UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

PROBLEMA GENERALIZADO DEL EMPAQUETAMIENTO DE CONTENEDORES: UNA COMPARACIÓN ENTRE DIFERENTES MÉTODOS DE SOLUCIÓN

POR

LUIS ÁNGEL GUTIÉRREZ RODRÍGUEZ

COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE MAESTRÍA EN CIENCIAS EN INGENIERÍA DE SISTEMAS

JULIO, 2019

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

SUBDIRECCIÓN DE ESTUDIOS DE POSGRADO

PROBLEMA GENERALIZADO DEL EMPAQUETAMIENTO DE CONTENEDORES: UNA COMPARACIÓN ENTRE DIFERENTES MÉTODOS DE SOLUCIÓN

POR

LUIS ÁNGEL GUTIÉRREZ RODRÍGUEZ

COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE MAESTRÍA EN CIENCIAS EN INGENIERÍA DE SISTEMAS

JULIO, 2019

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Subdirección de Estudios de Posgrado

Los miembros del Comité de Tesis recomendamos que la Tesis «Problema generalizado del empaquetamiento de contenedores: una comparación entre diferentes métodos de solución», realizada por el alumno Luis Ángel Gutiérrez Rodríguez, con número de matrícula 1484412, sea aceptada para su defensa como requisito parcial para obtener el grado de Maestría en Ciencias en Ingeniería de Sistemas .

El Comité de Tesis

Dr. Vincent André Lionel Boyer

Asesor

Dr. Igor Semionovich Litvinchev

Revisor

Dr. Norberto Alejandro Hernández Leandro

Revisor

Vo. Bo.

Dr. Simón Martínez Martínez

Subdirección de Estudios de Posgrado

SUBDIRECCIÓN DE ESTUDIOS DE POSGRADO
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA
U.A.N.L.
FIME

San Nicolás de los Garza, Nuevo León, México julio de 2019

A Dios.

A mis padres, Luis Carlos y Luz María,
y a mi hermano, Carlos Antonio,
por impulsarme y apoyarme
incondicionalmente.

A mi esposa Claudia Estefania
por ser mi compañera, confidente,
por ayudarme a salir adelante y
orientarme para no perder el rumbo.

Al Dr. Vincent, mi director de tesis,
por tenerme tanta paciencia.

"Todo cabe en un jarrito sabiéndolo acomodar"

-Refrán popular mexicano.

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

Agradezco a la Universidad Autónoma de Nuevo León (UANL) por la oportunidad que me brindó para realizar mis estudios de posgrado.

A la Facultad de Ingeniería Mecánica y Eléctrica (FIME) por el apoyo brindado durante mis estudios de maestría.

Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo económico otorgado mediante la beca de estudios de tiempo completo bajo la beca con el *Curriculum Vitae Único* (CVU) 854945.

Agradezco al Posgrado en Ingeniería de Sistemas (PISIS) por darme la oportunidad de realizar mis estudios de maestría.

Quiero expresar mi agradecimiento al Dr. Vincent André Lionel Boyer, director de la tesis, por haberme guiado durante todo este tiempo, darme consejos para mejorar mi investigación, aconsejarme en el rumbo a tomar en esta línea de investigación y ayudarme a desarrollar mi potencial académico y profesional, además de ser un gran compañero y agradecido de poder contar con su amistad.

Agradezco a todos los doctores del Posgrado por ser grandes mentores, por la gran calidad y desempeño aplicado a las clases y por ayudarme a desarrollar mis habilidades de análisis y de investigación científica.

Agradezco a mis compañeros de PISIS por su compañía y ayuda durante mis estudios.

Agradezco profundamente a mi familia: mis padres Luis Carlos y Luz María, así como a mi hermano Carlos Antonio, que siempre están apoyándome y estuvieron haciéndolo durante el transcurso de mis estudios tanto en el ámbito profesional y el personal, siempre ayudándome a enfrentar las adversidades y apoyándome cuando no veía la salida. Gracias a ustedes he podido completar esta meta, recordándome que *si alcanzo el triunfo aquí no habrá fronteras para mí.*

Por último, pero no menos importante, agradezco de todo corazón a mi esposa Claudia Pacheco, por comprenderme, ayudarme, apoyarme, no dejarme desistir, ser el apoyo inmediato que siempre necesité, la voz de la razón y mi motivo para seguir mejorando como profesional y como persona, ser mi compañera de vida y mi mejor amiga y además por siempre estar a mi lado incondicionalmente.

Resumen

Luis Ángel Gutiérrez Rodríguez.

Candidato para obtener el grado de Maestría en Ciencias en Ingeniería de Sistemas

Universidad Autónoma de Nuevo León.

Facultad de Ingeniería Mecánica y Eléctrica.

Título del estudio: Problema generalizado del empaquetamiento de contenedores: una comparación entre diferentes métodos de solución.

Número de páginas: 58.

Objetivos y método de estudio: El objetivo general de esta tesis es analizar modelos de propagación de restricciones para el problema generalizado del empaquetamiento de contenedores.

El problema generalizado de empaquetamiento de contenedores consta de dos clases de objetos que deben ser empacados, los obligatorios y los no obligatorios. Cada objeto tiene un beneficio por ser empaquetado y un volumen. Los contenedores tienen un costo por uso y una capacidad que no debe ser excedida.

El modelo de flujo en redes es uno de los mejores métodos resolviendo los problemas Bin Packing. Por ello utilizaremos este modelo para evaluar el desempeño de los modelos a ser resueltos por propagación de restricciones.

Desarrollamos tres modelos bajo el paradigma de la programación con restricciones para resolver el problema generalizado del empaquetamiento de contenedores basándonos en la librería CPOptimizer del ILOG Cplex 12.9. En esta librería se encuentra la restricción *Pack*, que fue diseñada para resolver los problemas de Bin Packing. En ésta restricción todos los objetos se empaquetan, debido a la definición del Bin Packing. Para poder adaptarlo al problema generalizado del empaquetamiento de contenedores fue necesario agregar restricciones para que los objetos tuvieran la oportunidad de ser descartados. Como *Pack* no permite el descarte de objetos decidimos crear contenedores ficticios para empaquetar ahí los objetos. De acuerdo a la forma de creación de los contenedores ficticios se implementaron dos formulaciones. La primera cuenta con un contenedor ficticio del tamaño de la suma de todos los objetos descartables y costo cero. La segunda genera un contenedor ficticio para cada objeto descartable y cada uno con costo cero. En el tercer modelo, se comprueba si es indispensable la restricción *Pack* ya que se aplica un modelo de asignación en lugar de la restricción antes mencionada.

CONTRIBUCIONES Y CONCLUSIONES: Se proponen tres modelos de programación con restricciones para el problema generalizado del empaquetamiento de contenedores. Los primeros dos con la misma base de empaquetar con la restricción *Pack*, y donde difieren es en la forma de generar contenedores ficticios. El tercero, realiza una asignación de objetos sin la restricción *Pack*.

Los modelos de programación con restricciones tiene un GAP promedio del 5 % con respecto a la solución óptima encontrada por los modelos matemáticos basados en flujo en redes. También se limitó la ejecución, ya que, al aumentar la cantidad de objetos, la convergencia al óptimo conocido se ve impactada. Aún con las limitaciones, se obtuvieron soluciones factibles en poco tiempo.

Firma del asesor: 

Dr. Vincent André Lionel Boyer

Capítulo 1

Introducción

El Generalized Bin Packing Problem (GBPP) es un problema NP-difícil, el cual es una variante general del problema de empaquetamiento de contenedores. Los contenedores cuentan con una capacidad para almacenar objetos y un costo por utilización, éstos pueden ser de diferentes tamaños. Los objetos cuentan con un volumen que ocupan en los contenedores y ofrecen un beneficio por ser empaquetados. Éstos, a su vez, se subdividen en dos grupos los obligatorios y no obligatorios. Los objetos obligatorios son aquellos que deben empaquetarse siempre sin importar su beneficio. Los objetos no obligatorios son opcionales de empaquetar en los contenedores. En este problema se debe de utilizar la menor cantidad de contenedores y se deben empaquetar los objetos no obligatorios que otorguen un mayor beneficio para incrementar la utilidad. El objetivo del GBPP está íntimamente relacionado con dos problemas clásicos de la investigación de operaciones, el Bin Packing Problem y el Knapsack Problem.

El Bin Packing Problem (BPP) es un problema NP-difícil, donde se tiene un conjunto de objetos y un conjunto de contenedores. Los objetos tienen un volumen mientras que los contenedores cuentan con un costo de uso y una capacidad. La capacidad no debe ser excedida por la suma de los volúmenes de los objetos que contiene. El objetivo del BPP es minimizar la cantidad de contenedores que debemos usar para empacar todos los objetos.

El Cutting Stock Problem (CSP) es un problema NP-difícil, donde se cuenta con patrones de varios tamaños de objetos para cortar un material. Se debe cumplir la demanda de objetos que deben cortarse. El objetivo del CSP es minimizar la cantidad de desperdicio al cortar diferentes objetos en un material. Se observa que el BPP y el CSP son problemas idénticos, ya que ambos buscan minimizar el desperdicio y ordenar los objetos en los recursos disponibles.

El Knapsack Problem (KP) es un problema NP-completo, donde se tienen varios objetos que pueden ser guardados en un contenedor. Todos los objetos son opcionales de empaquetar y tienen dos propiedades, un volumen y un beneficio por empaquetarse. El objetivo del KP es maximizar la ganancia de empaquetar ciertos objetos cuya suma de volúmenes no exceda la capacidad del contenedor. El BPP es una variante del KP, donde todos los objetos son obligatorios y hay más de un contenedor. En BPP el beneficio pierde relevancia en la toma de decisiones.

Si se toma el objetivo del BPP que es minimizar el desperdicio y el objetivo del KP donde se busca maximizar el beneficio por empaquetamiento de ciertos objetos, se obtiene el objetivo de GBPP. Por este motivo el GBPP resuelve problemas de KP, BPP y CSP, ya que el CSP es un problema idéntico al BPP.

En este trabajo se estudia el GBPP y se ponen a prueba los modelos matemáticos más recientes y potentes de la literatura. Además, se proponen tres modelos para la solución de GBPP a través de Constraint Programming (CP) los cuales arrojan soluciones factibles en poco tiempo.

Las aplicaciones del BPP son muy amplias y se encuentran desde el área del transporte y cargamento de paquetes, como llenar un barco mercante, organizar los paquetes de la carga de un camión de envíos. Así mismo, en el área de las tecnologías de información se usa para empaquetar los respaldos minimizando el espacio, compresión de archivos, inclusive la gestión de memoria RAM de Máquinas Virtuales en el equipo Host y también el almacenamiento de Máquinas Virtuales en la Nube.

Para ilustrar este estudio y sus diferentes formulaciones, usaremos el siguiente ejemplo. Supongamos que tenemos 7 contenedores con capacidad 6, y queremos empaquetar 13 objetos, 3 objetos con un volumen de 6, 4 objetos con un volumen de 3 y 6 objetos con un volumen de 2. En la Figura 1.1 podemos observar que contamos con dos grupos de elementos, a la izquierda podemos ver los contenedores y a la derecha los objetos que están ordenados de forma decreciente. Al aplicar la heurística Best-Fit Decreasing obtendremos la solución de la Figura 1.2. Podemos observar que tenemos exactamente la cantidad justa para empaquetar todos los objetos. En este caso la mínima cantidad de contenedores a utilizar es 7 contenedores, sin espacio desperdiciado.

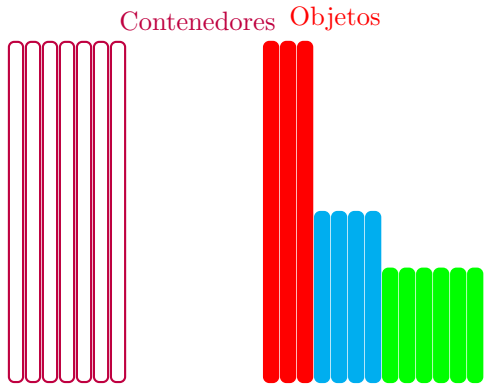

Figura 1.1: Grupos de contenedores y objetos

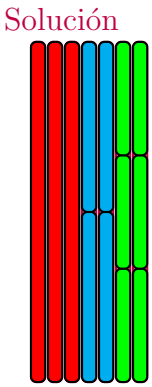

Figura 1.2: Solución usando el constructivo First-Fit Decreasing.

En el GBPP, se tienen objetos obligatorios y no obligatorios, por esto algunos objetos pueden quedar fuera de los contenedores. Para ejemplificar GBPP se retoma el ejemplo anterior y se aplican las nuevas condiciones. En la Figura 1.3 a la izquierda se ven los contenedores, debajo de éstos se encuentra el costo por uso de cada uno y a la derecha los objetos que están ordenados de forma decreciente y en su interior el beneficio si son empaquetados. Los objetos tienen un patrón si son obligatorios y se encuentran llenos de color si son no obligatorios. En este ejemplo, el costo por uso de contenedor es igual a su capacidad, pero ésto no siempre es así. Al resolver el GBPP obtenemos la solución de la Figura 1.4 donde podemos observar que tenemos

dos objetos descartados y un contenedor vacío.

Esto es debido a que la suma del beneficio es 4 mientras que el costo por contenedor es 6. Si los objetos fueran obligatorios serían empaquetados, aún con pérdidas. Como el GBPP se resuelve en términos de gasto por uso y beneficio por empaquetamiento, esta solución tiene un gasto de 36 y un beneficio de 87 dejando una utilidad de 51. Los objetos obligatorios no son considerados en la suma del beneficio ya que siempre tienen que empaquetarse y su beneficio se considera constante.

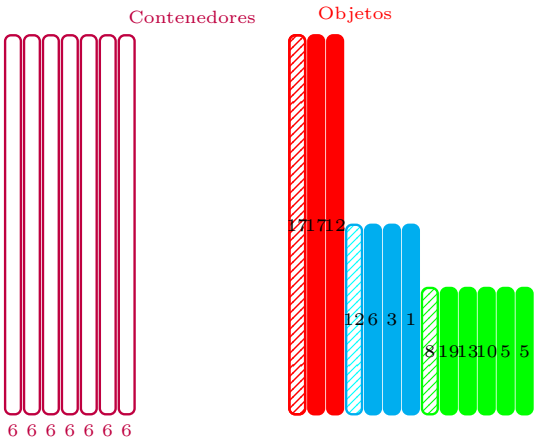

Figura 1.3: Contenedores con capacidad y objetos con volumen

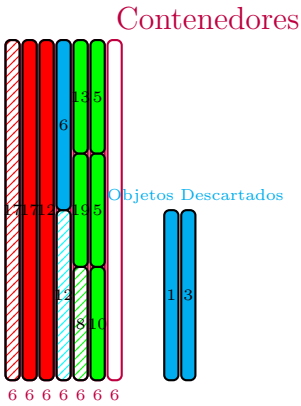

Figura 1.4: Solución óptima.

