolverlos y al ser intratables se tienen que recurrir a los métodos aproximados como las heurísticas y metaheurísticas, las cuales hallan buenas soluciones en tiempos razonables, computacionalmente hablando.

2.2 Clasificación de Corte y Empaquetado

En la literatura el problema de C&P aparece con varios nombres, entre los más relevantes se encuentran “cutting stock, trim loss, bin packing, dual bin packing, strip packing, vector packing, knapsack (packing), vehicle loading, pallet loading, container loading, car loading” (Dyckhoff, 1990), todos con la misma estructura lógica.

Ante esta diversidad de trabajos que se muestran, Dyckhoff (1990) presenta una tipología donde hace una clasificación de los problemas de C&P, a través de la definición de criterios establecidos de acuerdo con las características que existen entre ellos.

Esta tipología se basa en la estructura lógica de los problemas C&P, donde intervienen dos criterios importantes, el primer criterio son las figuras geométricas que definen el inventario de la empresa y la demanda del cliente; el segundo criterio es el corte y empaquetado de las figuras, de manera específica se puede establecer como sigue:

1) Se tiene dos grupos de datos básicos cuyos elementos están definidos por cuerpos geométricos (figuras), en un espacio de una o más dimensiones:

    1. Contendores: Material disponible en el almacén, el cual forma parte de un inventario.
    2. Artículos: Elementos pequeños que son parte de una demanda.

2) El corte y empaquetado se crea generando patrones, que son combinaciones geométricas de pequeños elementos asignados a contenedores, estableciendo un lugar a cada pieza. Los desperdicios son figuras que ocupan espacio en los patrones pero que no pertenecen al conjunto de piezas pequeñas demandadas.

Las características esenciales de los problemas C&P se dividen en cuatro:

1) **La dimensión,** la cual delimita la cantidad de dimensiones indispensables para crear la geometría de los patrones, las dimensiones pueden ser en una, dos, tres o n. Los problemas mayores a tres dimensiones surgen cuando se extienden a dimensiones no espaciales, por ejemplo, peso o tiempo.

2) **La asignación,** entre artículos y contenedores se puede dividir en dos formas, la primera es cuando se asignan todos los contenedores a una

selección de artículos y la segunda es cuando se asignan todos los artículos a una selección de contenedores.

3) **La variedad de contenedores,** se distinguen de la siguiente manera, un solo contenedor grande, muchos contendedores todos idénticos o muchos contenedores todos distintos. En algunos trabajos los contenedores son llamados objetos grandes.

4) **La variedad de artículos**, se describe la cantidad y forma de los artículos, los C&P pueden tener escasos artículos, artículos homogéneos, varios artículos de diversas formas disímiles y varios artículos de pocas formas diferentes.

Para complementar la clasificación es importante explicar la forma geométrica de los artículos que deben ser cortados y empaquetados los cuales se pueden dividir en dos tipos:

1) Regulares: Son figuras con escasos parámetros por lo general se tratan de figuras ortogonales.
2) Irregulares: Son formas no simétricas y concavidades.

La forma de cortar y empaquetar piezas regulares está dada por un conjunto de rectángulos que deben introducirse en un contenedor rectangular. El corte y empaquetado de piezas irregular trabaja con un conjunto de piezas cuyos lados y ángulos interiores no son iguales entre sí, estos deben introducirse a menudo en contenedores rectangulares.

En esta investigación abordamos los problemas con piezas regulares.

Según la naturaleza geométrica de los artículos a ser cortados y empaquetados existen dos tipos de patrones:

1) Patrones ortogonales: Cortes perpendiculares a los lados del contenedor, véase Figura 2.1.

   - Guillotina: Corte desde un borde de la placa hasta el borde opuesto, paralelo al borde restante, véase Figura 2.1 (a).

   - No guillotina: No importan las restricciones de guillotina un artículo se puede extraer sin necesidad de hacer cortes de borde

a borde en el contenedor disponible, cuidando siempre que no se traslapen los elementos Figura 2.1 (b).

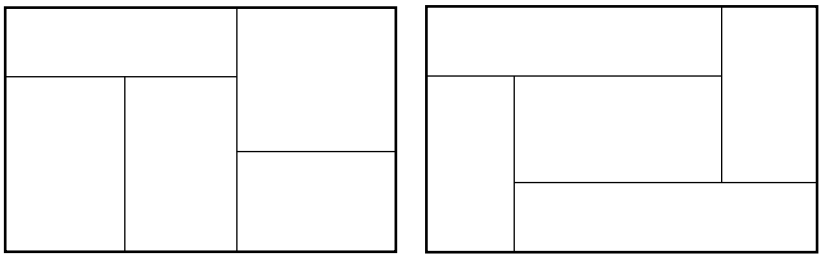


Figura 2.1 Ejemplo de cortes ortogonales.
Donde (a) muestra patrones de cortes en guillotina y (b) patrones de corte no guillotina. Elaboración propia.

2) No ortogonales: cortes que no se hacen en forma perpendicular a los lados del contenedor, Figura 2.2.

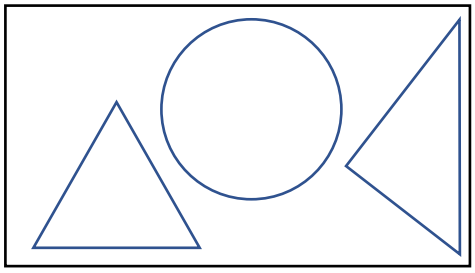


Figura 2.2 Cortes no ortogonales. Elaboración propia.

De toda la diversidad de problemas C&P que muestra la tipología de Dyckhoff, nos centraremos en los problemas de corte y empaquetado bidimensional. De los más estudiados se encuentran el problema de la mochila en dos dimensiones, carga de palets, minimización de área, corte de stock en dos dimensiones, empaquetado de tiras y empaquetado en contenedores bidimensionales (2BP, por sus siglas en inglés), haciendo especial énfasis en

este último, Lodi, Martello y Vigo (1999a) muestran las diferentes variantes del problema:

- 2BP|O|G: Los artículos están orientados (O), esto quiere decir que no pueden ser girados y para ser extraídos se necesitan cortes en guillotina (G).
- 2BP|R|G: Los artículos pueden rotarse noventa grados (R) y para ser extraídos se necesitan cortes en guillotina (G).
- 2BP|O|F: Los elementos están orientados (O) y para ser extraídos el corte es libre (F).
- 2BP|R|F: Los elementos pueden rotarse noventa grados (R) y para ser extraídos el corte es libre (F).

De estos tomaremos como referencia el problema 2BP|R|G que es el más apegado al problema que se aborda en la presente tesis ya que el tema de la investigación contiene aún más restricciones que las que se encuentran normalmente en la literatura.

2.3 Modelo matemático básico del problema Corte y Empaquetado

Gilmore y Gomory (1965), modelan por primera vez el problema de empaquetamiento bidimensional, a través de una amplificación del problema 1BP (empaquetado en contenedores unidimensional). Plantean un enfoque donde crean columnas basadas en la enumeración de todos los subconjuntos de artículos que se pueden empaquetar en un solo contenedor.

En la recopilación de diferentes modelos matemáticos que hicieron Lodi, Martello y Monaci (2002), describen el modelo básico de Gilmore y Gomory de la siguiente manera:

> Sea A_j un vector de columna binario de n elementos a_{ij} $(i = 1, \dots, n)$ tomando el valor 1 si el elemento i pertenece al patrón j-ésimo, y el valor 0 en caso contrario. El conjunto de todos los patrones posibles se representa mediante la matriz A, compuesta por todas las columnas A_j

posibles $(j = 1, \dots, M)$. Donde x_j toma el valor 1 si el patrón j pertenece a la solución de lo contrario toma el valor 0.

Función objetivo: $\min \sum_{j=1}^{M} x_j$

Sujeto a:

$$\sum_{j=1}^{M} a_{ij} x_j = 1 \qquad (i = 1, \dots, n), \qquad (1)$$

$$x_j \in \{0,1\} \qquad (j = 1, \dots, M), \qquad (2)$$

Función objetivo: busca minimizar la sumatoria de patrones que pertenezcan a una solución.
Donde:

- La restricción (1) impone que todos los patrones estarán en la solución

- La restricción (2) determina que x_j solo contiene valores binarios 1 si es parte de la solución y 0 en caso contrario.

2.4 Métodos de solución de Corte y Empaquetado bidimensional

El problema de corte en una dimensión fue propuesto por primera vez por el economista ruso Kantorovich (1939), desde ese momento los trabajos aumentaron y en la década de los setentas aparecieron las primeras encuestas donde hicieron notable una estrecha relación entre los problemas de corte que buscan minimizar la pérdida de material y los problemas de empaquetado que tienen como objetivo reducir el espacio desperdiciado, dando lugar al problema de corte y empaquetado (Dyckhoff, 1990).

Lodi, Martello y Vigo (1999b), mencionan que el problema es NP-duro porque se puede transformar al conocido problema de empaquetamiento de contenedores unidimensional el cual se encuentra en la clase de los NP-duros.

Dicho problema en forma general es abordado en diferentes sectores industriales por lo cual la literatura científica tiene muchos trabajos con diferentes variantes.

En esta revisión de literatura describimos seis diferentes métodos de solución para diferentes problemas de C&P bidimensionales con figuras regulares.

2.4.1 Programación dinámica con recursión

De acuerdo con Levitin (2000), la programación dinámica es una técnica para resolver problemas, dividiéndolos en subproblemas más pequeños y resolviendo cada uno de ellos una sola vez, los resultados se registran en una tabla desde la cual se puede obtener una solución al problema original, por otro lado, la recursividad según Goodrich y Tamassia (2002), es una técnica de programación donde una función se llama así misma como subrutina garantizando su terminación en algún punto. En esta sección describimos algunos trabajos que involucran la combinación de programación dinámica y recursividad.

Uno de los primeros trabajos que se realizó para el problema de corte de stock bidimensional (CSP, por sus siglas en inglés) utilizando guillotina fue el de Gilmore y Gomory (1965), generaron un modelo matemático donde el problema se dividió en dos etapas (cortes) cada una era un BP-unidimensional, las etapas consistían en cortar en forma paralela a un borde del contenedor para crear tiras y posteriormente cortar de forma individual cada tira en la orientación perpendicular al primer corte, utilizando programación dinámica con recursión en instancias creadas aleatoriamente.

Más tarde Hahn (1968), soluciona un CSP de guillotina de tres etapas, donde las hojas de material pueden tener defectos rectangulares en ciertas áreas. Utilizó la programación dinámica específicamente para maximizar los valores de los tamaños.

Años después Beasley (1985), encontró un error relacionado con la recursión de programación dinámica dada para el corte bidimensional en etapas dentro del trabajo de Gilmore y Gomory, desarrollando una recursión con programación dinámica correcta, explicando que el error aparece a causa de no poder desarrollar distintas funciones para distintas direcciones de corte de la primera etapa. El algoritmo corregido muestra que para el corte de la etapa-*i*, la orientación de corte de la etapa-1 es fija, la recursión de Gilmore y Gomory

altera la orientación del corte de la etapa-1 con respecto a la etapa-i. Un ejemplo de la recursión incorrecta es cuando i = 3, el corte en la etapa-1 la orientación es paralela al eje y, pero con i= 4, la orientación de corte en la etapa-1 es paralela al eje x. A pesar del error en la implementación del algoritmo recursivo los siguientes trabajos fueron basados en la idea de Gilmore y Gomory.

Fayard, Hifi y Zissimopoulos (1998), generaron una nueva versión de su mejor algoritmo BSC (Best Strip Cutting) implementado en el año 1995, para resolver una de las variantes del problema de corte bidimensional, la nueva versión del algoritmo se llamó GBSC (for General Best Strip Cutting) y estuvo diseñado para resolver todas las variantes del problema de corte bidimensional utilizando programación dinámica. El algoritmo consta de tres procedimientos, SGP (Strips Generation Procedure) el cual crea tiras horizontales de ancho fijo, FP (Filling Procedure) crea tiras verticales de altura fija y DP (Discretisation Procedure) este decide cual de esos cortes son los mejores para utilizar en el patrón de la solución.

2.4.2 Árboles AND/OR

Sahni y Horowitz (1978), mencionan que los árboles orientados AND/OR desglosan problemas complejos en varios subproblemas, representados por una distribución gráfica donde los nodos representan problemas y los descendientes de un nodo representan los subproblemas relacionados con él.

Morabito, Arenales y Arcaro (1992), presentaron un enfoque de un árbol orientado AND/OR para el problema de corte de guillotina sin etapas y sin restricciones, el trabajo fue extendido por Morabito y Arenales (1996), para el problema de corte de guillotina por etapas y restringido, el árbol genera un patrón de corte donde AND es un arco que representa un corte el cual une a un nodo (rectángulo) a sus dos sucesores y OR es un arco que une a un nodo final. Cuando la solución inicial se generó, una heurística hibridada de Morabito y Arenales (1994), es aplicada, la hibridación combina *back-tracking* (BT) para generar un subárbol AND/OR y *hill-climbing* (HC) para tomar la ruta más valiosa.

2.4.3 Árboles de enumeración

Christofides y Whitlock (1977), presentaron un enfoque del CSP de *n* etapas: crearon un algoritmo de búsqueda de árbol en el cual cada nodo representa el estado de un rectángulo recortado, y cada ramificación representa un corte. Un procedimiento enumerativo estuvo diseñado para generar los patrones de corte normales que significa desplazar todos los elementos al lugar más bajo a la izquierda. Para limitar la búsqueda, se calculó un límite superior en cada nodo, como sigue: Para los rectángulos que no se cortarán más, se resuelve el problema del transporte asociado; para todos los restantes rectángulos, se resuelve el CSP bidimensional sin restricciones utilizando, el principio de Gilmore y Gomory.

Lodi, Monaci y Pietrobuoni (2017), presentan un algoritmo heurístico basado en la enumeración parcial, para el problema 2BP|O|G, el algoritmo es de naturaleza heurística porque no prueba todas las combinaciones posibles para empaquetar los artículos. Los nodos que son hojas del árbol pertenecen a soluciones completas, donde todos los artículos se han empaquetado. Los nodos intermedios tienen un número de nodos descendientes, asociados con diferentes formas de empaquetar el siguiente artículo usando cortes guillotina. La forma de llenar las bandejas individualmente es aplicando una estrategia para empaquetar un artículo a la vez, mediante una regla de selección y una regla de división de guillotina definidas. La regla de selección establece el siguiente artículo a empaquetar y su posición en el contenedor, por otro lado, la regla de división de guillotina se utiliza para verificar que el patrón producido sea de tipo guillotina. Al experimentar con instancias de la literatura el algoritmo muestra que es capaz de resolver más del setenta y ocho por ciento de los problemas, por lo que se concluye que el algoritmo implementado tuvo un desempeño aceptable.

2.4.4 Búsqueda tabú

Laguna, Taillard y de Werra (1993), definen a la búsqueda tabú (TS, por sus siglas en inglés) como una metaheurística que guía un procedimiento de búsqueda heurística local para explorar el espacio de la solución fuera del optimo local. Se basa en utilizar memoria adaptativa y exploración sensible para poder calificar de inteligente una solución.

Lodi et al. (1999b), generaron un trabajo muy completo para el problema 2BP|R|G, crearon una heurística original que trabaja en dos fases, para poder

utilizarla posteriormente en una búsqueda tabú. El procedimiento de la heurística original es descrito a continuación, se inicializa un nivel en el contenedor colocando una pieza con su borde izquierdo tocando el borde izquierdo de la tira, en la posición más baja admisible: llamamos piso o techo del nivel a la línea horizontal tocada por el borde inferior arriba de tal pieza. Los algoritmos FFFRG y FBSRG de Lodi et al. (1999b), empaquetan las piezas exclusivamente en el piso. El algoritmo Piso-Techo (FCRG), en su lugar, empaqueta las piezas en el nivel ya sea con su borde inferior tocando el piso o con su borde superior tocando el techo, de tal manera que la guillotina puede cortar las piezas asignadas a los niveles. En el piso, las piezas se empaquetan de izquierda a derecha, mientras que en el techo se empaquetan de derecha a izquierda intentando llenar el espacio sobre las piezas más bajas empaquetadas en el piso. Una pieza empacada en el piso siempre tiene su borde izquierdo tocando el borde derecho de otra pieza empacada en el piso, mientras que las piezas empacadas en el techo pueden estar separadas por espacios vacíos para permitir el empaque de guillotina, véase Figura 2.3

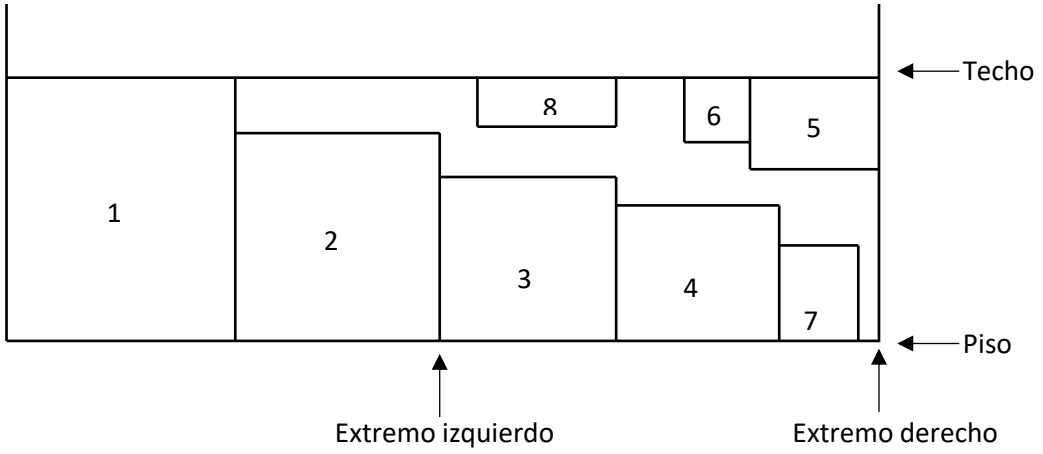


Figura 2.3 Patrones producidos por el algoritmo FCRG.(Lodi et al., 1999b).

FCRG es utilizada para trabajar en búsqueda tabú junto con un límite inferior calculado llamado LB, se define UB como el valor de la mejor solución encontrada, si $UB>LB$ búsqueda tabú se inicializa con la solución producida por el algoritmo FCRG. Donde z es número de contenedores usados en la

solución actual. Búsqueda tabú acepta movimientos que reducen a z o vuelven a distribuir las piezas entre las bandejas, los movimientos que empeoran a z

no son aceptados. La búsqueda finaliza si se determina una solución viable de valor LB, o si se alcanza un número máximo de iteraciones. Los vecindarios que tiene son dos:

Primer vecindario

Consiste en tomar cada pieza no tabú del contenedor más débil (contenedor con escasas piezas) de la solución actual, y calcular FCRG para todos los contenedores que no son débiles (contenedores con abundantes piezas), cuando se encuentra un espacio en algún contenedor no débil, la pieza se mueve y se inserta a la lista tabú, el proceso es repetido para las siguientes piezas. Existe un criterio de aspiración, que consiste en verificar el número de piezas que aún se encuentran en el contenedor débil si solo existe una entonces la búsqueda se realiza incluso si la pieza es tabú, pues el movimiento mejoraría a z. Cuando varios movimientos vacían al contenedor más débil, z es reducido y se determina el nuevo contenedor más débil para que la búsqueda continúe.

Segundo Vecindario

Consiste en tomar una pieza no tabú del contenedor más débil y elegir los dos contenedores siguientes más débiles de la solución calculando FCRG con estos elementos con el fin de disminuir a z.

Se corrió búsqueda tabú para casos individuales de hasta 164 piezas, con instancias de la literatura que son para el problema de cortes bidimensional pero que se trasformaron para 2BP|R|G en mucho de los casos la metaheurística logro llegar al óptimo de las instancias.

2.4.5 Algoritmos constructivos

Wang (1983), mostró dos algoritmos aproximados constructivos de n etapas para el CSP bidimensional restringido. Sus métodos trabajan de forma muy distinta a los tradicionales que dividen las hojas del stock, los algoritmos que implementa construyen repetidamente rectángulos verticales u horizontales para crear un patrón de guillotina más grande, dejando de lado los rectángulos que su desperdicio en porcentaje supere una cota máxima. En el primer

algoritmo el porcentaje de desperdicio es con relación al área de total la hoja del stock y en el segundo algoritmo es con respecto al área del rectángulo de guillotina más pequeño. De esta forma, se crean todos los rectángulos de guillotina factibles y se toma aquel que tenga la pérdida mínima.

Messaoud, Chu y Espinouse (2008), proponen un algoritmo llamado BMCE con una complejidad polinómica de $O(n^2)$, su objetivo es verificar si los elementos acomodados en un contenedor tienen patrones de corte en guillotina. A continuación, se describe la forma en que trabaja el algoritmo, el número de iteraciones que realiza se definen por el total de elementos a cortar, cada vez que el algoritmo itera divide un patrón, este podría ser inicial o un subpatrón obtenido en iteraciones anteriores, que incluye mínimo dos elementos en subpatrones separados realizando cortes de guillotina. El procedimiento es repetido hasta que todos los elementos correspondan a un subpatrón o no sea posible generar cortes de guillotina en al menos un patrón que tenga mínimo dos elementos. La división de un patrón dado que contiene al menos dos elementos en dos subpatrones dispone de dos pasos: el primero es determinar todos los segmentos horizontales en el eje x o verticales en el eje y, construyendo intervalos, en el paso dos, fusiona todos los intervalos de la misma naturaleza, es decir, todos los intervalos horizontales o verticales, para posteriormente determinar si sobre estos intervalos se pueden incluir cortes horizontales o verticales.

Charalambous y Fleszar (2011), presentaron un nuevo algoritmo heurístico para resolver el problema 2BP|*|G (con y sin rotación en los artículos). La heurística construye una solución empaquetando un contenedor a la vez. Mientras aun queden artículos por empaquetar, existe la posibilidad de abrir un nuevo contenedor, un patrón se construye utilizando un subconjunto de los artículos sin empaquetar y colocándolos en el contenedor sin que estos sobrepasen su capacidad. El patrón se representa por el subconjunto de elementos y sus coordenadas. Cada patrón se construye con el procedimiento genPattern el cual consiste en generar varios rectángulos en los espacios vacíos del contenedor para probar si algún artículo no empaquetado cabe en algún rectángulo vacío.

Côté y Lori (2018), generaron un nuevo principio para la construcción de patrones llamado MIM(Meet-in-the-Middle) con una idea muy diferente a la creación de patrones normales los cuales consisten en alinear en el fondo y a la izquierda del contenedor un elemento. Este tipo de patrones se utilizan en varias técnicas de C&P porque ayudan a reducir el espacio de búsqueda y al

mismo tiempo conservan la optimalidad, pero tienen una limitante, su número aumenta de manera constante cuando crece el número de elementos y el tamaño de la bandeja. En MIM se puede alinear en a) al fondo y a la izquierda o bien b) a la derecha y arriba. El principio MIM no aumenta el número de patrones, de la misma forma conserva la optimalidad y su cálculo tiene la misma complejidad de tiempo que la de los patrones normales. El algoritmo comienza determinando un umbral que puede tomar el valor desde uno hasta el ancho total del contenedor, posteriormente el umbral se fija en alguna parte de la dimensión con la que se determinó por ejemplo la mitad del contenedor, forzando a que todos los elementos que su borde izquierdo se encuentren a la izquierda del umbral se alineen de ese lado y obligando a que los elementos restantes se alineen a la derecha. Después el proceso se repite para la siguiente dimensión hasta terminar de construir la solución. Demostraron que la exploración de un óptimo puede restringirse a soluciones donde los elementos están empaquetados en un patrón MIM, que el número de patrones MIM siempre será menor que el de los patrones normales y que en la práctica los patrones MIM dan mejores resultados que los patrones normales porque su número generalmente es más pequeño y se pueden obtener con el mismo esfuerzo computacional.

Fleszar, K. (2013). Desarrolla tres heurísticas constructivas, la primera es una heurística de inserción de primer ajuste (FFIH), la segunda una heurística de inserción de mejor ajuste (BFIH), la tercera es una heurística de inserción de ajuste crítico (CFIH) y una nueva heurística de mejora de justificación para el problema 2BP|*|G (con o sin rotación en los artículos),comenzando con una solución inicial la cual consiste en generar un conjunto de árboles donde cada árbol representa un contenedor con sus patrones de corte verticales u horizontales, los cuales se construyen a través de tres tipos de inserción que se describen a continuación:

- AsSubnode se crea un subnodo en el nodo vertical u horizontal y de este no se crean nuevos subnodos.
- HorizRoot / VertRoot se crea un nuevo nodo vertical u horizontal para ser utilizado como nodo raíz.
- InParallelTo consiste en insertar un nodo vertical u horizontal en el subconjunto de nodos y si este requiere la creación de subnodos se crearán nuevos subconjuntos con subnodos.

Las heurísticas trabajan de la siguiente manera, la heurística de inserción de primer ajuste (FFIH) y la heurística de inserción de mejor

ajuste (BFIH) toman un artículo desempaquetado en cada etapa e intenta insertarlo en uno de los contenedores iniciados anteriormente. Si tal inserción es posible, FFIH adopta la mejor inserción en la primera bandeja en la que encaja el artículo, mientras que BFIH adopta la mejor inserción entre todas las posibles inserciones en todas las bandejas. Si el elemento no se puede insertar en ninguno de los contenedores iniciados, se agrega un nuevo contenedor y el elemento se inserta en él. La siguiente heurística de ajuste crítico (CFIH) toma en consideración todos los artículos que aun sobran en cada etapa, la idea es verificar cual elemento domina a otro es decir el artículo que sea mayor en largo y ancho será el dominante. Para cada elemento no dominado, CFIH determina la cantidad de contenedores en los que se puede insertar el elemento y la mejor inserción del elemento; luego, de todos los elementos no dominados se elige el elemento crítico, que es el articulo con el menor número de contenedores en los que es posible la inserción, el elemento crítico se inserta según su mejor inserción, o si no es posible la inserción en contenedores iniciados anteriormente, se agrega un nuevo contenedor y se inserta. Por último, la heurística de justificación intenta reducir el número de contenedores utilizados en una solución dada, eliminando repetidamente el ultimo patrón de la solución anterior e insertando los elementos del patrón destruido en una nueva solución si después de haber construido la nueva solución es factible y esta disminuye el número de contendedores, se reemplaza por la solución anterior. Todas las heurísticas tienen una complejidad computacional cuadrática del caso más desfavorable, excepto la heurística de inserción de ajuste crítico que tiene una complejidad computacional cúbica del caso más desfavorable.

2.4.6 Algoritmo glotón

De acuerdo con Cormen, Leiserson, Rivest y Stein (2009), un algoritmo codicioso siempre hace la mejor elección óptima en cada paso local con la esperanza de que esta elección conduzca a una solución global óptima.

Las características de un algoritmo glotón son, (Levitin, 2000):

- **Factible**, debe satisfacer las limitaciones del problema.

- **Óptimo a nivel local**, tiene que ser la mejor opción local de todas las opciones posibles en ese paso.
- **Irrevocable**, una vez tomada la opción no se puede cambiar en un paso posterior en el algoritmo.

Ventajas:

- Son fáciles de implementar.
- Producen soluciones eficientes.
- En algunas ocasiones encuentran la solución óptima.

Desventajas:

- No todos los problemas de optimización se solucionan con algoritmos glotones.
- La búsqueda de un óptimo local no implica encontrar un óptimo global.
- Es difícil encontrar un criterio de selección que garantice la elección óptima.

Elementos que intervienen:

- Conjunto de C candidatos.
- Criterio de selección.
- Función de factibilidad.
- Función de selección.
- Una función objetivo.

Brassard y Bradley (1997), describen el siguiente pseudocódigo del algoritmo glotón:

función codiciosa (C: conjunto): conjunto

{C es el conjunto de candidatos}

$S \leftarrow \varnothing$ {Construimos la solución en el conjunto S}

Mientras $C \neq \varnothing$ **y no** *solución (s)* **hacer**

$x \leftarrow$ *seleccionar* (C)

$C \leftarrow C \setminus \{x\}$

Si *factible* ($S \cup \{x\}$) **entonces** $S \leftarrow S \cup \{x\}$

Si solución(S) **entonces devolver** S

Sino devolver <<*no hay soluciones*>>

Lodi et al. (1999b), adaptan los algoritmos glotones de Berkey y Wang (1987), los cuales dan buenos resultados para el problema 2BP|O|G. Modificando dichos algoritmos para el problema 2BP|R|G, los algoritmos adaptados son:

FFFRG (Finite First Fit) crea filas formando niveles en los contenedores. El primer nivel es la base de un contenedor. Los elementos se empaquetan de izquierda a derecha en el nivel elegido. Cuando se necesita un nuevo nivel, se crea a lo largo de la línea horizontal que coincide con la parte superior de la primera pieza más alta empaquetada en el nivel inferior. La primera pieza que inicializa un nivel siempre se empaqueta horizontalmente, para minimizar la ocupación vertical del contenedor, cuando una pieza se empaqueta en un nivel existente, si ambas orientaciones son posibles, se le da prioridad a la vertical esto con el fin de minimizar la ocupación horizontal.

FBSRG (Finite Best Strip) funciona en dos fases. Primero, una tira de ancho W y largo infinito se divide en niveles horizontales de acuerdo con la siguiente estrategia. Las piezas se empaquetan en el nivel factible que tiene el espacio horizontal residual mínimo, cuando un nivel está lleno, se crea un nuevo nivel. Si la orientación de ambas piezas es factible, entonces se usa la vertical, para maximizar la probabilidad de empaquetar más piezas en el nivel. En la segunda fase, las bandejas se combinan en bandejas finitas utilizando el conocido algoritmo de reducción de ajuste óptimo.

Ambos algoritmos comienzan ordenando las piezas respecto a minimizar la altura del contenedor.

El trabajo de Lodi et al. (1999b) para el problema 2BP|R|G, muestra un punto de partida para poder resolver el Problema de Corte de Vidrio ROADEF con algoritmos glotones, se puede notar que esta clase de algoritmos son buenos para resolver problemas donde no solo intervienen combinaciones también es importante la geometría de los artículos para obtener soluciones con buena calidad, así que tomando en consideración estas características en el siguiente capítulo se abordará como se diseñaron criterios de selección para diferentes algoritmos glotones.

2.5 Problema de Corte de Vidrio ROADEF

Este problema tiene una similitud con el problema de 2BP|R|G, aunque por ser para una empresa en específico sus restricciones se adaptan a las necesidades de la industria, lo cual incrementa considerablemente la complejidad del problema.

El problema de Corte de Vidrio ROADEF plantea lo siguiente: se desea generar un conjunto de patrones de corte bidimensionales que permitan cortar todos los artículos it de un lote I utilizando un conjunto de contenedores finitos disponibles B={b_1, b_2, …, b_n}, de tal manera que se minimice la pérdida geométrica total del subconjunto de contenedores utilizados en una solución, respetando las restricciones del problema, estas pueden ser físicas, relacionadas con el corte del vidrio u organizacional para satisfacer las órdenes de los clientes.

A continuación, establecemos algunas definiciones importantes del problema y las restricciones que se deben cumplir en forma específica para cada solución dada.

Definiciones importantes:

Contenedor: Es una hoja de vidrio de tamaño estándar de ancho W=6,000 mm por largo H=3,210 mm, que sale del proceso de flotación. Estas hojas son almacenadas formando parte de un conjunto B, para después cortarse en piezas más pequeñas.

Artículo: Es una pieza de vidrio it para ser cortada, conformada por un ancho w y un largo h.

Pila: Es un conjunto S de artículos donde existe un nivel de precedencia para ser cortados es decir en una pila S= {it_1, it_2, it_3, …, it_n} un artículo it_k no puede ser cortado sin haber cortado antes un artículo it_l cuando k>l.

Lote: Es un conjunto I de pilas a cortar I= S_1 ∪ S_2 ∪ S_3, …, S_m

Defecto: Son imperfecciones en algunas áreas del contenedor definidas por d en una tupla (x_d, y_d, w_d, h_d) donde (x_d, y_d) son sus coordenadas de ubicación

en el contendor y (w_d, h_d) son las longitudes en ancho y largo respectivamente del defecto.

Pérdida: Son partes rectangulares de un contenedor que no es un artículo dentro de los patrones de corte.

Residuo: Esta representado por el sobrante a la derecha del último Corte-1 realizado en el ultimo contenedor utilizado. Dicha pieza es almacenanda para utilizarse en otras demandas.

Corte-α: Es un corte con guillotina.

Las restricciones que debe cumplir una solución se describen a continuación:

- Los artículos para cortar deben seguir el orden de la pila dado, pero se pueden cortar artículos de diferentes pilas siempre respetando la precedencia, por ejemplo, en la Figura 2.4, de las cuatro pilas que se muestran no se puede cortar un artículo it_3 sin haber cortado it_1 e it_2, pero si es válido cortar it_1, it_4, it_6 ó it_9 porque entre elementos de diferentes pilas el orden de precedencia no aplica.

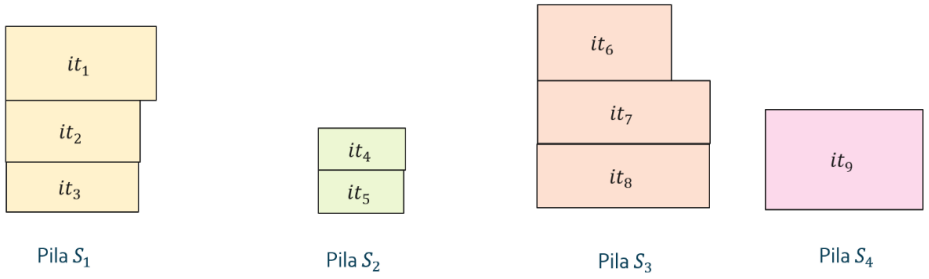
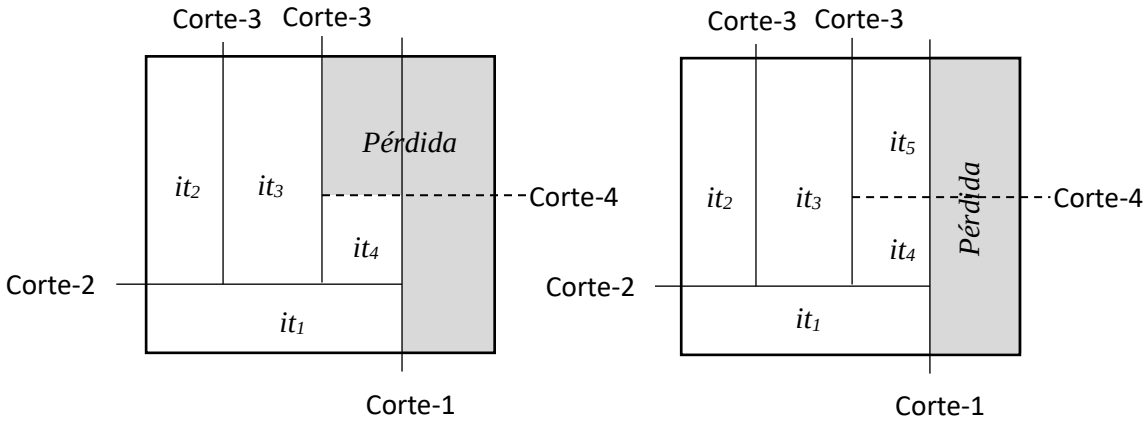


Figura 2.4 Pilas con diferentes números de artículos. Elaboración propia.

- Cada elemento pertenece a una pila S y no es permitido modificar el orden de los elementos.
- Solo se permite rotar noventa grados los artículos para acomodarse.
- Todos los artículos de un lote I deben ser cortados sin producir artículos demás en la solución.
- El traslape entre elementos no está permitido.
- Cada artículo cortado debe estar libre de defectos.

- Los contenedores no se pueden rotar y siempre están en forma horizontal, es decir, $W>H$.
- El área de pérdida mínima es de 20 mm por 20 mm.
- El número de etapas para extraer un artículo es de máximo cuatro, se definen de la siguiente manera:
  - ❖ Corte-1: Es un corte vertical que siempre parte de la hoja principal, el mínimo espacio entre dos Cortes-1 es de 100 mm y el máximo es de 3,500 mm
  - ❖ Corte-2: Es un corte horizontal que viene de un Corte-1, el mínimo espacio entre dos Cortes-2 es de 100 mm y el máximo lo delimita el largo del contenedor.
  - ❖ Corte-3: Es un corte vertical que viene de un Corte-2.
  - ❖ Corte-4: Es un corte horizontal que viene de un Corte-3. Este corte divide en dos partes a un rectángulo dado, una parte será un artículo y el otro desperdicio.

- Los patrones de corte o etapas que se pueden utilizar para extraer un artículo son de máximo tres, permitiendo hacer un cuarto corte cuando se trata de extraer específicamente una pérdida (área sombrada) y no un artículo it, la Figura 2.5 (a) muestra patrones de corte válidos y la Figura 2.5 (b) patrones de corte inválido.



(a) Patrón válido

(b) Patrón inválido

Figura 2.5 Patrones de corte para extraer un artículo.
En (a) muestra patrones de corte válidos, el Corte-4 se utiliza para obtener a it_4 y extraer la pérdida; en (b) los patrones de corte no son válidos porque de un Corte-4 se extrae el articulo it_4 e it_5. (Tlilane y Viaud, 2018).

Formulación matemática.

Sea P el conjunto de patrones de Corte-1 factibles. Para que un patrón de corte $p \in P$ sea factible se requiere lo siguiente:

a) Cada Corte-α está representado por cut_i^{α} y pertenece al conjunto CUT^{α} de cortes posibles que se pueden generar, donde $cut_i^{\alpha} \in CUT^{\alpha}$.

b) Si p tiene al menos un artículo it, el ancho p^w está en el rango [100 mm, 3500 mm], es decir $100 \leq p^w \leq 3500$. Si p no tiene ningún artículo (p será el residuo R si se trata del último patrón creado en el último contenedor, o será un desperdicio en caso contrario) $p^w \geq 20$.

c) Cada patrón p está conformado por 0, 1, 2 o más Cortes-2. Tiene 0 Corte-2, cuanto el patrón es un desperdicio, residuo o todo el Corte-1 representado por el patrón, pertenece a un artículo it. Si tiene uno o más cortes, p está definido por la siguiente fórmula: $p = \sum_{i=1}^{|CUT^2|} cut_i^2$. Donde cada Corte-2 está representado por la variable cut_i^2. A su vez, cada corte cut_i^2 está conformado por 0, 1, 2 o más Cortes-3. Tiene 0 Corte-3, cuando el patrón es un desperdicio o todo el Corte-2 representado por cut_i^2 pertenece a un artículo it. Si tiene uno o más cortes, el patrón cut_i^2 está definido $cut_i^2 = \sum_{i=1}^{|CUT^3|} cut_i^3$. La sumatoria del largo de cada Corte-2, deberá ser igual al largo H del contenedor. A su vez, cada Corte-3 tiene 0 ó 2 Cortes-4. Tiene 0 Corte-4, cuando el patrón es un desperdicio o todo el Corte-3 representado por cut_i^3 pertenece a un artículo it. Cuando cut_i^3 tiene 2 Cortes-4 $cut_i^3 = cut_{i_1}^4 + cut_{i_2}^4$.

d) Cada artículo $it \in p$, o desperdicio $\delta_i \in \Delta$ conjunto de desperdicio, guarda un orden $O_I(it)$ de acuerdo con el árbol de corte. Se enumeran las hojas del árbol de izquierda a derecha según aparecen en el árbol. El primer nodo hoja del árbol se numera con el uno, el siguiente nodo en aparecer se numera con el dos y así sucesivamente hasta numerar todas las hojas del árbol, entonces, para cada par de artículos it_i, it_j que pertenecen a la misma pila y pertenecen a p, $O_I(i) < O_I(j)$ si $S(it_i) < S(it_j)$, ver Figura 2.6.

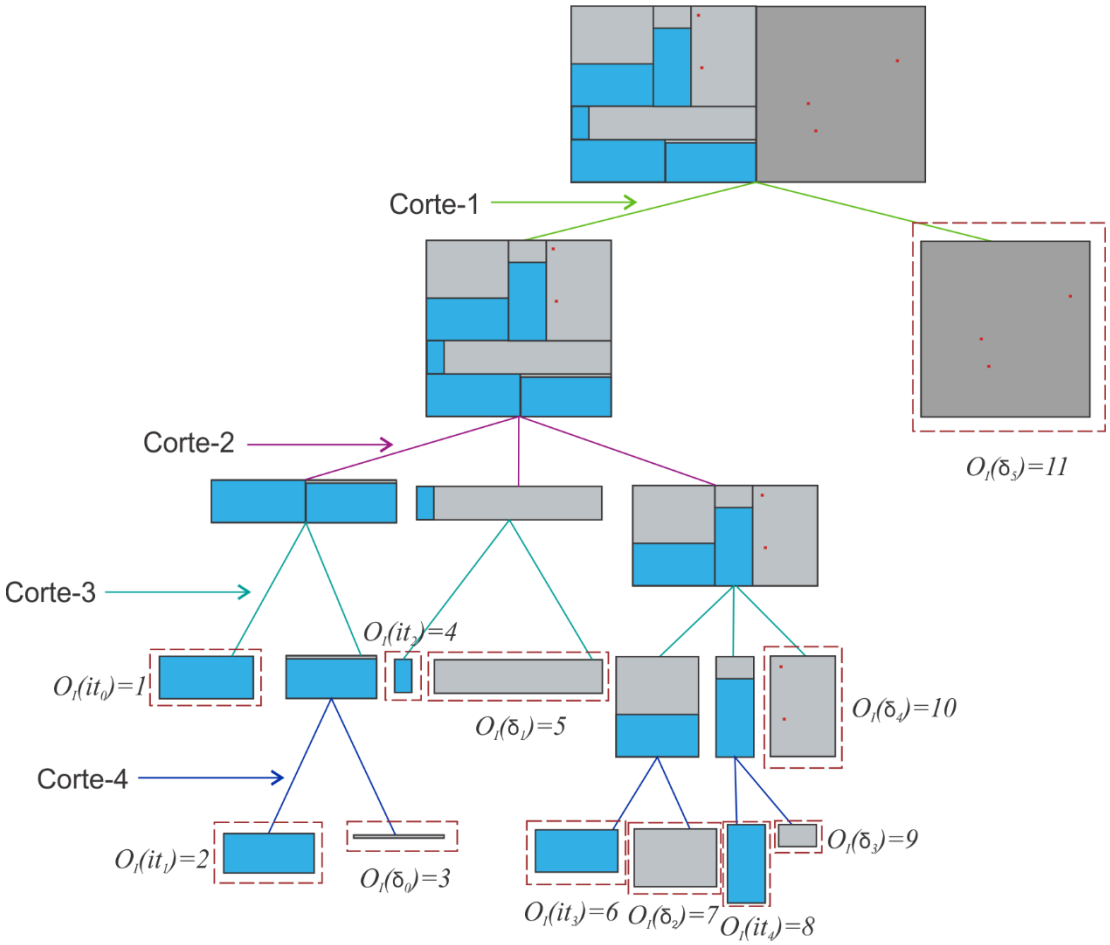


Figura 2.6 Árbol de corte.
Cada arco representa el tipo de corte hecho y cada nodo representa el estado del vidrio en cada subnivel del árbol, hasta llegar a las hojas que representan los artículos extraídos en azul, la pérdida generada en gris claro y el residuo en gris oscuro. Elaboración propia.

e) El conjunto de artículos de un patrón p está definido por p^{it}, dado un subconjunto de patrones $P' \in P$, definimos el subconjunto P' como una solución factible al problema de corte. Cuando utilizamos la función $O_I(it)$ en conjunto con el orden de los patrones P' nos da el número correspondiente al corte del artículo en un bosque (unión disjunta de árboles), es decir, si tenemos un bosque de corte de patrones ordenados, la función O_I, para el primer árbol nos dará el orden de 1 a n_1, para el segundo árbol nos dará el orden de n_1 + 1 hasta n_2, para el

tercer árbol nos dará el orden de n_2 + 1 hasta n_3 , y así sucesivamente hasta terminar el bosque. La función $O_P(p_i)$, nos dará el orden del patrón p_i en una solución dada. Es decir, si $p_i \epsilon P'$, $O_P(p_i)$ tendrá un número positivo, representando el orden en que se encuentra en la solución, 0 en caso contrario.

Modelo matemático.

$$\text{Min} f(Z) = \sum_{i=1}^{|P|} p_i^w \lambda_i$$

Sujeto a:

$$\sum_{i=a_k}^{a_{k+1}} p_i^w = W \quad a_k \in A(a_1, a_2, \dots, a_n) \text{ donde } k \text{ sea impar}$$
$$p_i^w \in P' \text{ y } O_P(p_i) < O_P(p_{i+1}) \ \forall\, i \in P' \tag{1}$$

$$d_{x\,ini, y\,ini} \notin it \ \forall\, it \in I \tag{2}$$

$$d_{x\,fin, y\,fin} \notin it \ \forall\, it \in I \tag{3}$$

$$\sum_{p \epsilon P'} |p^{it}| = |I| \tag{4}$$

$$p_i \cap p_j = 0 \quad \forall\, p_i, p_j \in P' \qquad \text{con } i \neq j \tag{5}$$

$$O_I(it_i) < O_I(it_j) \ \forall\, i, j \in S(it = (1, \dots, n)) \text{ donde } \forall\, i < j \in S \tag{6}$$

$$A, \text{a, i, j, p, w, R, d, x ini, y ini, x fin, y fin, it, I} \geq 0 \tag{7}$$

Dado:

A: Conjunto de patrones en una solución.

a_k: Patrón k en una solución donde $a \in A$ y $k = (1, \dots, n)$

p_i^w: Patrón i construido en cada Corte-1 donde $i = (1, \dots, P')$.

λ_i: Variable binaria que determina si un patrón pertenece a la solución

w: Ancho del patrón.

P: Conjuntos de patrones posibles para generar una solución.

P': Subconjunto de patrones que generan una solución.

W: Ancho del contenedor.

$d_{x\,ini}$: Coordenada inicial en x de un defecto en el contenedor.

$d_{y\,ini}$: Coordenada inicial en y de un defecto en el contenedor.

$d_{x\,fin}$: Coordenada inicial en x de un defecto en el contenedor.

$d_{y\,fin}$: Coordenada final en y de un defecto en el contenedor.

it: Artículo a cortar de un lote.

I: Lote de piezas de vidrio a cortar.

Función objetivo: Minimizar la sumatoria en longitudes de todos los patrones que pertenecen a una solución.

A continuación se describen las restricciones:

- (1) especifica que los patrones utilizados en cada contenedor deben sumar el ancho W (6000 mm).
- (2) y (3) especifican que no debe haber ningún defecto en las coordenadas donde fueron asignados los artículos.
- (4) impone que todos los artículos cortados en una solución deben ser igual a los solicitados.
- (5) especifica que para cada par de patrones utilizados no debe de contener ningún artículo igual, es decir los patrones utilizados en la solución debe ser conjuntos disjuntos.
- (6) indica que se debe respetar el orden de precedencia al cortar los artículos.
- (7) especifica el dominio de las variables.

Ejemplo:

Se tiene un lote a cortar con los datos de la instancia A1 que se encuentran en la tabla 2.1.

Tabla 2.1 Datos de la instancia A1.

id(it)	*w_it*	*h_it*	*S*	*Secuencia*
0	758	1578	0	1
1	1550	738	0	2
2	276	581	0	3
3	1396	781	0	4
4	648	1426	0	5

Se sabe que existe un conjunto de P patrones para poder cortar los artículos solicitados en el lote, respetando todas las restricciones. A continuación, se muestra cómo se eligen los patrones para crear una solución factible:

La Figura 2.7 muestra el posible conjunto de P patrones a tomar para una solución, cada patrón generado respeta los incisos del a) al e) que se encuentran en la formulación matemática, para generar una solución factible.

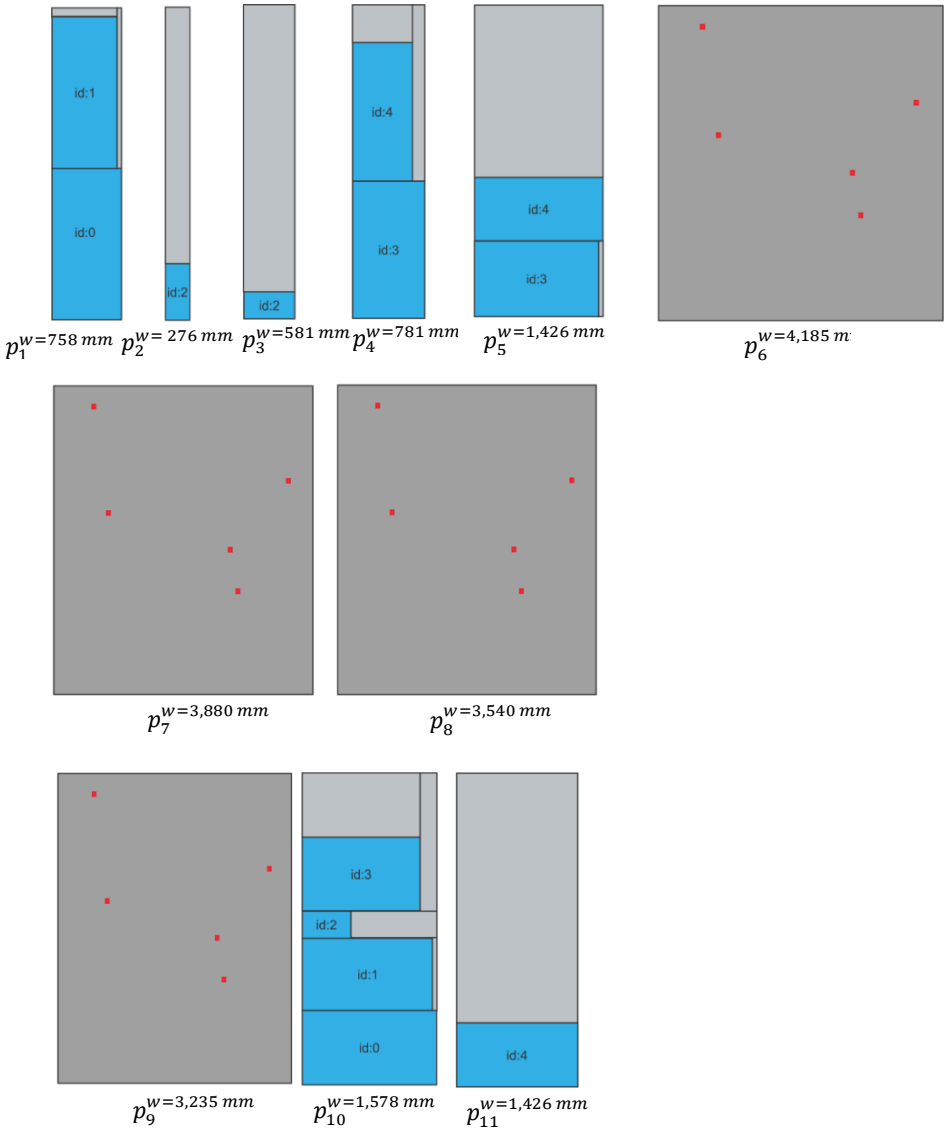


Figura 2.7 Posible conjunto de P patrones.
Cada patrón de corte muestra una manera distinta de acomodar los artículos de la instancia A1 y los posibles residuos que podrían quedar al cubrir la demanda. Elaboración propia.