En el siguiente capítulo, se presenta el estado del arte del GBPP, se mostrarán las variantes de las cuales provienen, los métodos de solución, donde haremos énfasis en la formulación pseudo polinómica de Arc-Flow Model. En el capítulo 3, analizaremos los métodos de solución para cada variante de BPP y se presentarán las condiciones de los modelos de asignación y su homólogo basado en flujo en redes. Mostraremos los mejores modelos conocidos hasta el momento y los modelos basados en programación con restricciones propuestos en este trabajo.

Capítulo 2

Revisión de Literatura

En la década de los treinta del siglo XX, se sabía que un método para aumentar la productividad era mejorando la tecnología y la calidad de los recursos a procesar. Existía también otro método poco estudiado en la época, que consistía en mejorar la organización y planificación de la producción con este mismo objetivo.

Los problemas de Cutting and Packing (C&P) son todos aquellos problemas de optimización donde se tienen dos conjuntos de entidades, las entidades *contenedor* y las entidades *objeto* que deben ordenarse para poder entrar en las entidades contenedoras. Un ejemplo de los problemas de corte sería el CSP, donde se busca ordenar los patrones de ropa (entidad objeto) para cortar un rollo de tela (entidad contenedor) buscando minimizar el desperdicio de tela. Un ejemplo de los problemas de empaquetamiento sería el BPP, donde se busca ordenar los objetos (entidad objeto) para empaquetarlos en un contenedor (entidad contenedor) buscando minimizar el espacio sin utilizar dentro de los contenedores.

En [Kantarovich, 1939] se sentaron las bases de los problemas (C&P), ya que se realizaron análisis matemáticos sobre la planificación de varias industrias para alcanzar el objetivo de producción. Para ello, se debía lograr la producción más alta posible basándose en la utilización óptima de los recursos disponibles. Se buscaba minimizar la chatarra del proceso de producción y también se buscaba maximizar

la utilización de las materias primas. En [Scholl et al., 1997] se aborda el BPP y se aprecia claramente que cuando se minimiza el desperdicio o se maximiza la utilización, se está tratando con las dos versiones del objetivo de los problemas de BPP y CSP.

Posteriormente en el libro *Knapsack problems* de [Martello and Toth, 1990a] en el capítulo 8 los autores recopilan las heurísticas usadas hasta el momento para resolver BPP y además proponen un posible modelo matemático de asignación para abordarlo. Por parte de los métodos exactos en un principio se trabajó con Programación Dinámica y relajaciones lineales.

Después de la década de los sesenta del siglo XX, se desarrollan formulaciones exactas basadas aplicaciones para la industria, como el One-cut, el DP-Flow y el Arc-Flow. En [Dyckhoff, 1981] el One-cut es un modelo desarrollado para poder resolver el CSP, basándose en ordenar los moldes de los objetos para minimizar pérdidas. En [Cambazard and O'Sullivan, 2010] se propone un algoritmo para resolver problemas de flujo en redes usando la programación dinámica, conocido como DP-Flow, donde se toma en cuenta la consistencia y propagación de restricciones de KP aplicándolas a BPP. El Arc-Flow, se basa en la formulación del BPP en términos de grafos [Valério De Carvalho, 1999], donde cada arco representa un objeto, los nodos representan la capacidad del contenedor y el grafo representa el contenedor con todas las combinaciones posibles de los objetos que contiene.

2.1 Tipología de los problemas Cutting and Packing

En [Dyckhoff, 1990] se presenta una tipología para integrar los distintos problemas de C&P tomando en cuenta la estructura lógica de estos. Esta estructura consiste en que se tienen dos grupos de datos básicos, cuyos elementos definen *figuras* de uno o más dimensiones espaciales. Las figuras donde se va a almacenar son los

contenedores y las figuras que se van a ordenar son los *objetos*. Otra particularidad de los problemas C&P es que el proceso de corte o embalaje consiste en realizar combinaciones geométricas de los objetos asignados a los contenedores. Las figuras que no corresponden a los tamaños de objeto generalmente son tratados como pérdidas, como el espacio vacío en BPP o los residuos del corte en CSP. La estructura lógica de los problemas se representan en cuatro directrices para clasificar un problema de C&P como se ve en la Tabla 2.1.

Dimensión	
(1)	una dimensión
(2)	dos dimensiones
(3)	tres dimensiones
(n)	n-dimensiones con n >3
Tipo de Asignación	
(B)	Una selección de objetos a todos los contenedores
(V)	Todos los objetos a una selección de contenedores
Conjunto de contenedores	
(O)	Un solo contenedor
(I)	Contenedores idénticos
(D)	Contenedores de diferentes formas
Conjunto de objetos	
(F)	Pocos objetos de diferentes formas
(M)	Muchos objetos de muchas formas diferentes
(R)	Muchos objetos de pocas formas diferentes (no congruentes)
(C)	Muchos objetos de formas congruentes

Tabla 2.1: Clasificación de Dyckhoff

Aplicando la tipología de Dyckhoff podemos representar los problemas antes mencionados con esta clasificación, como se ve en la Tabla 2.2. El BPP clásico es un problema de una dimensión, donde deben agregarse todos los objetos a una selección de contenedores. Los contenedores tienen una forma idéntica y los objetos tienen muchos tamaños diferentes. En la tipología de Dyckhoff se clasifica como 1/V/I/M.

Problema	Dimensión	Asignación	Contenedores	Objetos
Knapsack Problem	1	B	O	F
Cutting Stock Problem	1	V	I	R
Bin Packing Problem	1	V	I	M
Generalized Bin Packing Problem	1	-	D	M

Tabla 2.2: Clasificación de Cutting and Packing Problems

Al utilizar esta clasificación el GBPP quedaría sin una clase de asignación. Para solucionar ésto, es necesario ampliar esta clasificación. Se debe agregar un tipo de asignación nuevo donde se pueda asignar una selección de objetos en una selección de contenedores. Al hacer esto podemos clasificar aún más problemas de C&P.

El GBPP es un problema de una dimensión, donde deben agregarse una selección de objetos a una selección de contenedores. Los contenedores tienen formas diferentes y los objetos tienen muchos tamaños diferentes. En la tipología de Dyckhoff se clasifica como 1/-/D/M. En este trabajo se utilizará la clasificación de Dyckhoff para caracterizar las variantes del BPP.

2.2 Variantes del Bin Packing Problem

En BPP existen múltiples variantes, donde cada una de ellas aportan elementos al problema clásico. Estas se distinguen entre sí por su planteamiento ya que engloban y generalizan el problema clásico. Dentro de este trabajo solo se consideran las variantes que permiten ir desde BPP hasta el GBPP. Se hace énfasis en aquellas cuyo planteamiento podía relacionarse a un modelo de flujo en redes.

2.2.1 Bin Packing Problem

Clasificación: 1/V/I/M

El Bin-Packing Problem(BPP) es uno de los problemas más famosos de la optimización combinatoria. Su estructura y aplicaciones han sido estudiadas desde la década de 1930, ver [Kantarovich, 1939]. En [Gilmore and Gomory, 1961] para este tipo de problemas se introdujo el algoritmo de *Generación de columnas*, derivado de las ideas de [Ford and Fulkerson, 1958] y [Dantzig and Wolfe, 1960]. En [Martello and Toth, 1990a] se presenta el BPP clásico explicado en términos del KP. Se define como un conjunto de objetos con volumen y un conjunto de contenedores con una capacidad, donde se guardan todos los objetos. Se propuso un modelo matemático de asignación, el cual fue la base para el desarrollo del modelo flujo en redes. También se habla del problema generalizado de asignación cuya formulación es similar al GBPP. Existen artículos dedicados a demostrar y realizar una aplicación de este problema a situaciones similares en la industria, podremos encontrar esto en [Koch et al., 2009], [Stadtler, 1990], [Gradišar et al., 1997],[Kallrath et al., 2014], [Morabito and Garcia, 1998], entre otros.

En la industria metalúrgica, en Alemania [Stadtler, 1990] se busca encontrar la mejor forma de decidir cómo cortar el aluminio para hacer marcos para ventanas. En ésta publicación comparan los resultados que se obtienen usando la heurística First-Fit Decreasing y se propone una solución obtenida por la generación de columnas de [Gilmore and Gomory, 1961]. Es la primera vez que se reporta en la literatura una aplicación de la industria calculando desde una computadora personal.

En la industria textil, en [Gradišar et al., 1997] se aplicaron las heurísticas de First-Fit Decreasing y Best-Fit Decreasing, donde se busca minimizar la pérdida de tela al momento de cortar patrones de ropa y se define un modelo matemático que se aproxima a un BPP. Se tiene como objetivo disminuir la pérdida hasta un 0.1 %. Se realizan pruebas sobre 2 conjuntos de datos se determina que en el primer

conjunto tiene como resultado el 0.03 % de pérdida en el mejor caso y 0.17 % en el peor caso, excediendo la restricción del 0.1 %; En el segundo conjunto de datos se obtuvo un 0.01 % de pérdida en el mejor caso y 0.06 % de pérdida en el peor caso. Se programó una solución en FORTRAN llamada *Computerized Laying out* (COLA), la cual además de poder estar corriendo en una computadora personal, entregaba resultados en menos de 10 segundos.

En la industria maderera, en Brasil [Morabito and Garcia, 1998] tratan el problema de corte de tablones y derivados de la madera, donde se determina que en ciertas configuraciones de cortes se produce un mayor ahorro de material. Se considera que al combinar ciertos objetos se ahorra aún más si existe compatibilidad entre los objetos.

En la industria papelera, [Kallrath et al., 2014] se utiliza un enfoque de generación de columnas para CSP, un método de enumeración de columnas y un método exhaustivo. Después se compararon los resultados de los tres métodos, donde comprueba que el método exhaustivo que propone es el que les da un mayor beneficio.

2.2.2 Variable Size Bin Packing Problem

Clasificación: 1/V/D/M

El Variable Size Bin Packing Problem (VSBPP) se introduce en[Friesen and Langston, 1986] y es una generalización del BPP donde se agrega la posibilidad de tener diferentes tamaños de contenedores. Todos los objetos deben ser almacenados en la menor cantidad de contenedores posibles. Se considera también una cantidad restringida de contenedores de cada tipo.

En [Kang and Park, 2003] se proponen dos algoritmos basados en los constructivos *First-Fit Decreasing* y *Best-Fit Decreasing*. Se modifican las cotas de los constructivos y se introducen en una meta-heurística *iterated greedy*. Fueron nombrados

Iterated First-Fit Decreasing(IFFD) e *Iterated Best-Fit Decreasing*(IBFD) respectivamente. También, en [Haouari and Serairi, 2009] se proponen cuatro heurísticas basadas en el Subset sum problem (SSP) y sus variantes. Además, se propone una heurística basada en el Set-Covering Problem (SCP) y otra basada en algoritmos genéticos. El algoritmo genético fue el que mejor se desempeñó. En [Correia et al., 2008], se abordan los métodos de solución del VSBPP y se desarrolla un modelo discreto para VSBPP se basa en el modelo propuesto por [Gouveia, 1995] para el problema de *Capacitated Minimum Spanning Tree*, posteriormente se demuestra su eficacia en la nueva formulación en [Gouveia and Saldanha-da Gama, 2006].

2.2.2.1 Variable Cost and Size Bin Packing Problem

El Variable Cost and Size Bin Packing Problem (VCSBPP) se introduce en [Crainic et al., 2011] y es una variante del VSBPP donde el costo del contenedor no tiene relación con la capacidad del mismo, ya que en varias publicaciones anteriores se considera que existe una. Además, se le agrega un beneficio a cada objeto por ser empaquetado y todos los objetos son obligatorios de empaquetar. Debido a que el costo de los contenedores es variable, se busca empaquetar los objetos en los contenedores que ayuden a disminuir el impacto de uso. El objetivo de esta variante es minimizar la diferencia entre costo-beneficio contemplando también la capacidad del contenedor y el volumen de los objetos como en las variantes anteriores.

En [Monaci, 2003] se aborda un problema de calendarización basándose en el trabajo de [Caprara et al., 2001] desde un enfoque VCSBPP. Específicamente para el problema de una compañía de trenes italiana, donde se aplicaba el Crew Planning Problem (CPP). El cual consiste en que se debe definir un conjunto de grupos de conductores que cubran un conjunto determinado de servicios de trenes todos los días, minimizando el costo de servicio.

2.2.3 Generalized Bin Packing Problem

Clasificación: 1/-/D/M

El Generalized Bin Packing Problem(GBPP) es introducido por [Baldi et al., 2012] como una generalización del VCSBPP, en el cual se consideran conjuntos de objetos que tienen volumen y beneficio, y un conjunto de contenedores de varios tipos con volumen y costo. Es importante saber que lo que difiere este problema con sus antecesores es el que algunos objetos no obligatorios pueden quedar fuera de la asignación si el costo de empaquetamiento es mayor al beneficio que proporciona. [Brandão, 2017] en su tesis doctoral introduce el modelo del Generalized Vector Packing Problem(GVPP) que también da solución a los problemas del GBPP. En GVPP considera que los objetos son vectores de 1-Dimensión y se compara con la solución del modelo de [Baldi et al., 2012].

2.2.3.1 Generalized Bin Packing Problem with bin-dependent item profits

El Generalized Bin Packing Problem with bin-dependent item profits(GBPPI) [Baldi et al., 2019] es una extensión de GBPP donde los beneficios de los objetos varían si son empaquetados junto a otros con los cual hay una compatibilidad. Esta relación de empaquetado está dada por una función que representa la afinidad entre los objetos y nos arroja un mayor beneficio a mayor afinidad.

2.2.4 Otras variantes

En este trabajo se abordan solo las variantes de BPP de una dimensión. Sin embargo existen variantes que exploran el problema de dos dimensiones y sus variantes que podemos observar en “Two-dimensional packing problems: A survey”

[Lodi et al., 2002] y el problema de tres dimensiones, con sus variantes en "The three-dimensional bin packing problem" [Martello et al., 2000]. Además en "Approximation algorithms for bin packing problems: A survey" [Garey and Johnson, 1981] se plantean heurísticas para la solución del problema y se propone el *Vector packing problem*, el cual es el BPP de n dimensiones y cada dimensión corresponde a los atributos de los objetos a empaquetar.

2.3 Métodos de Solución

Debido a que el BPP es uno de los problemas combinatorios más famosos, se ha tratado de resolver de diversas formas, en un inicio con algoritmos de aproximación, dónde se han desarrollado heurísticas, meta-heurísticas e hyper-heurísticas para encontrar una solución factible en un tiempo de cómputo razonable. También existen métodos de solución exacta, dónde se ha utilizado algoritmos de enumeración, formulaciones pseudo-polinómicas y un modelo basado en la programación con restricciones.

2.3.1 Algoritmos de aproximación

En un principio, el BPP solo se había resuelto con algoritmos de aproximación, los cuales van desde la definición de cotas mínimas hasta la aplicación de constructivos básicos.

2.3.1.1 Cotas Inferiores

[Martello and Toth, 1990a] proponen las cotas inferiores L_1, L_2 y L_3 para el BPP. Una cota extra fue propuesta por [Labbé et al., 1991] denominada L_{2LLM} la cual tiene una definición parecida a la L_2.

L1

Esta dada por la suma de los volúmenes de todos los objetos dividida entre la capacidad del contenedor más grande.

L2

En esta cota se considera dividir los objetos en dos grupos, los grandes, que son mayores a la mitad de la capacidad del contenedor y los pequeños. Debemos tener un contenedor por cada objeto grande y se busca la compatibilidad entre los objetos grandes y pequeños para minimizar el desperdicio. Si algún objeto pequeño no es compatible se agrega en otro contenedor nuevo.

L2LLM

Esta cota se propuso en [Labbé et al., 1991] donde se generaliza el procedimiento de la cota L_2 con la adaptación del constructivo First Fit de [Coffman et al., 1984]. Además, se consideran objetos mayores a cualquiera que exceda un tercio de la capacidad del contenedor.

L3

Fue propuesta por [Crainic et al., 2007] y es calculada considerando el peor caso de la cota L_2. Esta se obtuvo de la cota calculada en el procedimiento reducido de Martello-Toth (MTRP) después de hacer $n - 1$ repeticiones.

2.3.1.2 Heurísticas

Los algoritmos de aproximación principales fueron descritos por [Scholl et al., 1997]. Son la base para las heurísticas existentes. Tienen dos versiones llamadas On-line y Off-line. La diferencia es que el On-line los objetos a empaquetar no están ordenados de ninguna forma y el Off-line los objetos están ordenados de forma decreciente. Se agrega una D al final del acrónimo para indicar que que los objetos están en orden decreciente.

Next Fit (NF)

El primero de los elementos es empacado en el primer contenedor, cada elemento siguiente será empaquetado en el mismo lugar que el elemento anterior si la capacidad residual del contenedor es suficiente, sino se empieza a guardar los elementos en un nuevo contenedor.

First Fit (FF)

Por cada elemento a empaquetar se asigna al primer contenedor que esté parcialmente lleno y tenga la capacidad residual suficiente para soportar al elemento.

Best Fit (BF)

Durante cada paso, el elemento actual es asignado a un contenedor que cumple con las restricciones de ser el que tienen menor capacidad pero suficiente para almacenar el elemento. Si no se encuentra un contenedor con estas características, se inicia un nuevo contenedor. De esta forma se van complementando elementos que juntos podrían cubrir la capacidad total del contenedor.

Worst Fit (WF)

En oposición al BF ésta heurística selecciona al contenedor con la capacidad residual más grande y suficiente para empacar el elemento, si no existe algún contenedor, se empieza a utilizar uno nuevo.

2.3.1.3 Meta-heurísticas

Recocido simulado y búsqueda Tabú

La heurística *Simulated Annealing clásico* se implementa para el BPP por [Kämpke, 1988], mientras que una variante, llamada *Weight annealing* fue propuesto por [Loh et al., 2008]. En [Scholl et al., 1997] se usa la heurística *Tabu Search* para acelerar su algoritmo exacto conocido como BISON (**bi**n packing **so**lutio**n** procedure). En [Alvim et al., 2004] incluye una *Tabu Search* en una heurística híbrida para BPP.

Heurísticas basadas en poblaciones

El primer algoritmo genético para BPP fue propuesto en [Falkenauer and Delchambre, 1992], donde se muestra que el algoritmo genético clásico no tiene soluciones de calidad para BPP, pero presentan una variante *The grouping genetic algorithm*, capaz de producir un buen comportamiento computacional. Después en [Falkenauer, 1996] mejoró el método a través de la hibridación usando un criterio de dominancia que se propone en [Martello and Toth, 1990b]. También, se propone un conjunto de instancias como punto de referencia que posteriormente adoptarían otros autores para pruebas computacionales de los algoritmos de BPP.

A pesar de que en [Gent, 1998] se demuestra que la mayoría de las instancias son muy fáciles de resolver, se siguieron usando para probar algoritmos genéticos como en [Bhatia and Basu, 2004], [Singh and Gupta, 2007], [Ülker et al., 2008] y [Stawowy, 2008]. Otros algoritmos genéticos son propuestos por [Poli et al., 2007] y [Rohlfshagen and Bullinaria, 2010]. En la actualidad existe un algoritmo genético muy efectivo propuesto por [Quiroz-Castellanos et al., 2015] ya que logra resolver a optimalidad 1602 de 1615 instancias de la literatura. Mejorando los resultados de [Alvim et al., 2004] con 1587 y [Fleszar and Charalambous, 2011] con 1590. [Liang et al., 2002] propuso un algoritmo de programación evolutiva para el CSP y algunas variantes incluidas el BPP. En [Levine and Ducatelle, 2004] se usa un algoritmo de colonia de hormigas, combinado con una búsqueda local para resolver problemas de BPP.

Hibridación extrema y búsqueda de vecindario variable

También se han obtenido nuevas heurísticas modificando las ya existentes en la literatura. Por ejemplo, [Fleszar and Hindi, 2002] se modifica la heurística de [Gupta and Ho, 1999] y se obtiene una *búsqueda de vecindario variable*. En [Gómez-Meneses and Randall, 2009] se propone un algoritmo de *hibridación extrema* con una búsqueda local para BPP.

2.3.1.4 Hiper-heurísticas

En [Burke et al., 2003] se plantea que las hiper-heurísticas funcionan como las meta-heurísticas evolutivas. Las meta-heurísticas evolutivas se encargan de dirigir una población de soluciones para resolver un problema. La hiper-heurísticas dirigen a una población de meta-heurísticas para resolver un problema a lo largo de varias generaciones. En las hiper-heurísticas, la población de meta-heurísticas evoluciona para resolver el problema.

En [Ross et al., 2002] y [Ross et al., 2003] se trata el BPP a través de la combinación de algoritmos genéticos e hiper-heurísticas. Otras combinaciones de algoritmos evolutivos e hiper-heurísticas para BPP son propuestas por [Burke et al., 2006], [López-Camacho et al., 2011] y [Sim et al., 2012]. En [Bai et al., 2012] se prueban instancias de BPP en una hiper-heurística con el *Simulated Annealing* y en [Sim and Hart, 2013] se usa una hiper-heurística con un algoritmo genético generativo, para crear heurísticas.

2.3.2 Métodos Exactos

Los primeros intentos por resolver exactamente el BPP y el CSP fueron desarrollados entre la década de 1950 y 1960 usando relajaciones lineales y programación dinámica como en [Eisemann, 1957], [Gilmore and Gomory, 1961], [Gilmore and Gomory, 1963] y [Gilmore and Gomory, 1965]. A inicios de la década de 1970, la investigación en el área se enfocó en Branch-and-Bound.

2.3.2.1 Algoritmos de enumeración

Branch-and-bound

El primer algoritmo de Branch-and-Bound para BPP se propone en [Eilon and Chris-

tofides, 1971], donde se adaptó el esquema general enumerativo propuesto por [Balas, 1965] para resolver problemas con variables binarias. En este algoritmo se utilizaba el constructivo *Best Fit Decreasing*(BFD) y se producía un árbol de decisión binaria donde genera dos nodos descendentes por asignación de objetos a contenedores. Más tarde se dasarrollaron meta-heurísticas basadas en FF y BF, las cuales tenían un mejor rendimiento que los constructivos. Esto se aprovechó para crear el método MTP (**M**artello and **T**oth **p**rocedure), desarrollado por [Martello and Toth, 1990a] donde la cota inferior es la cota L_1. Después se desarrolló el método MTRP, agregando reducciones al método MTP y se usó la cota inferior L_2. Años después [Scholl et al., 1997] desarrolló un branch-and-bound conocido como BISON, basándose en el MTP. En él se adapta la robustez del MTP combinado con cotas inferiores calculadas por una búsqueda tabú, obteniendo un método exacto mejorado para BPP. [Schwerin and Wäscher, 1998] mejoró el método con el que se calculaba la cota inferior y aplicó uno basado en el método de generación de columnas desarrollado por [Gilmore and Gomory, 1961] para CSP.

Branch-and-price

El algoritmo *Branch-and-price* fue propuesto por [Vance et al., 1994] para el CSP. Este se enfoca en la generación de precios y no en la generación de columnas como el Branch-and-bound. En cada nodo de decisión, el algoritmo considera aquellas opciones para las cuales la variable de decisión es fraccionaria y del par de elementos que derivan de ese nodo se selecciona el de mayor peso total. Los principales algoritmos Branch-and-price para BPP y CSP están basados en el trabajo de [Gilmore and Gomory, 1961] y [Gilmore and Gomory, 1963]. En [Barnhart et al., 1998] se determina que *branch-and-price* puede ser usado para resolver problemas muy grandes de programación entera.

2.3.2.2 Formulaciones Pseudo-polinómicas

Existen tres principales formulaciones basadas en métodos matemáticos. La primera formulación pseudo-polinómica fue desarrollada para el CSP pero igual tiene aplicación y buenos resultados resolviendo problemas BPP como se puede observar en [Martinovic et al., 2018]. El *Dynamic Programming Flow* fue basado en los trabajos de Gilmore y Gomory [Gilmore and Gomory, 1965]. El Arc-Flow está basado en el trabajo J.M. Valério de Carvalho [Valerio de Carvalho, 2002].

One-Cut

Esta formulación está basada en las ideas propuestas por [Rao, 1976] y [Dyckhoff, 1981]. Este modelo para CSP consiste en simular el proceso físico del corte. Primero se dividen en un caso ideal un contenedor con dos piezas (la pieza izquierda y la pieza derecha). La pieza izquierda es del tamaño del objeto que deseamos cortar y la pieza derecha es el residuo y puede ser reutilizado para producir otros objetos. Este proceso se repite en el corte de los residuos o nuevos contenedores, hasta que la demanda ha sido cubierta. En [Dyckhoff, 1981] el One-cut fue un modelo desarrollado para poder resolver el CSP, basándose en ordenar los moldes de los objetos para minimizar pérdidas.

Dynamic Programing Flow

Este modelo se obtiene de asociar los objetos a las variables de decisión del problema clásico de *programación dinámica* (DP). En [Cambazard and O'Sullivan, 2010] se presenta como *DP-flow*. En él se considera la forma de construcción de un grafo dirigido usada por [Trick, 2003] para formar un propagador de la restricción del problema de mochila usando programación dinámica. Cada solución del BPP corresponde a un problema de flujo mínimo, con restricciones adicionales.

Arc-Flow

Es una formulación pseudo-polinómica efectiva para CSP y BPP, se propone en [Valério De Carvalho, 1999], donde usa un algoritmo branch-and-price. En esta formulación cada objeto es representado por una arista, el tamaño del objeto depende

de la distancia que separa el nodo de salida y el nodo de llegada. El grafo resultante representa todas las posibles combinaciones de objetos y la forma en que pueden ser ordenados dentro de un contenedor. En [Valerio de Carvalho, 2002] se propone una generalización de este modelo ya que agrega la capacidad al grafo de almacenar diferentes tamaños de contenedor. [Brandao and Pedroso, 2016] se basa en los trabajos Valério de Carvalho y propuso una compresión más eficiente del grafo, y es conocida como *General Arc-Flow Model* (GAFM), siendo el mejor método para resolver el GBPP en la actualidad. Esta formulación se presentará más a detalle en el siguiente capítulo.

2.3.2.3 Programación con restricciones

El Constraint Programming (CP) un nuevo paradigma de formulación de problemas difíciles ha sido propuesto y está tomando fuerza en el área de las investigación de operaciones. En él se plantean algunas propuestas para resolver el BPP. En [Shaw, 2004] se presenta una nueva restricción dedicada al BPP, y posteriormente implementada en el **CPOptimizer** como la restricción *IloPack*. Esta restricción se basa en reglas de corte y propagación de nodos tomando en cuenta la cota L_2. En [Cambazard and O'Sullivan, 2010] integra una formulación pseudo-polinómica con el enfoque CP de [Shaw, 2004]. [Dupuis et al., 2010] usan la cota inferior L_{2LLM} de [Labbé et al., 1991] y agregan un algoritmo de reducción. Finalmente, en [Schaus et al., 2012] se incluye una regla de filtrado basada en consideraciones de cardinalidad. En este trabajo utilizamos el CP para dar solución al GBPP, utilizando la restricción *Pack* y adaptándola para considerar los objetos no obligatorios. A su vez, se busca minimizar el impacto del uso de *Pack* para converger al objetivo del GBPP.

En el siguiente capítulo se desarrollan todos los modelos de programación lineal y programación con restricciones que se utilizan en la experimentación. Se explica su método de generación, sus parámetros, la estructura de cada modelo y las restricciones de los mismos, para cada una de las variantes del BPP.

Capítulo 3

Métodos de solución para variantes del Bin Packing

En este capítulo se verán los modelos propuestos para las variantes del BPP, la definición de sus variables de decisión, la función objetivo y las restricciones. En la literatura se han propuesto diferentes modelos para el BPP. Se presentan a detalle los modelos más importantes que contribuyeron para llegar a la formulación generalizada del GBPP. Se presenta también el modelo de flujo en redes con grafo compacto de [Brandão, 2017] ya que es capaz de resolver la mayoría de las instancias de la literatura. También se presentan tres modelos de programación con restricciones que proponemos para resolver el GBPP.

3.1 Modelo de flujo en redes para Bin Packing Problem

En [Valério De Carvalho, 1999] se propone el modelo de flujo en redes el cual se basa en una representación en grafo de todas las posibles combinaciones de los objetos en el contenedor. Las aristas de este grafo representan los objetos y los vértices la capacidad del contenedor. La Tabla 3.1 presenta un resumen de los parámetros que

se utilizan en este modelo.

Para poder generar el grafo que se utilizará en el modelo se utilizó el pseudocódigo que puede observarse en el Algoritmo 1.

Algoritmo 1: Construcción del grafo para BPP

Entrada: m: número de tamaños de los objetos; *l*: lista de tamaños únicos de los objetos; L: capacidad del contenedor.