En la Figura 2.8 se muestra la selección de patrones para el subconjunto de patrones P', con el fin de minimizar la sumatoria de sus longitudes en una solución factible, donde los patrones pueden ser divididos en subconjuntos que suman el ancho W del contenedor, cumplir de esta manera con la restricción (1). Cada artículo que pertenece a un patrón está libre de defectos respetando

las restricciones (2) y (3); Los artículos cortados son exactamente lo que pide la demanda, cumpliendo la restricción (4), ningún artículo se repite en los patrones elegidos en una solución, ya que se tratan de conjuntos disjuntos lo que hace que la restricción (5) se cumpla, los patrones están ordenados de izquierda a derecha, comenzando a cortar con it_0, it_1,…, it_3 lo que demuestra que la restricción (6) se cumple.

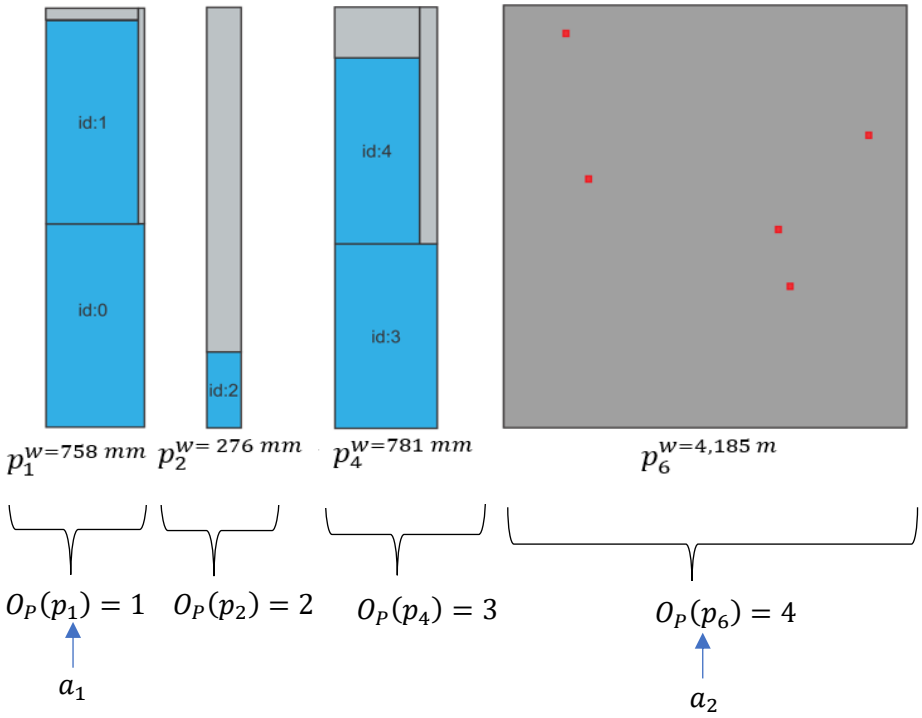


Figura 2.8 Subconjunto de patrones P'. Elaboración propia.

Un árbol de corte en la Figura 2.9 muestra cómo se extraen los artículos para la solución antes descrita utilizando la función $O_I(it)$.

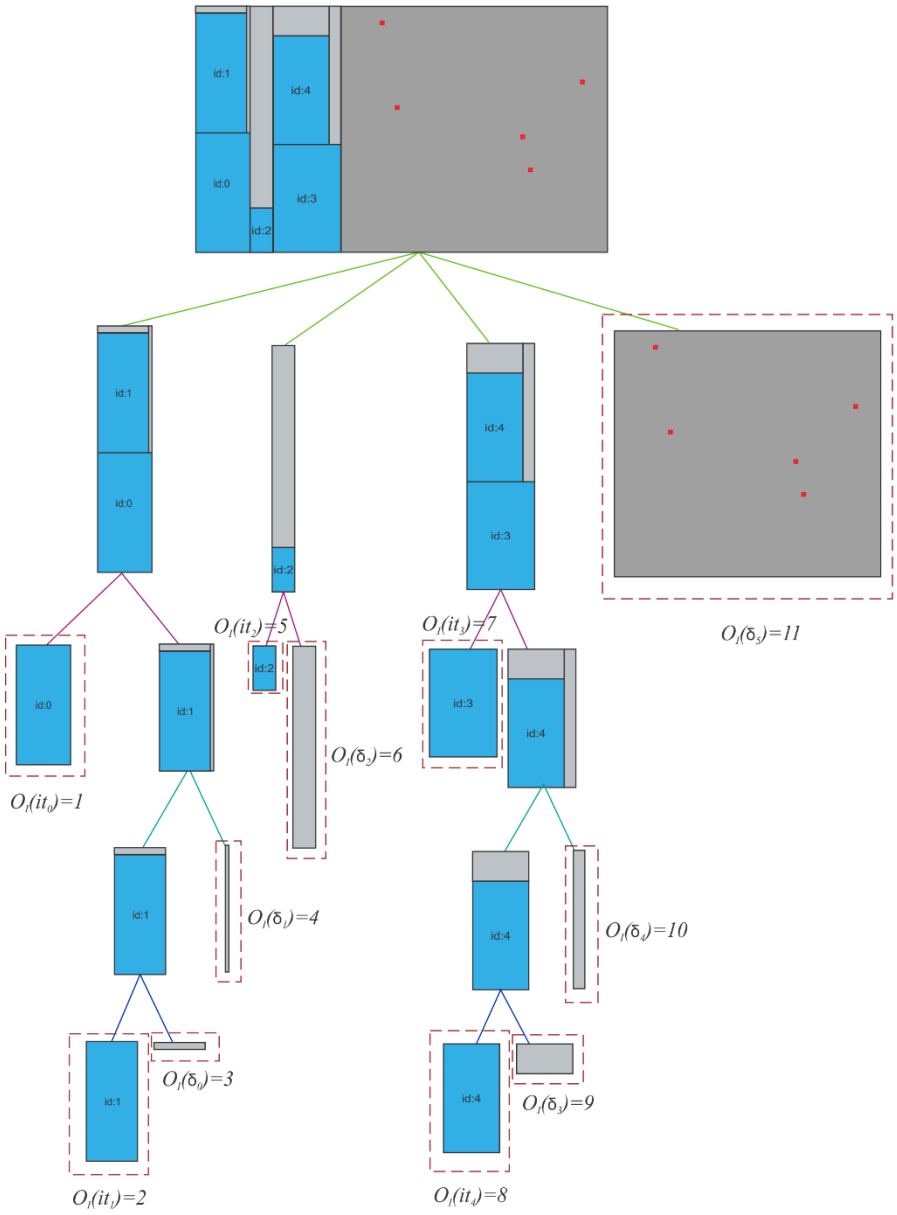


Figura 2.9 Árbol de corte que muestra la solución de A1. Elaboración propia.

Calculamos la función Objetivo, de la solución el cuarto patrón ordenado ($O_P(p_6) = 4$) no es considerado como un patrón a sumar, porque está a la derecha del último Corte-1 hecho en el último contendedor, así que lo tomaremos como un residuo R que será guardado para ser usado en cubrir futuras demandas.

$$\min f(Z) = \sum_{i=1}^{|P|} p_i^w \lambda_i$$

$$\min f(Z) = 758 + 276 + 781$$

$$f(Z) = 1{,}815\ mm$$

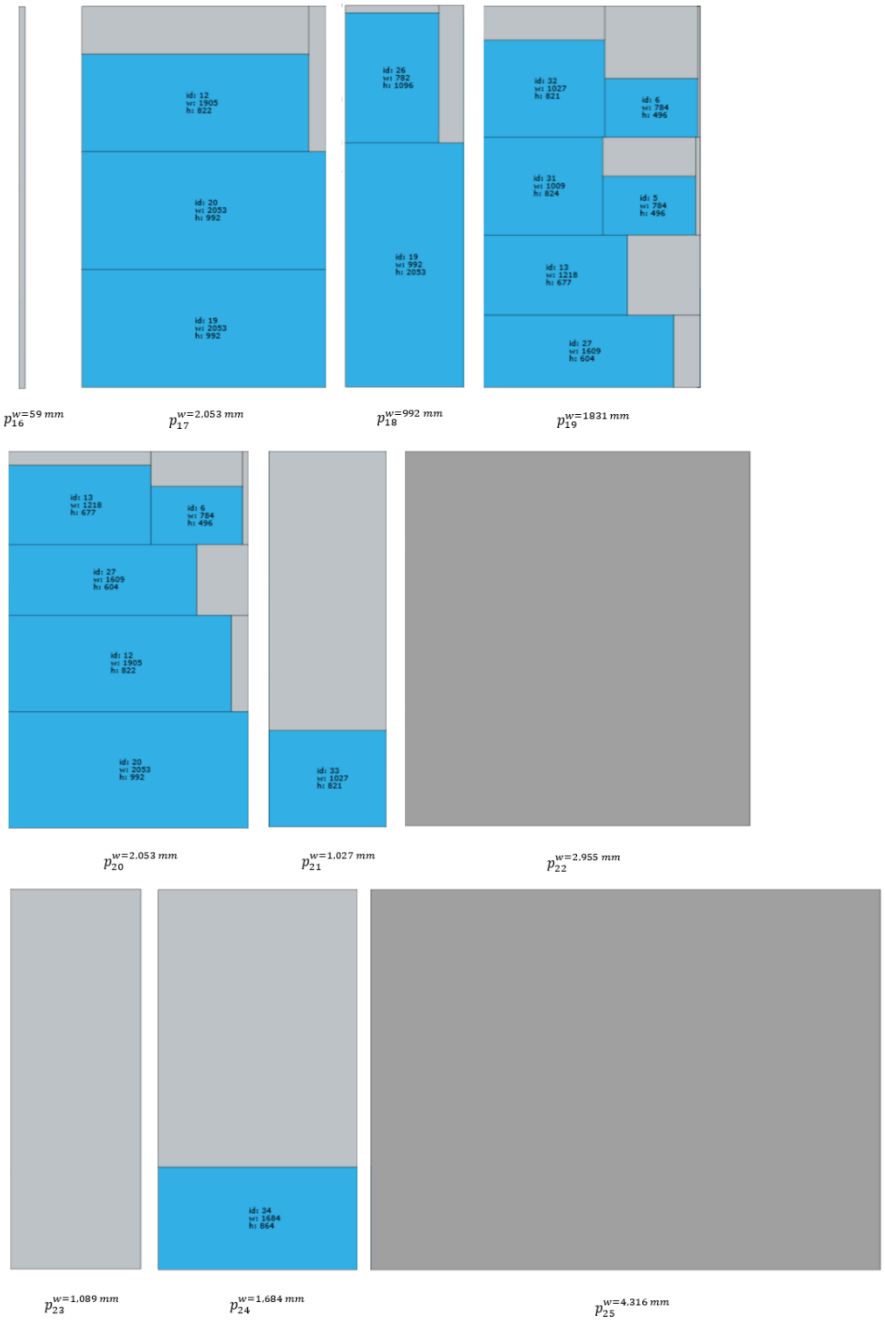
A continuación se muestra otro ejemplo con la instancia A6 de la Tabla 2.2, omitiendo la restricción (2) y (3) de los defectos, para tener una idea clara de cómo crear soluciones sin defectos.

Tabla 2.2 Datos de la instancia A6.

ID_ARTICULO (*it*)	LARGO (*h*)	ANCHO (*w*)	PILA (*S*)	SECUENCIA
0	610	1752	0	1
1	298	1929	0	2
2	798	1929	0	3
3	2128	1259	0	4
4	296	584	0	5
5	496	784	0	6
6	496	784	0	7
7	677	1614	1	1
8	1398	2166	1	2
9	1398	2166	1	3
10	728	2058	1	4
11	354	1905	1	5
12	822	1905	1	6
13	677	1218	1	7
14	824	1609	2	1
15	382	1717	2	2
16	382	1717	2	3
17	947	2013	2	4
18	947	2013	2	5
19	992	2053	2	6
20	992	2053	2	7
21	864	1684	3	1
22	864	1684	3	2
23	864	1684	3	3
24	864	1684	3	4
25	212	1096	3	5
26	782	1096	3	6
27	604	1609	3	7
28	821	843	4	1
29	824	843	4	2
30	402	1009	4	3
31	824	1009	4	4
32	821	1027	4	5
33	821	1027	4	6
34	864	1684	4	7
35	321	691	5	1
36	402	843	5	2

Con los datos se genera un posible conjunto de P patrones a considerar para construir una solución. La Figura 2.10 (a) y (b) muestra algunos patrones con la longitud en el ancho del contendor que ocupa.

(a)



(b)

Figura 2.10 Ejemplo de patrones de la instancia A6. Elaboración propia.

Del conjunto P de patrones válidos se selecciona un subconjunto P' que representa la solución. En las Figuras 2.11, 2.12 y 2.13 podemos ver la solución P', que respeta las restricciones del modelo matemático. Podemos ver la restricción (1) cumplida porque los patrones pueden ser divididos en subconjuntos que suman el ancho W del contenedor. El ancho del contenedor es de 6,000 mm. Se puede notar que los 37 artículos de la demanda se encuentran en la solución y ninguno se repite respetando la restricción (4) y (5). La manera en que son cortados guarda el orden de precedencia como lo indica la restricción (6).

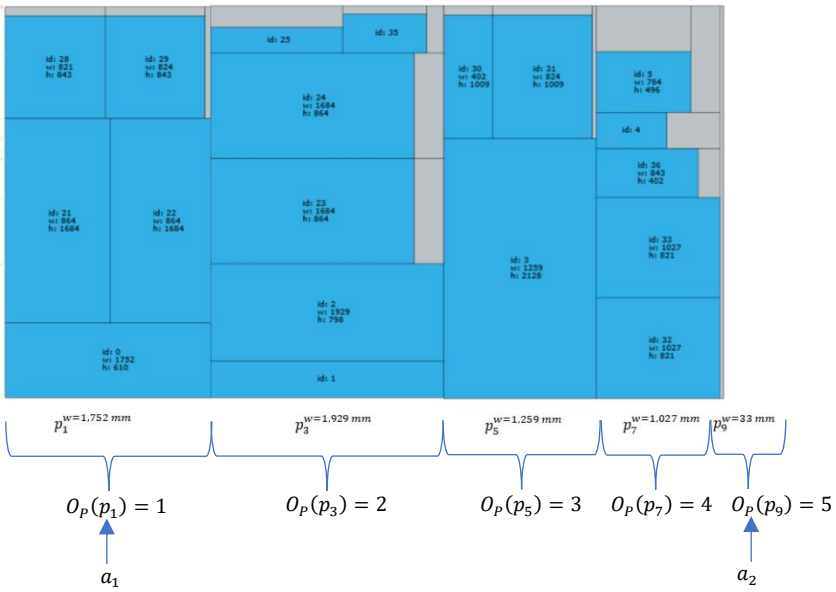


Figura 2.11 Primer contenedor de la solución para A6. Elaboración propia.

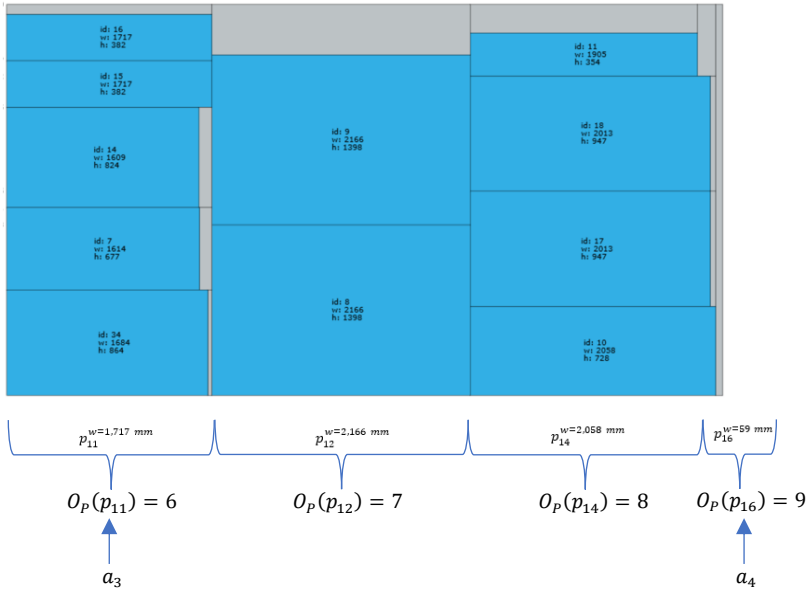


Figura 2.12 Segundo contenedor de la solución para A6. Elaboración propia.

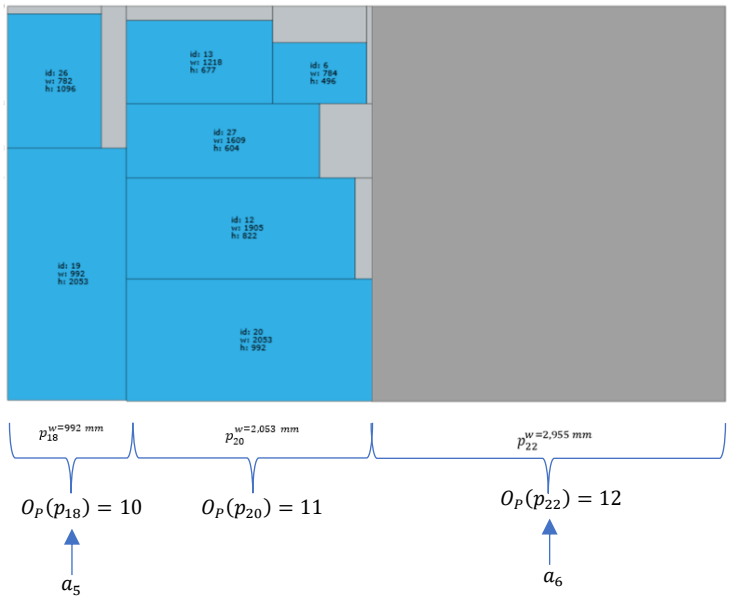


Figura 2.13 Tercer contenedor de la solución para A6. Elaboración propia.

La solución construida pasa por la función objetivo calculando el ancho total de los patrones dando como resultado 15,045 mm, dejando un residuo de 2,955 mm que se guarda para usarse en demandas posteriores.

Capítulo 3 Metodología

En este capítulo se explican los algoritmos glotones implementados, la manera en que se hibridan y los parámetros sintonizados que ocupan. Es importante señalar que para resolver el problema de corte de vidrio ROADEF no se consideró la restricción de los defectos en los contenedores. Es decir, suponemos que todos los contenedores que utilizamos para cortar artículos se encuentran libres de defectos, esto con el fin de simplificar el problema.

"La lógica te llevará desde A a B. La imaginación te llevará a todas partes". Albert Einstein

3.1 Algoritmo Min_Max

La implementación del algoritmo Min_Max, parte de un criterio de selección que se utiliza para asignar un artículo dentro de un contenedor en cada paso local, la solución se construye con un algoritmo glotón, este algoritmo considera la longitud más corta entre el ancho w y el largo h de cada artículo disponible it, $\beta^{it} = \min(w, h)$ en un conjunto de candidatos C, para luego elegir de ellos el valor máximo, $\beta^{max} \leftarrow max\ \beta^{it}, \forall\ it\ \in C$ y evaluar si es un candidato que no viola ninguna de las restricciones, si esto se cumple se asigna al contenedor de tal manera que cada asignación va construyendo una solución. La Figura 3.1 muestra el pseudocódigo del algoritmo glotón Min_Max.

Entrada: (*I, H, W*); *I*-Lote $I = \{S_1, S_2, \ldots, S_m\}$ donde cada pila $S = \{it_1, it_2, \ldots, it_n\}$ e *it* con dimensiones (h_{it}, w_{it}) donde $it \in S$ y $S \in I$, *H*- largo del contenedor y *W*- ancho del contendor.
Salida: *X*- Una solución representada por una lista de patrones
$X \leftarrow \emptyset$;
mientras $I \neq \emptyset$ **hacer**
    C = Calcula_Candidatos();
    Min_Max(*C*);//función de ordenamiento para los artículos *it* del *C* subconjunto de candidatos disponibles
        si *it* =mejor_artículo_a_la derecha **entonces**
                genera patrón p_{it} en la derecha;
        si no
            si *it*= mejor_artículo_arriba **entonces**
                genera patrón p_{it} arriba;
            si no
                si *it*= mejor_artículo_en_nuevo_corte_uno **entonces**
                    genera patrón p_{it} en nuevo_corte_uno;
                si no
                    si *it*= mejor_artículo_en_nueva_hoja **entonces**
                        genera patrón p_{it} en nueva_hoja;
        si *it* ∈ nuevo_corte o nueva_hoja **entonces**
                CALCULA un umbral *U* a la derecha;
    $X \leftarrow X \cup p_{it}$;
    $I = I - it$;
fin de mientras;
retorna *X*;
fin glotón;

Figura 3.1 Pseudocódigo del algoritmo glotón Min_Max. Elaboración propia.

El algoritmo comienza con una entrada de datos, supongamos que tenemos un lote I de cuatro pilas S, $I=\{S_1, S_2, S_3, S_4\}$ donde la pila S_1 contiene tres artículos $S_1=\{it_1, it_2, it_3\}$, la pila S_2 contiene dos artículos $S_2=\{it_4, it_5\}$, la pila S_3 contiene tres artículos $S_3=\{it_6, it_7, it_8\}$ y la pila S_4 contiene un artículo $S_4=\{it_9\}$, cada artículo tiene dos dimensiones ancho w y largo h, (para fines prácticos no se especifican las longitudes en este ejemplo porque el objetivo es explicar con detalle el comportamiento del algoritmo Min_Max);los últimos datos de entrada son las dimensiones del contenedor en su ancho W y su largo H. La salida del algoritmo es una solución representada por una lista de patrones de corte para todos los artículos de un lote, llamémosla X, la cual comienza vacía y conforme el ciclo del algoritmo itera se agregan patrones que en conjunto formaran una solución. El ciclo termina en el momento que las pilas del lote estén vacías.

Cada vez que el algoritmo entra en el ciclo comienza calculando todos los posibles artículos que se pueden cortar en ese momento con la función Calcula_Candidatos(), la cual de una manera muy sencilla toma el primer artículo disponible de cada pila (que no haya sido asignado) y lo agrega a una lista de candidatos disponibles para cortar. La Figura 3.2 ejemplifica

visualmente el comportamiento de la función con los datos de entrada del lote I, mostrando en cinco pasos como se van agregando artículos a la lista de candidatos. El primer paso representa como está organizado el lote I según los datos de entrada, en los pasos del dos al cinco, muestra cómo se van incorporando los primeros elementos de cada pila a la lista de candidatos representada por el rectángulo que encierra a los artículos seleccionados.

Figura 3.2 Comportamiento de la función Calcula_Candidatos().
La imagen muestra cómo se incorporan los primeros artículos de cada pila a la lista de candidatos, esto con el fin de respetar la restricción de precedencia entre artículos de una misma pila. Elaboración propia.

La lista de candidatos disponibles que se muestra en el paso cinco es mandada a la función Min_Max(). La Figura 3.3 contiene el pseudocódigo donde compara las longitudes del ancho w y el largo h de cada artículo para elegir de estos el menor, almacenando en una nueva lista la longitud mínima y el número de artículo al que pertenece, este procedimiento se repite hasta que toda la lista de artículos disponibles se evalúa.

Posteriormente las longitudes de la nueva lista son ordenadas de mayor a menor por el método de inserción, retornando al algoritmo glotón la nueva lista de longitudes ordenadas.

Entrada: (*C)*; *C* – Subconjunto de artículos candidatos *it*
Salida: *M*- Un conjunto de dimensiones ordenadas para ser cortados con su respectivo número de articulo
M← ∅;
for i←0 **hasta** *C*
  si $h_{it} \leq w_{it}$
    M← h_{it};
  sino
    M← w_{it};
fin de for;
inserción(*M*);
retorna *M*;
fin de Min_Max;

Figura 3.3 Pseudocódigo de la función Min_Max(). Elaboración propia.

Cada longitud en la nueva lista ordenada trae consigo el número de artículo a cortar. Tomando la lista de candidatos disponibles de la Figura 3.2 que se muestra en el paso cinco, la Figura 3.4 ejemplifica visualmente cómo queda la nueva lista de longitudes ordenadas al pasar por la función Min_Max(), para poder determinar cuál será el primer artículo a considerar en la construcción de la solución.

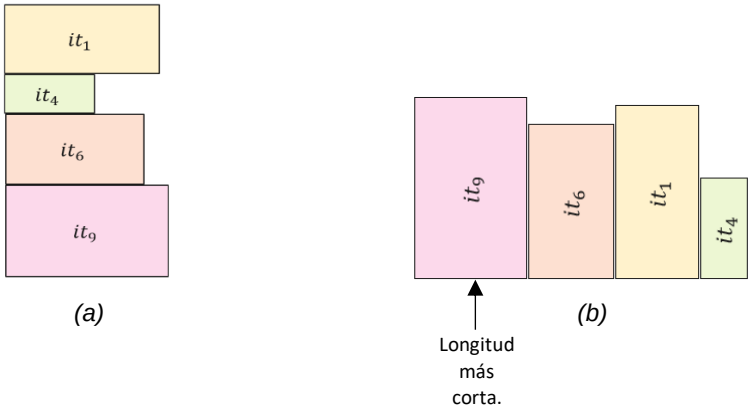


Figura 3.4 Comportamiento de la función Min_Max().
En (a) se muestran la lista de candidatos que manda la función Calcula_Candidatos(), en (b) la función Min_Max() ordena los artículos de izquierda a derecha de mayor a menor según su longitud más corta. Elaboración propia.

Para construir una solución es importante mencionar que los artículos se cortan de izquierda a derecha, comenzando en la esquina inferior izquierda en todos los algoritmos implementados, así que para elegir al mejor artículo en cada paso se toma en consideración todas las posibles posiciones en las que puede asignarse un artículo al contenedor. Cuando una hoja está vacía existen dos posibilidades de acomodar un artículo esto sucede porque los elementos pueden girarse noventa grados, por lo tanto se acomodan en una orientación original o trasversal. La Figura 3.5 hace notar que el primer artículo it_9 de la nueva lista de longitudes ordenadas tiene dos formas de asignarse a la nueva hoja. Es necesario señalar que la función Min_Max() determinó la longitud más corta de ambas dimensiones, también determinó la orientación de cada artículo y las guardo en la nueva lista de longitudes ordenadas, así que cuando es asignado el artículo al contenedor trae consigo la información sobre la orientación, por lo tanto será con una orientación original.

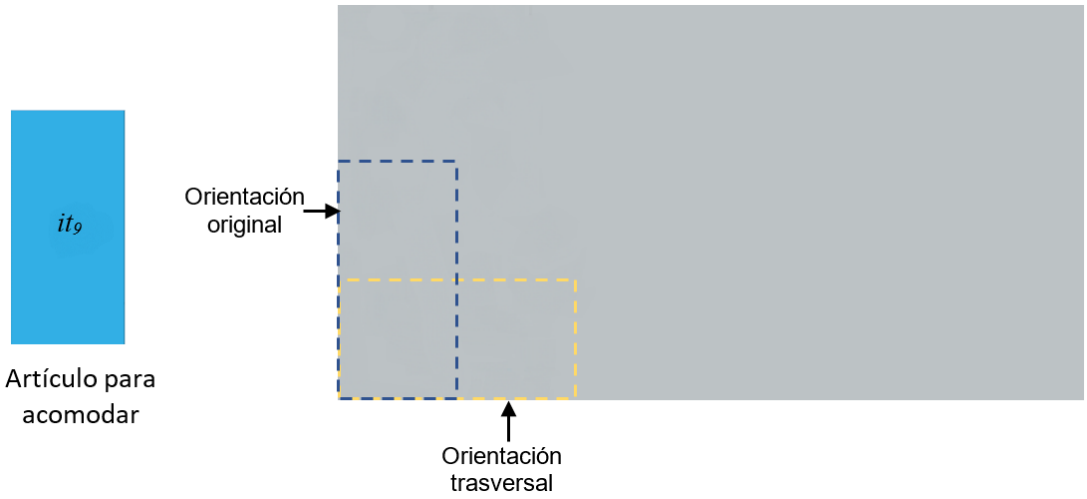
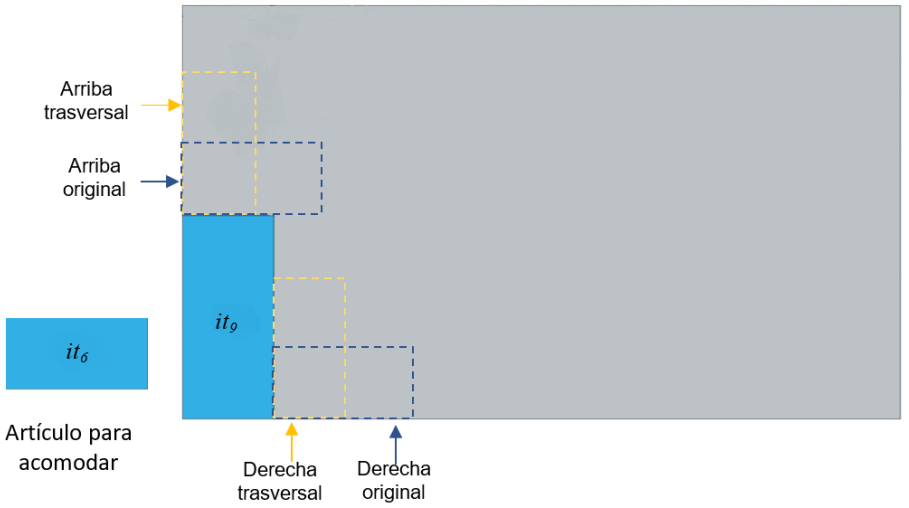


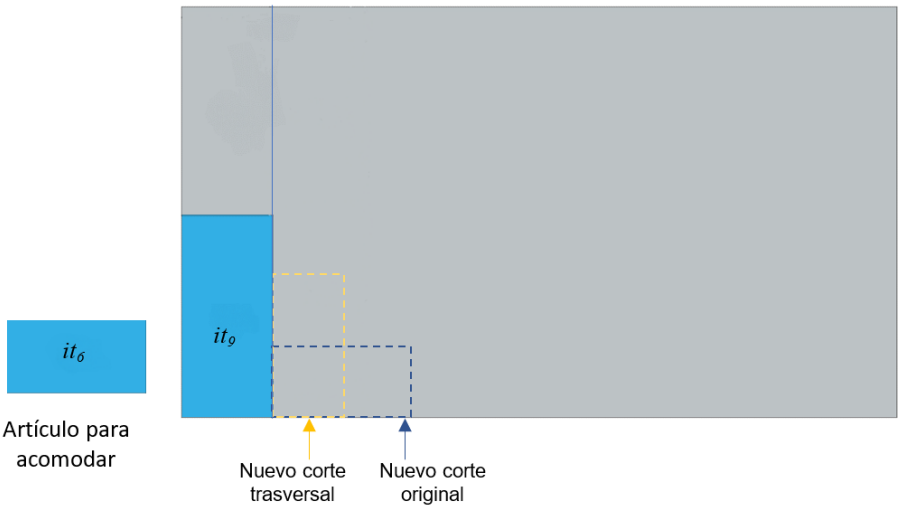
Figura 3.5 Posibles posiciones de un artículo para construir una solución en un nuevo contenedor. Elaboración propia.

Una vez asignado el artículo it_9 se toma el siguiente artículo de la nueva lista ordenada, las posibles posiciones para el nuevo artículo en este caso it_6 aumentan a seis (máximo), tomando como referencia el último artículo asignado que es it_9, se puede poner a su derecha, arriba o en un nuevo corte, estas tres posibilidades se duplican porque por cada posible posición existen

dos posibles orientaciones para el artículo (original y trasversal) véase Figura 3.6.



(a)



(b)

Figura 3.6 Posibles posiciones de un artículo cuando el contender tiene elementos. En (a) se muestran las posiciones para asignar un nuevo artículo a la derecha y arriba del artículo it_9 que ya fue asignado y en (b) muestra las posiciones en las que se puede asignar un nuevo artículo en un nuevo corte, es decir, al contenedor se le dividieron en dos subhojas. Elaboración propia.

Cuando un artículo es asignado a una nueva hoja o un nuevo corte un umbral U se genera para que segmente el espacio donde se puede acomodar los siguientes artículos, esto con el fin de aprovechar al máximo esa sección de vidrio, el valor del umbral está dado por un porcentaje aleatorio que puede estar entre cero por ciento hasta un cien por ciento sobre la longitud w (ancho) del artículo, donde $0 \leq U \leq 100$, asignado a una nueva hoja o en un nuevo corte sobre el eje x del contenedor y los elementos que se acomoden a la derecha y arriba no deben pasar ese límite.

La Figura 3.7 muestra como al ser asignado el artículo it_9 a una nueva hoja se genera un valor aleatorio para U que se encuentra en un intervalo de 0 a 50 por ciento sobre el valor en el ancho del artículo it_9.

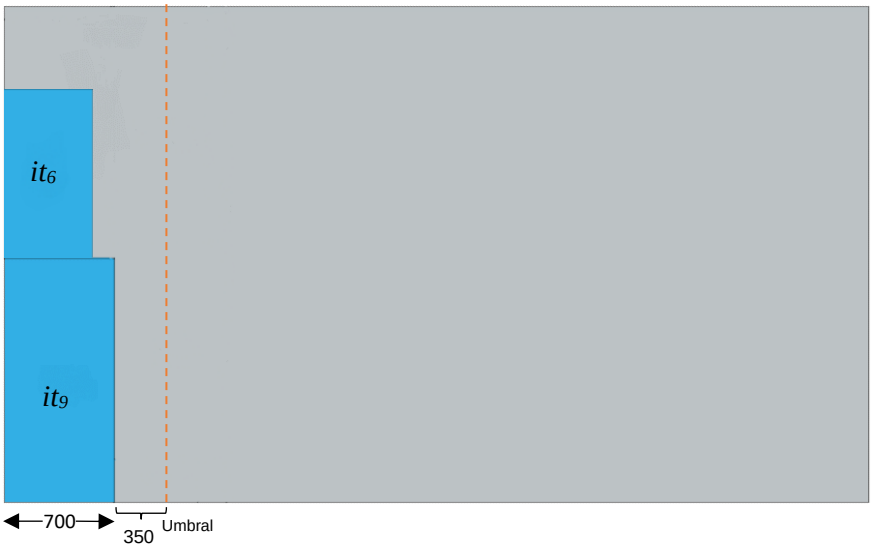


Figura 3.7 Artículos asignados bajo el umbral máximo de 50%. Elaboración propia.

En el ejemplo, U está definido por un valor aleatorio máximo del cincuenta por ciento sobre el ancho w=700 mm de it_9, por lo tanto al probar it_6 a la derecha de it_9 el artículo no cabe, así que se prueba arriba de it_9 donde la posición es factible. Cada vez que un artículo se coloque en un nuevo contenedor o en un nuevo corte se determina un nuevo umbral U de acuerdo con la longitud del artículo asignada al ancho del contenedor, por consecuencia el valor de U también cambia.

A continuación, se da un ejemplo completo de cómo se comporta el algoritmo glotón en una de las instancias de (ROADEF, 2018), específicamente en A20, la Tabla 3.1 muestra los datos de entrada para el algoritmo Min_Max.

La tabla completa representa el lote A20, la primera columna muestra el número de artículo en el lote, comenzando con it_0, it_1, it_2, …, it_{16}, la segunda y tercera columna definen las longitudes de cada artículo it en largo h y ancho w, la cuarta columna determina a que pila S pertenece cada artículo it y el número de pilas en el lote I, la quinta columna describe la secuencia en la que se deben cortar los artículos de cada pila con el fin de respetar un orden de precedencia entre artículos de una misma pila.

Tabla 3.1 Datos de la instancia A20.

ID_ARTICULO (it)	*LARGO (h)*	*ANCHO (w)*	*PILA (S)*	*SECUENCIA*
0	*1084*	*430*	*0*	*1*
1	*618*	*1604*	*0*	*2*
2	*549*	*1262*	*0*	*3*
3	*498*	*1365*	*0*	*4*
4	*960*	*895*	*0*	*5*
5	*405*	*1911*	*0*	*6*
6	*1126*	*2052*	*0*	*7*
7	*426*	*1655*	*0*	*8*
8	*395*	*1675*	*1*	*1*
9	*482*	*1700*	*2*	*1*
10	*381*	*2085*	*3*	*1*
11	*733*	*2085*	*4*	*1*
12	*522*	*1171*	*5*	*1*
13	*522*	*1171*	*6*	*1*
14	*1077*	*460*	*7*	*1*
15	*1077*	*827*	*8*	*1*
16	*482*	*1700*	*9*	*1*

La representación visual de la instancia se muestra en la Figura 3.8, donde el lote A20 contiene diez pilas A20$= \{S_0, S_1, S_2, \ldots, S_9\}$, la pila S_0 contiene ocho artículos $S_0 = \{it_0, it_1, it_2, \ldots, it_7\}$, por ser la única que contiene varios elementos debe respetar el orden de precedencia entre sus artículos, por lo tanto no se

podrá cortar un it_1 sin haber cortado un it_0. Las pilas S_1, S_2, S_3, …, S_9 solo contienen un artículo por pila.

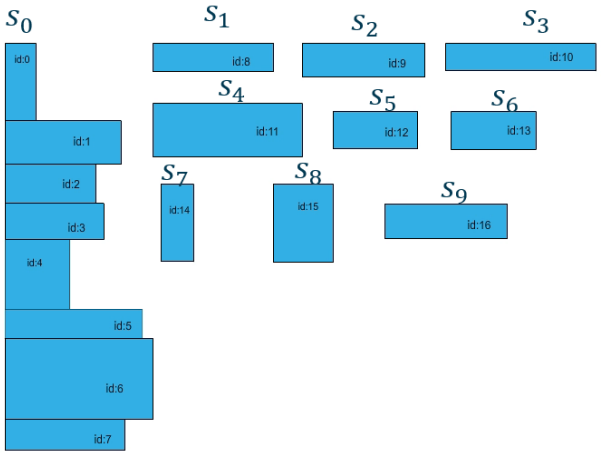


Figura 3.8 Artículos de la instancia A20.Elaboración propia

Los datos de entrada son mandados a la función Calcula_Candidatos() para crear una lista de candidatos (artículos) disponibles a cortar (véase Figura 3.9 (a)). La lista de candidatos disponibles es mandada a la función Min_Max() para ordenarla y arrojar la nueva lista de longitudes ordenadas, como se nota en la parte inferior derecha de la Figura 3.9 (b) y comenzar a asignar artículos al contenedor vacío.

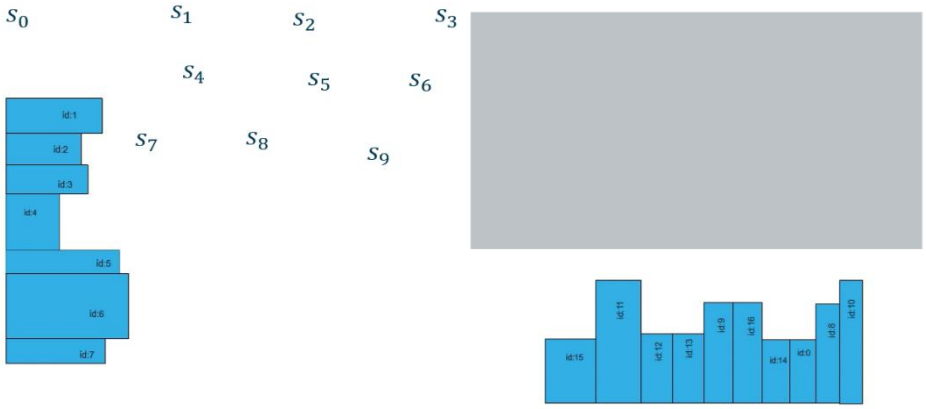


Figura 3.9 Selección de los primeros candidatos a cortar al comenzar el algoritmo. Elaboración propia.

El primer artículo asignado al nuevo contenedor es it_{15}, como su asignación fue a una nueva hoja, sobre su valor w es generado un umbral U máximo aleatorio del 30% específico para este ejemplo, (véase Figura 3.10(a)). Una vez determina el valor de U, el algoritmo regresa a evaluar los candidatos disponibles, notemos que la pila S_8 a la que pertenecía el artículo it_{15} se encuentra vacía (Figura 3.9(a)), por lo tanto ningún nuevo elemento entrara a la lista para ordenarlo, haciendo de esa manera que el siguiente artículo a evaluar sea el it_{11}, la Figura 3.10 (b) muestra como dicho artículo es evaluado a la derecha de it_{15} excediendo el límite del umbral U, impidiendo con esto que el articulo it_{11} se asigne.

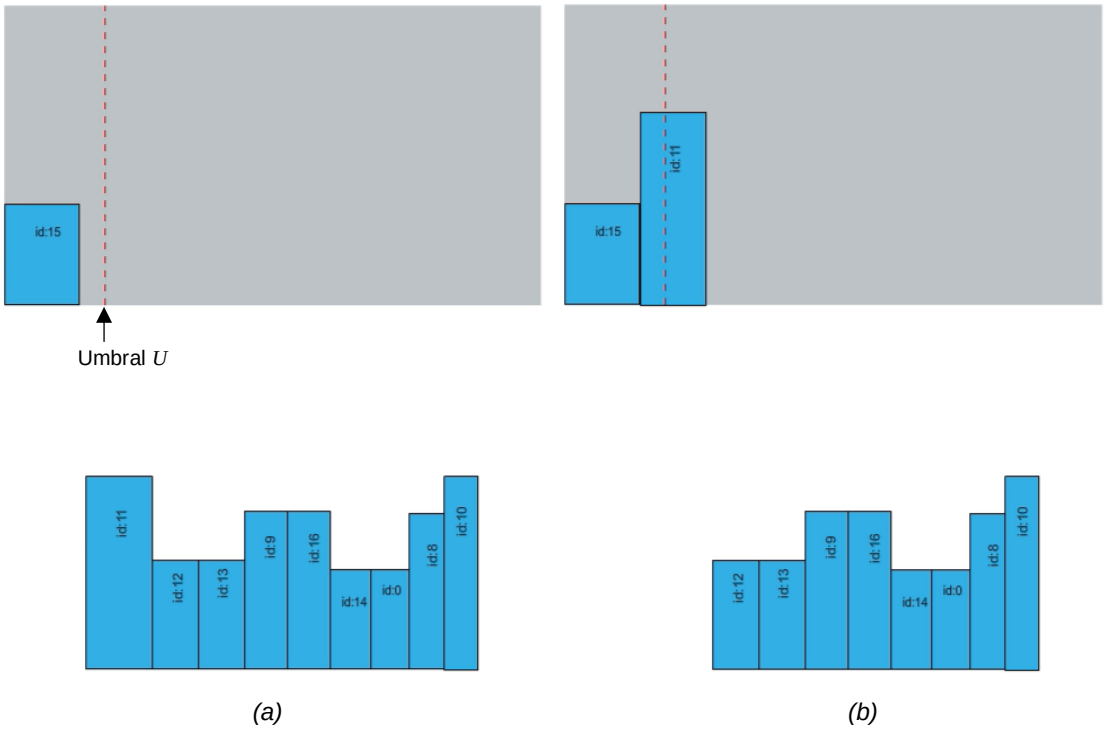


Figura 3.10 Primera asignación de artículo y cálculo de umbral.
En (a) se muestra la asignación de it_{15} y se determina el umbral. En (b) se comienza a probar it_{11} a la derecha. Elaboración propia.

El siguiente artículo de la nueva lista de longitudes ordenadas es it_{12}, el cual se trata de asignar a la derecha de it_{15} pero el umbral no permite su asignación (véase Figura 3.11(a)), así que se evalúa cada uno de los artículos de la nueva

lista de longitudes ordenadas hasta encontrar al mejor candidato a la derecha, pero como no se encontró se busca en una nueva posición arriba del artículo it_{15}, comenzando a probar una vez más toda la nueva lista de longitudes ordenadas, cuando se evalúa el artículo it_{11} cabe arriba de it_{15} sin violar ninguna restricción asignándolo al contenedor como se muestra en la Figura 3.11(b), y tomándolo como referencia para poner más artículos a su derecha y arriba.

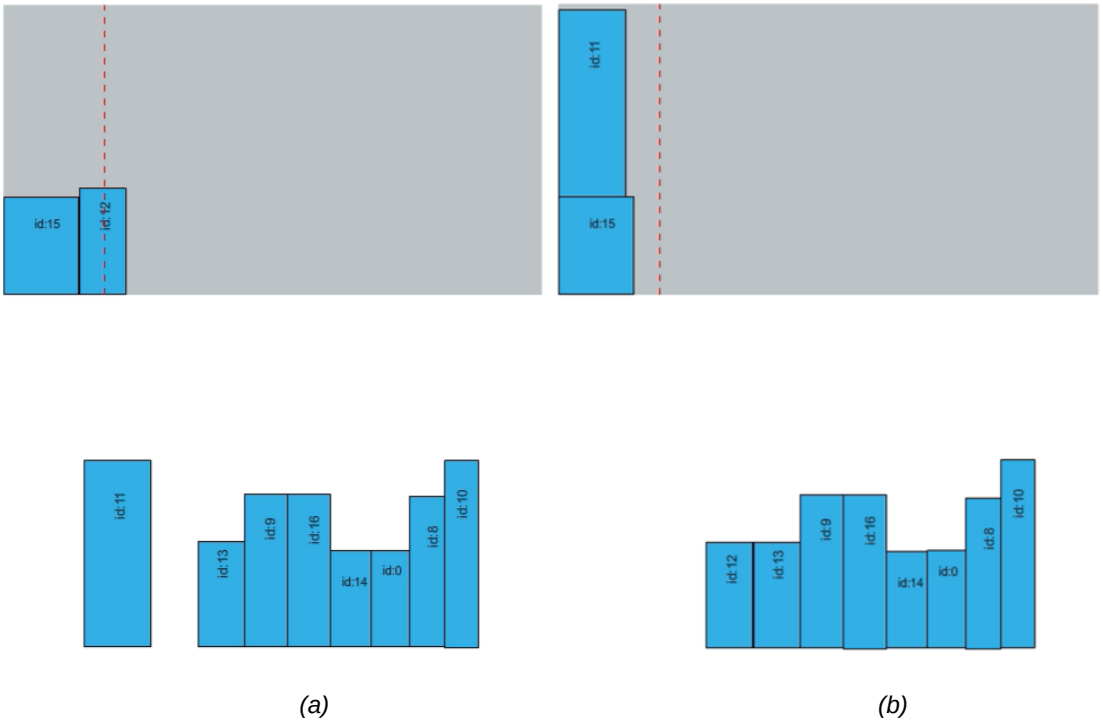


Figura 3.11 Asignación de artículos.
En (a) prueba el siguiente elemento it_{12} de la lista. En (b) después de probar todos los elementos de la lista a la derecha comienza a probarlos arriba de it_{15}. Elaboración propia.

La nueva referencia es el artículo it_{11}, ahora se intenta poner a it_{12} a la derecha del articulo it_{11} pero el umbral no lo permite como se muestra en la Figura 3.12 (a), nuevamente se evalúan todos los artículos de la lista sin tener éxito al tratar de asignarlos a la derecha, por lo tanto, se evalúan arriba del articulo it_{11} y de la misma manera ninguno cabe, porque el largo del contenedor no lo permite, así que se genera un nuevo corte (Corte-1) y con ello sus cortes consecutivos como se indica en la Figura 3.12 (b), luego un nuevo artículo es

evaluado, cuando se asigna, un nuevo umbral es determinado y todo el proceso se repite hasta terminar de construir una solución.

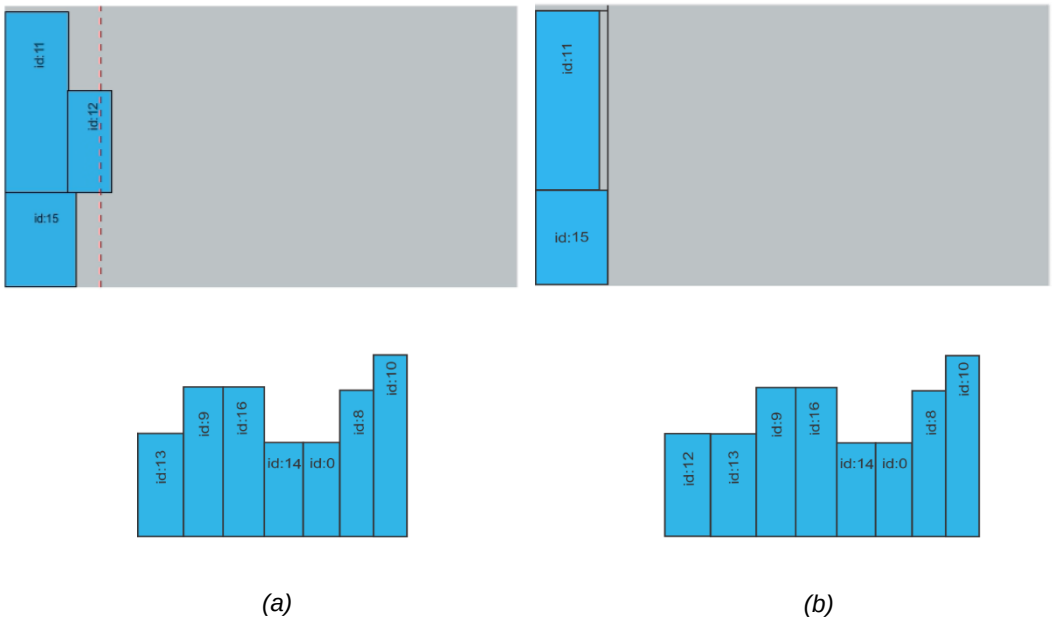


Figura 3.12 Creación del primer patrón de corte.
En (a) muestra la asignación de it_{11} y la prueba de it_{12} a la derecha de it_{11}. En (b) después de probar todos los elementos en derecha y arriba, genera un nuevo corte. Elaboración propia.

Es importante tener presente que después de que un artículo es asignado las posibles posiciones del siguiente artículo vuelven a ser seis (máximo); las funciones Calcula_Candidatos y Min_Max se calculan, para verificar la incorporación de un nuevo artículo a la lista de candidatos posibles y que este se encuentre ordenado en la nueva lista de longitudes ordenadas.

La solución se termina de construir cuando las pilas del lote A20 están vacías, y todos los artículos forman parte de una solución sin violar ninguna restricción descrita en la formulación y el modelo matemáticos. En la Figura 3.13 se visualiza del lado izquierdo la solución factible con los patrones de corte para los artículos del lote A20, la parte gris clara es la pérdida y la parte gris oscura es el residuo que se guarda para utilizarse en otros lotes a cortar, este residuo se obtiene en el último Corte-1 hecho en el último contenedor utilizado en la solución; del lado derecho todas las pilas están vacías, por lo tanto, la

demanda fue cubierta y el algoritmo glotón Min_Max regresa la solución construida.