Salida : V: conjunto de vértices; A: conjunto de aristas.

```
1  Función generarGrafoBPP(m, l, L): es
2      V ← {0, ..., L};
3      A ← ∅ ;
4      A ← A ∪{(0, t)|t ∈ l};
5      para cada u ∈ V hacer
6          T ← ∅;
7          mla ← 0 ;                                          // mla: MaxLengthArc
8          para cada (i, j) ∈ A hacer
9              si u = j and j − i > mla entonces mla ← j − i ;
10         para i = 1 a m hacer
11             si l_i <= mla entonces T ← T ∪{u + l_i} ;
12         para cada v ∈ T hacer
13             si v > L entonces salir de para ;
14             en otro caso A ← A ∪{(u, v)} ;
15         A ← A ∪{(u, L)}
16     devuelve (G = (V,A)) ;
```

Parámetro	Descripción
$\mathcal{T}$	Conjunto de contenedores
C	Capacidad del contenedor
$\mathcal{K}$	Conjunto de tipos de objetos
w_k	Tamaño del objeto tipo $k \in \mathcal{K}$
b_k	Demanda de objetos tipo $k \in \mathcal{K}$
$\mathcal{G} = (\mathcal{V}, \mathcal{A})$	Grafo dirigido asociado al BPP

Tabla 3.1: Parámetros del modelo de flujo en redes para Bin Packing

Variables de decisión

$$z = \text{cantidad de contenedores a usar}$$

$$x_{de} = \text{cantidad de objetos de tamaño } e - d$$

Función Objetivo

$$\text{mín } z \tag{3.1}$$

Restricciones

$$\sum_{(d,e)\in A} x_{de} - \sum_{(e,f)\in A} x_{ef} = \begin{cases} -z & \text{si } e = 0 \\ 0 & \text{si } 1 \leq e \leq C-1 \\ z & \text{si } e = C \end{cases} \tag{3.2}$$

$$\sum_{(d,e)\in A: w_k = e-d} x_{de} \geq b_k \quad \forall k \in \mathcal{K} \tag{3.3}$$

$$x_{de} \in \mathbb{Z}^+ \quad \forall (d,e) \in \mathcal{A} \tag{3.4}$$

En este modelo las aristas representan un objeto de tamaño $e-d$. El contenedor pasa a ser un grafo donde las aristas representan todas las posibles configuraciones de los objetos. El objetivo (3.1) es minimizar la variable de flujo z que está definida por las variables x_{ed} de la ecuación de conservación de flujo (3.2). La ecuación (3.3) nos indica que la demanda de los objetos que debe ser cubierta.

Ejemplo

En la Figura 3.1 se ve la equivalencia entre un objeto que cada objeto de tamaño w_j y una arista es representada por el flujo $x_{ed}, (e,d) \in \mathcal{A}$, donde $w_k = e - d$.

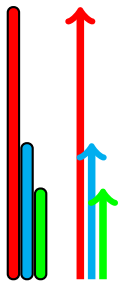

Figura 3.1: Relación entre objetos y aristas

Finalmente, en la Figura 3.2 podemos ver el grafo completo, donde se pueden

ver todas las configuraciones posibles de los objetos dentro de los contenedores de tamaño $c = 7$.

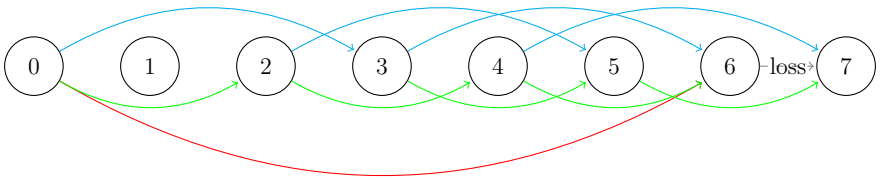

Figura 3.2: Grafo generado para Bin Packing, cuenta con 11 aristas y 8 nodos.

3.2 Modelo de flujo en redes para Variable Size Bin Packing Problem

El VSBPP se introduce en [Friesen and Langston, 1986] donde se desarrolla un modelo para resolverlo con heurísticas basadas en el constructivo FF. Originalmente el objetivo de esta variante era minimizar el desperdicio considerando la capacidad del contenedor y el volumen de los objetos, más tarde esto cambiaría a la minimización del costo por uso de contenedores. En esta sección se habla sobre el modelo de flujo en redes para VSBPP que propuso en [Valerio de Carvalho, 2002] y el cual esta basado en su trabajo [Valério De Carvalho, 1999]. Se considera un conjunto de objetos al igual que en BPP. Existe un límite de contenedores que se pueden utilizar por cada tipo. Las aristas del grafo de este modelo se generan de la misma forma que el modelo de flujo en redes para BPP. En cambio, la cantidad de vértices se define como la capacidad del contenedor más grande disponible. En [Correia et al., 2008] se propone la formulación de un modelo de asignación para el VSBPP. La Tabla 3.2 se presenta un resumen de los parámetros utilizados en el modelo.

Para poder generar el grafo que se utilizará en el modelo es necesario definir el pseudocódigo que puede observarse en la Algoritmo 2.

Algoritmo 2: Construcción del grafo para VSBPP

Entrada: m: número de tamaños de los objetos; l: lista de tamaños únicos de los objetos; C: lista de capacidades únicas de los contenedores.

Salida : V: conjunto de vértices; A: conjunto de aristas.

Función *generarGrafoVSBPP(m, l, C):* **es**
  L $\leftarrow$ máx$\{c_i : i \in C\}$;
  V $\leftarrow \{0, ..., L\}$;
  A $\leftarrow \emptyset$;
  A $\leftarrow$ A $\cup\{(0, t)|t \in l\}$;
  para cada $u \in V$ **hacer**
    T $\leftarrow \emptyset$;
    mla $\leftarrow 0$; `// mla: MaxLengthArc`
    para cada $(i, j) \in A$ **hacer**
      si $u = j$ *and* $j - i >$ *mla* **entonces** mla $\leftarrow j - i$;
    para $i = 1$ **a** m **hacer**
      si $l_i <=$ *mla* **entonces** T $\leftarrow$ T $\cup\{u + l_i\}$;
    para cada $v \in T$ **hacer**
      si $v > L$ **entonces** salir de **para** ;
      en otro caso $A^x \leftarrow A^x \cup\{(u, v)\}$;
    para cada $c \in C$ **hacer**
      A $\leftarrow$ A $\cup\{(u, c)\}$
  devuelve (G = (V,A)) ;

Parámetro	Descripción
$\mathcal{T}$	Conjunto de tipos de contenedores
C_t	Capacidad del contenedor de tipo $t \in \mathcal{T}$
C_{max}	Capacidad del contenedor más grande
U_t	Cantidad máxima de contenedores de tipo $t \in \mathcal{T}$
$\mathcal{K}$	Conjunto de tipos de objeto
w_k	Tamaño de los objetos de tipo $k \in \mathcal{K}$
b_k	Demanda de los objetos de tipo $k \in \mathcal{K}$
$\mathcal{G} = (\mathcal{V}, \mathcal{A})$	Grafo dirigido asociado al VSBPP

Tabla 3.2: Parámetros del modelo de flujo en redes para Variable Size Bin Packing

Variables de decisión

$$z_t = \text{cantidad de contenedores a usar de tipo } t$$

$$x_{de} = \text{cantidad de objetos de tamaño } e - d$$

Función Objetivo

$$\text{mín} \sum_{t \in \mathcal{T}} C_t z_t \tag{3.5}$$

Restricciones

$$-\sum_{(d,e)\in A} x_{de} + \sum_{(e,f)\in A} x_{ef} = \begin{cases} \sum_{t\in\mathcal{T}} \quad z_t & \text{si } e = 0, \\ -z_t & \text{para } e = C_t, \quad t \in \mathcal{T} \\ 0 & \text{si } 1 \le e \le C_{max} - 1, \quad \forall e \neq C_t \end{cases} \tag{3.6}$$

$$\sum_{(d,e)\in A : w_k = e-d} x_{de} \ge b_k, \quad \forall k \in \mathcal{K} \tag{3.7}$$

$$z_t \le U_t, \quad t \in \mathcal{T} \tag{3.8}$$

$$x_{de} \in \mathbb{Z}^+ \quad \forall (d,e) \in A \tag{3.9}$$

$$z_t \in \mathbb{Z}^+ \quad t \in \mathcal{T} \tag{3.10}$$

Este modelo actualiza al modelo propuesto en [Valério De Carvalho, 1999] para BPP y lo generaliza adaptándolo para los problemas con contenedores de diferentes tamaños. En la ecuación (3.6) se observa que la conservación de flujo del grafo es la misma que se propone en [Valério De Carvalho, 1999], la diferencia es que ahora se tienen múltiples vértices terminales. Un vértice se considera terminal si su identificador corresponde a alguna capacidad de contenedor. En la ecuación (3.7) se puede ver que la cantidad de objetos que tengamos en el grafo pueden ser mayores a la demanda de estos por cada tipo. En la ecuación (3.8) se observa que se tiene una cantidad máxima de contenedores que pueden ser utilizados.

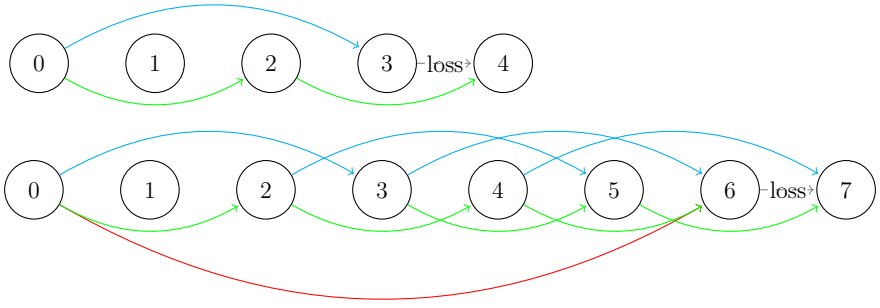

Figura 3.3: Grafos para cada tipo de contenedor

Ejemplo

Ahora tenemos las aristas para los dos tipos de contenedores, pero como podemos observar en la Figura (3.3) las aristas del contenedor pequeño ya están representadas en el contenedor grande, así que podemos descartar el grafo del contenedor pequeño y solo usar el de mayor capacidad. Para saber cuántos contenedores pequeños se necesitarán tendremos que permitir que el vértice terminal en el grafo pequeño también lo sea en el más grande, aceptando así múltiples vértices terminales. El modelo equivalente con un solo grafo está representado en la Figura 3.4 que tiene como valor objetivo $z = 50$, donde $z_1 = 6$ y $z_2 = 2$, también tenemos 8 espacios vacíos, hay 6 para z_1 y 2 para z_2.

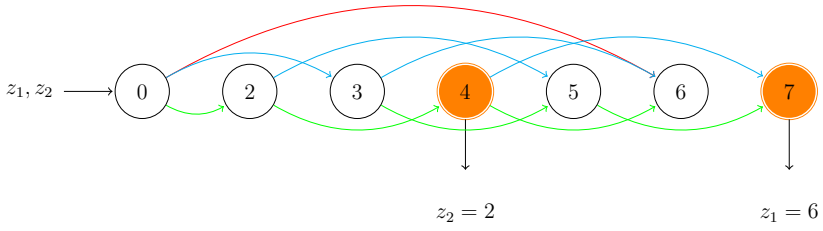

Figura 3.4: Grafo generado para el Variable Size Bin Packing, cuenta con 17 aristas y 7 nodos.

3.3 Modelos de flujo en redes para Generalized Bin Packing Problem

En esta sección se tratan los modelos propuestos para GBPP. El modelo basado en flujo de redes de usando la compresión de grafo General Arc-Flow Model (GAFM) de [Brandão, 2017]. Además, se presenta el modelo de flujo en redes de VSBPP adaptado para resolver GBPP. Se presentan los parámetros de cada modelo y su relevancia.

3.3.1 Modelo de flujo en redes usando General Arc-Flow Model

En [Brandão, 2017] se propone la siguiente formulación matemática para el *Generalized Vector Packing Problem* (GVPP). En este modelo se considera un conjunto de objetos y un conjunto de tipos de objeto. La Tabla 3.3 se presenta un resumen de los parámetros utilizados en el modelo. Este modelo busca encontrar la solución exacta de las instancias usando el modelo general de flujo en redes.

Generación del grafo

[Brandao and Pedroso, 2016] propuso el General Arc-Flow Model (GAFM) que está basado en los trabajos de [Valério De Carvalho, 1999] y [Valerio de Carvalho, 2002]. Este modelo fue desarrollado para resolver el VPP. La principales diferencias que este modelo tiene con respecto a los anteriores son:

- Cada nodo es más general que los anteriores ya que representa múltiples dimensiones.
- Puede haber más de una arista entre dos vértices ya que es un multigrafo.
- Las demandas pueden satisfacerse en exceso, pero en algunas dimensiones es necesario que se satisfagan exactamente.
- Las aristas tienen una cota superior igual a la demanda total del objeto asociado a este.
- Las aristas no están forzadas a corresponder con el peso del objeto asociado.

Para la generación de este grafo se siguen reglas de compresión que se basan en el algoritmo de camino más corto. En este modelo, al trabajar sobre un multigrafo, las aristas están representadas por la tupla (d, e, k), donde d es el vértice del cual sale la arista, e el vértice de llegada y k el tipo de objeto asociado a la arista. En

los modelos anteriores se definen los tipos de objetos en base a sus volúmenes. Para GBPP el tipo de objeto depende del volumen y beneficio del objeto. Es decir, si tenemos dos objetos del mismo volumen pero diferente beneficio, estos pertenecen a dos tipos de objetos diferentes.