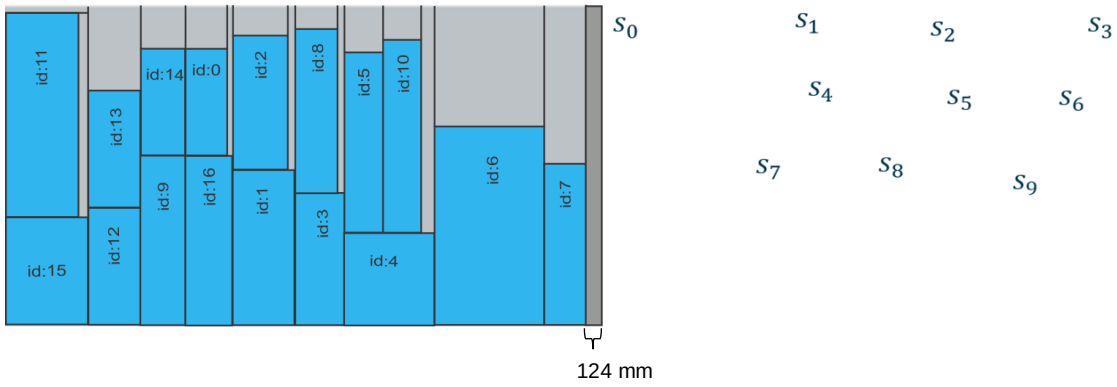


Figura 3.13 Solución de A20 con Min_Max.
Se muestra la solución gráfica y las pilas vacías. Elaboración propia.

Cuando se tiene una solución construida la función objetivo que proporciona el ROADEF la evalúa de la siguiente manera: se desea minimizar el área de pérdida de los contenedores utilizados en una solución, la función objetivo comienza calculando el área total de todos los contenedores utilizados en una solución, es decir multiplicar H (largo del contenedor) por W (ancho del contenedor) por m(número total de contenedores en una solución), a esto se le resta el área del residuo que se calcula multiplicando el H (largo) del contenedor por el r_m (ancho del residuo), que se encuentra a la derecha del último Corte-1 realizado en el último contenedor, en la Figura 3.13 r_m= 124 mm, menos la sumatoria del área total de los artículos en el lote A20, el resultado es la pérdida total que tiene la solución, donde existen un solo contenedor, la función objetivo da como resultado 4,151,485 mm^2 de pérdida. El cálculo se hace de la siguiente manera:

$$\min \mathrm{f(Z)} = \quad HWm - Hr_m - \sum_{i \in I} w_i h_i$$

$\min f(Z)$= ((3,210 * 6,000) (1)) - ((3,210*124)) - (14,710,475)

$\min f(Z)$= 19,260,000 – 398,040 – 14,710,475

$f(Z)$= 4,151,485 mm^2

3.2 Algoritmo Max_Max

La implementación del algoritmo Max_Max parte de un criterio de selección que se utiliza para asignar un artículo dentro de un contenedor en cada paso local, este algoritmo considera la longitud más larga entre el ancho w y el largo h de cada artículo disponible it, $\gamma^{it} = \max(w, h)$ en un conjunto de candidatos C, para luego elegir de ellos el valor máximo, $\gamma^{max} \leftarrow max\ \gamma^{it}, \forall\ it \in C$ y evaluar si es un candidato que no viola ninguna de las restricciones, si esto se cumple se asigna al contenedor. La Figura 3.14 muestra el pseudocódigo del algoritmo glotón trabajando con un criterio de selección Max_Max.

Entrada:(*I, H, W*); *I*-Lote *I*= {$S_1, S_2, \ldots, S_m$} donde cada pila *S*= {$it_1, it_2, \ldots, it_n$} e *it* con dimensiones (h_{it}, w_{it}) donde *it* ϵ *S* y *S* ϵ *I*, *H*- largo del contenedor y *W*- ancho del contendor.
Salida: *X*- Una solución representada por una lista de patrones
X← ∅;
mientras *I*≠∅ **hacer**
    C = Calcula_Candidatos ();
    Max_Max (*C*);//función de ordenamiento para los artículos *it* del *C* subconjunto de candidatos disponibles
        si *it* =mejor_artículo_a_la derecha **entonces**
                genera patrón p_{it} en la derecha;
        si no
            si *it*= mejor_artículo_arriba **entonces**
                genera patrón p_{it} arriba*;*
            si no
                si *it*= mejor_artículo_en_nuevo_corte_uno **entonces**
                    genera patrón p_{it} en nuevo_corte_uno;
                si no
                    si *it*= mejor_artículo_en_nueva_hoja **entonces**
                        genera patrón p_{it} en nueva_hoja*;*
        si *it* ϵ nuevo_corte o nueva_hoja **entonces**
                CALCULA un umbral *U* a la derecha;
     X←*X* ∪ p_{it}*;*
    I = *I* –*it*;
fin de mientras;
retorna *X*;
fin glotón;

Figura 3.14 Pseudocódigo del algoritmo glotón Max_Max. Elaboración propia.

Este algoritmo es muy similar al algoritmo Min_Max su diferencia radica en el criterio de selección que utiliza cada uno, mientras en Min_Max los artículos disponibles son tomados por su longitud mínima y ordenados de mayor a menor bajo ese valor para comenzar a cortar. En este algoritmo Max_Max los

artículos disponibles son tomados por su longitud máxima y ordenados bajo ese valor de mayor a menor para comenzar su asignación en el contenedor.

Específicamente la función Max_Max() inicia recibiendo la lista de candidatos disponibles que le manda la función Calcula_Candidatos(), comparando los valores del ancho w y largo h de cada artículo para elegir de estos el mayor, agregando en una nueva lista el valor máximo y el número de artículo al que pertenece, este procedimiento se repite hasta que toda la lista de candidatos disponibles se evalúa, en seguida las longitudes de la nueva lista son ordenadas de mayor a menor por el método de inserción, retornando al algoritmo glotón la nueva lista de longitudes ordenadas. En la Figura 3.15 se muestra el pseudocódigo de la función Max_Max().

Entrada: (C*)*; *C* – Subconjunto de artículos candidatos *it*
Salida: *M*- un conjunto de dimensiones ordenadas para ser cortados con su respectivo número de artículo
$M \leftarrow \varnothing$;
for i← 0 **hasta** *C*
  si $h_{it} \geq w_{it}$
    $M \leftarrow h_{it}$;
  sino
    $M \leftarrow w_{it}$;
fin de for;
inserción(*M*);
retorna *M*;
fin de Max_Max;

Figura 3.15 Pseudocódigo de la función Max_Max(). Elaboración propia.

Para tener claro el comportamiento de la función Max_Max() en la Figura 3.16 (a), se retoma la lista de candidatos disponibles que la función Calcula_Candidatos() genera en el paso cinco de la Figura 3.2, para ser ordenada mediante el criterio de selección de la función Max_Max(), creando la nueva lista de longitudes ordenadas como se muestra en la Figura 3.16 (b).

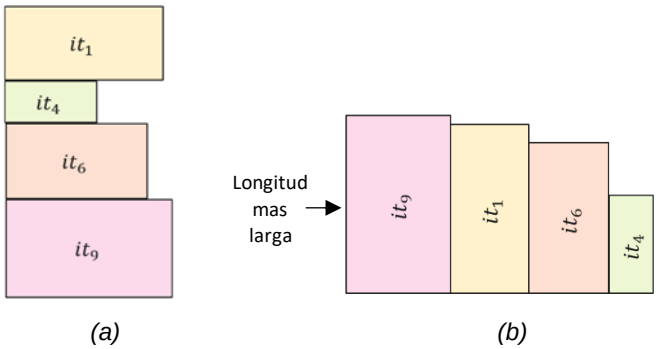


Figura 3.16 Comportamiento de la función Max_Max().
En (a) se muestran la lista de candidatos que manda la función Calcula_Candidatos(). En (b) la función Max_Max() ordena los artículos de izquierda a derecha de mayor a menor según su longitud más larga. Elaboración propia.

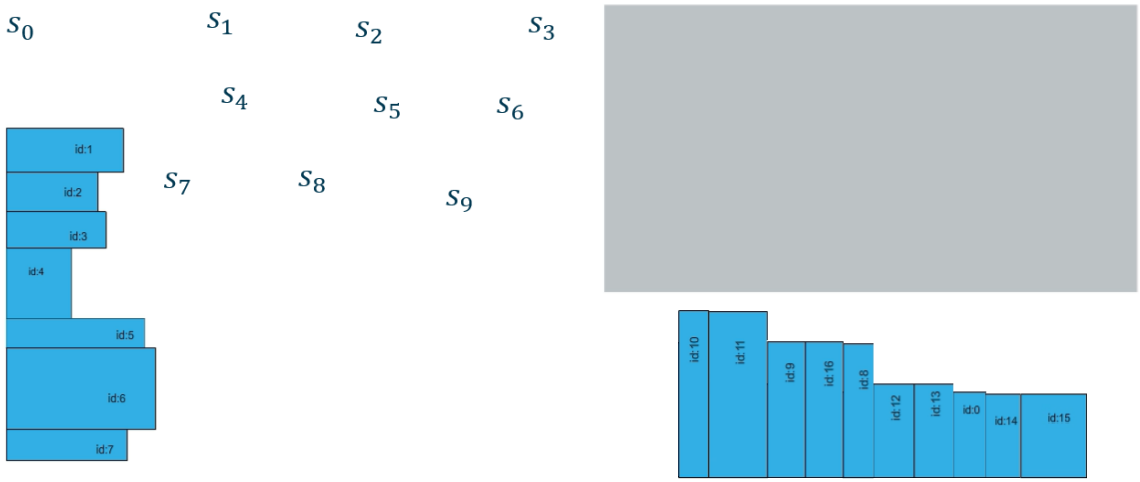
Cuando los artículos comienzan a evaluarse para ser asignados, la longitud más larga de cada artículo determina la orientación en la que será colocado en el eje x del contenedor. Las formas de asignar un artículo a un contenedor son las mismas que las mencionadas para el glotón Min_Max en las Figuras 3.5 y 3.6, un artículo puede ser asignado en una nueva hoja, a la derecha de un artículo, arriba de un artículo y en un nuevo corte, en dos posiciones, original y transversal. El algoritmo también contiene un umbral aleatorio U y su valor es calculado con base en la longitud máxima del artículo puesto en una nueva hoja o en un nuevo corte.

Al ser cortado un artículo, si la pila a la que pertenecía aun contiene elementos, la función Calcula_Candidatos() toma el siguiente artículo y lo incorpora a la lista de candidatos disponibles, mandándola a la función Max_Max() que retorna una nueva lista de longitudes ordenadas.

Un ejemplo de cómo se comporta el algoritmo Max_Max se da a continuación con la misma instancia A20 que se describe en la Tabla 3.1 y la Figura 3.8, esos datos son introducidos al algoritmo de la misma manera que lo hace el

algoritmo Min_Max pero la solución es totalmente diferente por el criterio de selección y el valor de U.

El algoritmo comienza eligiendo de cada pila al primer elemento que contienen (véase Figura 3.17(a)), mediante la función Calcula_Candidatos() genera una lista de candidatos disponibles para cortar. La lista se manda a la función Max_Max(), donde toma la dimensión máxima entre el ancho y el largo de cada artículo, después hace un ordenamiento de mayor a menor de esas longitudes y crea una nueva lista de longitudes ordenadas, que contiene los artículos ordenados para comenzar a asignarse en el contenedor vacío (véase Figura 3.17(b)).



(a) *(b)*

Figura 3.17 Selección de los primeros candidatos a cortar al comenzar el algoritmo Max_Max. En (a) se muestran el estado de las pilas después de generar la lista de candidatos disponibles. En (b) la función Max_Max() hizo un ordenamiento a los artículos para comenzar a asignarse. Elaboración propia.

El primer artículo asignado al nuevo contenedor es it_{10}, sobre su valor w es generado un umbral aleatorio máximo del 25% específico para este ejemplo (véase Figura 3.18 (a)). El algoritmo regresa a evaluar los candidatos disponibles, notemos que la pila S_3 (Figura 3.17 (a)) a la que pertenecía el artículo it_{10} se encuentra vacía, por lo tanto, ningún nuevo elemento está disponible para cortar, haciendo de esa manera que el siguiente artículo a

evaluar sea el it_{11}, la Figura 3.18 (b) muestra como dicho artículo es evaluado a la derecha del articulo it_{10} para verificar si es una posición factible, pero el límite del umbral impide la asignación de it_{11}.

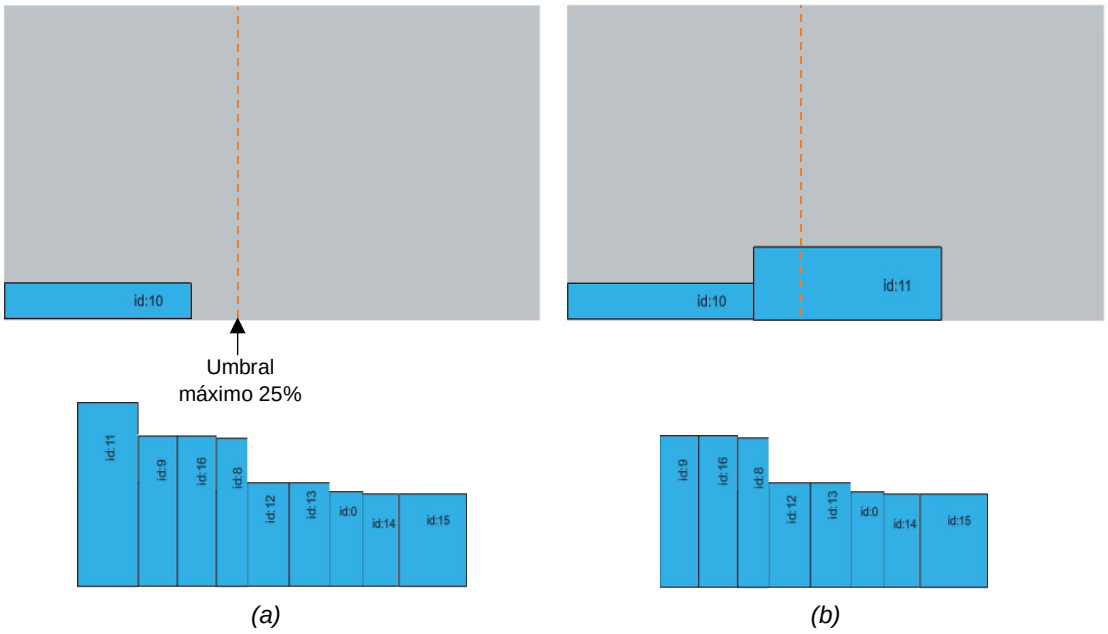


Figura 3.18 Asignación del primer artículo y cálculo de umbral.
En (a) se muestra la asignación de it_{10} y la determinación del umbral. En (b) se comienza a probar it_{11} a la derecha. Elaboración propia.

El siguiente artículo en la nueva lista de longitudes ordenadas es it_9 el cual es tomado para evaluarse a la derecha de it_{10} como se muestra en la Figura 3.19 (a), pero el límite del umbral impide su asignación, así que se evalúan cada uno de los siguientes artículos de la nueva lista de longitudes ordenadas, hasta encontrar al mejor candidato a la derecha, pero como este no se encontró se busca en una posición arriba del articulo it_{10}, comenzando a probar otra vez cada artículo de la nueva lista de longitudes ordenadas, el primer artículo it_{11} es asignado arriba de it_{10} sin violar ninguna restricción como lo muestra la Figura 3.19 (b). El articulo it_{11} se convierte en la nueva referencia para posibles asignaciones de más artículo a su derecha y arriba, pero el umbral que se utiliza sigue siendo el que se determinó cuando se asignó el articulo it_{10} a la

nueva hoja y seguirá siendo mientras no se determine un nuevo corte o contenedor.

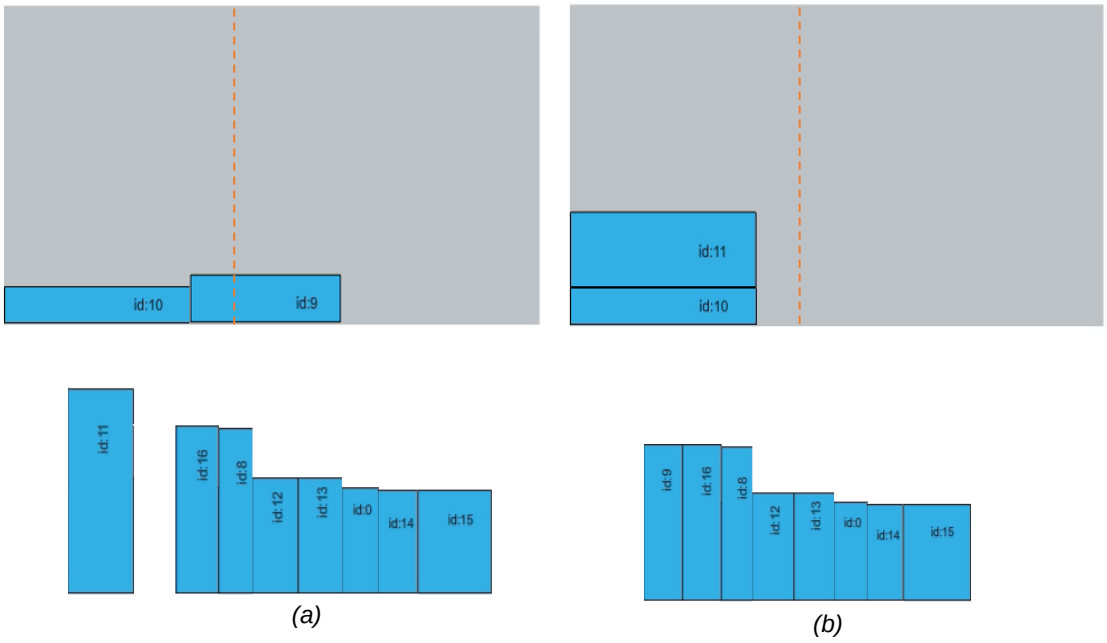
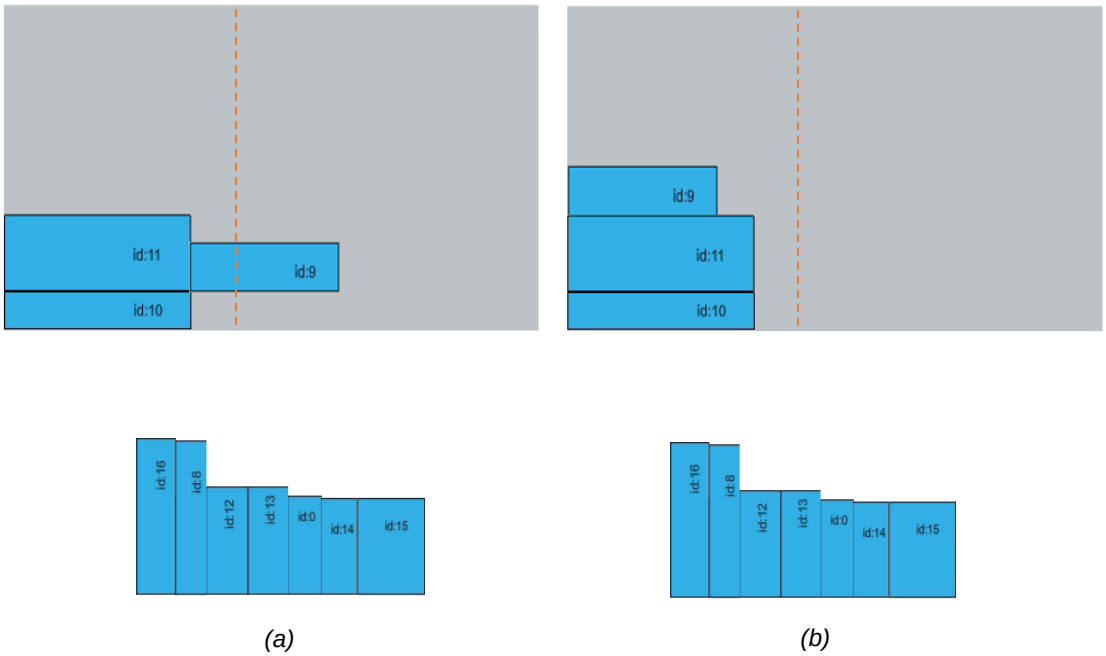


Figura 3.19 Asignación de artículos al contenedor.
En (a) prueba el siguiente elemento it_9 de la nueva lista de artículos ordenados. En (b) después de probar todos los elementos de la lista a la derecha comienza a probarlos arriba de it_{10}. Elaboración propia.

Siguiendo con la construcción de la solución se toma el articulo it_9 para intentar asignarlo a la derecha de it_{11}, pero el umbral no lo permite como se muestra en la Figura 3.20 (a), nuevamente se evalúan todos los artículos de la lista pero ninguno cabe a la derecha, por lo tanto, se intenta en una nueva posiciónarriba del artículo it_{11}, comenzando otra vez con el artículo it_9 donde su asignación arriba es válida, porque no exceder el límite del umbral ni viola ninguna restricción como se muestra en la Figura 3.20 (b).



(a) *(b)*

Figura 3.20 Asignación de artículos.
En (a) muestra la asignación de it_{11} y la prueba de it_9 a la derecha de it_{11} sin ser asignado porque excede el umbral. En (b) después de probar todos los elementos a la derecha, comienza a probar arriba de it_{11} asignando a it_9. Elaboración propia.

De esta manera se sigue construyendo la solución hasta que todas las pilas del lote A20 se encuentren vacías, cada vez que exista un nuevo corte u hoja el valor del límite del umbral cambia, hasta llegar a construir toda la solución como lo muestra la Figura 3.21 donde se utilizan dos contenedores para generarla, dejando como pérdida la parte gris clara y como residuo la parte gris oscura, el cual se guarda para utilizarse en cubrir demandas posteriores. El valor para esta solución al pasar por la función objetivo es de 11,136,445 mm^2.

Figura 3.21 Solución completa de la instancia A20 con el algoritmo Max_Max. Elaboración propia.

3.3 Algoritmo Ω.

Este algoritmo glotón está diseñado con un criterio de selección hibridado con Max_Max y Min_Max de una manera muy peculiar. Cuando una nueva hoja o corte se genera el artículo es colocado siguiendo el criterio de selección del algoritmo Max_Max, al igual que el umbral aleatorio U. El resto de los artículos que se coloquen en el Corte-1 será de acuerdo con el criterio de selección Min_Max. El criterio de selección Ω se creó con la finalidad de tratar de aprovechar el mayor espacio en el ancho del contenedor, cuando se pone por primera vez un artículo o se genera un nuevo corte, para después aprovechar el mayor espacio en el largo del contenedor. La Figura 3.22 muestra el pseudocódigo del algoritmo Ω.

Entrada:(*I, H, W*); *I*-Lote *I*= {S_1, S_2, …, S_m} donde cada pila *S*= {it_1, it_2,…, it_n} e *it* con dimensiones (h_{it}, w_{it}) donde *it* ϵ *S* y *S* ϵ *I*, *H*- largo del contenedor y *W*- ancho del contendor.
Salida: X- Una solución representada por una lista de patrones
X← ∅;
mientras *I*≠∅ **hacer**
C = Calcula_Candidatos ();
Ω (*C*); //función de ordenamiento para los artículos *it* del *C* subconjunto de candidatos disponibles
si *it* =mejor_artículo_a_la derecha **entonces**
genera patrón p_{it} en la derecha;
si no
si *it*= mejor_artículo_arriba **entonces**
genera patrón p_{it} arriba;
si no
si *it*= mejor_artículo_en_nuevo_corte_uno **entonces**
genera patrón p_{it} en nuevo_corte_uno;
si no
si *it*= mejor_artículo_en_nueva_hoja **entonces**
genera patrón p_{it} en nueva_hoja;
si *it* ϵ nuevo_corte o nueva_hoja **entonces**
CALCULA un umbral *U* a la derecha;
X←X ∪ p_{it};
I = *I* –*it*;
fin de mientras;
retorna X;
fin glotón;

Figura 3.22 Pseudocódigo del algoritmo Ω. Elaboración propia.

El algoritmo comienza con la misma entrada de valores que los algoritmos anteriores, un lote I de artículos it con sus respectivas dimensiones y las longitudes en el ancho W y largo H del contenedor. Su salida es una lista de patrones de corte para los artículos it sobre el contenedor, la lista comienza vacía, en seguida un ciclo itera mientras el lote no este vacío, cada que el ciclo itera calcula la lista de candidatos disponibles y bajo el criterio de selección de Ω comenzará a asignarlos en el contenedor.

La función Ω que se muestra en el pseudocódigo de la Figura 3.23 recibe los valores de la lista de candidatos disponibles para cortar, con sus dimensiones en ancho y largo, la salida es una nueva lista de longitudes ordenadas M con sus respectivos números de artículos, que definen como se van asignando los elementos al contenedor, M comienza vacía. Siempre que se haga un nuevo corte o se use una nueva hoja el criterio a utilizar es Max_Max, de lo contrario el criterio usado es Min_Max.

Entrada: (*C)*; *C* – Subconjunto de artículos candidatos *it,*
Salida: *M*- un conjunto de dimensiones ordenadas para determinar el corte de los artículos y su respectivo número
M← ∅;
  si nuevo_corte_uno **o** nueva_hoja
    it ← Max_Max(*C*);
  sino
    it ← Min_Max(*C*)*;*
inserción(*M*);
retorna *M*;
fin de *Ω*;

Figura 3.23 Pseudocódigo de la función Ω(). Elaboración propia.

Para tener más claro lo que hace la función Ω, la Figura 3.24 (a) muestra la lista de candidatos disponibles que la función Calcula_Candidatos() genera en el paso cinco de la Figura 3.2. La nueva lista de longitudes ordenadas queda como se muestra en la Figura 3.24 (b), donde se toma el articulo it_9 como el elemento con la longitud global mayor que calcula la función Max_Max () y los artículos it_6, it_1 e it_4 son elementos ordenados por la función Min_Max ().