Parámetro	Descripción
$\mathcal{T}$	Conjunto de contenedores de tipo t
$\mathcal{K}$	Conjunto de objetos de tipo k
C_t	Capacidad del contenedor de tipo $t \in \mathcal{T}$
c_t	Costo por uso del contenedor de tipo $t \in \mathcal{T}$
U_t	Cantidad máxima de contenedores de tipo t
b_k	Demanda de objetos obligatorios de tipo $k \in \mathcal{K}$
β_k	Demanda de objetos no obligatorios de tipo $k \in \mathcal{K}$
p_k	Beneficio de empaquetar el objeto de tipo k
w_k	Volumen del objeto tipo $k \in \mathcal{K}$
$\mathcal{G} = (\mathcal{V}, \mathcal{A})$	Multigrafo dirigido asociado al GBPP
S	vértice origen ($S \in \mathcal{V}$)
T_t	vértice terminal asociado a los contenedores de tipo t ($T_t \in \mathcal{V}$)

Tabla 3.3: Parámetros del modelo de flujo en redes para Generalized Bin Packing

Variables de decisión

$$x^k_{de} = \text{cantidad de objetos de tipo } k \text{ que están entre } d \text{ y } e$$

$$\delta_k = \text{cantidad de objetos no obligatorios de tipo } k$$

Función Objetivo

$$\text{mín} \sum_{t \in \mathcal{T}} c_t x^0_{T_t S} - \sum_{k \in \mathcal{K}} p_k \delta_k \tag{3.11}$$

Restricciones

$$\sum_{(d,e,k) \in A} x^k_{de} - \sum_{(e,f,k) \in A} x^k_{ef} = 0, \quad v \in \mathcal{V} \tag{3.12}$$

$$\sum_{(d,e,k) \in A} x^k_{de} = b_k + \delta_k, \quad k \in \mathcal{K} \tag{3.13}$$

$$x^0_{T_t S} \leq U_t, \quad t \in \mathcal{T} \tag{3.14}$$

$$0 \leq \delta_k \leq \beta_k, \quad k \in \mathcal{K} \tag{3.15}$$

$$x_{de}^{k} \in \mathbb{Z}^{+}, \quad (d,e,k) \in A \tag{3.16}$$

$$\delta_k \in \mathbb{Z}^{+}, \quad k \in \mathcal{K} \tag{3.17}$$

La función objetivo (3.11) representa que minimizamos el costo por uso de los contenedores y maximizamos el beneficio por empaquetar objetos no obligatorios. En la restricción (3.12) podemos ver la versión simplificada de la ecuación de conservación de flujo del *General Arc-Flow Model* que introduce en [Brandão, 2017], donde cada variable de flujo x tiene dos sub-índices y un super-índice; Los sub-índices representan el vértice de origen y de destino de la arista x; El super-índice indica el tipo de objeto k que representa la arista, y donde las variables con $k = 0$ representan las variables donde se recolecta el flujo dependiendo del T_t del que provengan. La restricción (3.13) es una simplificación de las restricciones que representa la demanda total de los objetos, donde b_k representa la demanda de los objetos obligatorios y β_k representa la demanda de los objetos no obligatorios, además la desigualdad está regulada con una variable de holgura β_k que a su vez nos sirve como variable para maximizar el beneficio de empaquetar los objetos k.

Ejemplo de compresión de GAFM

Para ilustrar el algoritmo de compresión del GAFM, en este ejemplo se muestran los grafos de las cuatro etapas de compresión. En la Figura 3.5 se muestra el grafo más grande del primer paso, el cual se genera considerando la demanda de los objetos. Cuenta con un nodo inicial **S** y tantos nodos terminales **T** como tipos de contenedores se tengan.

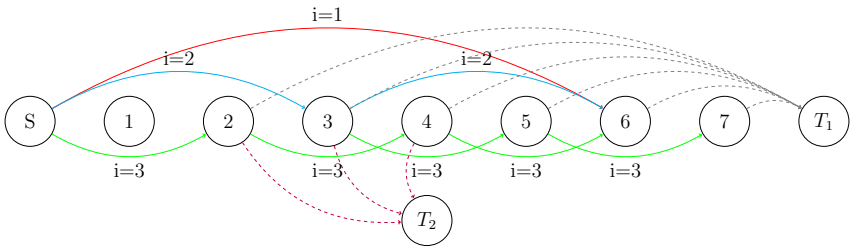

Figura 3.5: Paso 1 de la compresión del General Arc-Flow Model cuenta con 17 aristas y 10 nodos.

Como la generación del grafo depende de la demanda de cada tipo de objeto, hay casos en los que la demanda es menor al mínimo de objetos para completar un contenedor, este tipo de llenado se considera no factible. Para eliminar caminos no factibles y repetidos, el segundo paso del GAFM genera el grafo de la Figura 3.6.

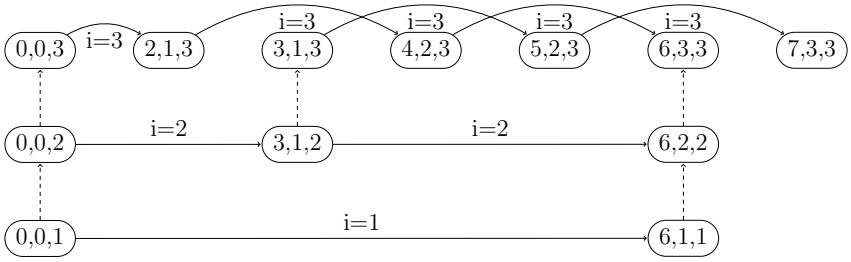

Figura 3.6: Paso 2 de la compresión del General Arc-Flow Model.

En el grafo de la Figura 3.6 se insertan las aristas del mismo tipo por niveles. Cada nivel representa un tamaño de objeto diferente, yendo de forma decreciente.

Para el tercer paso se utiliza el grafo del paso anterior. En la Figura 3.7 observamos que se eliminaron los nodos que no son necesarios y los caminos no factibles. En este ejemplo no hay ninguna arista eliminada y se elimina el nodo 1. El grafo del tercer paso se utiliza en el cuarto y último paso, donde se busca combinar los nodos que tengan aristas de entrada del mismo nodo de origen y también aristas de salida que vayan a vértice destino.

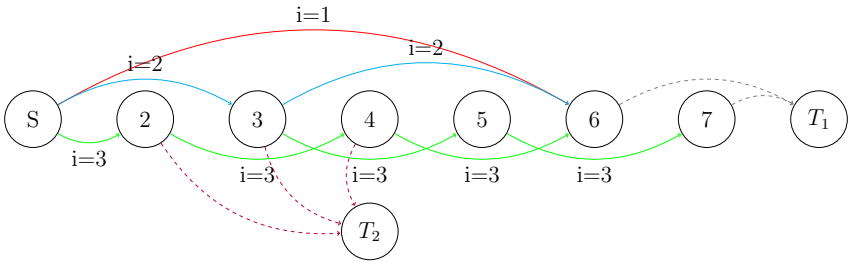

Figura 3.7: Paso 3 de la compresión del General Arc-Flow Model, cuenta con 13 aristas y 9 vértices.

El método de compresión consiste en aplicar el algoritmo de camino más corto

al grafo de la Figura 3.7 y se fusionan los vértices con la misma distancia desde el vértice origen. El resultado de esta compresión se observa en la Figura 3.8. Se observa que los vértices 2 y 3 se combinaron y también los vértices 6 y 7. En este ejemplo el grafo del paso uno tiene 17 aristas y 10 vértices y después del proceso de compresión de GAFM pasó a tener 11 aristas y 7 vértices.

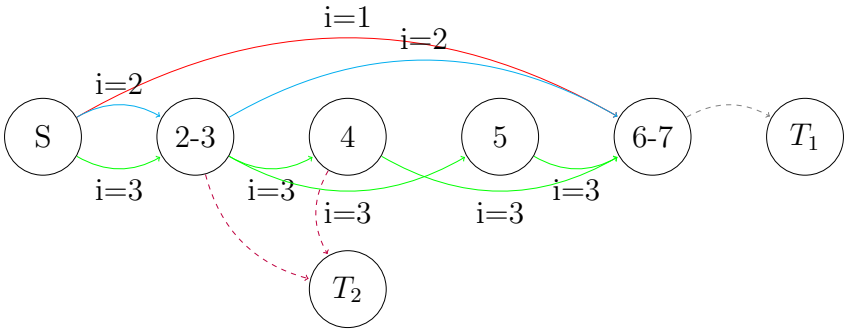

Figura 3.8: Paso 4 de la compresión del General Arc-Flow Model, cuenta con 11 aristas y 7 vértices.

Ejemplo

Tomando en cuenta el grafo del paso final de la compresión del GAFM, podemos compararlo con el grafo de la figura (3.3). Se puede notar que el grafo de la Figura 3.8 tiene menos vértices y menos aristas, lo cual al momento de procesar el modelo significan menos variables. Ahorrando el espacio en memoria y reduciendo la cantidad de variables y restricciones del modelo matemático. Al resolver el problema con este modelo de flujo en redes se obtiene como valor objetivo $z = 48$, donde $T_1 = 6$ y $T_2 = 0$ y se descarta un objeto de volumen 3 y beneficio 1.

3.3.2 Modelo de flujo en redes sin compresión

Se desarrolla un modelo matemático basado en la generación del grafo del VSBPP, se utiliza el modelo de la Sección 3.2. En este modelo no se considera la demanda para la generación del grafo y no se consideran vértices terminales externos.

Los parámetros de este modelo se muestran en la Tabla 3.4.

Parámetro	Descripción
$\mathcal{T}$	Conjunto de contenedores de tipo t
$\mathcal{K}$	Conjunto de objetos
$\mathcal{O}_k$	Subconjunto de objetos no obligatorios de tipo k
C_t	Capacidad del contenedor tipo t
c_t	Costo por uso del contenedor de tipo t
C_{max}	Capacidad del contenedor más grande
U_t	Cantidad máxima de contenedores de tamaño t
w_k	Volumen del objeto tipo k
p_k	Beneficio del objeto tipo k
b_k	Demanda de los objetos obligatorios de tipo k
$\mathcal{A}^x$	Conjunto de aristas que representan objetos
$\mathcal{A}^y$	Conjunto de aristas que representan espacio vacío
$\mathcal{G} = (\mathcal{V}, \mathcal{A}^x \cup \mathcal{A}^y)$	Grafo dirigido asociado al GBPP

Tabla 3.4: Parámetros del modelo de flujo en redes desarrollado para Generalized Bin Packing

Variables de decisión

$$z_t = \text{cantidad de contenedores a usar de tipo } t$$

$$x_{de} = \text{cantidad aristas de objetos de tamaño } e - d$$

$$y_{de} = \text{cantidad aristas de espacios vacíos de tamaño } e - d$$

$$e_i = \begin{cases} 1 & \text{si el objeto no obligatorio } i \in \mathcal{O}_k \text{ es empaquetado} \\ 0 & \text{en cualquier otro caso} \end{cases}$$

Función objetivo

$$\text{mín} \sum_{t \in \mathcal{T}} c_t z_t - \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{O}_k} p_k e_i \tag{3.18}$$

Restricciones

$$\sum_{(d,e)\in A^x} x_{de} + \sum_{(d,e)\in A^y} y_{de} - \sum_{(e,f)\in A^x} x_{ef} - \sum_{(e,f)\in A^y} y_{ef} = \begin{cases} z_t \quad si \quad e = C_t \\ -\sum_{t\in\mathcal{T}} z_t \quad e = 0 \\ 0 \quad para \quad e \neq 0 \end{cases} \tag{3.19}$$

$$\sum_{(d,e)\in A^x} x_{de} = b_k + \sum_{i\in\mathcal{O}_k} e_i \quad \forall k \in \mathcal{K} \tag{3.20}$$

$$z_t \leq U_t, \quad \forall t \in \mathcal{T} \tag{3.21}$$

$$\sum_{(v,C_t)\in A^y} y_{v,C_t} \leq z_t, \quad \forall t \in \mathcal{T} \tag{3.22}$$

$$e_i \in \{0,1\} \quad \forall k \in \mathcal{K}, \quad \forall i \in \mathcal{O}_k \tag{3.23}$$

$$x_{de} \in \mathbb{Z}^+ \quad \forall (d,e) \in A^x \tag{3.24}$$

$$y_{de} \in \mathbb{Z}^+ \quad \forall (d,e) \in A^y \tag{3.25}$$

El objetivo de este modelo (3.18) está en términos de utilidad como en el VSBPP. La utilidad está dada por la diferencia entre el costo de utilización de los contenedores y el beneficio de empaquetar los objetos no obligatorios. En la restricción (3.19) se puede observar la estructura de la conservación de flujo. La demanda está definida como la suma de la demanda de objetos obligatorios y la suma de la demanda de los objetos no obligatorios como se ve en la restricción (3.20), una diferencia es que en el modelo con GAFM la demanda de los objetos obligatorios es entera y en este modelo se toma una decisión por cada objeto, siendo una variable binaria. La restricción (3.21) nos ayuda a que la solución no debe exceder la cantidad de contenedores disponibles por cada tipo de contenedor. La restricción (3.22) indica que la cantidad de aristas vacías que lleguen a un vértice terminal pueden ser menores o iguales a la cantidad de aristas que llegan al vértice terminal, lo que nos ayuda a mantener la conservación del flujo de las aristas de espacios vacíos.

Ejemplo

El modelo equivale a un solo grafo como el de la Figura 3.9, las aristas con línea continua representan las variables x_{de} que son los objetos dentro del contenedor; Las aristas de línea punteada representan las variables y_{de} que son los espacios vacíos al final de contenedor. Para la solución del GBPP con este grafo se tiene como valor objetivo $z = 48$, donde $z_1 = 6$ y $z_2 = 0$, también tenemos 3 espacios vacíos, hay 6 para z_1 y 0 para z_2. Se descarta un objeto de volumen 3 y beneficio 1 ya que cuesta más empacarlo que la ganancia que proporciona.

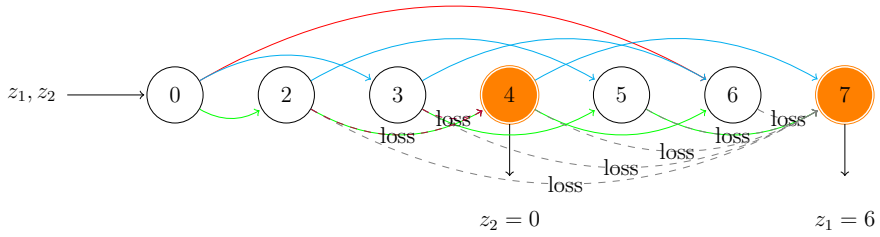

Figura 3.9: Grafo generado para el Variable Size Bin Packing, cuenta con 17 aristas y 7 vértices.

3.4 Modelos basados en Constraint Programming

Dado que en la literatura no existen modelos de *CP* para abordar el problema de GBPP proponemos tres modelos. Dos de ellos utilizan la restricción *Pack* dedicada para BPP. *Pack* fue propuesta por [Shaw, 2004] y fue implementada en la librería *CPOptimizer de CPLEX*. En esta restricción se empaquetan todos los objetos y usa la cota inferior L_2 para calcular la cantidad mínima de contenedores a utilizar. Esta restricción empaqueta todos los objetos que se le indiquen, ya que es una propiedad del BPP hacerlo. Al resolver para GBPP se tienen objetos no obligatorios, los cuales pueden no ser empaquetados. Es necesario tener la posibilidad de descartar objetos, por este motivo en los modelos CP_1 y CP_2 se crearon contenedores ficticios.