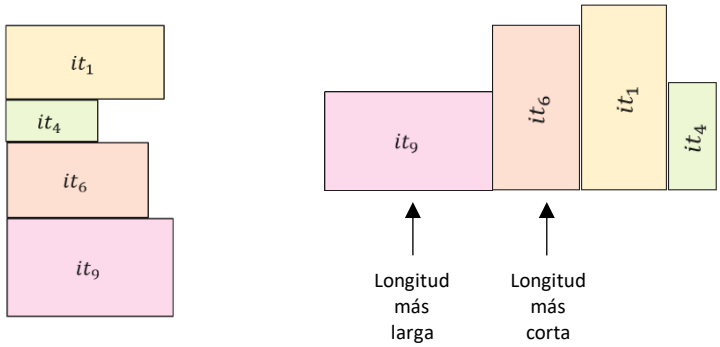


Figura 3.24 Comportamiento de la función Ω().
En (a) se muestran los artículos disponibles que arrojó la función Calcula_Candidatos() de la Figura 3.2. En (b) la función Ω() ordena la lista de candidatos disponibles. Elaboración propia.

Cuando la asignación comienza se puede hacer a la derecha de un artículo, arriba del artículo, en un nuevo corte y en una nueva hoja como ya se explicó en las Figuras 3.5 y 3.6. El umbral es calculado como se hace en el algoritmo Max_Max, porque su criterio de selección se utiliza siempre que se asigne un

artículo en un nuevo corte u hoja. El patrón de corte generado siempre se agrega a la solución y esta solución se retorna hasta que se generen patrones para todos los artículos del lote.

Tomando una vez más los datos de la instancia A20, especificados en la Tabla 3.1 que se visualizan en la Figura 3.8, se crea un ejemplo completo de cómo se comporta el algoritmo Ω.

El algoritmo comienza tomando al primer artículo de cada pila S (véase Figura 3.25(a)), para crear una lista de candidatos disponibles con la función Calcula_Candidatos(), la lista es manda a la función Ω que comienza a ordenar los artículo de la siguiente manera: el contenedor a utilizar es una nueva hoja, por consecuencia la función Max_Max() pone al artículo it_{10} en primer lugar dentro de la nueva lista de longitudes ordenadas, porque es el de mayor longitud global, en seguida la función Min_Max() evalúa los demás artículos y los agrega después del it_{10} como se muestra en la Figura 3.25 (b).

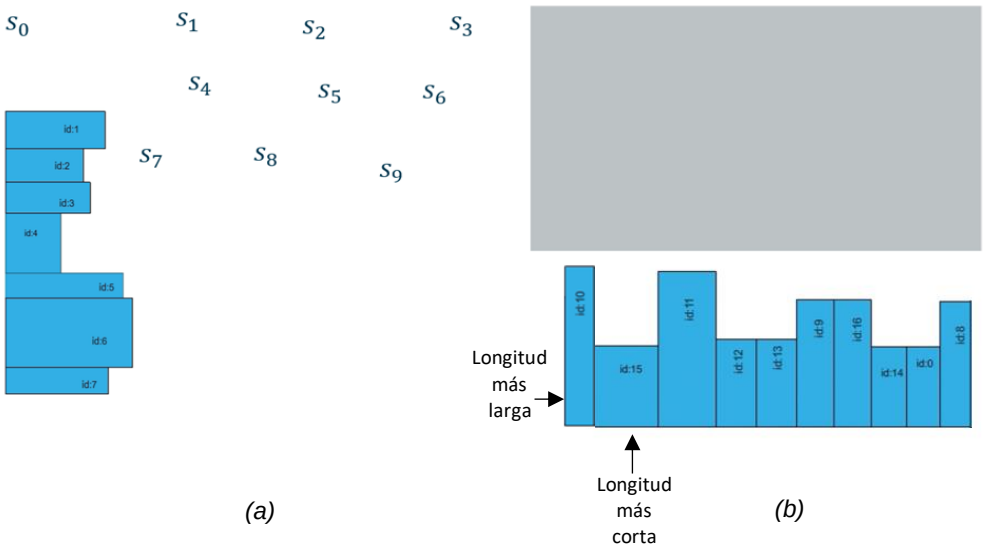


Figura 3.25 Selección y ordenamiento de los primeros artículos a cortar con criterio de selección Ω. En (a) muestra el estado de las pilas después de crear la lista de candidatos. En (b) se encuentra la nueva lista de longitudes ordenadas. Elaboración Propia.

El primer artículo asignado es it_{10}, de tal forma que la longitud más larga del elemento quede sobre el ancho del contenedor, de esta se genera un umbral aleatorio máximo del 45% como ejemplo para este caso(véase Figura 3.26 (a)), después de que fue determinado el umbral los artículos aun no cortados

serán asignados por su lado más corto, así que el siguiente artículo asignado es el it_{15} a la derecha del artículo it_{10} esto porque el umbral lo permite y ninguna restricción se viola (véase Figura 3.26 (b)).

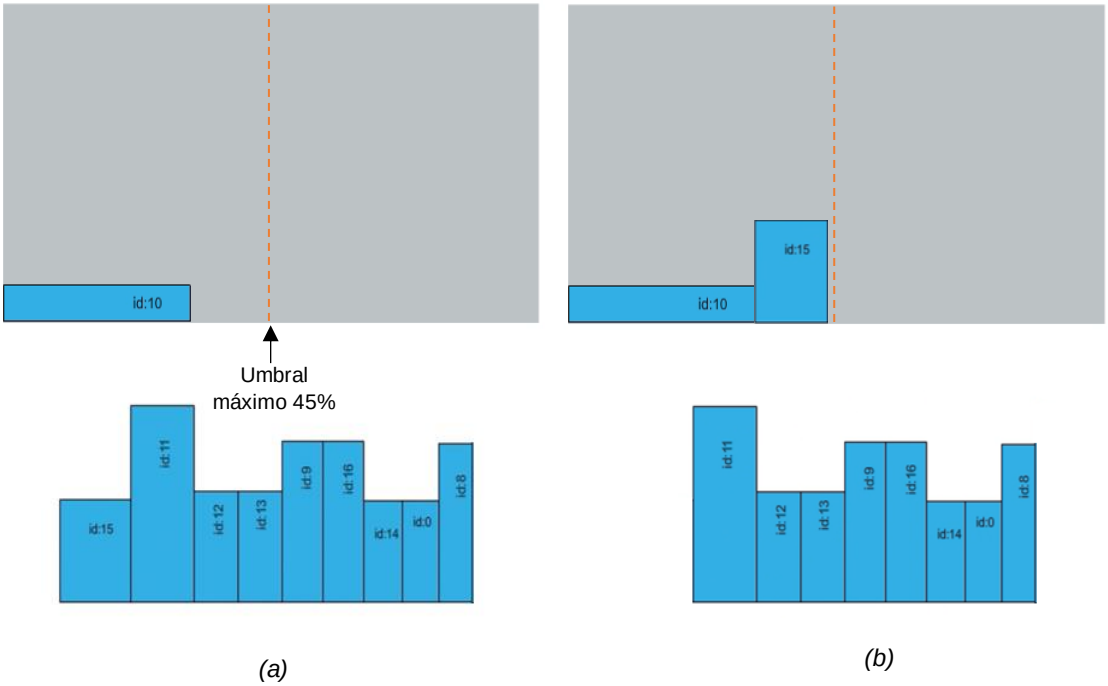


Figura 3.26 Asignación de los primeros artículos.
En (a) se asigna el primer artículo it_{10} a la nueva hoja generando un umbral, que segmenta al contenedor. En (b) el articulo it_{15} es asignado a la derecha de it_{10} por su lado más corto sin pasar el límite del umbral. Elaboración propia.

Es importante señalar que el ordenamiento de la nueva lista de longitudes ordenadas no cambiará, hasta que un nuevo artículo este dentro de la lista de candidatos y sea mandada a la función Ω para ordenarla, así que el siguiente articulo a evaluar es el it_{11} pero no cabe a la derecha del artículo it_{15} porque el umbral no lo permite, entonces se evalúa toda la nueva lista de longitudes ordenadas sin tener éxito, en consecuencia se busca asignar arriba como lo muestra la Figura 3.27 (a). Por restricciones de corte, el artículo it_{11} que se pone arriba de los artículos it_{10} y it_{15} estará a la altura máxima de los dos, en cuanto al eje de las “y” y se encontrará en la misma coordenada “x” del artículo que se encuentre más a la izquierda, en este caso el artículo it_{10}.Todos los

artículos asignados a la derecha de it_{11} se colocan por su lado más corto hasta generar un nuevo corte como lo muestra la Figura 3.27 (b).

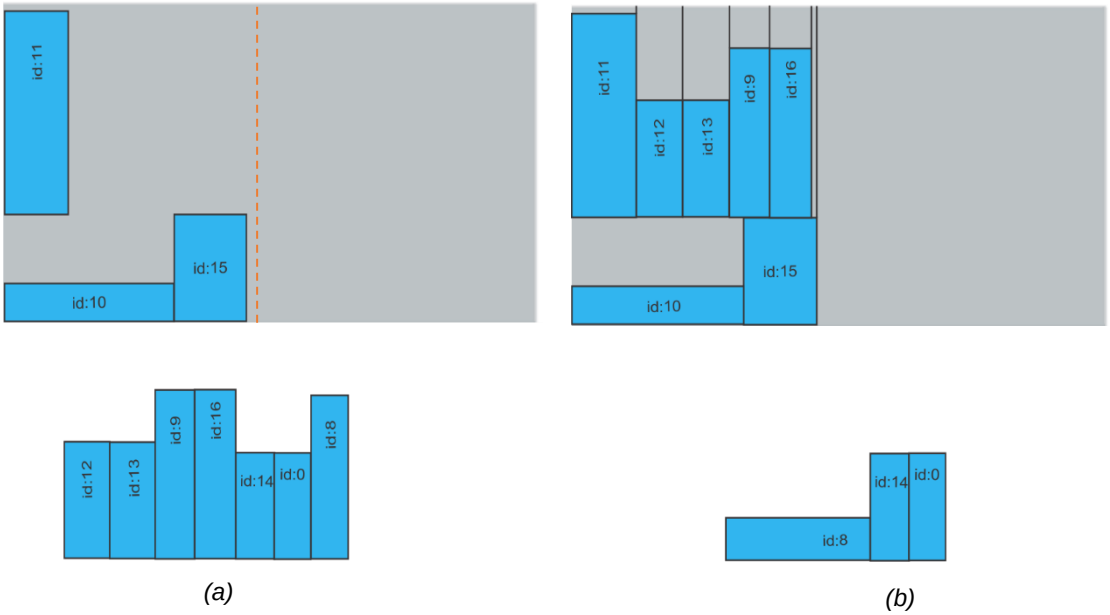


Figura 3.27 Construcción del primer patrón con el algoritmo Ω.
En (a) se asigna it_{11} arriba de it_{10} e it_{15}, esto para respetar el número de tipos de corte en una solución.
En (b) se genera toda la solución del primer patrón, utilizando la función Ω(). Elaboración propia.

El criterio Ω se aplica de manera repetitiva para construir cada patrón de corte. En la figura 3.27 (b) se puede notar que aún quedan más elementos por cortar, así que se repite el procedimiento hasta llegar a la solución que muestra la Figura 3.28. La solución ocupa dos contenedores, la parte gris clara representa la pérdida y la parte gris oscura contiene el residuo, su valor al pasar por la función objetivo es de 10,683,835 mm^2.

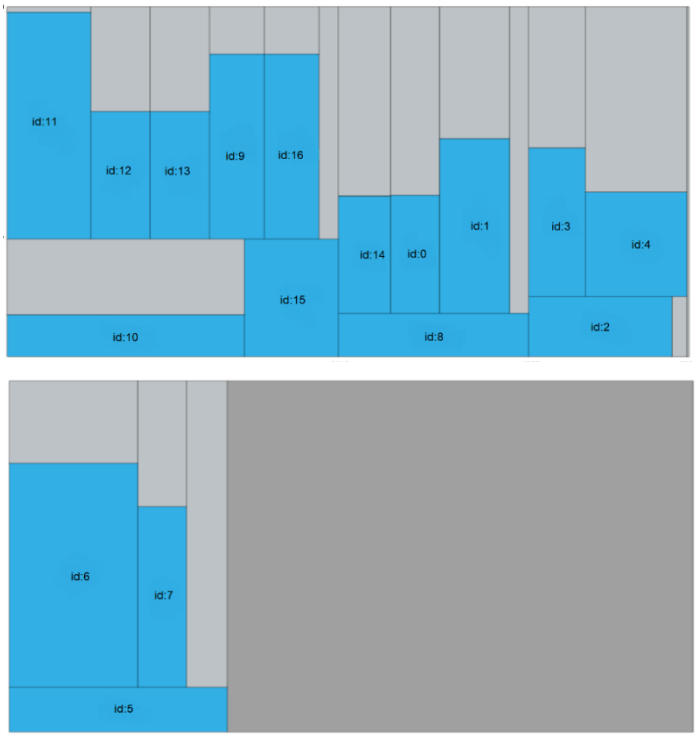


Figura 3.28 La solución completa construida con el algoritmo Ω para la instancia A20. Elaboración propia.

3.4 Algoritmo Ψ.

En esta sección se explica la implementación de un nuevo algoritmo glotón, hibridado con los tres criterios de selección Min_Max, Max_Max y Ω, basado en la conjetura de que la hibridación aleatoria de criterios glotones, al proporcionar más variedad en el acomodo de artículos, se encontrarán soluciones que mejoren la calidad de los glotones de forma individual.

La idea comenzó con la hibridación de Min_Max y Max_Max, la explicación especifica de cómo se implementó y su comportamiento no es necesaria, porque su estructura lógica es la misma que el algoritmo glotón Ψ, hibridado con los tres criterios de selección explicado a continuación.

El algoritmo Ψ, se estructuró pensando en generar patrones de corte válidos mediante los tres criterios de selección, que pueden ser utilizados de manera aleatoria cada vez que un contenedor nuevo sea ocupado o cuando se haga

un nuevo corte, el umbral U sigue usándose en este algoritmo dependiendo de cuál sea el criterio utilizado. La Figura 3.29 muestra el pseudocódigo del algoritmo glotón Ψ.

```
Entrada:(I, H, W); I-Lote I= {S1, S2, …, Sm} donde cada pila S= {it1, it2, …, itn} e it con dimensiones (hit, wit)
donde it ϵ S y S ϵ I, H- largo del contenedor y W- ancho del contendor
Salida: X- una solución representada por una lista de patrones
X← ∅;
φ=rand()%3;
mientras I≠∅ hacer
    C = Calcula_Candidatos ();
    {ordena bajo un criterio φ, el C subconjunto, φ puede valer 0=Min_Max(), 1=Max_Max() y 2=Ω}
        si it =mejor_artículo_a_la derecha entonces
                genera patrón p_i en la derecha;
        si no
           si it= mejor_artículo_arriba entonces
              genera patrón p_i arriba;
           si no
              si it= mejor_artículo_en_nuevo_corte_uno entonces
                 φ =rand()%3;
                 genera patrón p_i en nuevo_corte_uno;
              si no
                 si it= mejor_artículo_en_nueva_hoja entonces
                    φ =rand()%3;
                    genera patrón p_i en nueva_hoja;
          si it ϵ nuevo_corte o nueva_hoja entonces
                 CALCULA un umbral U a la derecha;
     X←X ∪ p_i;
     I = I –it;
fin de mientras;
retorna X;
fin glotón;
```

Figura 3.29 Pseudocódigo del algoritmo glotón Ψ. Elaboración propia.

El algoritmo comienza con la entrada de datos de un lote I y las longitudes de ancho W y largo H del contenedor. La salida es una lista X de patrones de corte, la lista comienza vacía y un valor aleatorio entre cero y dos es generado para asignarlo a φ, el valor determina el criterio de selección a utilizar. Un ciclo comienza a iterar y termina hasta que el lote este vacío, en cada iteración se calcula la lista de candidatos disponibles con la funcion Calcula_Candidatos(), esto por la precedencia que tiene cada pila para cortar sus artículos, después de generar la lista de candidatos disponibles, se ordenan de acuerdo al criterio definido por φ creando una nueva lista de longitudes ordenadas con los artículos para ser cortados.

El algoritmo glotón comienza a evaluar en qué posición pueden ser acomodados los artículos, estos pueden asignarse a una nueva hoja, a la derecha de un artículo, arriba de un artículo y en un nuevo Corte-1 como ya se explicó en las Figuras 3.5 y 3.6, es necesario tener en claro que no importa que criterio se utilice todos toman como prioridad una longitud y justo esa longitud siempre se posicionará en el ancho del contenedor. Cada que se genere un nuevo corte o se ocupe una nueva hoja, el criterio de selección puede cambiar según el valor aleatorio que sea generado para φ, de la misma manera el umbral se volverá a calcular cada vez que esto suceda, porque los artículos son de diferentes longitudes y el umbral toma su valor de acuerdo con un porcentaje aleatorio sobre el artículo que se encuentre después de un nuevo corte uno o una nueva hoja. Cada que un nuevo patrón de corte se genere este se agregará a la lista de patrones, para que al final de las iteraciones se retorne una solución completa de como cortar el lote de artículos.

Para tener la idea más clara se muestra un ejemplo retomando los datos de entrada de la instancia A20, que se especifican en la Tabla 3.1 y se visualizan en la Figura 3.8.

El algoritmo comienza generando el valor de 0 para φ definiendo con esto que la función a utilizar es Min_Max, y tomando el primer artículo de cada pila S con la función Calcula_Candidatos() (véase Figura 3.30(a)), creando una lista de candidatos disponibles que son ordenadas con la función Min_Max(), generando una nueva lista de longitudes ordenadas la Figura 3.30(b) muestra dicha lista con los artículos ordenados para comenzar a asignarse en una nueva hoja.

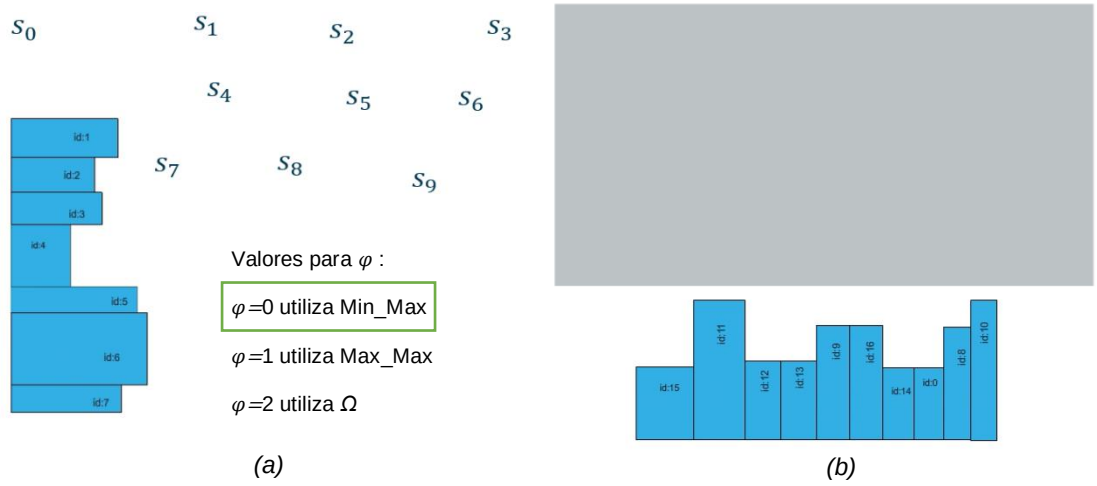


Figura 3.30 Ordenamiento de la lista de candidatos disponibles con el criterio Min_Max. En (a) se define el valor para φ y se toman los primeros artículos de cada pila. En (b) los artículos son ordenandos con el criterio que define φen este caso Min_Max. Elaboración propia.

La construcción del primer patrón es de acuerdo con el algoritmo Min_Max, asignando los artículos it_{15} e it_{11} al contenedor, respetando el umbral que segmenta a la hoja de vidrio, esta delimitación siempre existe aunque en las figuras no se muestra para fines prácticos. Como el umbral no permite asignar más artículos al espacio segmentado se genera un nuevo Corte-1, donde φ cambia de valor a 2 definiendo con esto que la función a utilizar es $\Omega()$, ordenando nuevamente los artículos bajo este criterio, guardando en la lista de longitudes ordenadas al primer artículo por su longitud más larga y al resto por su longitud más corta, para comenzar a asignarlos al contenedor en el nuevo Corte-1 (véase Figura 3.31(a)).

El segundo patrón de corte es construido con el algoritmo Ω, cuando la construcción llega a un nuevo Corte-1 el valor para φ cambia a 1, dando lugar a que la función Max_Max() ordene los artículos restantes bajo ese criterio, si observamos con atención notaremos que el articulo it_0 fue cortado, por lo tanto el articulo it_1 se encuentra disponible para cortar, cuando este sea cortado el articulo it_2 estará disponible y así sucesivamente hasta que la pila a la que pertenecen se encuentre vacía (véase Figura 3.31(b)).

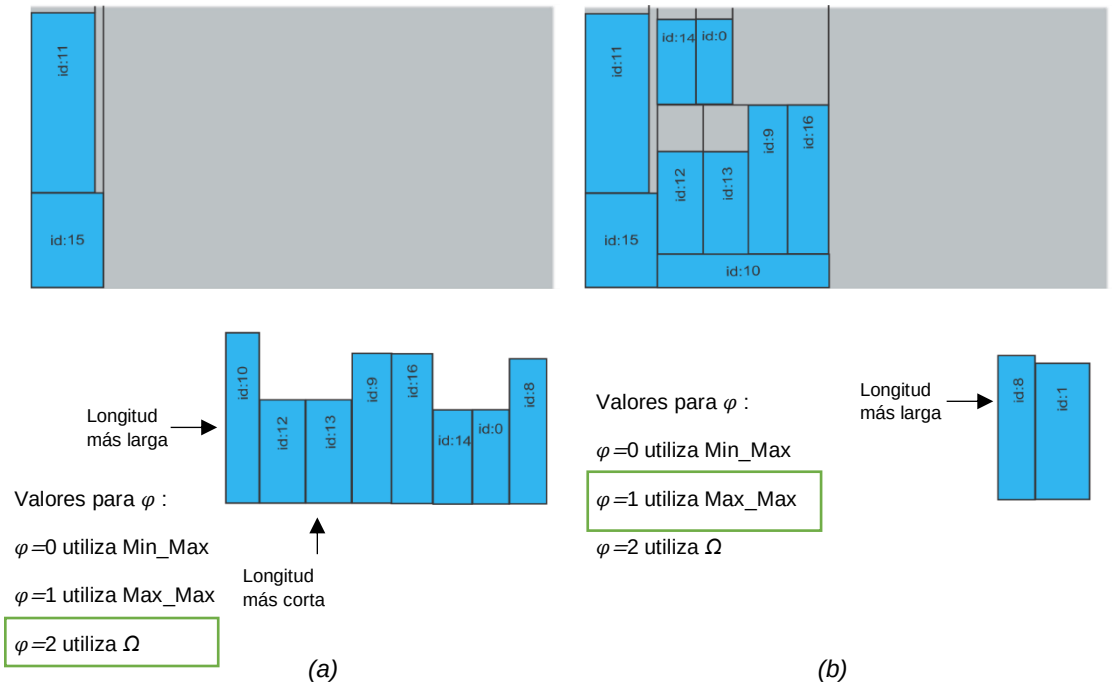


Figura 3.31 Construcción del primer y segundo patrón de corte con el algoritmo Ψ. En (a) la construcción del primer patrón está hecha con el criterio Min_Max, cuando se genera un nuevo corte φ toma el valor de 2 que corresponde a Ω, ordenando bajo este criterio a la lista de candidatos. En (b) se construye un segundo patrón bajo el criterio Ω y se genera un nuevo valor para φ. Elaboración propia.

El tercer patrón de corte es construido con el algoritmo Max_Max, cuando la construcción llega a un nuevo Corte-1, φ cambia de valor a 0 lo que equivale a utilizar la función Min_Max() que ordena el único artículo disponible en este momento por su longitud más corta. La Figura 3.32 (a) muestra la construcción y el ordenamiento descrito.

La Figura 3.32 (b) contiene el cuarto patrón de corte construido, el artículo it_5 es asignado y el siguiente artículo it_6 de la pila a la que pertenecía es incorporado a la lista de candidatos disponibles, ordenándola con la misma función Min_Max() y tratando de asignar el articulo it_6 a la derecha de it_5, pero el umbral no lo permite y cuando se trata de poner arriba de it_5 el largo del contenedor lo impide así que se genera un nuevo Corte-1, cambiando con esto el valor de φ, si recordamos el valor es aleatorio, puede caer en el mismo

criterio de selección, como sucede en este caso pero aun así el articulo it_6 no se puede asignar porque excede el ancho del contenedor.