Para los modelos de CP se considera que las variables que representan un contenedor tienen un dominio que es igual la capacidad máxima del contenedor más grande, que luego es limitada por la cantidad disponible de contenedores. Las variables que representan los contenedores ficticios tienen un dominio igual al volumen del objeto asociado. Las variables que representan objetos tienen un dominio que es igual a la ubicación donde se empaqueta dentro del contenedor. El objetivo de estos modelos es minimizar el uso de los contenedores y maximizar el beneficio por empaquetar los objetos no obligatorios, como el objetivo definido en GBPP. En los modelos CP los contenedores y los objetos no están agrupados en tipos, como en modelos pasados. En la Tabla 3.5 se presenta un resumen de los parámetros utilizados en los modelos.

Parámetro	Descripción
$\mathcal{T}$	Conjunto de contenedores
$\mathcal{T}^*$	Conjunto de contenedores ficticios
$\mathcal{K}$	Conjunto de objetos
$\mathcal{R}$	Subconjunto de objetos obligatorios
$\mathcal{O}$	Subconjunto de objetos no obligatorios
C_t	Capacidad del contenedor t
c_t	Costo de uso del contenedor t
p_k	Beneficio del objeto k
w_k	Volumen del objeto k
m	Cardinalidad del conjunto $\mathcal{T}$

Tabla 3.5: Parámetros de los modelos de programación con restricciones

Variables de decisión

$$z_t = \begin{cases} 1 & \text{si el contenedor } t \text{ es usado} \\ 0 & \text{en cualquier otro caso} \end{cases}$$

$where_k =$ indica el número de contenedor donde se empaqueta el objeto k

Función Objetivo

$$\text{mín} \sum_{t \in \mathcal{T}} c_t z_t - \sum_{k \in \mathcal{O}} p_k (where_k \leq m) \tag{3.26}$$

Restricciones

$$\sum_{k \in \mathcal{K}} w_k (where_k == t) \leq C_t z_t \quad \forall t \in \mathcal{T} \cup \mathcal{T}^* \tag{3.27}$$

$$where_k \in \mathcal{T} \cup \mathcal{T}^*, \quad \forall k \in \mathcal{O} \tag{3.28}$$

$$where_k \in \mathcal{T}, \quad \forall k \in \mathcal{R} \tag{3.29}$$

$$z_t \in \{0, 1\} \quad \forall t \in \mathcal{T} \cup \mathcal{T}^* \tag{3.30}$$

La restricción (3.27) representa la restricción *Pack*. La restricción (3.28) indica el dominio de las variables que representan los objetos no obligatorios y la restricción (3.29) indica el dominio de las variables que representan los objetos obligatorios. En CP_1 y CP_3 la cardinalidad del conjunto de contenedores ficticios es 1. En CP_2 la cardinalidad del conjunto de contenedores ficticios es igual a la cardinalidad del conjunto de objetos no obligatorios.

En el modelo CP_1, se considera la creación de un contenedor ficticio donde se almacenarán los objetos que serán descartados de la solución. El tamaño del contenedor ficticio es la suma de todos los volúmenes de los objetos no obligatorios.

Se busca comprobar si la capacidad del contenedor ficticio generado en CP_1 afecta en el cálculo de la cota L_2 que se realiza dentro de la restricción Pack. Para comprobar ésto se planteó el modelo CP_2 en cual se crea un contenedor ficticio para cada objeto no obligatorio de capacidad igual a cada objeto opcional relacionado.

Finalmente, se busca saber si la restricción Pack es realmente necesaria para resolver problemas de GBPP. Para poder comprobar esto se diseñó el modelo CP_3 basándose en modelo de asignación de CP_1. Sin embargo, en este modelo no se utiliza el paquete Pack para las restricciones (3.27) y se consideran solo las restricciones de capacidad para los contenedores no ficticios.

En el capítulo siguiente se prueban los modelos de flujo en redes de GBPP, el modelo basado en AFM se identificará como $MILP_1$ y el modelo basado en GAFM se identificará como $MILP_2$ y se comparan entre sí. También se compara su desempeño con el de los modelos de programación con restricciones, los cuales ya se han sido definidos e identificados como CP_1, CP_2 y CP_3.

Capítulo 4

Resultados Computacionales

En esta sección se presentan los resultados computacionales de los modelos que se introdujeron en el capítulo anterior. Primero se explica el entorno de pruebas en la sección 4.1, después compararemos los tres modelos CP entre sí, para evaluar si es necesaria la restricción *IloPack*. También compararemos el modelo $MILP_1$ y el $MILP_2$ contra el mejor modelo CP.

4.1 Entorno de pruebas

El experimento fue realizado en una Workstation HP Z230, que cuenta con un procesador Intel Xeon CPU E3-1245 v3 @ 3.40GHz x 8 y una RAM de 15,4 GiB, la versión usada de CPLEX fue la 12.9.0 que incluye el CPOptimizer. En CPLEX se limitó el procesamiento a un thread, y en CPOptimizer se limitó a un worker. El tiempo límite para la ejecución de ambos solucionadores fue de 300 segundos.

Entorno de pruebas de Brandão, 2017

Se utilizó una computadora con un procesador Intel Xeon 2.66GHz Quad-Core. Sistema operativo Mac OS X 10.11.6, con 16GB de RAM. El algoritmo para generar el grafo se desarrollo en *C++*, los modelos se construyeron en *Python 2.7* y se resolvió en Gurobi 7.0.2.

4.2 Instancias de prueba

El conjunto de instancias disponible para GBPP fue generada en [Baldi et al., 2012], y están agrupadas en tres clases:

Class 0: son 300 instancias que originalmente fueron creadas para el VCSBPP, existen 10 instancias para cada combinación posible de los 4 parámetros siguientes:

- Número de objetos: 25, 50, 100, 200 y 500.
- Volumen del objeto: I1:[1,100]; I2:[20,100]; I3:[50,100]
- Beneficio del objeto: 200 para todos los objetos, todos obligatorios
- Tipo de contenedor:
  - 3 tipos: 100, 120 y 150
  - 5 tipos: 60, 80, 100, 120, 150

El costo por uso es igual a la capacidad del contenedor. Para cada tipo de contenedor existe una cota inferior y una cota superior, las cotas inferiores son 0 y las cotas superiores están dadas $\lceil V_{TOT}/V_t \rceil$ donde V_{TOT} es el volumen total de los objetos, y V_t es la capacidad del contenedor tipo t.

Class 1: son las mismas instancias que la Class 0, pero todos los objetos no son obligatorios, y los beneficios de empaquetar los objetos esta dada una distribución uniforme $p_i \in \lceil \mathbb{U}(0.5, 3)w_i \rceil$

Class 2: son las mismas instancias que la Class 0, pero todos los objetos son No obligatorios, y los beneficios de empaquetar los objetos esta dada una distribución uniforme $p_i \in \lceil \mathbb{U}(0.5, 4)w_i \rceil$

Class 3: son una selección de 12 instancias grandes (500 objetos) de la clase 1 y clase 2, con una mezcla representativa de características en términos de volumen de objetos, beneficio de objetos, ganancias de artículos y tipos de contenedores. Para

cada instancia se obtuvieron aleatoriamente cinco instancias con 0 %, 25 %, 50 %, 75 % y 100 % de artículos obligatorios, para un total de 60 instancias.

4.3 Modelos de programación lineal de enteros mixtos

Los dos modelos MILP se comparan en la Tabla 4.1 donde se pueden ver el número de instancias en las cuales se logró encontrar el valor óptimo y el tiempo de ejecución de los mismos, además de la cantidad de nodos que recorrieron los modelos y la cantidad de variables y restricciones de cada modelo. La información del modelo $MILP_2$ se reporta en [Brandão, 2017].

Tabla 4.1: Comparativo entre los modelos $MILP_1$ y $MILP_2$.

Clase	Bin	Objetos	Tiempo		Nodos		Restricciones		Variables		Óptimas	
			MILP1	MILP2	MILP1	MILP2	MILP1	MILP2	MILP1	MILP2	MILP1	MILP2
0	3	25	0.15	0.05	63.63	4.67	122	61.67	750.33	257.37	30	30
		50	0.17	0.29	16.83	19.37	152.53	104.47	1326.33	728	30	30
		100	1.33	0.93	250.33	41.07	181.4	143.23	2268.23	1550.57	30	30
		200	0.72	2.09	86.16	109.73	198.93	166.57	3200.23	2425.17	30	30
		500	0.75	4.32	77.86	189.03	206.26	176.8	3912.7	2916.4	30	30
	5	25	0.09	0.03	1.96	0	126.46	71.13	866.06	310.33	30	30
		50	0.13	0.13	8.5	4.73	155.5	109.87	1433.36	792.73	30	30
		100	0.49	0.24	69.93	5.10	180.8	143.8	2250.43	1548.27	30	30
		200	0.99	2.08	103.03	84.3	200.76	169	3272.26	2481.07	30	30
		500	1.16	2.31	173.06	95.03	208.06	179.2	4000.3	2971.17	30	30
			0.59	1.24	85.13	55.30	173.27	132.57	2328.02	1598.10	300	300
1	3	25	0.11	0.05	56.46	6.53	122	61.67	750.33	282.37	30	30
		50	2.29	0.72	1020.3	201.63	152.53	104.47	1326.33	778	30	30
		100	48.12	3.87	18947.2	241.5	181.4	143.23	2268.23	1650.57	27	30
		200	58.62	8.12	28696.73	358.3	198.93	166.57	3200.23	2625.17	25	30
		500	0.74	20.33	73.93	1755.6	206.26	176.8	3912.7	3416.4	30	29
	5	25	3.45	0.04	2258.13	0	126.46	71.13	866.06	335.33	30	30
		50	18.84	1.49	4954.8	352.87	155.5	109.87	1433.36	842.73	29	30
		100	72.19	3.06	29425.73	431.43	180.8	143.8	2250.43	1648.27	24	30
		200	36.34	4.08	16193.5	170.43	200.76	169	3272.26	2681.07	27	30
		500	1.31	20.26	165.13	1220.87	208.06	179.2	4000.3	3471.17	30	29
			24.20	6.20	10179.19	473.91	173.27	132.57	2328.02	1773.10	282	298
2	3	25	0.81	0.06	566.96	15.07	122	61.67	750.33	282.37	30	30
		50	12.13	0.75	5013.3	87.7	152.53	104.47	1326.33	778	29	30
		100	11.83	10.30	2399.03	1184	181.4	143.23	2268.23	1650.57	30	30
		200	39.65	12.09	17362.13	902.13	198.93	166.57	3200.23	2625.17	28	30
		500	1.40	42.93	83.56	4704.57	206.26	176.8	3912.7	3416.4	30	28
	5	25	0.5	0.05	244.03	17.03	126.46	71.13	866.06	335.33	30	30
		50	30.43	0.61	13446.5	38.53	155.5	109.87	1433.36	842.73	27	30
		100	33.29	4.93	8962.43	618.43	180.8	143.8	2250.43	1648.27	27	30
		200	66.63	15.48	26465.8	1210.10	200.76	169	3272.26	2681.07	24	29
		500	1.45	9.02	148.13	704.53	208.06	179.2	4000.3	3471.17	30	30
			19.81	9.62	7469.19	948.2	173.27	132.57	2328.02	1773.10	285	297
		En General	14.86	5.68	5911.17	492.47	173.27	132.57	2328.02	1714.76	867	897

En la Tabla 4.1, la columna Tiempo indica el tiempo de resolución; La columna Nodos indica la cantidad de nodos que exploró el branch-and-bound de CPlex en $MILP_1$ y Gurobi en $MILP_2$; La columna Restricciones indica la cantidad de restricciones promedio en cada modelo; La columna Variables indica la cantidad de variables promedio en cada modelo; La columna Óptima indica la cantidad de instancias en las que se obtuvieron resultados óptimos.

Tabla 4.2: Comparativo entre los modelos $MILP_1$ y $MILP_2$ en instancias clase 3.

% Obligatorio	Tiempo		Nodos		Restricciones		Variables		Óptimas	
	MILP1	MILP2	MILP1	MILP2	MILP1	MILP2	MILP1	MILP2	MILP1	MILP2
0 %	1.20	33.43	138.16	2304.67	215.25	190.67	4385.66	3897.67	12	11
25 %	2.78	5.03	858	275.58	215.25	190.67	4260.66	3772.67	12	12
50 %	126.60	39.84	44434.91	2565	215.25	190.67	4135.66	3647.67	7	11
75 %	92.68	44.56	29736.08	2972.25	215.25	190.67	4010.66	3522.67	9	11
100 %	1.88	4.40	223.25	149.58	215.25	190.67	3885.66	3397.67	12	12
En General	45.08	25.45	15078.08	1653.41	215.25	190.67	4135.66	3647.67	52	57

La Tabla 4.2 cuenta con las mismas columnas que la Tabla 4.1 y en ella se muestra la comparación de los modelos sobre las instancias de clase 3. Podemos observar que en las instancias donde el porcentaje de objetos obligatorios es 0 %, 25 % y 100 % los tiempos de procesamiento del $MILP1$ son menores que los de $MILP_2$, aunque en general el $MILP_2$ tarda menos en promedio. Otro punto a destacar es que en promedio el modelo $MILP_1$ tiene menos variables que el $MILP_2$.

A pesar de que los modelos no fueron ejecutados en el mismo equipo, ambos modelos logran encontrar una gran cantidad de valores óptimos para las instancias de la literatura, dentro de un tiempo razonable. Sin embargo, se nota que el método de compresión del $MILP_2$ permite una reducción importante de la cantidad de variables y restricciones lo que resulta en un tiempo de procesamiento menor.

Se puede observar que el modelo $MILP_1$ se comportó casi tan eficiente como el mejor modelo de la literatura propuesto por [Brandão, 2017]. Por este motivo, en lo que sigue, validamos los valores óptimos obtenidos en cada modelo de CP con el valor de la función objetivo de este modelo.

Algo interesante de remarcar es que el modelo $MILP_1$ encuentra soluciones óptimas en las instancias de 500 objetos mucho antes del $MILP_2$, se tiene que revisar la estructura de los modelos, para confirmar esta situación peculiar. Una posible explicación es que el $MILP_2$ en su algoritmo de compresión descarta caminos en los cuales no hay suficiente demanda de objetos pequeños, por lo cual estos objetos se

les busca compatibilidad con otros más grandes. En $MILP_1$ se utilizan las aristas de espacio vacío para hacer factibles estos caminos y conservar el flujo con estas aristas de peso cero. Otra posible explicación sería que al ser una cantidad grande de objetos, estos objetos pequeños en grupo son suficientes para llenar un contenedor por ellos mismos, sin necesidad de buscar la compatibilidad y sin tener que pertenecer a un camino que pudiera ser descartado por los algoritmos de compresión de GAFM.

4.4 Modelos de programación con restricciones

Tabla 4.3: Resultados de los modelos de programación con restricciones

Clase	Bin	Objetos	CP1		CP2		CP3	
			OPT	GAP	OPT	GAP	OPT	GAP
0	3	25	26	0.04 %	28	0.02 %	25	0.05 %
		50	14	0.12 %	14	0.09 %	10	0.2 %
		100	3	0.38 %	2	0.33 %	3	0.46 %
		200	0	1.31 %	0	0.7 %	0	0.8 %
		500	0	5.98 %	0	1.58 %	0	1.59 %
	5	25	25	0.05 %	27	0.03 %	25	0.05 %
		50	11	0.19 %	13	0.13 %	10	0.22 %
		100	2	50.51 %	1	0.37 %	0	0.48 %
		200	0	1.63 %	0	0.71 %	0	0.78 %
		500	0	5.24 %	0	1.71 %	0	1.49 %
			81	6.54 %	85	0.56 %	73	0.61 %
1	3	25	15	0.16 %	22	0.08 %	23	0.06 %
		50	0	1.41 %	0	0.8 %	1	0.67 %
		100	0	4.1 %	0	2.83 %	0	1.61 %
		200	0	9.55 %	0	5.99 %	0	4.7 %
		500	0	21.23 %	0	9.26 %	0	9.05 %
	5	25	20	0.23 %	22	0.14 %	21	0.2 %
		50	0	1.67 %	0	1.02 %	1	1.06 %
		100	0	4.22 %	0	2.41 %	0	2.55 %
		200	0	9.56 %	0	5.96 %	0	6.63 %
		500	0	19.28 %	0	9.68 %	0	9.04 %
			35	7.14 %	44	3.81 %	46	3.55 %
2	3	25	25	0.08 %	26	0.04 %	23	0.09 %
		50	2	0.86 %	4	0.51 %	6	0.4 %
		100	0	1.89 %	0	1.39 %	0	1.03 %
		200	0	5.19 %	0	2.94 %	0	2.53 %
		500	0	12.24 %	0	4.73 %	0	4.49 %
	5	25	20	0.12 %	24	0.07 %	23	0.08 %
		50	3	0.64 %	4	0.52 %	3	0.52 %
		100	0	2.63 %	0	1.46 %	0	1.48 %
		200	0	5.35 %	0	3.13 %	0	3.65 %
		500	0	11.19 %	0	5.12 %	0	4.88 %
			50	4.02 %	58	1.99 %	55	1.91 %
	En General		166	5.80 %	187	2.12 %	174	2.02 %

Los tres modelos de CP se comparan en la Tabla 4.3, la columna OPT representa la cantidad de instancias que llegaron a optimalidad; la columna GAP representa el gap con respecto a los valores óptimos encontrados por el $MILP_1$. Se observa

que el modelo CP_2 converge al óptimo en más instancias, lo que nos indica que el tamaño del contenedor ficticio de CP_1 si afecta en el proceso de búsqueda de la solución óptima. Además, si se compara el mejor modelo con la restricción Pack, que es CP_2 y el modelo sin Pack CP_3. Se aprecia que CP_2 logra que más instancias converjan al óptimo del $MILP_1$ pero, CP_3 tiene un gap menor al valor óptimo de $MILP_1$ que CP_2.

En éste caso podemos observar en la Tabla 4.3 que a pesar de que CP_3 no logró que sus instancias llegaran al valor óptimo conocido logró entregar soluciones factibles que tiene en promedio un gap menor que CP_2. Esto nos indica que CP_3 tiene una mejor convergencia al valor óptimo. Entonces no se puede determinar si la restricción *Pack* es necesaria. También, se observa que en las instancias de clase 1 es donde CP_3 tiene un mejor desempeño que los otros modelos. Es necesario recordar que *CP* es un método exacto y en el tiempo límite que se le asignó a los modelos se obtuvieron soluciones factibles.

En la figura 4.1 se observa el número de instancias donde se encontró el valor óptimo, tal como se muestra en los datos de la Tabla 4.3. En los tres modelos no encontró el valor óptimo de la función objetivo en las categorías que tienen 200 y 500 objetos. En la Figura 4.2 se observa que el modelo CP_1 es el que presenta el mayor gap y los modelos CP_2 y CP_3 casi se encuentran a la par en la mayoría de las categorías. También se observa que en las instancias de la clase 1, es donde el CP_1 tiene mayor gap y donde CP_3 tiene mejores resultados que CP_2.

Los modelos de CP no son capaces de confirmar lo óptimo de las soluciones encontradas en el tiempo limite de 5 minutos a diferencia del $MILP_1$ y $MILP_2$ que en la mayoría de las instancias lograron esto en menos de 5 minutos.

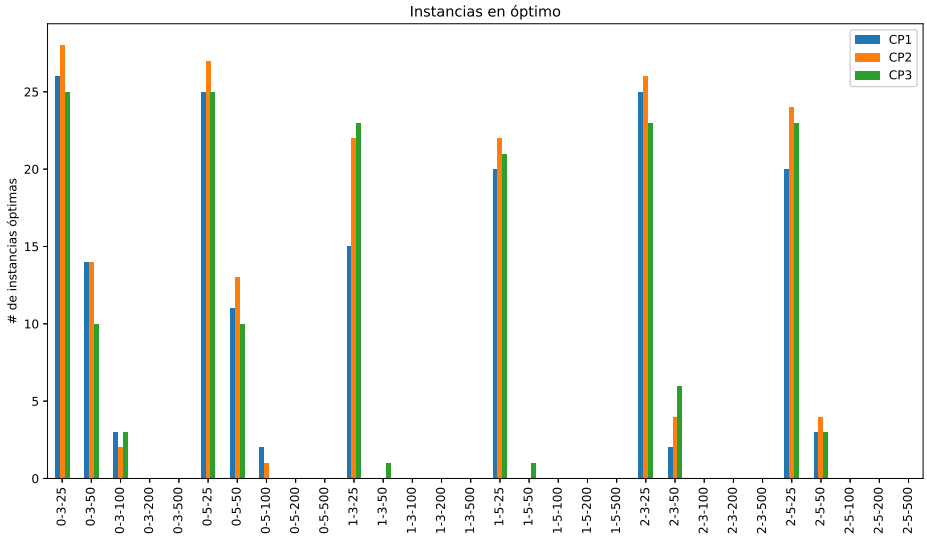

Figura 4.1: Gráfica de instancias que lograron el valor óptimo por categoria de instancia

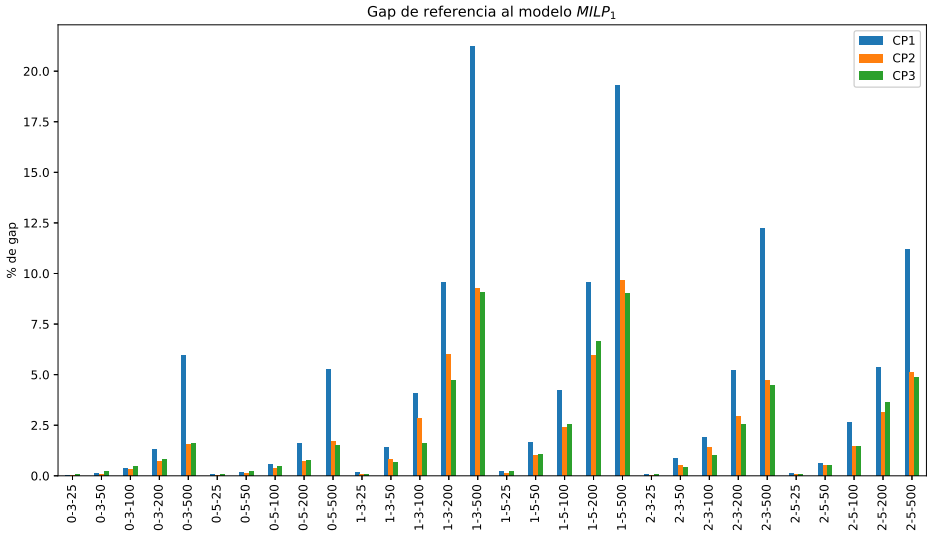

Figura 4.2: Gráfica de comportamiento de gap por categoria de instancia

Capítulo 5

Conclusiones

Sabiendo que la formulación pseudo-polinómica de AFM es una de las mejores resolviendo las variantes del BPP de una dimensión y el VSBPP. Se desarrolló un método basado en modelo de flujo en redes para VSBPP y comprobamos que tiene buenos resultados también para la variante GBPP presentando resultados casi tan buenos como el mejor modelo conocido basado en flujo en redes, el GAFM, propuesto por [Brandão, 2017].

Se proponen tres modelos de CP para abordar el GBPP. Dos de ellos se basan en la restricción IloPack de CPOptimizer del ILOG Cplex 12.9 e implementamos nuevas restricciones para almacenar los objetos descartados en un contenedor ficticio lo suficientemente grande para almacenar todos los objetos no obligatorios en CP_1 y generando un conjunto de contenedores ficticios de tamaño igual a cada uno de los objetos no obligatorios donde se almacenaran los objetos descartados en CP_2. Obtuvimos que el modelo CP_2 tiene mejor convergencia al óptimo, debido a que generamos contenedores ficticios del tamaño de los objetos opcionales y que al momento en que la restricción *IloPack* calcula la cota L_2, el tamaño de estos contenedores ficticios no afectan tanto como el contenedor ficticio de CP_1. Además se propuso el modelo CP_3 para comprobar si es necesaria la restricción *IloPack*, y obtuvimos que a pesar de no converger en más instancias al valor óptimo conocido, si tuvo una mejor gap con respecto al modelo $MILP_1$.

Al comparar los modelos CP contra el modelo $MILP_1$ se encontró que los modelos de Constraint Programming tienen un gap menor a 5 % con respecto a la solución óptima encontrada el modelo $MILP_1$ en las instancias en que logró encontrar el óptimo y en promedio un gap cercano al 2 %. A todos los modelos se les otorgó el mismo límite de tiempo y todos obtuvieron soluciones factibles en menos de cinco minutos.

Para trabajo futuro se buscará aplicar el General Arc-Flow Model de GVPP en GBPPI y verificar si aún se conserva el desempeño del modelo con la nueva condición que aumenta el beneficio en función de la compatibilidad de objetos. Se indagará a profundidad en los métodos de propagación de restricciones específicos para el GBPP. Se buscará entender los métodos de propagación de restricciones planteados en *IloPack* para que pueda soportar los objetos no obligatorios. Se buscará proponer una mejor versión de la restricción *Pack* implementando otra cota diferente a la L_2 y adaptándola para distintos capacidades de contenedor para ayudar a agilizar la convergencia al óptimo. Finalmente se buscará comprobar si los modelos CP mejoran su rendimiento utilizando múltiples procesadores.

Bibliografía

[Alvim et al., 2004] Alvim, A. C., Ribeiro, C. C., Glover, F., and Aloise, D. J. (2004). A hybrid improvement heuristic for the one-dimensional bin packing problem. *Journal of Heuristics*, 10(2):205–229.

[Bai et al., 2012] Bai, R., Blazewicz, J., Burke, E. K., Kendall, G., and McCollum, B. (2012). A simulated annealing hyper-heuristic methodology for flexible decision support. *4OR*, 10(1):43–66.

[Balas, 1965] Balas, E. (1965). An additive algorithm for solving linear programs with zero-one variables. *Operations Research*, 13(4):517–546.

[Baldi et al., 2012] Baldi, M. M., Crainic, T. G., Perboli, G., and Tadei, R. (2012). The generalized bin packing problem. *Transportation Research Part E: Logistics and Transportation Review*, 48(6):1205–1220.

[Baldi et al., 2019] Baldi, M. M., Manerba, D., Perboli, G., and Tadei, R. (2019). A generalized bin packing problem for parcel delivery in last-mile logistics. *European Journal of Operational Research*, 274(3):990–999.

[Barnhart et al., 1998] Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W., and Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3):316–329.

[Bhatia and Basu, 2004] Bhatia, A. and Basu, S. K. (2004). Packing bins using multi-chromosomal genetic representation and better-fit heuristic. In *International Conference on Neural Information Processing*, pages 181–186. Springer.

[Brandão, 2017] Brandão, F. (2017). *Cutting & Packing Problems: General Arc-flow Formulation with Graph Compression*. PhD thesis, Faculdade de Ciências da Universidade do Porto, Portugal.

[Brandao and Pedroso, 2016] Brandao, F. and Pedroso, J. P. (2016). Bin packing and related problems: general arc-flow formulation with graph compression. *Computers & Operations Research*, 69:56–67.

[Burke et al., 2003] Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., and Schulenburg, S. (2003). Hyper-heuristics: An emerging direction in modern search technology. pages 457–474.

[Burke et al., 2006] Burke, E. K., Hyde, M. R., and Kendall, G. (2006). Evolving bin packing heuristics with genetic programming. In *Parallel Problem Solving from Nature-PPSN IX*, pages 860–869. Springer.

[Cambazard and O'Sullivan, 2010] Cambazard, H. and O'Sullivan, B. (2010). Propagating the bin packing constraint using linear programming. In *International Conference on Principles and Practice of Constraint Programming*, pages 129–136. Springer.

[Caprara et al., 2001] Caprara, A., Monaci, M., and Toth, P. (2001). A global method for crew planning in railway applications. In *Computer-Aided Scheduling of Public Transport*, pages 17–36. Springer.

[Coffman et al., 1984] Coffman, E. G., Garey, M. R., and Johnson, D. S. (1984). Approximation algorithms for bin-packing—an updated survey. In *Algorithm design for computer system design*, pages 49–106. Springer.

[Correia et al., 2008] Correia, I., Gouveia, L., and Saldanha-da Gama, F. (2008). Solving the variable size bin packing problem with discretized formulations. *Computers & Operations Research*, 35(6):2103–2113.

[Crainic et al., 2007] Crainic, T. G., Perboli, G., Pezzuto, M., and Tadei, R. (2007). Computing the asymptotic worst-case of bin packing lower bounds. *European journal of operational research*, 183(3):1295–1303.

[Crainic et al., 2011] Crainic, T. G., Perboli, G., Rei, W., and Tadei, R. (2011). Efficient lower bounds and heuristics for the variable cost and size bin packing problem. *Computers & Operations Research*, 38(11):1474–1482.

[Dantzig and Wolfe, 1960] Dantzig, G. B. and Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research*, 8(1):101–111.

[Dupuis et al., 2010] Dupuis, J., Schaus, P., and Deville, Y. (2010). Consistency check for the bin packing constraint revisited. In *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*, pages 117–122. Springer.

[Dyckhoff, 1981] Dyckhoff, H. (1981). A new linear programming approach to the cutting stock problem. *Operations Research*, 29(6):1092–1104.