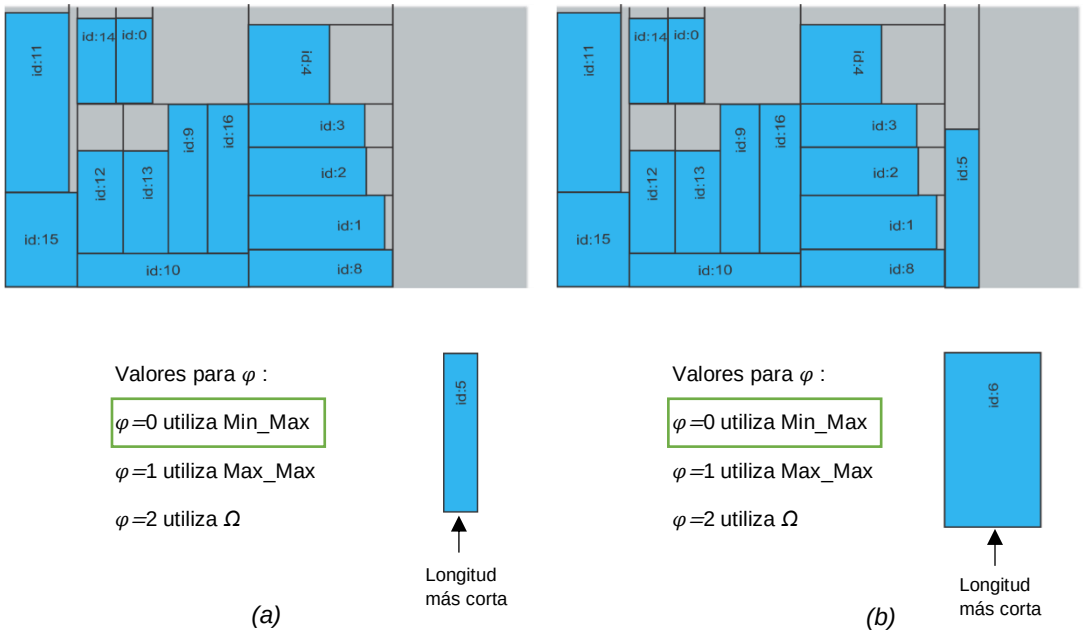
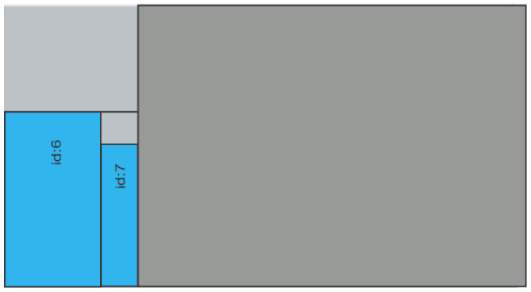


Figura 3.32 Construcción del tercer y cuarto patrón de corte con el algoritmo Ψ.
En (a) un tercer patron es generado utilizando el criterio Max_Max, cuando se genera un nuevo corte, φ toma un valor de 0 que corresponde a Min_Max ordenando a it_{15} por su lado más corto. En (b) se construye un cuarto patrón bajo el criterio Min_Max y se genera un nuevo valor para φ que es 0. Elaboración propia.

Como aún faltan artículos por cortar se incorpora una nueva hoja a la solución y con ello un nuevo valor para φ, el valor aleatorio se repite y construye el ultimo patrón de corte con el criterio Min_Max como se muestra en la Figura 3.33, en este caso el umbral generado con la longitud mínima del articulo it_6 asignada en el ancho del contenedor, permite que el articulo it_7 sea asignado a la derecha del it_6.



Valores para φ :

φ =0 utiliza Min_Max

φ =1 utiliza Max_Max

φ =2 utiliza Ω

Figura 3.33 Construcción del quinto patrón de corte con el algoritmo Ψ. Elaboración propia.

La solución completa consta de dos contenedores que se muestra en la Figura 3.34, donde el área gris oscura representa el residuo que está a la derecha del último Corte-1 en el último contenedor ocupado, y la parte gris clara es la pérdida de vidro al generar los patrones de corte. El valor de la solución al pasar por la función objetivo es de 9,531,445 mm^2.

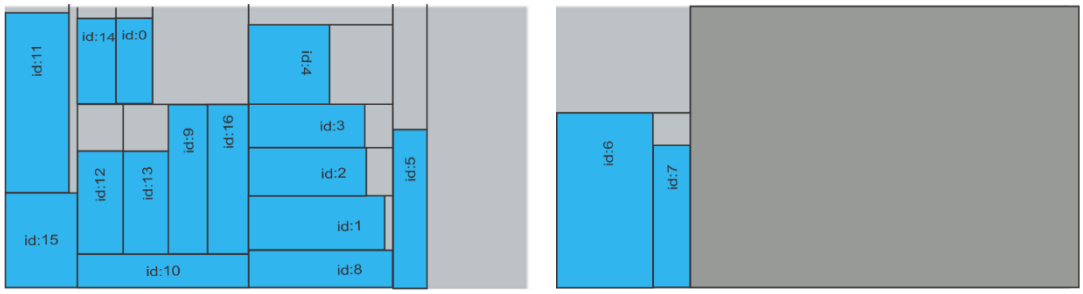


Figura 3.34 Solución completa utilizando el algoritmo Ψ. Elaboración propia.

3.5 Sintonización de parámetros

En esta sección se sintonizan los dos parámetros del algoritmo glotón Ψ, el primero es el umbral U donde se busca encontrar el mejor porcentaje aleatorio para segmentar un contenedor, el segundo parámetro es el número total de corridas para el algoritmo glotón hibridado, donde las soluciones estén más cercanas a las mejores soluciones conocidas del problema con defectos.

Se decidieron tomar cuatro diferentes instancias del tipo A para sintonizar los parámetros, ya que algunas contienen muchos elementos divididos en varias pilas como la instancia A14 y A8, también porque algunos elementos tienen longitudes muy largas para poder ser acomodados fácilmente, como la instancias A2 y peculiarmente la instancia A1 tiene una sola pila con elementos, lo que hace que no tenga manera de combinar artículos para su asignación.

La Tabla 3.2 muestra los resultados de A14 de diferentes números de corridas que comienzan desde 100 hasta 1,000,000 de corridas, cada conjunto de corridas prueba diferentes porcentajes para el umbral aleatorios U, el mejor resultado es de 37,816,038 mm^2 que corresponde a 1,000,000 de corridas con un umbral del 10%.

Tabla 3.2 Resultados de A14 con diferentes umbrales y corridas.

A14					
Corridas / U %	100	1,000	10,000	100,000	1,000,000
10	48,521,388	46,194,138	45,722,268	41,016,408	37,816,038
20	50,665,668	49,949,838	43,016,238	42,021,138	39,668,208
30	60,735,438	51,647,928	49,102,398	47,770,248	46,136,358
40	59,377,608	58,215,588	52,399,068	49,680,198	47,770,248
50	61,810,788	61,216,938	57,191,598	46,444,518	47,433,198
60	67,068,768	64,121,988	56,164,398	54,816,198	50,222,688
70	73,215,918	67,030,248	58,494,858	56,183,658	48,906,588
80	67,781,388	61,300,398	59,201,058	57,807,918	54,270,498
90	72,272,178	61,197,678	60,684,078	55,284,858	53,917,398
100	74,451,768	61,168,788	64,982,268	57,214,068	53,413,428

La Tabla 3.3 muestra los resultados de A2 en donde la mejor solución es de 7,599,929 mm^2, con un umbral U=10% y un número total de corridas de 1,000,000.

Tabla 3.3 Resultados de A2 con diferentes umbrales y corridas.

A2					
Corridas / U %	100	1,000	10,000	100,000	1,000,000
10	13,092,239	10,084,469	9,150,359	8,752,319	7,599,929
20	13,092,239	10,610,909	9,680,009	8,174,519	8,161,679
30	12,068,249	11,679,839	9,192,089	9,150,359	8,829,359
40	13,477,439	12,135,659	10,646,219	9,365,429	8,919,239
50	13,541,639	12,591,479	9,705,689	10,216,079	9,192,089
60	16,719,539	12,077,879	11,666,999	10,376,579	9,233,819
70	16,469,159	12,751,979	11,066,729	10,524,239	9,801,989
80	12,077,879	13,304,099	11,265,749	10,261,019	10,020,269
90	16,844,729	13,705,349	11,612,429	10,787,459	10,174,349
100	13,952,519	13,477,439	12,244,799	10,655,849	10,036,319

En la Tabla 3.4 los resultados de la instancia A8 al ejecutar el algoritmo con diferente número de corridas y umbrales, dejan ver que la mejor solución es de 26,836,454 mm^2 encontrada en un umbral del 10% con 1,000,000 corridas.

Tabla 3.4 Resultados de A8 con diferentes umbrales y corridas.

A8					
Corridas / U %	100	1,000	10,000	100,000	1,000,000
10	37,224,014	32,858,414	28,961,474	28,919,744	26,836,454
20	38,700,614	33,002,864	29,802,494	28,447,874	26,836,454
30	39,653,984	33,760,424	30,925,994	29,526,434	27,122,144
40	39,653,984	34,335,014	31,808,744	30,874,634	28,752,824
50	42,266,924	34,951,334	32,993,234	32,145,794	30,232,634
60	33,551,774	34,537,244	33,676,964	31,905,044	30,977,354
70	42,648,914	39,451,754	34,476,254	32,405,804	28,855,544
80	38,947,784	34,659,224	37,451,924	32,267,774	31,314,404
90	43,634,384	38,796,914	34,591,814	33,278,924	29,555,324
100	42,863,984	38,119,604	34,184,144	33,956,234	32,633,714

Para la instancia A1 que se muestra en la tabla 3.5, al ejecutar el algoritmo con diferentes umbrales y corridas, el primer resultado que alcanza el óptimo de 425,486 mm^2 corresponde a un umbral U=40% y un número de corridas igual a 1,000. El algoritmo se termina de ejecutar con los umbrales del 50 al 100 por ciento y las corridas de 10,000 a 1,000,000, arrojando como resultado el valor del óptimo de manera constate, así que no es necesario tanto desgaste computacional para encontrar el óptimo en esta instancia específicamente.

Tabla 3.5 Resultados de A1 con diferentes umbrales y corridas.

A1					
Corridas / U %	100	1,000	10,000	100,000	1,000,000
10	1,311,446	1,311,446	1,311,446	1,311,446	1,311,446
20	1,311,446	1,311,446	1,311,446	1,311,446	1,311,446
30	1,311,446	1,311,446	1,311,446	1,311,446	1,311,446
40	1,831,466	425,486	425,486	425,486	425,486
50	425,486	425,486	425,486	425,486	425,486
60	425,486	425,486	425,486	425,486	425,486
70	1,247,246	425,486	425,486	425,486	425,486
80	1,311,446	425,486	425,486	425,486	425,486
90	1,247,246	425,486	425,486	425,486	425,486
100	425,486	425,486	425,486	425,486	425,486

Analizando cada una de las mejores soluciones de cada instancia se decidió ejecutar todas las instancias con un umbral U=10% y las corridas =1000,000, a excepción de A1 que se ejecuta con un umbral U=40% y las corridas =1000, porque el resultado llego al óptimo con esos valores en los dos parámetros del algoritmo glotón Ψ.

Capítulo 4 Resultados

En este capítulo se hace una descripción de las cincuenta instancias con las que se trabajaron y la comparación de los algoritmos implementados.

"La mente es como un paracaídas… Solo funciona si la tenemos abierta". Albert Einstein

4.1 Descripción de las instancias

En el presente trabajo de investigación se experimenta con un conjunto de 50 instancias en formato Excel, pertenecientes al challenge ROADEF/EURO 2018-Gobain (ROADEF, 2018), que lanza la Sociedad Francesa de Investigación Operativa y de Apoyo a la Decisión (ROADEF), dedicada a promover el desarrollo de la Investigación Operativa y el Apoyo a la Decisión en Francia, para difundir el conocimiento a la industria y promover su enseñanza en la formación inicial y la educación continua, en junto con la Sociedad Europea de Investigación Operativa (EURO) dedicada a promover esta disciplina.

Las instancias están divididas en 3 tipos, A que contiene 20, B que contiene 15 y X que contiene 15, donde la principal diferencia entre las tres clases de instancias son el número de pilas y artículos que contiene cada lote I.

En las instancias tipo A el número de artículos distribuidos en las pilas en su mayoría no exceden los cien elementos, por lo que son instancias pequeñas y solo A8, A13, A14 y A15 son grandes. La Tabla 4.1 muestra el tamaño de cada una. Si observamos con atención la instancia A3 y A4 se repiten, esto es porque el problema original contiene un tercer factor que son los defectos que tiene cada contenedor a utilizar, las instancias son diferentes en el número de defectos que contienen, pero para esta investigación no se toma en consideración dichos defectos, por consecuencia las instancias son iguales, así que solo se dará el resultado de A4.

Las instancias tipo B son de tamaño grande, a excepción de B1 que es pequeña. Particularmente la instancia B7 contiene el mismo tamaña de pilas y artículos, lo que hace que no exista precedencia en las pilas y las combinaciones de artículos al crear patrones sean mayores (Véase Tabla 4.2).

Tabla 4.1 Instancias tipo A.

Instancias	Pilas (S)	Artículos (it)
A1	1	5
A2	72	72
A3	7	68
A4	7	68
A5	12	97
A6	6	37
A7	9	57
A8	8	129
A9	9	63
A10	9	86
A11	10	86
A12	8	50
A13	11	271
A14	12	361
A15	14	392
A16	5	38
A17	2	21
A18	6	73
A19	6	47
A20	10	17

Tabla 4.2 Instancias tipo B.

Instancias	Pilas (S)	Artículos (it)
B1	68	68
B2	13	383
B3	12	332
B4	12	261
B5	2	207
B6	9	204
B7	241	241
B8	12	334
B9	12	247
B10	13	214
B11	15	274
B12	18	439
B13	18	656
B14	14	267
B15	21	431

Las instancias tipo X en su totalidad son de tamaño grande y la única que no tiene precedencia en sus pilas es X2, lo que hace que existan más posibles patrones de corte, la Tabla 4.3 muestran el tamaño de cada instancia.

Tabla 4.3 Instancias tipo X.

Instancias	Pilas (S)	Artículos (it)
X1	10	300
X2	247	247
X3	12	258
X4	18	371
X5	16	124
X6	18	412
X7	10	215
X8	2	173
X9	12	233
X10	15	375
X11	14	362
X12	14	302
X13	16	344
X14	12	253
X15	12	296

La estructura de cada instancia está conformada por un conjunto de datos que representan los artículos a cortar en un lote I, donde cada artículo contiene un ID que lo identifica dentro del lote, su largo h y su ancho w en milímetros, la pila S a la que pertenece y la secuencia en que debe ser cortado según el orden de precedencia.

La Tabla 4.4 muestra como están distribuidos en columnas los datos de la instancia A20.

Tabla 4.4 Datos de la instancia A20 instancia.

ID_ARTICULO (it)	LARGO (h)	ANCHO (w)	PILA (S)	SECUENCIA
0	1084	430	0	1
1	618	1604	0	2
2	549	1262	0	3
3	498	1365	0	4
4	960	895	0	5
5	405	1911	0	6
6	1126	2052	0	7
7	426	1655	0	8
8	395	1675	1	1
9	482	1700	2	1
10	381	2085	3	1
11	733	2085	4	1
12	522	1171	5	1
13	522	1171	6	1
14	1077	460	7	1
15	1077	827	8	1
16	482	1700	9	1

4.2 Comparación de los algoritmos implementados

En esta sección se muestran los resultados y tiempos promedios de ejecución de cada algoritmo glotón implementado de forma individual o hibridado, comparándose entre sí para determinar cuál de ellos está más cerca a la mejor solución conocida. Los valores sombreados en verde indican que fue el mejor resultado, esto se aplica para todas las instancias. La tabla con los valores sombreados en azul destaca el porcentaje de error entre la solución encontrada y la mejor solución conocida (cota ROADEF).

La mejor solución encontrada y sus tiempos de ejecución, con la que se comparan los resultados de cada instancia se tomaron del challenge ROADEF/EURO 2018-Gobain (ROADEF, 2018), de la sección de resultados (véase Tabla 4.5).

Tabla 4.5 Mejor solución conocida en ROADEF para cada instancia.

Instancia	Mejor solución conocida	Instancias	Mejor solución conocida	Instancia	Mejor solución conocida	Tiempos de ejecución (s)
A1	425,486	B1	2,661,318	X1	14,127,797	3600
A2	7,686,599	B2	13,674,125	X2	5,434,667	3600
A4	3,396,600	B3	18,191,093	X3	7,473,076	3600
A5	3,662,433	B4	8,269,045	X4	11,405,252	3600
A6	3,312,600	B5	72,155,615	X5	4,712,147	3600
A7	4,832,160	B6	11,195,257	X6	10,363,613	3600
A8	9,518,504	B7	8,355,819	X7	21,127,260	3600
A9	3,441,096	B8	16,067,959	X8	24,788,661	3600
A10	4,472,791	B9	17,484,577	X9	20,167,935	3600
A11	5,382,919	B10	21,951,533	X10	17,824,952	3600
A12	2,184,904	B11	22,584,380	X11	12,417,552	3600
A13	13,751,463	B12	13,958,707	X12	10,583,545	3600
A14	14,020,308	B13	24,471,375	X13	13,533,042	3600
A15	14,937,701	B14	8,656,330	X14	8,013,212	3600
A16	3,380,333	B15	24,517,031	X15	11,682,204	3600
A17	3,617,251					3600
A18	5,317,458					3600
A19	3,599,804					3600
A20	1,467,925					3600

Las soluciones de los algoritmos glotones Min_Max, Max_Max y Ω son comparadas entre sí. La Tabla 4.6 contiene los resultados de las instancias tipo A, donde se nota que el algoritmo glotón Min_Max es el mejor de los tres, porque sus resultados son mejores que los de Max_Max y Ω en un 50% de las instancias.

Tabla 4.6 Resultados de los algoritmos glotones para las instancias tipo A.

Instancia	*Pérdida mm^2 Glotón Min_Max*	*Pérdida mm^2 Glotón Max_Max*	*Pérdida mm^2 Glotón Ω*	*Promedio de los tiempos de ejecución(s)*
A1	1,311,446	5,128,136	2,919,656	0.03629
A2	26,009,279	14,283,149	31,867,529	0.06856
A4	13,013,760	14,057,010	9,684,990	0.06133
A5	11,414,583	16,248,843	30,918,543	0.03711
A6	18,300,090	19,930,770	46,361,910	0.03831
A7	29,382,240	17,463,510	41,769,630	0.03522
A8	59,308,814	48,500,744	68,338,544	0.04012
A9	16,785,066	12,275,016	22,466,766	0.03736
A10	23,967,121	24,207,871	40,334,911	0.04502
A11	16,486,309	17,433,259	40,278,829	0.05684
A12	10,312,624	16,732,624	16,944,484	0.0449
A13	43,293,093	43,623,723	107,592,603	0.03885
A14	58,042,248	54,469,518	162,884,058	0.04205
A15	69,164,231	59,116,931	146,457,821	0.0413
A16	15,969,953	14,246,183	15,992,423	0.03107
A17	11,038,771	16,075,261	15,751,051	0.03828
A18	25,158,468	19,415,778	54,876,648	0.06829
A19	10,806,254	13,062,884	22,320,524	0.05145
A20	4,151,485	11,136,445	10,683,835	0.04979

La tabla 4.7 muestra los resultados de cada algoritmo glotón implementado en forma individual para las instancias tipo B, al compararse entre ellos el que encontró mejores resultados fue Max_Max en un 80% de las instancias.

Tabla 4.7 Resultados de los algoritmos glotones para las instancias tipo B.

Instancia	*Pérdida mm^2 Glotón Min_Max*	*Pérdida mm^2 Glotón Max_Max*	*Pérdida mm^2 Glotón Ω*	*Promedio de los tiempos de ejecución(s)*
B1	28,687,998	14,377,818	27,763,518	0.1331
B2	103,493,135	75,505,145	144,574,715	1.113
B3	125,347,313	79,264,553	142,446,983	1.163
B4	37,913,395	41,604,895	73,143,145	1.121
B5	195,178,865	175,122,785	227,253,185	0.2504
B6	60,494,437	50,540,227	72,650,707	1.113
B7	41,200,539	31,917,219	56,017,899	1.136
B8	150,191,389	68,962,339	152,775,439	1.067
B9	147,537,727	66,780,547	167,279,227	0.3039
B10	128,517,113	86,414,753	110,663,093	0.2724
B11	109,989,470	86,414,753	99,739,940	1.123
B12	63,800,377	65,042,647	111,507,397	1.122
B13	126,828,645	123,117,885	208,991,805	1.125
B14	38,891,320	44,123,620	60,501,040	0.2897
B15	142,076,861	121,099,511	148,500,071	0.7991

En las instancias tipo X sucede algo similar, en el 54.33% de las instancias Max_Max obtiene mejores resultados que los otros dos algoritmos glotones (véase Tabla 4.8).

Tabla 4.8 Resultados de los algoritmos glotones para las instancias tipo X.

Instancia	*Pérdida mm^2 Glotón Min_Max*	*Pérdida mm^2 Glotón Max_Max*	*Pérdida mm^2 Glotón Ω*	*Promedio de los tiempos de ejecución(s)*
X1	65,876,207	61,481,717	91,543,367	1.17
X2	22,165,187	33,252,527	41,229,377	1.132
X3	32,514,286	32,434,036	54,493,156	0.7363
X4	75,685,502	58,633,982	85,813,052	1.14
X5	26,463,107	23,272,367	24,870,947	0.1824
X6	55,242,623	55,409,543	103,787,453	1.122
X7	203,875,770	109,100,520	111,466,290	1.062
X8	82,709,901	76,931,901	76,501,761	0.2259
X9	194,856,135	81,511,035	165,237,465	0.288
X10	110,879,642	88,448,162	152,571,122	1.126
X11	55,547,112	61,472,772	103,276,602	1.116
X12	76,966,345	63,972,265	99,002,995	0.337
X13	67,252,392	58,765,152	104,299,002	1.165
X14	45,557,372	48,263,402	65,321,342	0.2812
X15	70,325,694	72,851,964	96,609,174	1.092

Los algoritmos glotones se hibridan para tratar de alcanzar la mejor solución conocida, primero Min_Max y Max_Max, posteriormente se agrega a la hibridación Ω este algoritmo con tres criterios es llamado Ψ, aunque no existe mucha diferencia entre las hibridaciones, sus soluciones son distintas.

La Tabla 4.9 muestra los resultados y tiempo promedio de ejecución del algoritmo hibridado con dos criterios y el algoritmo Ψ. Al compararse las soluciones los valores que se encuentran sombreados son de mejor calidad. Podemos notar que el algoritmo Ψ obtiene mejores resultados en un 91.83% de las instancias tipo A y encuentra el valor óptimo para A1 señalado con un asterisco.

Tabla 4.9 Resultados de los algoritmos hibridados para las instancias tipo A.

Instancia	*Pérdida* mm^2 *Algoritmo hibridado con Min_Max y Max_Max*	*Tiempo promedio por corrida* (s)	*Pérdida* mm^2 *Algoritmo* Ψ	*Tiempo promedio por corrida* (s)
A1	1,311,446	0.000730	**425486***	0.000026
A2	13,756,709	0.003514	7,449,059	0.001004
A4	9,707,460	0.001130	6,343,380	0.000141
A5	11,632,863	0.001760	8,958,933	0.000217
A6	6,567,540	0.001285	5,039,580	0.000083
A7	13,730,280	0.001100	12,282,570	0.000146
A8	31,847,264	0.001650	27,006,584	0.000276
A9	9,029,706	0.001461	6,683,196	0.000154
A10	16,542,391	0.001160	12,440,011	0.000201
A11	14,415,859	0.001160	10,377,679	0.000211
A12	7,625,854	0.000906	4,842,784	0.000108
A13	34,308,303	0.002586	33,720,873	0.000701
A14	47,770,248	0.003139	39,815,868	0.000859
A15	48,462,941	0.003161	44,167,961	0.001211
A16	7,200,233	0.000508	6,895,283	0.000080
A17	6,686,011	0.000592	6,804,781	0.000050
A18	15,749,958	0.000830	10,331,478	0.000145
A19	9,358,544	0.000980	7,313,774	0.000098
A20	9,531,445	0.001207	4,151,485	0.000312

Algo similar ocurre con los resultados de las instancias tipo B que se muestran en la Tabla 4.10, donde la mayoría de los valores sombreados perteneces al algoritmo glotón Ψ, obteniendo un mejor resultado en el 86.66% de las instancias.

Tabla 4.10 Resultados de los algoritmos hibridados para las instancias tipo B.

Instancia	*Pérdida mm^2 Algoritmo hibridado con Min_Max y Max_Max*	*Tiempo promedio por corrida (s)*	*Pérdida mm^2 Algoritmo Ψ*	*Tiempo promedio por corrida (s)*
B1	9,029,958	0.002898	7,280,508	0.001003
B2	58,906,235	0.004784	56,042,915	0.001009
B3	66,119,603	0.004908	60,890,513	0.000889
B4	24,768,445	0.003133	23,481,235	0.000683
B5	129,935,615	0.002954	155,862,785	0.000440
B6	33,366,727	0.003446	32,483,977	0.000555
B7	22,948,479	0.036420	16,785,279	0.027265
B8	53,913,859	0.004742	56,295,679	0.000972
B9	58,829,377	0.003809	45,013,537	0.000805
B10	73,032,263	0.004176	71,549,243	0.000770
B11	72,349,010	0.004443	60,208,790	0.000891
B12	46,135,747	0.005260	44,887,057	0.001443
B13	97,251,705	0.008940	92,680,665	0.002047
B14	30,994,720	0.003393	30,256,420	0.000858
B15	92,591,501	0.005292	81,825,161	0.001285

La Tabla 4.11 muestra la misma tendencia en los resultados para las instancias tipo X, notándose que el algoritmo glotón Ψ genera una mejor solución en el 93.33% de las instancias.

Tabla 4.11 Resultados de los algoritmos hibridados para las instancias tipo X.

Instancia	*Pérdida mm^2 Algoritmo hibridado con Min_Max y Max_Max*	*Tiempo promedio por corrida (s)*	*Pérdida mm^2 Algoritmo Ψ*	*Tiempo promedio por corrida (s)*
X1	49,521,257	0.003116	48,760,487	0.000963
X2	17,860,577	0.035700	13,459,667	0.027958
X3	28,392,646	0.003035	25,712,296	0.000865
X4	40,927,622	0.003967	40,324,142	0.001049
X5	14,566,847	0.001525	10,547,927	0.000574
X6	46,954,403	0.004148	44,633,573	0.001701
X7	81,799,470	0.002299	79,452,960	0.000841
X8	57,103,731	0.002162	63,449,901	0.000497
X9	75,203,385	0.003569	62,186,835	0.000914
X10	61,631,822	0.001751	59,076,662	0.001412
X11	48,157,692	0.003322	43,991,112	0.001249
X12	48,329,935	0.001449	42,458,845	0.001132
X13	45,568,842	0.003507	42,695,892	0.001318
X14	31,478,312	0.001486	29,260,202	0.000847
X15	44,507,664	0.003047	43,316,754	0.000815

Al tener claro que el algoritmo con mejores soluciones es Ψ se muestra en la Tabla 4.12, la mejor solución conocida, el porcentaje de error que existe entre la mejor solución conocida y la mejor solución encontrada, la peor solución, la media, la desviación estándar y el tiempo promedio por corrida de cada instancia tipo A.