[Dyckhoff, 1990] Dyckhoff, H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research*, 44(2):145–159.

[Eilon and Christofides, 1971] Eilon, S. and Christofides, N. (1971). The loading problem. *Management Science*, 17(5):259–268.

[Eisemann, 1957] Eisemann, K. (1957). The trim problem. *Management Science*, 3(3):279–284.

[Falkenauer, 1996] Falkenauer, E. (1996). A hybrid grouping genetic algorithm for bin packing. *Journal of heuristics*, 2(1):5–30.

[Falkenauer and Delchambre, 1992] Falkenauer, E. and Delchambre, A. (1992). A genetic algorithm for bin packing and line balancing. pages 1186–1192.

[Fleszar and Charalambous, 2011] Fleszar, K. and Charalambous, C. (2011). Average-weight-controlled bin-oriented heuristics for the one-dimensional bin-packing problem. *European Journal of Operational Research*, 210(2):176 – 184.

[Fleszar and Hindi, 2002] Fleszar, K. and Hindi, K. S. (2002). New heuristics for one-dimensional bin-packing. *Computers & operations research*, 29(7):821–839.

[Ford and Fulkerson, 1958] Ford, L. R. and Fulkerson, D. R. (1958). A suggested computation for maximal multi-commodity network flows. *Management Science*, 5(1):97–101.

[Friesen and Langston, 1986] Friesen, D. and Langston, M. (1986). Variable sized bin packing. *SIAM Journal on Computing*, 15(1):222–230.

[Garey and Johnson, 1981] Garey, M. R. and Johnson, D. S. (1981). Approximation algorithms for bin packing problems: A survey. In *Analysis and design of algorithms in combinatorial optimization*, pages 147–172. Springer.

[Gent, 1998] Gent, I. P. (1998). Heuristic solution of open bin packing problems. *Journal of Heuristics*, 3(4):299–304.

[Gilmore and Gomory, 1965] Gilmore, P. and Gomory, R. E. (1965). Multistage cutting stock problems of two and more dimensions. *Operations research*, 13(1):94–120.

[Gilmore and Gomory, 1961] Gilmore, P. C. and Gomory, R. E. (1961). A linear programming approach to the cutting-stock problem. *Operations Research*, 9(6):849–859.

[Gilmore and Gomory, 1963] Gilmore, P. C. and Gomory, R. E. (1963). A linear programming approach to the cutting stock problem—part ii. *Operations research*, 11(6):863–888.

[Gómez-Meneses and Randall, 2009] Gómez-Meneses, P. and Randall, M. (2009). A hybrid extremal optimisation approach for the bin packing problem. In *Australian Conference on Artificial Life*, pages 242–251. Springer.

[Gouveia, 1995] Gouveia, L. (1995). A 2n constraint formulation for the capacitated minimal spanning tree problem. *Operations research*, 43(1):130–141.

[Gouveia and Saldanha-da Gama, 2006] Gouveia, L. and Saldanha-da Gama, F. (2006). On the capacitated concentrator location problem: a reformulation by discretization. *Computers & operations research*, 33(5):1242–1258.

[Gradišar et al., 1997] Gradišar, M., Jesenko, J., and Resinovič, G. (1997). Optimization of roll cutting in clothing industry. *Computers and Operations Research*, 24(10):945 – 953.

[Gupta and Ho, 1999] Gupta, J. N. and Ho, J. C. (1999). A new heuristic algorithm for the one-dimensional bin-packing problem. *Production planning & control*, 10(6):598–603.

[Haouari and Serairi, 2009] Haouari, M. and Serairi, M. (2009). Heuristics for the variable sized bin-packing problem. *Computers & Operations Research*, 36(10):2877–2884.

[Kallrath et al., 2014] Kallrath, J., Rebennack, S., Kallrath, J., and Kusche, R. (2014). Solving real-world cutting stock-problems in the paper industry: Mathematical approaches, experience and challenges. *European Journal of Operational Research*, 238(1):374 – 389.

[Kämpke, 1988] Kämpke, T. (1988). Simulated annealing: use of a new tool in bin packing. *Annals of Operations Research*, 16(1):327–332.

[Kang and Park, 2003] Kang, J. and Park, S. (2003). Algorithms for the variable sized bin packing problem. *European Journal of Operational Research*, 147(2):365–372.

[Kantarovich, 1939] Kantarovich, L. (1939). Mathematical methods in the organization and planning of production. *Publication House of the Leningrad State University.[Translated in Management Sc. vol 66, 366-422].*

[Koch et al., 2009] Koch, S., König, S., and Wäscher, G. (2009). Integer linear programming for a cutting problem in the wood-processing industry: a case study. *International Transactions in Operational Research*, 16(6):715–726.

[Labbé et al., 1991] Labbé, M., Laporte, G., and Mercure, H. (1991). Capacitated vehicle routing on trees. *Operations Research*, 39(4):616–622.

[Levine and Ducatelle, 2004] Levine, J. and Ducatelle, F. (2004). Ant colony optimization and local search for bin packing and cutting stock problems. *Journal of the Operational Research society*, 55(7):705–716.

[Liang et al., 2002] Liang, K.-H., Yao, X., Newton, C., and Hoffman, D. (2002). A new evolutionary approach to cutting stock problems with and without contiguity. *Computers & Operations Research*, 29(12):1641–1659.

[Lodi et al., 2002] Lodi, A., Martello, S., and Monaci, M. (2002). Two-dimensional packing problems: A survey. *European Journal of Operational Research*, 141(2):241 – 252.

[Loh et al., 2008] Loh, K.-H., Golden, B., and Wasil, E. (2008). Solving the one-dimensional bin packing problem with a weight annealing heuristic. *Computers & Operations Research*, 35(7):2283–2291.

[López-Camacho et al., 2011] López-Camacho, E., Terashima-Marín, H., and Ross, P. (2011). A hyper-heuristic for solving one and two-dimensional bin packing problems. In *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*, pages 257–258. ACM.

[Martello et al., 2000] Martello, S., Pisinger, D., and Vigo, D. (2000). The three-dimensional bin packing problem. *Operations Research*, 48(2):256–267.

[Martello and Toth, 1990a] Martello, S. and Toth, P. (1990a). *Knapsack problems: Algorithms and computer implementations*. Wiley and Sons, Chichester.

[Martello and Toth, 1990b] Martello, S. and Toth, P. (1990b). Lower bounds and reduction procedures for the bin packing problem. *Discrete Applied Mathematics*, 28(1):59 – 70.

[Martinovic et al., 2018] Martinovic, J., Scheithauer, G., and de Carvalho, J. V. (2018). A comparative study of the arcflow model and the one-cut model for one-dimensional cutting stock problems. *European Journal of Operational Research*, 266(2):458–471.

[Monaci, 2003] Monaci, M. (2003). Algorithms for packing and scheduling problems. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 1(1):85–87.

[Morabito and Garcia, 1998] Morabito, R. and Garcia, V. (1998). The cutting stock problem in a hardboard industry: A case study. *Computers and Operations Research*, 25(6):469 – 485.

[Poli et al., 2007] Poli, R., Woodward, J., and Burke, E. K. (2007). A histogram-matching approach to the evolution of bin-packing strategies. pages 3500–3507.

[Quiroz-Castellanos et al., 2015] Quiroz-Castellanos, M., Cruz-Reyes, L., Torres-Jimenez, J., Gómez, C., Huacuja, H. J. F., and Alvim, A. C. (2015). A grouping genetic algorithm with controlled gene transmission for the bin packing problem. *Computers & Operations Research*, 55:52–64.

[Rao, 1976] Rao, M. (1976). On the cutting stock problem.

[Rohlfshagen and Bullinaria, 2010] Rohlfshagen, P. and Bullinaria, J. A. (2007,2010). A genetic algorithm with exon shuffling crossover for hard bin packing problems. pages 1365–1371.

[Ross et al., 2003] Ross, P., Marín-Blázquez, J. G., Schulenburg, S., and Hart, E. (2003). Learning a procedure that can solve hard bin-packing problems: A new ga-based approach to hyper-heuristics. In *Genetic and Evolutionary Computation Conference*, pages 1295–1306. Springer.

[Ross et al., 2002] Ross, P., Schulenburg, S., Marín-Bläzquez, J. G., and Hart, E. (2002). Hyper-heuristics: learning to combine simple heuristics in bin-packing problems. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pages 942–948. Morgan Kaufmann Publishers Inc.

[Schaus et al., 2012] Schaus, P., Régin, J.-C., Van Schaeren, R., Dullaert, W., and Raa, B. (2012). Cardinality reasoning for bin-packing constraint: application to a tank allocation problem. In *Principles and Practice of Constraint Programming*, pages 815–822. Springer.

[Scholl et al., 1997] Scholl, A., Klein, R., and Jürgens, C. (1997). Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. *Computers and Operations Research*, 24(7):627–645.

[Schwerin and Wäscher, 1998] Schwerin, P. and Wäscher, G. (1998). *A new lower bound for the bin-packing problem and its integration into MTP*. Martin-Luther-Univ. Halle-Wittenberg, Wirtschaftswiss. Fak.

[Shaw, 2004] Shaw, P. (2004). A constraint for bin packing. In *International conference on principles and practice of constraint programming*, pages 648–662. Springer.

[Sim and Hart, 2013] Sim, K. and Hart, E. (2013). Generating single and multiple cooperative heuristics for the one dimensional bin packing problem using a single node genetic programming island model. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 1549–1556. ACM.

[Sim et al., 2012] Sim, K., Hart, E., and Paechter, B. (2012). A hyper-heuristic classifier for one dimensional bin packing problems: Improving classification accuracy by attribute evolution. In *International Conference on Parallel Problem Solving from Nature*, pages 348–357. Springer.

[Singh and Gupta, 2007] Singh, A. and Gupta, A. K. (2007). Two heuristics for the one-dimensional bin-packing problem. *OR Spectrum*, 29(4):765–781.

[Stadtler, 1990] Stadtler, H. (1990). A one-dimensional cutting stock problem in the aluminium industry and its solution. *European Journal of Operational Research*, 44(2):209 – 223. Cutting and Packing.

[Stawowy, 2008] Stawowy, A. (2008). Evolutionary based heuristic for bin packing problem. *Computers & Industrial Engineering*, 55(2):465–474.

[Trick, 2003] Trick, M. A. (2003). A dynamic programming approach for consistency and propagation for knapsack constraints. *Annals of Operations Research*, 118(1):73–84.

[Ülker et al., 2008] Ülker, Ö., Korkmaz, E. E., and Özcan, E. (2008). A grouping genetic algorithm using linear linkage encoding for bin packing. In *International Conference on Parallel Problem Solving from Nature*, pages 1140–1149. Springer.

[Valério De Carvalho, 1999] Valério De Carvalho, J. (1999). Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research*, 86:629–659.

[Valerio de Carvalho, 2002] Valerio de Carvalho, J. (2002). Lp models for bin packing and cutting stock problems. *European Journal of Operational Research*, 141(2):253 – 273.

[Vance et al., 1994] Vance, P. H., Barnhart, C., Johnson, E. L., and Nemhauser, G. L. (1994). Solving binary cutting stock problems by column generation and branch-and-bound. *Computational optimization and applications*, 3(2):111–130.

Resumen autobiográfico

Luis Ángel Gutiérrez Rodríguez

Candidato para obtener el grado de
Maestría en Ciencias en Ingeniería de Sistemas

Universidad Autónoma de Nuevo León
Facultad de Ingeniería Mecánica y Eléctrica

Tesis:
Problema generalizado del empaquetamiento de contenedores: una comparación entre diferentes métodos de solución

Nací un 16 de Marzo de 1993 en la ciudad de Monterrey, Nuevo León, México. Hijo primogénito de Luis Carlos Gutiérrez Martínez y Luz María Rodríguez Valero, ambos abogados de profesión. Durante mis primeros años de vida, mi madre me ayudó mucho en el desarrollo del pensamiento matemático, ya que nos enseñaba a mi hermano Carlos Antonio Gutiérrez Rodríguez y a mi a hacer operaciones de matemáticas, además de leer y escribir. Por otro lado, mi padre nos ayudo a desarrollar el pensamiento analítico mucho antes de siquiera aprender dicho concepto. Lo llamábamos lógica o sentido común, algo que era correcto y debía darse por sentado. Sin saberlo estaban formando a dos personas apasionadas por el saber. Todo ésto antes de ingresar a primaria.

Años más tarde, en 2010 a mis 17 años, ingresé a la Facultad de Ciencias Físico Matemáticas de la Universidad Autónoma de Nuevo León, a la Licenciatura en Ciencias Computacionales, donde acudí a varios foros de investigación como observador, generando en mi interés por la ciencia e investigación. Fui becario del Dr. Romeo Selvas en el laboratorio de Óptica por un semestre. También lo fui del Dr. Francisco Hernandez Cabrera durante un semestre en el laboratorio de Experimentos. Además, fui becario del Dr. Aurelio Ramirez Granados en los laboratorios de telecomunicaciones y base de datos del CSI-FCFM-UANL, donde me desempeñé como entrenador de competidores de los torneos de robótica que iniciaban durante el 2013, en octubre de 2014 me ofrecieron la oportunidad de trabajar como maestro de robótica en una secundaría. En julio de 2016 concluí mis estudios de licenciatura.

Entre octubre de 2014 y julio 2017 me desempeñé como profesor de secundaría de las materias de Física, Informática y Robótica en las Secundaria # 33 Çentro de alto rendimiento académicoz la Secundaria Federal # 4 Reforma", entrenando y preparando alumnos con conocimientos mas allá de los planes básicos de la educación secundaria, obteniendo como resultado, alumnos concursando en olimpiadas estatales y nacionales, concursando en eventos internacionales. En Física teniendo 30 alumnos concursando quedando entre los primeros 100 del estado de Nuevo León en 2016. En informática teniendo 120 alumnos participando en las olimpiadas de informática del estado de Nuevo León, clasificando a 12 de mis alumnos de los 15 seleccionados para prepararse a la competencia nacional. Quedando uno de mis alumnos en primer lugar nacional ganando Oro en el año 2016. En Robótica, participando en el torneo nacional de 2017 llevado a cabo en la Ciudad de México en el colegio Cristóbal Colón de La Salle, concursando con dos equipos, uno de cada secundaría donde laboré, en las categorías de Rescue line y Dance, quedando en rescue line dentro de los mejores 8 de la competencia, con la secundaría # 33 y con la Secundaría Federal # 4 en la categoría Dance en el 4to lugar de 84 grupos concursantes.

En 2018 contraje nupcias con Claudia Estefania Pacheco Aguirre en una ceremonia católica y civil.