Tabla 4.12 Resultados y comparación con la mejor solución conocida para las instancias A.

Instancia	Mejor solución conocida con un tiempo de 3600 segundos	Mejor solución encontrada	Porcentaje de error entre la mejor solución conocida y la mejor solución encontrada	Peor solución encontrada	Media	Desviación	Tiempo promedio por corrida (s)
A1	425,486	425,486	0.00	26,172,896	10,605,462.49	5,604,265.79	0.000026
A2	7,686,599	7,449,059	-3.09	74,184,959	25,954,568.18	7,673,773.29	0.001004
A4	3,396,600	6,343,380	86.76	51,492,030	23,080,800.80	6,201,715.08	0.000141
A5	3,662,433	8,958,933	144.62	55,600,233	26,408,028.14	6,094,085.83	0.000217
A6	3,312,600	5,039,580	52.13	59,876,010	21,705,632.50	6,678,757.84	0.000083
A7	4,832,160	12,282,570	154.18	77,484,090	32,751,063.13	7,260,212.93	0.000146
A8	9,518,504	27,006,584	183.73	96,413,204	52,075,766.34	7,561,688.06	0.000276
A9	3,441,096	6,683,196	94.22	57,918,006	22,449,132.67	5,818,453.93	0.000154
A10	4,472,791	12,440,011	178.13	60,638,161	31,159,186.43	6,421,219.07	0.000201
A11	5,382,919	10,377,679	92.79	56,884,159	28,397,023.65	6,380,418.29	0.000211
A12	2,184,904	4,842,784	121.65	42,537,814	17,734,500.27	5,700,950.17	0.000108
A13	13,751,463	33,720,873	145.22	98,306,073	62,560,242.90	7,869,291.50	0.000701
A14	14,020,308	39,815,868	183.99	116,884,758	70,387,428.97	8,435,806.33	0.000859
A15	14,937,701	44,167,961	195.68	120,443,981	72,571,855.58	7,972,407.82	0.001211
A16	3,380,333	6,895,283	103.98	54,303,773	22,880,197.22	6,595,423.72	0.00008
A17	3,617,251	6,804,781	88.12	62,206,171	23,153,700.75	6,921,105.82	0.00005
A18	5,317,458	10,331,478	94.29	69,106,578	32,872,886.74	7,474,462.62	0.000145
A19	3,599,804	7,313,774	103.17	48,559,064	22,585,313.39	6,280,735.02	0.000098
A20	1,467,925	4,151,485	182.81	48,382,075	18,661,656.11	5,964,034.60	0.000312

La Tabla 4.13 contiene datos similares que la tabla anterior pero para las instancias tipo B.

Tabla 4.13 Resultados y comparación con la mejor solución conocida para las instancias B.

Instancia	Mejor solución conocida con un tiempo de 3600 segundos	Mejor solución encontrada	Porcentaje de error entre la mejor solución conocida y la mejor solución encontrada	Peor solución encontrada	Media	Desviación	Tiempo promedio por corrida (s)
B1	2,661,318	7,280,508	173.57	68,739,168	29,650,899.87	7,743,069.74	0.001003
B2	13,674,125	56,042,915	309.85	143,133,425	90,159,995.61	9,533,709.15	0.001009
B3	18,191,093	60,890,513	234.73	142,800,083	95,233,201.19	9,335,885.34	0.000889
B4	8,269,045	23,481,235	183.97	100,961,005	49,891,304.64	7,696,193.99	0.000683
B5	72,155,615	155,862,785	116.01	326,377,985	232,171,489.93	18,404,492.57	0.00044
B6	11,195,257	32,483,977	190.16	102,009,367	60,924,871.63	7,903,971.97	0.000555
B7	8,355,819	16,785,279	100.88	80,497,359	42,079,589.64	7,481,570.71	0.027265
B8	16,067,959	56,295,679	250.36	155,375,539	100,097,232.80	11,007,052.34	0.000972
B9	17,484,577	45,013,537	157.45	157,491,937	87,755,260.94	11,693,100.56	0.000805
B10	21,951,533	71,549,243	225.94	169,332,263	114,051,531.12	10,358,341.66	0.00077
B11	22,584,380	60,208,790	166.59	135,980,840	93,588,443.53	8,337,976.43	0.000891
B12	13,958,707	44,887,057	221.57	131,675,827	77,682,143.97	8,887,624.40	0.001443
B13	24,471,375	92,680,665	278.73	189,667,605	136,061,765.25	10,962,019.04	0.002047
B14	8,656,330	30,256,420	249.53	91,625,200	56,329,214.07	7,706,234.23	0.000858
B15	24,517,031	81,825,161	233.75	189,604,121	130,066,636.15	11,271,540.83	0.001285

Por último se muestran datos semejantes a las tablas anteriores para las instancias tipo X, en la Tabla 4.14.

Tabla 4.14 Resultados y comparación con la mejor solución conocida para las instancias X.

Instancia	Mejor solución conocida con un tiempo de 3600 segundos	Mejor solución encontrada	Porcentaje de error entre la mejor solución conocida y la mejor solución encontrada	Peor solución encontrada	Media	Desviación	Tiempo promedio por corrida (s)
X1	14,127,797	48,760,487	245.14	117,868,577	76,803,863.38	8,142,045.56	0.000963
X2	5,434,667	13,459,667	147.66	62,884,037	34,258,019.29	6,958,653.22	0.027958
X3	7,473,076	25,712,296	244.07	86,217,586	49,570,392.98	7,002,412.47	0.000865
X4	11,405,252	40,324,142	253.56	111,078,962	65,531,301.86	7,644,111.97	0.001049
X5	4,712,147	10,547,927	123.85	61,538,777	29,541,217.63	5,908,080.92	0.000574
X6	10,363,613	44,633,573	330.68	116,919,563	73,318,712.53	8,371,645.56	0.001701
X7	21,127,260	79,452,960	276.07	178,144,410	117,339,887.10	10,483,074.10	0.000841
X8	24,788,661	63,449,901	155.96	170,930,331	111,042,952.74	12,331,015.51	0.000497
X9	20,167,935	62,186,835	208.35	190,294,725	121,227,870.60	13,302,728.62	0.000914
X10	17,824,952	59,076,662	231.43	146,972,882	96,705,852.97	9,566,786.73	0.001412
X11	12,417,552	43,991,112	254.27	119,808,102	78,100,268.15	8,918,132.58	0.001249
X12	10,583,545	42,458,845	301.18	129,125,635	73,277,480.67	8,529,880.74	0.001132
X13	13,533,042	42,695,892	215.49	117,829,152	74,135,424.27	8,409,245.20	0.001318
X14	8,013,212	29,260,202	265.15	97,440,602	58,801,883.69	7,993,734.61	0.000847
X15	11,682,204	43,316,754	270.79	129,550,194	77,512,665.05	8,620,408.28	0.000815

4.3 Análisis de resultados

En las tablas de resultados se muestran la mejor y la peor solución encontrada en un millón de corridas, para las instancias tipo A, B y X. Para cada instancia, se obtuvo la media de la calidad de la solución, la desviación estándar, y los tiempos de ejecución, en segundos, que muestran la eficacia del algoritmo. La mejor solución conocida del problema con defectos se utiliza como cota inferior porque no existen resultados del problema sin utilizar la restricción de los defectos en el vidrio. Para saber qué tan eficiente es el algoritmo Ψ se comparan sus resultados con los mejores publicados en el ROADEF, esto da una idea de que tan alejado se encuentra de la mejor solución conocida.

El promedio en el porcentaje de error resulta ser alto cuando se comparan las soluciones de los algoritmos glotones individuales con la cota ROADEF. Para el algoritmo Min_Max es de 428.37%, para el algoritmo Max_Max es de 376.34% y para el algoritmo Ω es de 651.54%.

Cuando los criterios Min_Max y Max_Max se hibridaron, se mejoraron las soluciones en 47 instancias de 50, con respecto a utilizar cualquiera de los criterios de forma separada. En cuanto a calidad, mejoró en promedio 19.75% y el porcentaje de error con respecto a la cota ROADEF disminuyo a 228.99%. Cuando el algoritmo Ψ se ejecutó, mejoró los resultados obtenidos por la hibridación de dos criterios de selección, haciendo que el margen de error en promedio fuera de 179.98%. El algoritmo glotón Ψ, mejoró en 46 de 50

instancias con respecto a utilizar el glotón con dos criterios. En cuanto a calidad, mejoró en promedio 14.35%. Incluso llegó a encontrar la solución óptima de la instancia A1 y mejoró la solución en A2, aunque hay que tomar en cuenta que la solución comparada considera defectos y nuestra versión no.

Los parámetros sintonizados favorecen más al tipo de instancias A, posiblemente debido a que en la sintonización solo se utilizaron este tipo de instancias.

Los algoritmos se corrieron en una computadora de escritorio de la marca Dell con procesador Intel(R) Core (TM)i7-4770 CPU 3.40GHz, RAM de 16.0 GB y un sistema Operativo Windows 8.1 de 64 bits.

Capítulo 5 Conclusiones y trabajos futuros

En este capítulo se abordan las conclusiones y los trabajos futuros de la investigación.

"La medida de la inteligencia es la capacidad de cambiar". Albert Einstein

5.1 Conclusiones

En esta tesis de maestría se presentó una solución al problema de corte de vidrio mediante la hibridación de varios algoritmos glotones. La idea principal fue diseñar varios criterios de selección para elegir uno de los artículos disponibles para construir la solución. En cada paso de selección se utiliza de manera aleatoria un umbral que no debe de traspasar el artículo elegido. Se diseñaron en total 3 algoritmos glotones que al ser hibridados dan soluciones de calidad superior comparados con los glotones originales sin hibridar. Los resultados de cada algoritmo glotón, ya sea simple o hibridado son muy rápidos, ya que en el peor caso tarda menos de 3 centésimas de segundo y en la mayoría de los casos el tiempo promedio por corrida se encuentra en factores de 10^{-3} (milisegundos) o en 10^{-4} (décimas de milisegundos).

El enfoque presentado de hibridar tres glotones tiende a generar buenos resultados, por lo que podemos intuir que si se crearán más criterios de selección para los algoritmos glotones y nuevos umbrales quizá podrían mejorar los resultados encontrados.

En conclusión podemos afirmar que la hipótesis H_0 planteada se cumplió, porque es posible generar soluciones factibles utilizando algoritmos glotones que ayuden a minimizar la pérdida de vidrio.

5.2 Trabajos futuros

Analizando los datos obtenidos se determinó que este tipo de algoritmo es eficiente y eficaz para resolver el problema de corte de vidrio ROADEF, pero aún le falta incorporar más criterios de selección. Recordemos que este trabajo de investigación no contiene la restricción de los defectos de ROADEF, por lo tanto, se proponen los siguientes trabajos futuros:

- Sintonizar el algoritmo Ψ experimentando con instancias tipo A, B y X.
- Generar más criterios de selección para incorporarlos al glotón hibridado.
- Crear un nuevo umbral para delimitar el largo del contenedor (y), esto con el fin de no dejar mucho espacio en el largo del contenedor.
- Buscar nuevos umbrales con el fin de mejorar la calidad de las soluciones en las instancias tipo B y X.
- Incorporar al algoritmo la restricción de los defectos en los contenedores.
- Utilizar otras metaheurísticas, como búsqueda tabú y recocido simulado para resolver el problema de corte vidrio ROADEF.
- Generar un algoritmo con base en el desperdicio parcial que se genere al acomodar los vidrios.

Referencias

Alvarez-Valdés, R., Parajón, A., & Tamarit, J. M. (2002). A tabu search algorithm for large-scale guillotine (un)constrained two-dimensional cutting problems. Computers & Operations Research, Vol. 29(7), pp.925–947. doi:10.1016/s0305-0548(00)00095-2

Avila, D., & Sumactika, R. (2007). Un estudio algorítmico del problema de corte y empaquetado 2D.

Beasley, J. E. (1985). Algorithms for Unconstrained Two-Dimensional Guillotine Cutting. Journal of the Operational Research Society, Vol. 36(4), pp. 297–306. doi:10.1057/jors.1985.51

Ben Messaoud, S., Chu, C., & Espinouse, M.-L. (2008). Characterization and modelling of guillotine constraints. European Journal of Operational Research, Vol. 191(1), pp. 112–126. doi:10.1016/j.ejor.2007.08.029

Berkey, J. O., & Wang, P. Y. (1987). Two-Dimensional Finite Bin-Packing Algorithms. Journal of the Operational Research Society, Vol. 38(5), pp.423–429. doi:10.1057/jors.1987.70

Brassard, G., & Bradley, P. (1997). Fundamentos de Algoritmia.

Bratley, P., & Brassard, G. (1997). Fundamentos de algoritmia. Madrid: Pearson Prentice-Hall.

Charalambous, C., & Fleszar, K. (2011). A constructive bin-oriented heuristic for the two-dimensional bin packing problem with guillotine cuts. Computers & Operations Research, Vol. 38(10), pp. 1443–1451. doi: 10.1016/j.cor.2010.12.013

Christofides, N., & Whitlock, C. (1977). An Algorithm for Two-Dimensional Cutting Problems. Operations Research, Vol. 25(1), pp. 30–44. doi:10.1287/opre.25.1.30

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to algorithms. MIT press.

Côté, J.-F., & Iori, M. (2018). The Meet-in-the-Middle Principle for Cutting and Packing Problems. Journal on Computing, Vol. 30(4), pp. 646–661. doi:10.1287/ijoc.2018.0806

Dyckhoff, H. (1990). A typology of cutting and packing problems. European Journal of Operational Research, Vol. 44(2), pp. 145–159. doi:10.1016/0377-2217(90)90350-k

Fayard, D., & Zissimopoulos, V. (1995). An approximation algorithm for solving unconstrained two-dimensional knapsack problems. European Journal of Operational Research, Vol. 84(3), pp. 618–632. doi:10.1016/0377-2217(93)e0221-i

Fayard, D., Hifi, M., & Zissimopoulos, V. (1998). An efficient approach for large-scale two-dimensional guillotine cutting stock problems. Journal of the Operational Research Society, Vol. 49(12), pp. 1270–1277. doi:10.1057/palgrave.jors.2600638

Fleszar, K. (2013). Three insertion heuristics and a justification improvement heuristic for two-dimensional bin packing with guillotine cuts. Computers & Operations Research, Vol. 40(1), pp.463–474. doi:10.1016/j.cor.2012.07.016

Gilmore, P. C., & Gomory, R. E. (1965). Multistage Cutting Stock Problems of Two and More Dimensions. Operations Research, Vol. 13(1), pp. 94–120. doi:10.1287/opre.13.1.94

Goodrich, M. T., & Tamassia, R. (2002). Estructuras de datos y algoritmos en Java. Compañía Editorial Continental.

Hahn, S. G. (1968). On the Optimal Cutting of Defective Sheets. Operations Research, Vol. 16(6), pp. 1100–1114. doi:10.1287/opre.16.6.1100

Kantorovich, L. V. (1960). Mathematical Methods of Organizing and Planning Production. Management Science, Vol. 6(4), pp. 366–422. doi:10.1287/mnsc.6.4.366

Laguna, M., Taillard, E., & de Werra, D. (1993). Tabu search. F. Glover (Ed.). Basel: Baltzer.

Levitin, A. (2000). Design and analysis of algorithms reconsidered. ACM SIGCSE Bulletin, Vol. 32(1), pp. 16–20. doi:10.1145/331795.331802

Lodi, A., Martello, S., & Vigo, D. (1999). Heuristic and Metaheuristic Approaches for a Class of Two-Dimensional Bin Packing Problems. Journal on Computing, Vol. 11(4), pp. 345–357. doi:10.1287/ijoc.11.4.345

Lodi, A., Martello, S., & Vigo, D. (1999). Neighborhood Search Algorithm for the Guillotine Non-Oriented Two-Dimensional Bin Packing Problem. Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, pp. 125–139. doi:10.1007/978-1-4615-5775-3_9

Lodi, A., Martello, S., & Monaci, M. (2002). Two-dimensional packing problems: A survey. European Journal of Operational Research, Vol. 141(2), pp. 241–252. doi:10.1016/s0377-2217(02)00123-6

Lodi, A., Martello, S., & Vigo, D. (2002). Recent advances on two-dimensional bin packing problems. Discrete Applied Mathematics, Vol. 123(1-3), pp. 379–396. doi:10.1016/s0166-218x(01)00347-x

Lodi, A., Monaci, M., & Pietrobuoni, E. (2017). Partial enumeration algorithms for Two-Dimensional Bin Packing Problem with guillotine constraints. Discrete Applied Mathematics, Vol. 217, pp. 40–47. doi:10.1016/j.dam.2015.09.012

Martínez Oropeza, A. (2015). Algoritmo genético cooperativo paralelizado en ambiente grid para el problema de ruteo vehicular con ventanas de tiempo. Universidad Autónoma del Estado de Morelos.

Morabito, R. N., Arenales, M. N., & Arcaro, V. F. (1992). An and—or-graph approach for two-dimensional cutting problems. European Journal of Operational Research, Vol. 58(2), pp. 263–271. doi:10.1016/0377-2217(92)90212-r

Morabito, R. (1994). An AND/OR-graph approach to the container loading problem. International Transactions in Operational Research, Vol. 1(1), pp. 59–73. doi:10.1016/0969-6016(94)90046-9

Morabito, R., & Arenales, M. N. (1996). Staged and constrained two-dimensional guillotine cutting problems: An AND/OR-graph approach. European Journal of Operational Research, Vol. 94(3), pp. 548–560. doi:10.1016/0377-2217(95)00128-x

Peña, D. A., Orejuela, J. P., González, G., & Andrés, C. (2017). El Problema de patrones de corte, clasificación y enfoques. Prospectiva, Vol. 15, N°1, 112-126. Doi: http://dx.doi.org/10.15665/rp.v15i1.718

ROADEF, 2018. Société française de Recherche Opérationnelle et Aide à la Décision (Sociedad Francesa de Investigación Operativa y Apoyo a la

decisión), (ROADEF, por sus siglas en francés) (2018). Referencia electrónica. Recuperado el 1 de Julio de 2019. liga: http://www.roadef.org/challenge/2018/en/index.php

Sahni, S., & Horowitz, E. (1978). Fundamentals of computer algorithms. Computer Science Press.

Tlilane, L. & Viaud, Q. (2018). Cutting Optimization Problem Description. Saint-Gobain, p.10. Challenge ROADEF / EURO

Wang, P. Y. (1983). Two Algorithms for Constrained Two-Dimensional Cutting Stock Problems. Operations Research, Vol. 31(3), pp. 573–586. doi:10.1287/opre.31.3.573

Witenberg, J. P. (1999). Métodos y modelos de investigación de operaciones (Vol. 1). Editorial Limusa.

Cuernavaca, Morelos a 30 de octubre del 2019.

DRA. LAURA PATRICIA CEBALLOS GILES
DIRECTORA DE LA FCAeI
PRESENTE

En mi carácter de revisor de Tesis, hago de su conocimiento que he leído con interés la tesis para obtener el grado Maestría en Optimización y Cómputo Aplicado, de la estudiante Berenice Alonso Muñoz, con matrícula 10010392, con el título Diseño e implementación de un algoritmo heurístico para el problema en el corte de vidrio, por lo cual, me permito informarle que después de una revisión cuidadosa de dicha tesis, concluyo que el trabajo se caracteriza por el establecimiento de objetivos académicos pertinentes y una metodología adecuada para su logro. Además, construye una estructura coherente y bien documentada, por lo cual considero que los resultados obtenidos contribuyen al conocimiento del tema tratado.

Con base en los argumentos precedentes me permito expresar mi **VOTO APROBATORIO** por lo que de mi parte no existe inconveniente para que la estudiante continúe con los trámites que esta Secretaría de Investigación y Posgrado tenga establecidos para obtener el grado mencionado.

Atentamente
Por una humanidad culta
Una universidad de excelencia

Dr. Federico Alonso Pecina
Profesor- Investigador
Facultad de Contaduría, Administración e Informática

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

FACULTAD DE CONTADURÍA, ADMINISTRACIÓN E INFORMÁTICA

Cuernavaca, Morelos a 23 de septiembre del 2019.

DRA. LAURA PATRICIA CEBALLOS GILES
DIRECTORA DE LA FCAeI
PRESENTE

En mi carácter de revisor de Tesis, hago de su conocimiento que he leído con interés la tesis para obtener el grado Maestría en Optimización y Computo Aplicado, del estudiante Berenice Alonso Muñoz, con matrícula 10010392, con el título Diseño e implementación de un algoritmo heurístico para el problema en el corte de vidrio, por lo cual, me permito informarle que después de una revisión cuidadosa de dicha tesis, concluyo que el trabajo se caracteriza por el establecimiento de objetivos académicos pertinentes y una metodología adecuada para su logro. Además construye una estructura coherente y bien documentada, por lo cual considero que los resultados obtenidos contribuyen al conocimiento del tema tratado.

Con base en los argumentos precedentes me permito expresar mi **VOTO APROBATORIO** por lo que de mi parte no existe inconveniente para que la estudiante continúe con los trámites que esta Secretaría de Investigación y Posgrado tenga establecidos para obtener el grado mencionado.

Atentamente
Por una humanidad culta
Una universidad de excelencia

Dr. Martín Gerardo Martínez Rangel
Profesor- investigador
Facultad de Contaduría, Administración e Informática

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

FACULTAD DE CONTADURÍA, ADMINISTRACIÓN E INFORMÁTICA

Cuernavaca, Morelos a 23 de septiembre del 2019.

DRA. LAURA PATRICIA CEBALLOS GILES
DIRECTORA DE LA FCAeI
PRESENTE

En mi carácter de revisor de Tesis, hago de su conocimiento que he leído con interés la tesis para obtener el grado Maestría en Optimización y Computo Aplicado, del estudiante Berenice Alonso Muñoz, con matrícula 10010392, con el título Diseño e implementación de un algoritmo heurístico para el problema en el corte de vidrio, por lo cual, me permito informarle que después de una revisión cuidadosa de dicha tesis, concluyo que el trabajo se caracteriza por el establecimiento de objetivos académicos pertinentes y una metodología adecuada para su logro. Además construye una estructura coherente y bien documentada, por lo cual considero que los resultados obtenidos contribuyen al conocimiento del tema tratado.

Con base en los argumentos precedentes me permito expresar mi **VOTO APROBATORIO** por lo que de mi parte no existe inconveniente para que la estudiante continúe con los trámites que esta Secretaría de Investigación y Posgrado tenga establecidos para obtener el grado mencionado.

Atentamente
Por una humanidad culta
Una universidad de excelencia

Dr. José Crispín Zavala Diaz
Profesor- investigador
Facultad de Contaduría, Administración e Informática

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

FACULTAD DE CONTADURÍA, ADMINISTRACIÓN E INFORMÁTICA

Cuernavaca, Morelos a 23 de septiembre del 2019.

DRA. LAURA PATRICIA CEBALLOS GILES
DIRECTORA DE LA FCAeI
PRESENTE

En mi carácter de revisor de Tesis, hago de su conocimiento que he leído con interés la tesis para obtener el grado Maestría en Optimización y Computo Aplicado, de la estudiante Berenice Alonso Muñoz, con matrícula 10010392, con el título Diseño e implementación de un algoritmo heurístico para el problema en el corte de vidrio, por lo cual, me permito informarle que después de una revisión cuidadosa de dicha tesis, concluyo que el trabajo se caracteriza por el establecimiento de objetivos académicos pertinentes y una metodología adecuada para su logro. Además construye una estructura coherente y bien documentada, por lo cual considero que los resultados obtenidos contribuyen al conocimiento del tema tratado.

Con base en los argumentos precedentes me permito expresar mi **VOTO APROBATORIO** por lo que de mi parte no existe inconveniente para que la estudiante continúe con los trámites que esta Secretaría de Investigación y Posgrado tenga establecidos para obtener el grado mencionado.

Atentamente
Por una humanidad culta
Una universidad de excelencia

Dr. Martín Heriberto Cruz Rosales
Profesor- investigador
Facultad de Contaduría, Administración e Informática

Cuernavaca, Morelos a 30 de octubre del 2019.

DRA. LAURA PATRICIA CEBALLOS GILES
DIRECTORA DE LA FCAeI
PRESENTE

En mi carácter de revisor de Tesis, hago de su conocimiento que he leído con interés la tesis para obtener el grado Maestría en Optimización y Cómputo Aplicado, de la estudiante Berenice Alonso Muñoz, con matrícula 10010392, con el título Diseño e implementación de un algoritmo heurístico para el problema en el corte de vidrio, por lo cual, me permito informarle que después de una revisión cuidadosa de dicha tesis, concluyo que el trabajo se caracteriza por el establecimiento de objetivos académicos pertinentes y una metodología adecuada para su logro. Además construye una estructura coherente y bien documentada, por lo cual considero que los resultados obtenidos contribuyen al conocimiento del tema tratado.

Con base en los argumentos precedentes me permito expresar mi **VOTO APROBATORIO** por lo que de mi parte no existe inconveniente para que la estudiante continúe con los trámites que esta Secretaría de Investigación y Posgrado tenga establecidos para obtener el grado mencionado.

Atentamente
Por una humanidad culta
Una universidad de excelencia

Dr. José Alberto Hernández Aguilar
Profesor- investigador
Facultad de Contaduría, Administración e Informática